Digital Whisper

גליון 102, ינואר 2019

מערכת המגזין:

מייסדים: אפיק קסטיאל, ניר אדר

מוביל הפרויקט: אפיק קסטיאל

עורכים: אפיק קסטיאל

כתבים: דן פייגין, Dvd848, YaakovCohen88 וטל בלום

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

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

דבר העורכים

את השנה החולפת פקדו לא מעט אירועים הקשורים לתחומנו, אך ככל הנראה האירוע שיזכר יותר מכל הוא כניסתה של רגולציית ה-GDPR לתוקף. ובתקווה, נשמע בשנים הקרובות פחות ופחות על מקרים שבהם חברות פשוט הפקירו מאגרי מידע על כל מליוני רשומותיהם חשופים לכל.

איכשהו, ברב השיחות שיצא לי לנהל אודות ה-GDPR, חזרו ועלו תמיד טענות כלפי הרשת החברתית פייסבוק, על איך הם שיתפו מידע עם חברה X, ועל איך לחברה Y הייתה גישה לקרוא את... כבר די נמאס לי לשמוע.

ואם זה לא מספיק, אז ממש לאחרונה פרסם ה-New York Times מחקר אשר במסגרתו ראיין מעל ל-60 עובדי חברת Facebook, ועבר על מעל ל-270 מסמכים פנימיים של החברה. מהמחקר עולה כי חברות שונות קיבלו שלל הרשאות לצפייה (ואף לעריכה?) של מידע, והתכתבויות שונות של משתמשים באותה הרשת. וכמובן, שגם הפעם, בדיוק כמו במקרה של Cambridge Analytica עלו אינסוף גינויים כלפי החברה.

אין לי מניות בפייסבוק, נשבע לכם, ולמעשה, יהיה קשה מאוד להגדיר אותי כמשתמש פעיל ברשת, אך מקריאת הגרסה העברית של ה-EULA של פייסבוק, **די עושה רושם שהם כותבים #000000 על גבי #FFFFFF שזה בדיוק מה שהם מתכוונים לעשות עם המידע שמשתמשי הרשת מספקים לה**.

למשל, מהשורה: "*אנו אוספים תוכן, התקשרויות ופרטי מידע נוספים שאתה מספק בעת השימוש במוצרינו, כולל כשאתה נרשם לחשבון, יוצר או משתף תוכן, ושולח הודעות או מתקשר עם אחרים*" - אני יכול להניח **שפייסבוק ככל הנראה אוספת עלי מידע**, כגון כל מה שאני משתף, וכל הודעה פרטית שאני בוחר לשלוח באחת מהפלטפורמות שלה.

ומהשורות: "*אנחנו עוסקים במחקר ומשתפים פעולה עם אחרים כדי לשפר את המוצרים שלנו. אחת הדרכים לכך היא על ידי ניתוח הנתונים העומדים לרשותנו והבנת האופן שבו אנשים משתמשים במוצרים שלנו*" ו-"*השותפים מקבלים את הנתונים שלך כשאתה מבקר בשירותים שלהם או משתמש בהם, או דרך גורמי צד שלישי שהם עובדים איתם.*" אני רק יכול לנחש **שיש להם שותפים שהם אינם פייסבוק**, ושהם כנראה **משתפים איתם פעולה ונותנים להם לנתח את הנתונים שאני בחרתי לשים במערכת שלהם**.

ורק כדי להסיר את הספק האחרון, במידה ואני משתמש סופר תמים, כתבו במיוחד לי: "*אנו אוספים מידע מתוך ואודות המחשבים, הטלפונים, הטלוויזיות המחוברות ומכשירים אחרים עם חיבור לאינטרנט שאתה*

משתמש בהם בשילוב עם המוצרים שלנו, ואנו משלבים מידע זה ממגוון המכשירים השונים שבהם אתה משתמש." - ולכן אני לא יכול שלא לנחש שהם אוספים עלי את המידע **בכל דרך שהם יכולים...**

אני לא מכיר את המודל העסקי של פייסבוק לפרטיו, אבל אני כן יודע לקרוא, ואם הם אומרים שהם עושים את זה - הם ככל הנראה עושים את זה. ויש שם כמובן עוד אינספור דוגמאות בסיגנון, ככה שקצת מוזר לי שמישהו מעיז להרים גבה בעניין הזה.

ה-GDPR הוא רפורמה מבורכת, אבל יש לה חסרון אחד מובהק (לפחות לדעתי) - היא מאפשרת למשתמשי האינטרנט להמשיך לפזר את המידע שלהם מבלי לעצור לרגע ולחשוב. הרי ברור שיש מי שינסה לעבור על החוק כדי להרוויח עוד קצת כסף, ותמיד יהיו את אלו שיתרשלו ובגללם שלל רשומות ה-DB יחשפו ויפורסמו באינטרנט. ואם הם יתפסו - החוק גם ידאג שהם ישלמו את הקנס. אבל לא משנה מה, הוא לא יוכל להחזיר לכם את הפרטיות שנפגעה.

GDPR או לא GDPR, את המידע הפרטי שלי לא שמתי בפייסבוק, ולכן לא משנה מה הם כותבים שהם עושים עם מה שיש להם ביד או מה הם לא כותבים וכן עושים עם מה שיש להם ביד - בהצלחה, הפרטיות שלי לא תלויה בשום חוק כזה או אחר.

ואני יודע שבמקרה הזה אני כנראה "משכנע את המשוכנעים", עובדה, אתם קוראים כרגע במגזין שעוסק בפרטיות, ושאת שאר האינטרנט כנראה לא נצליח לחנך. אבל אולי חבר'ה עם כח כמו מי שהצליח לחוקק חוק כמו ה-GDPR היו יכולים לעשות זאת...

ככל הנראה, שנת 2018 תזכר כשנה שבה ה-GDPR נכנס לתוקף. שזה טוב לנו בתור משתמשים. אבל לי קצת צובט הלב, כי אני יודע שביקום אחר, שנת 2018 הייתה יכולה להזכר בתור השנה שבה משתמשי האינטרנט היו מתחילים להיות קצת יותר קנאים לפרטיות שלהם.

קריאה נעימה,

אפיק קסטיאל וניר אדר

תוכן עניינים

דבר העורכים	2
תוכן עניינים	4
BGP Hijacking - או עד כמה קל להפיל את האינטרנט?	5
אתגר השב"כ 2018	12
Windows Notification Facility	48
דברי סיכום	60

BGP Hijacking - או עד כמה קל להפיל את האינטרנט?

מאת דן פייגין

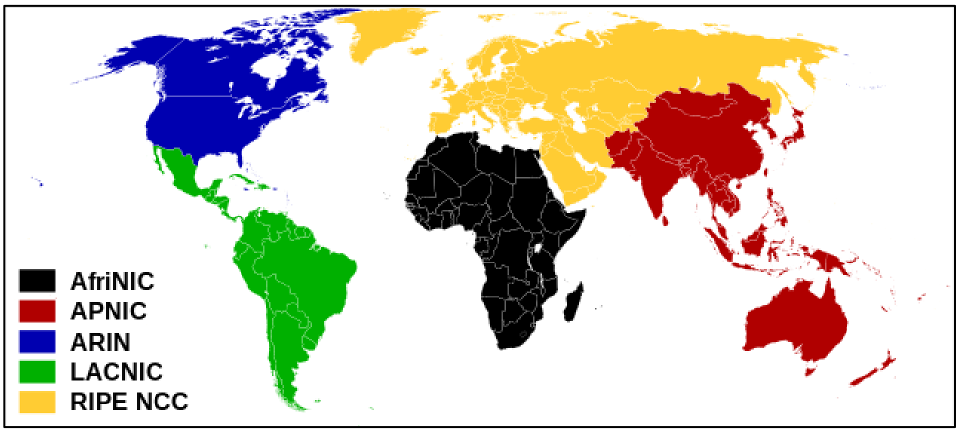
הקדמה

רבים מקוראי המגזין מכירים כיצד פועלות רשתות ביתיות, איך עובדים רכיבים כמו Router או Switch, וכנראה אף מכירים מספר סוגי תקיפות אשר רשתות כאלה והפרוטוקולים המנחים אותן מאפשרים. חבילת מידע אשר נשלחת ממחשב אל עבר הרשת אליה הוא מחובר, מועברת דרכה לספקית האינטרנט (ISP), נדרשת להגיע איכשהו למחשב אחר אשר נמצא ב-ISP אחר. על ניתוב החבילות הללו בין ה-ISPs, אשר חוצה מדינות ויבשות, מושל פרוטוקול ה-BGP (Border Gateway Protocol).

במאמר זה נכיר את הפרוטוקול ואת אופן הפעולה שלו, את מידת האבטחה שלו, ואף נראה סוג מתקפות לדוגמא ודוגמאות אמיתיות למתקפות BGP אשר תועדו בעבר.

IP Addresses and ASes

במישור ניהול האינטרנט, העולם מחולק לחמישה אזורים, ובכל אחד מהם קיים גוף אשר אחראי לניהול והקצאת כתובות IP (RIR - Regional Internet Registry). מרחב כתובות ה-IP העולמי מנוהל ע"י גוף בינלאומי בשם IANA (Internet Assigned Numbers Authority). גוף זה אחראי על הקצאת כתובות IP ל-RIRs השונים.



[מקור- ויקיפדיה:RIR]

היחידה הבסיסית לתיאור רשת אשר מחזיקה ברשותה טווח כתובות IP היא Autonomous System (AS). AS יכולה להיות ISP או כל ארגון גדול אחר אשר מחובר ישירות לאינטרנט, כאשר לכל AS מסופק מספר ייחודי (AS Number - ASN). ברשת האינטרנט, כל AS כזו מוגדרת כצומת ניתוב, כאשר בתוך הרשת

הפנימית שלה, הניתוב יכול להתבצע כראות עיניה, אך על מנת להתממשק מול העולם החיצון, על ה-AS להחזיק נתבי BGP.

ניתן דוגמא: כאן בתל אביב, מוקצה לי IP (חיצוני) מספקית האינטרנט שלי, Primo Communications LTD, לצורך העניין - 185.120.125.5. ה-IP הזה הוא חלק ממרחב כתובות גדול יותר - 185.120.125.0/24 (על CIDR notation ניתן לקרוא כאן). טווח זה מכונה Prefix. Prefix זה הינו בבעלות Primo, והוא מוכרז כחלק מ-AS8948. Primo מכריזה על prefixes נוספים, כאשר 185.120.125.0/24 הוא רק אחד מהם. כעת, נניח ואני רוצה לשלוח חבילת מידע לצידו השני של העולם. בסופו של דבר החבילה תעזוב את הרשת של ה-ISP שלי, ועל כן יש צורך בניתובה ל-ISP אחר. כאן BGP נכנס לתמונה.

Border Gateway Protocol

BGP בגרסתו הנוכחית (BGP4) נמצא בשימוש באינטרנט מאז 1994, ובבסיסו מורכב משני פרוטוקולים:

- **Interior BGP / iBGP**: פרוטוקול בין שני נתבי BGP בתוך AS, שמטרתו ללמד את הנתבים הפנימיים לגבי כתובות IP מחוץ ל-AS.
- **Exterior BGP / eBGP**: פרוטוקול בין שני נתבי BGP של ASes שכנים (BGP Peers). זה הפרוטוקול בו נתמקד על מנת להבין את הניתוב ברמת האינטרנט העולמי.

מכיוון שכל AS מחזיקה טווח כתובות IP, מטרת פרוטוקול ה-BGP היא כפולה:

1. לאפשר ל-AS להבין, בהינתן כתובת IP, לאיזה AS שייכת כתובת ה-IP, ומהו ה-path של ASes שדרכו על התעבורה לעבור על מנת שבסופו של דבר התעבורה תגיע ל-AS הנכונה.
2. לאפשר ל-AS להכריז בפני שכניה (BGP Speaker) אילו כתובות IP נמצאות ברשותה (origin), על מנת לקבל תעבורה המיועדת לכתובות IP אלו.

וכעת נצלול לפרוטוקול

כאשר AS כלשהי, למשל AS x, רוצה להכריז בפני שכניה, y ו-z, שיש ברשותה את טווח כתובות ה-IP a.b.c.d/16, היא שולחת חבילת BGP Announcement אשר מכילה את ה-prefix ואת ה-ASN שלה ל-peers שלה. כעת ה-peers של AS x יודעים שכדי לשלוח תעבורה לכתובת IP תחת ה-prefix a.b.c.d/16, עליהם להעביר תעבורה ישירות ל-AS x. כעת AS y ו-AS z יכולים לחלחל את הידע הזה הלאה לשכניהם, ע"י שרשור ה-ASN שלהם להודעה. כך ה-peers שלהם ידעו שכדי להגיע לכתובת ה-IP תחת ה-prefix הנ"ל יש להעביר תעבורה ל-y/z. על מנת להימנע ממעגלים, ASes מפילות BGP Announcements שמכילות את ה-ASN שלהן עצמן.

BGP Policies

מה קורה כאשר BGP peer מקבל שתי הצעות ניתוב לאותו prefix? התשובה כאן היא שבאופן עקרוני - אין לדעת. לכל AS יש policy משלה לגבי איזה BGP Announcement היא מעדיפה לקבל ואיך היא מעדיפה

לנתב את התעבורה שלה. ה-policies האלה מהוות סוד מסחרי של ה-AS, אך באופן אמפירי נראה שיש העדפה למסלולים קצרים ולכאלה שמיטיבים באופן כלכלי עם ה-AS (בין ASes יכולים להיות יחסי Provider-Customer, בהם תעבורה עוברת בתשלום, או יחסי Peer-Peer, בהם התעבורה אינה עוברת בתשלום).

BGP Sessions

BGP הוא פרוטוקול בשכבת האפליקציה בין שני BGP-enabled routers, כאשר מתחת לכובע הוא ממומש מעל TCP Session בפורט 179, כאשר כל BGP Speaker שולח הודעת Keep-Alive של 19 בתים מדי 60 שניות על מנת לשמור על חיבור.

כיצד ניתן להתחזות לנתב BGP? ובכן, ראשית יש לאתר שני נתבי BGP אשר קיים ביניהם session פעיל. כאמור, BGP רץ מעל TCP, ולכן יש צורך לבצע TCP Hijacking. לא נרחיב כיצד מבצעים TCP Hijacking על מנת שלא להיכנס ליותר פרטים. רק נציין שכדי לבצע זאת יש לשלוח חבילה עם פרמטרים נכונים, ובפרט sequence number נכון, אחרת החבילה תיפול (להרחבה ניתן לקרוא כאן). לכן אם ניתן להסניף את התעבורה של ה-Session או אם יש יכולת להיות MITM, זה הרבה יותר קל. לאחר מכן, כדי להכריז הודעת BGP כוזבת, ניתן להכין payload בינארי של BGP Update בעזרת כלים פומביים להרכבת חבילות (IPsend, Spoof), ולצרף אותו כ-payload לחבילת ה-TCP.

הנ"ל כמובן מתייחס למצב שבו התוקף אינו AS אלא מתפרץ ל-Session קיים. כאשר התוקף (במזיד או בטעות) הוא AS, אזי מעבר ליכולות פילטור עצמאיות ומשתנות של ASes אחרות, אין דבר המונע מ-AS להכריז הודעה כוזבת ולשבש תעבורה (דוגמאות בהמשך).

Prefix Hijacking

BGP הוא פרוטוקול מבוזר, כלומר, אין בסיס מידע ראשי שממנו שואבים כל ה-AS את מסלולי הניתוב האופטימאליים, והחלטות הניתוב יכולות להשתנות באופן דינאמי על בסיס המידע הנתון לכל AS באותו רגע.

Prefix Hijacking הוא שיטת השתלטות על קבוצת IPs במזיד או בטעות, אשר יכולה להיגרם מכל אחד מהתרחישים הבאים:

- BGP Speaker מכריז שהוא המקור של קבוצת IPs שאינה באמת ברשותו.
- BGP Speaker מכריז על prefix יותר ספציפי מה-prefix שהוכרז ע"י AS אחרת (prefix A מוגדר להיות יותר ספציפי מ-prefix B אם כל כתובות ה-IP של A מוכלות ב-prefix B. למשל, 8.8.8.8/20 יותר ספציפי מאשר 8.8.8.8/12). עבור כתובת IP ששייכת הן ל-prefix A והן ל-prefix B, העדפת הניתוב הדיפולטית היא ל-AS שהכריז על prefix יותר ספציפי.
- BGP Speaker מכריז על path המוביל ל-AS אשר יותר קצר (אינו בהכרח כזה שקיים) מכל ה-paths הזמינים.

בכל המקרים הללו, נתב eBGP יוסיף באופן דיפולטיבי את הכללים הנ"ל ל-routing table של ה-AS, ויחלחל באופן דיפולטיבי את מירב ההודעות שקיבל הלאה לנתבי eBGP אחרים.

באופן כזה, ניתן להשתלט על כלל התעבורה המיועדת לטווח כתובות IP מסוים. ניתן להעלות את השאלה : מדוע לתקוף בשיטה הזו? לכך יש מספר סיבות נפוצות:

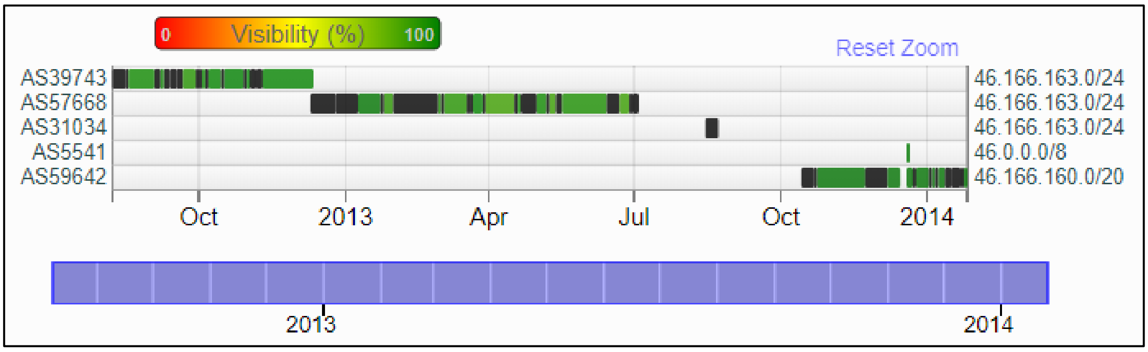
- גניבת כתובות IP "נקיות" לשם Spam או DDOS.
- לעיתים קשה לשחזר היסטוריה ולהבין מה היה ה-state של טבלת הניתוב וקשה למצוא תוקפים.
- MITM בוטה ופשוט - ניתן לכוון כנגד prefix, AS (ארגון) או קבוצת ASes (מדינה).

דוגמאות אמיתיות

1. עפ"י נתונים שנחשפו ע"י Wikileaks, ארגון ROS (ארגון בממשלת איטליה הנלחם בארגוני פשע) משתמש בתוכנת RAT (Remote Access Tool), תוכנה המאפשרת שליטה מרוחקת על מחשב, על מנת לעקוב אחר ארגוני הפשיעה במדינה. התוכנה מורכבת מ-client ו-server. ביולי 2013, נתקל הארגון בבעיה - ה-Prefix בו ישב ה-server אליו התחברו ה-clients, 46.166.163.0/24, הפך unreachable. על מנת לפתור את הבעיה במהירות, ולאפשר ל-clients להתחבר ל-IP תקין, ROS עבדה ביחד עם ספקית האינטרנט האיטלקית Aruba S.p.A (AS31034), ושתיהן הרימו server חדש עם כתובת ה-IP שאליה היו אמורים להתחבר ה-clients.

הן השיגו זאת ע"י כך ש-AS31034 הכריזה על BGP Announcement ומכילה את ה-Prefix שבו ישב ה-server, ומשכה אליה תעבורה שהייתה מיועדת ל-46.166.163.0/24 (לא כל התעבורה הגיעה אליה, שכן היו ASes שפילטרו את ההכרזות בהבנה שמדובר בהכרזות פיקטיביות). עם זאת, הפעולה עדיין הצליחה, וה-clients התחברו ל-server החדש, והיה ניתן לקנפג אותם להתחבר ל-IP חדש.

נקודה למחשבה - כיצד ניתן לאמת את הנתונים של המקרה? למזלנו, קיימים מאגריים עולמיים אשר מנטרים תעבורת BGP. מאגרים אלו מכונים collectors והם פזורים ברחבי העולם. בעזרת שירותי ויזואליזציה של מאגרי ניתוב שונים (ThousandEyes, BGPlay, וכו') ניתן לבחון היסטורית מה היה הסטטוס של כל prefix. אכן, ניתן לראות כי עבור תקופת הזמן הרלוונטית - קיץ 2013, ועבור ה-prefix הרלוונטי, קיימת קפיצה חדה בכמות ה-peers שחושבים שה-origin שלו הוא AS31034.



[מקור-ripestat.ripe.net]

2. בשנת 2008, ממשלת פקיסטן מחליטה לחסום את הגישה לאתר YouTube במדינה. אופן יישום ההחלטה הוא ע"י Prefix Hijacking. ה-IPs שפקיסטן רצתה לחסום היו: 208.65.153.238, 208.65.153.251, ו-208.65.153.23. על מנת לממש את החסימה, הספקית Pakistan Telecom (AS17557) הכריזה את ה-prefix 208.65.153.0/24 (כל הכתובות בטווח 208.65.153.255-208.65.153.0). עם זאת, באותה תקופה היה בבעלות YouTube (AS36561) ה-prefix 208.65.152.0/22, כלומר כל הכתובות בטווח 208.65.152.0-208.65.155.255. עם זאת, ה-prefix 208.65.153.0/24 הכיל את כל שרתי ה-DNS ושרתי ה-Web של החברה. מובן ש-/24 הוא יותר ספציפי מ-/22, וכאמור, נתבים מעדיפים באופן דיפולטי prefixes שהם יותר ספציפיים, ולכן ה-BGP Announcement חלחל הלאה, וכלל תעבורת YouTube הגיעה ל- Pakistan Telecom. גם כש-YouTube הכריזה על 208.65.153.0/24 זה לא עזר לפתרון הבעיה, שכן היא עדיין התחרתה עם Pakistan Telecom, ומי שזכה בתחרות הניתוב היה זה בעל הנתיב הקצר ביותר. התעבורה חזרה ל-YouTube רק כאשר היא הכריזה על ה-prefix 208.65.153.128/25, שהוא ספציפי יותר, וכך הנתיב אליה היה מועדף. בסה"כ YouTube הייתה ב-downtime של למעלה משעתיים. ניתן לשים לב בממקרה זה שתוקף יכול להרוג valid paths ע"י יצירה של bogus paths מתאימים.

קיימות דוגמאות רבות נוספות, החל ממעקב וצנזורה כנ"ל (חסימת Telegram ביולי 2018 ופורנוגרפיה בינואר 2017 ע"י איראן), עבור בגניבה של מטבעות וירטואליים (ע"י ישיבה כ-MITM על תעבורה רחבה - גניבת תעבורה של שרתי DNS של Amazon באפריל 2018 ושל ISPs קנדיים בפברואר 2014), וכלה בהשבתה המונית של חלקים נרחבים מהאינטרנט (ע"י סין ב-2010) והן מתרחשות באופן תדיר, חלקן במכוון וחלקן בטעות.

BGPsec ו-RPKI

מתקפות המנצלות את פרוטוקול ה-BGP קשות למניעה, בעיקר בגלל האופן שבו הפרוטוקול תוכנן, תכנון אשר אינו מאפשר את אימות דיוק נתוני הניתוב. על כן, קיימים מאמצים לשפר את BGP. שיפור משמעותי לכך מגיע בדמותה של BGPsec. הרחבה לפרוטוקול הכוללת מספר הגנות שנועדו להגדיל את רמת האבטחה של BGP, בעיקר ע"י שילוב יכולות קריפטוגרפיות. כל AS תחזיק רשימה חתומה דיגיטלית אשר מכילה מיפוי של IP prefixes ל-origin AS (פיצ'ר זה מכונה RPKI - Resource Public Key Infrastructure). בנוסף, כל AS תחזיק certificate על מנת לאפשר חתימה על paths, כך שבכל קפיצה בשרשרת הניתוב ה-AS תחתום על ה-path. בנוסף, BGPsec דורש את הוספת ה-ASN של ה-peer שההודעה מיועדת אליו, וגם חלק זה חתום דיגיטלית. כלומר, בכל פעם ש-AS תקבל BGP Announcement, היא תוכל לוודא שה-path נכון ושההודעה אכן מיועדת אליה, ע"י וידוא החתימה ע"י כל ה-ASes הרשומות ב-path, וכשתרצה לשלוח BGP Announcement חדש, היא תשרשר את ה-ASN שלה ושל ה-ASN של ה-peer אליו היא רוצה לשלוח את ההודעה, ותחתום על ה-path החדש עם המפתח פרטי שלה.

עם זאת, שיעור האימוץ של BGPsec די נמוך. BGP הוא פרוטוקול דינאמי שמנצל משאבי ריצה רבים, ועל כן תוכנן להיות יעיל. הודעות BGP updates מאובטחות הן בהכרח גדולות יותר, בגלל המידע הנלווה ובגלל תוספת החתימות. בנוסף, זמן עיבוד כל הודעה יגדל, בגלל שהוא דורש וידוא של מספר חתימות (כאורך ה-path), וזאת בכל hop בדרך. רק לשם המחשה, עם טבלת ניתוב גלובאלית בגודל של יותר מחצי מיליון IPv4 prefixes ועם path באורך ממוצע של 4 AS hops, מדובר בוידוא של יותר מ-2 מיליון חתימות באתחול של נתב BGP. על כן, ASes תדרשנה בחומרה מתאימה יותר לחישובים קריפטוגרפיים מהירים.

בנוסף, שימוש ביכולת אבטחתית כזו מתאפשר רק ב-deployment שבו קיימת שרשרת מקצה לקצה של ASes שאימצו את השיטה, אחרת לא ניתן יהיה לוודא את נכונות ה-path. רק החלפת החומרה לבדה מהווה הבטחה ל-deployment איטי של שדרוגי אבטחה מסוג זה, כך שנוסף על הצורך בהשקעה משותפת של ASes, זה לא מפתיע ש-ASes גדולים ומרכזיים לא ששים להחליף את כל נתביהם ברחבי

העולם. משום כך גם אימוצו של BGPsec ע"י ASes קטנים יותר אינו כדאי מבחינה כלכלית, ועל כן סך שיעור האימוץ נשאר קטן.

ובכל זאת - מה ניתן לעשות?

ב-BGP4 יש כשלים אבטחתיים חמורים, ולא ניתן לראות את אימוץ הפתרונות להם באופק הקרוב. על כן, כפתרון ביניים, חברות מאמצות מנגוני פיקוח - זיהוי אנומליות וזיהוי הסטת תעבורה, ניטור Sessions, כל זאת על מנת שבהינתן מתקפת BGP, יהיה ניתן לזהותה בזמן אמת ולתקנה (ע"י קנפוג ידני של הנתבים) לפני שהשפעתה על החברה ועל הלקוחות תורגש.

סיכום

BGP מאפשר לחבר בין רשתות שונות ברחבי העולם, והוא זה ששם את המילה Inter ב-Internet. עם זאת, הפרוטוקול תוקנן כשהמודעות לאבטחה הייתה בחיתוליה. על כן המצב כיום הוא שרוב תשתיות הניתוב העולמי מיושנות ובעלות כשלים אבטחתיים חמורים בשל השימוש ב-BGP. מכיוון שהפרוטוקול שזור בתשתית האינטרנט העולמי וקיים במספר רב של רשתות אשר מתופעלות ומתוחזקות ע"י מספר רב של ארגונים, אימוץ תיקונים אבטחתיים הוא מאתגר הן ברמה הטכנית והן ברמה האבטחתית.

קישורים

1. פירוט ה-prefix תחת כל AS:

https://www.dan.me.uk/bgplookup

2. ניתוח מקרה חטיפת BGP ע"י ראוטר מספקית קנדית לטובת מעקב / אחר כורי ביטקויין:

https://www.secureworks.com/research/bgp-hijacking-for-cryptocurrency-profit

3. ניתוח מקרה חטיפת BGP ע"י ספקית סינית לטובת גניבת וציטוט תעבורה מערבית:
https://scholarcommons.usf.edu/cgi/viewcontent.cgi?article=1050&context=mca

4. רשימת ה-ASes:

http://bgp.potaroo.net/cidr/autnums.html

5. אתר לתשאול מאגרי ניתוב ולמידע חדשותי בנושאי תקיפות BGP:

https://stat.ripe.net

אתגר השב"כ 2018

מאת Dvd848 ו-YaakovCohen88

הקדמה

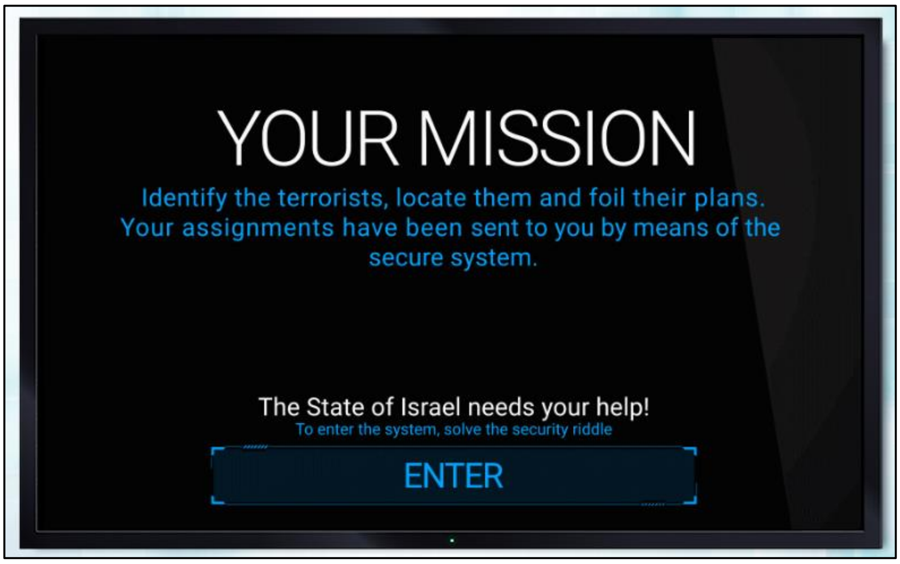
בתחילת דצמבר 2018 שחרר השב"כ סדרת אתגרים כחלק מקמפיין גיוס עבור אנשי טכנולוגיה. במאמר זה נציג את הפתרון שלנו לשניים מתוך ארבעת המסלולים של סדרת האתגרים.

חשוב לנו להגיד: אנחנו אומנם לא מכירים את היוצרים, אך הצלחנו להשיג מהם אישור לפרסום הפתרונות לפני מועד סגירת האתגר. ללא אישורם - מסמך זה לא היה מתפרסם.

סיפור רקע

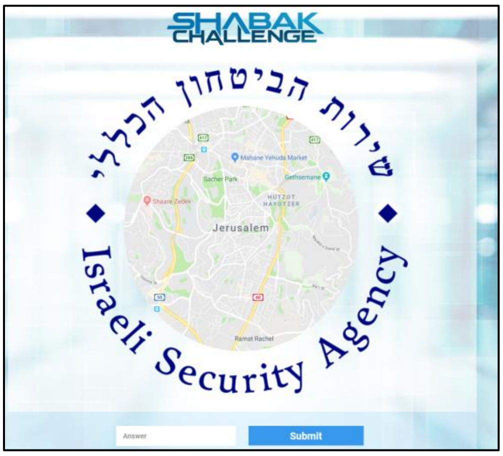
Hello Special agent A from the Israeli Security Agency (ISA) Technological Unit

“White September” (WS) is a group of arch-terrorists. They are connected to the global Jihadist movement, and are funded by Iran and Hezbollah. Several weeks ago, they used the darknet to declare their intentions of carrying out a mega terror attack in Israel. They nicknamed the operation “Israeli September 11th”. These people are highly sophisticated and utterly merciless. We at the ISA have received a tip that some of the terrorists have already infiltrated the country. Our agents have launched an operation to halt them before they can carry out their plot.

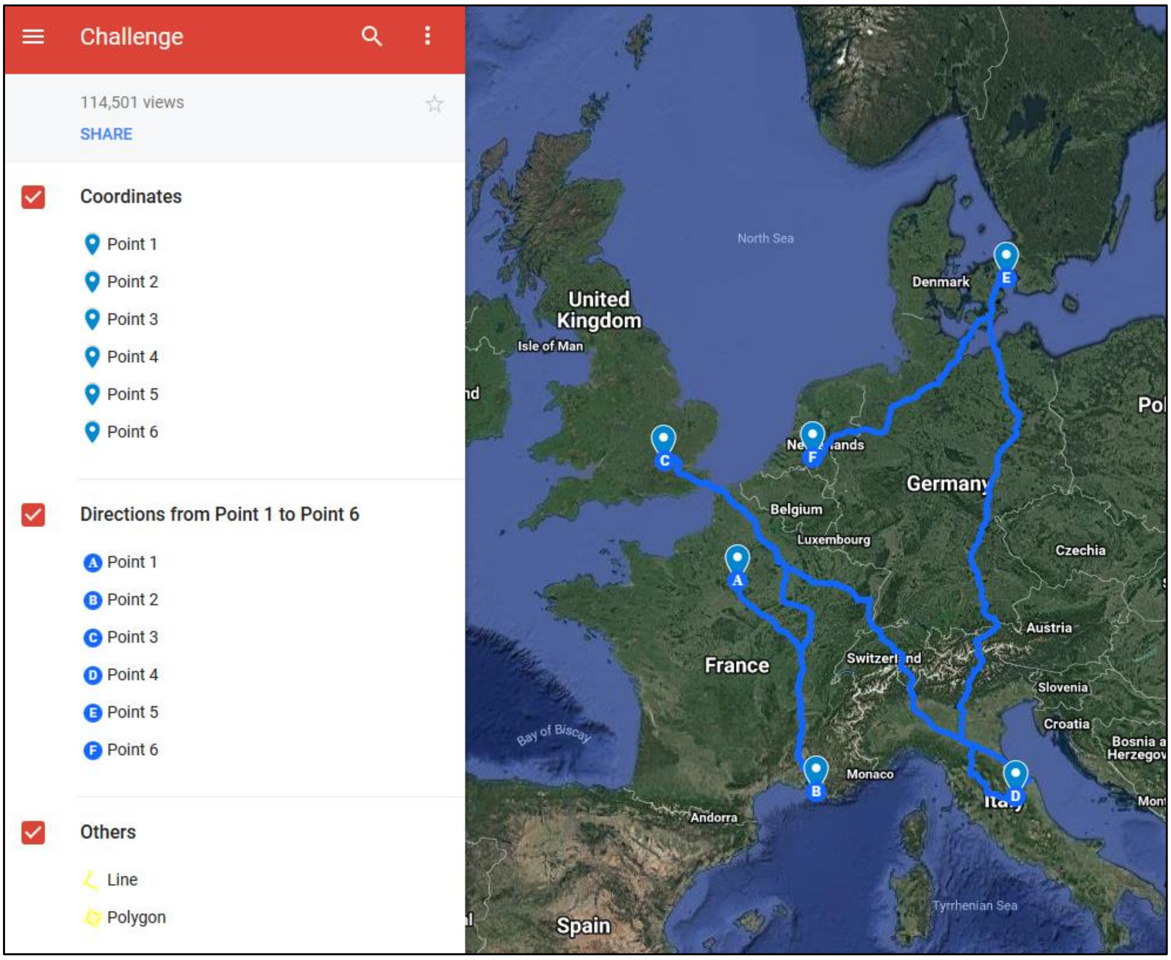


כניסה לאתגר

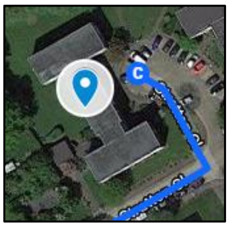
בדומה למה שראינו בשנים קודמות באתגרי המוסד, גם אתגר השב"כ הנוכחי נפתח עם תרגיל חימום שרק לאחריו ניתן להגיע אל רשימת האתגרים:

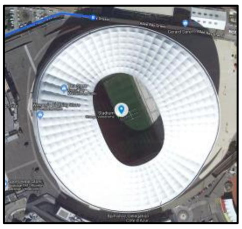


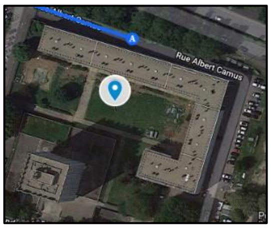
לחיצה על הלוגו מובילה אל מפה ב"גוגל מפות" שעליה שש נקודות ציון, ומסלול מהנקודה הראשונה אל האחרונה:

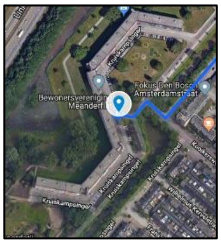


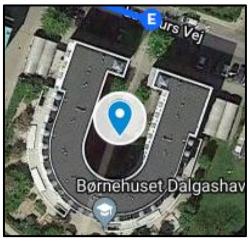
נבקר בנקודות השונות:





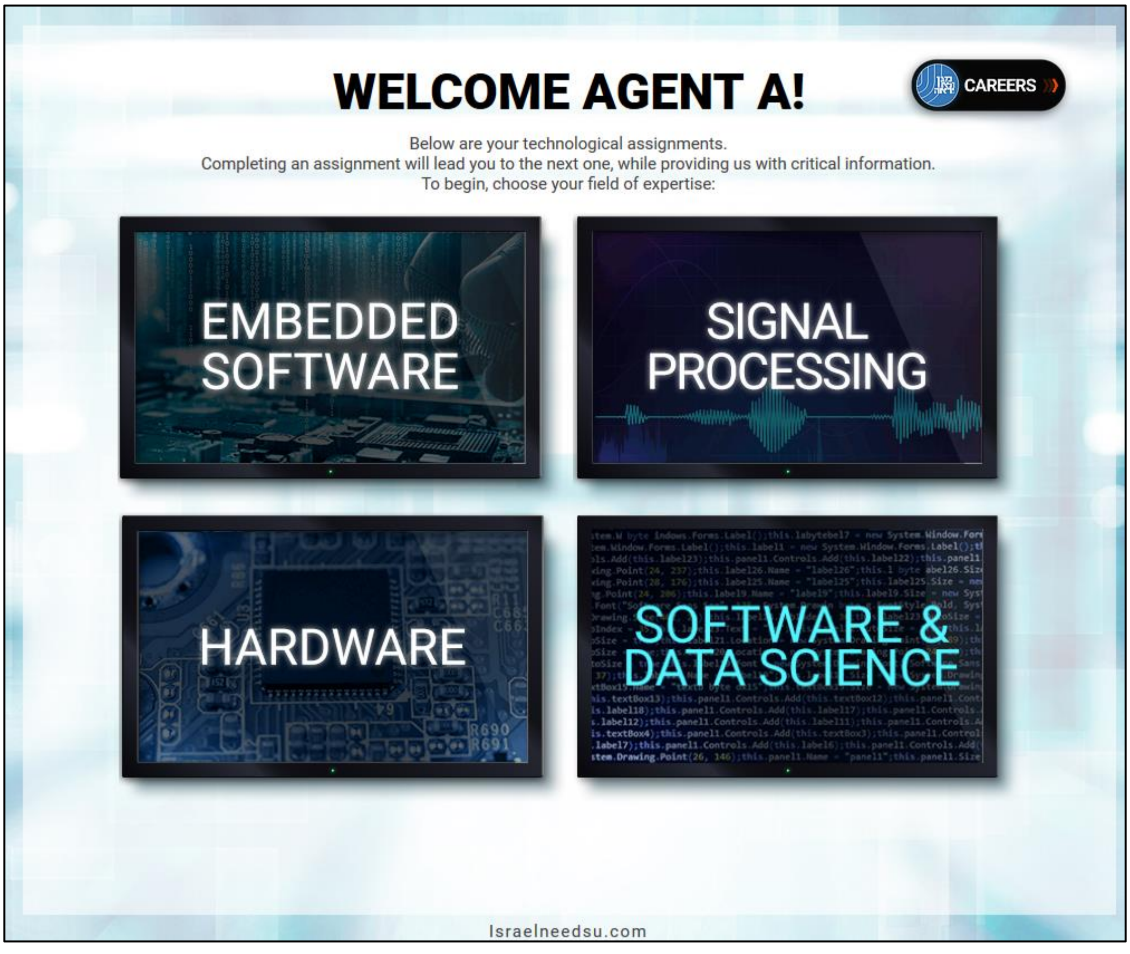






כל נקודה מציינת מבנה או סימן-נוף בצורה של אות באנגלית. אם נחבר את האותיות נקבל את התשובה: **joinus**.

התשובה מובילה אל ארבעת מסלולי האתגר:



מסלול Embedded Software

שאלה #1: Cat and Mouse

תיאור האתגר:

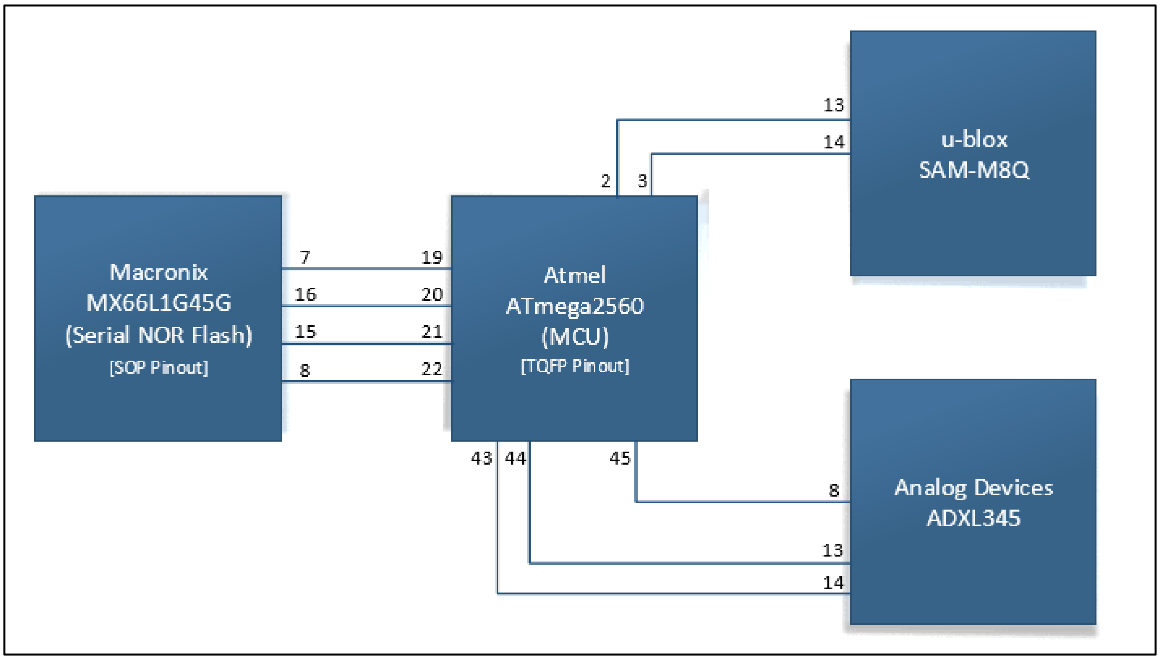
A routine counter-surveillance check of a senior Minister of Defense's vehicle revealed an electronic device in the undercarriage. We suspected it was a tracking device, and sent it to the Technological Department for an in-depth analysis.
After reverse-engineering the product, the following information was uncovered:

1. A partial scheme of the electrical circuit and its components (see the attached electronic_scheme.pdf file).
2. A disassembly of the code programmed to the micro controller (see the attached program.c file).
3. The memory dump of the external memory component (see attached external_mem_dump.bin file).

Earlier that week, we had intercepted a suspicious SMS sent by a suspected member of White September. The suspect's message read "package received".
The message was received on 30/10/18, at 01:21 AM UTC. Our analysis team analyzed the data and determined that the message had most likely been sent by the same WS member who had installed the device in the senior official's vehicle. The analysts suspect he retrieved it from a WS dead drop.
Our engineers believe that when the message was sent, the device was online, and that therefore the location of the dead drop could potentially be extracted from it.

Your mission:
Find the exact coordinates of the dead drop.

הקבצים המצורפים כללו, כאמור, תרשים של המכשיר:



קוד המקור:

```
#include <stdio.h>
#include <memory.h>
#include <stdlib.h>
#include <avr/io.h>
#include <avr/interrupt.h>

#define TRUE 1
#define FALSE 0

#define PARSER_INVALID 0
#define PARSER_TYPE_1 1
#define PARSER_TYPE_2 2

#define PARSER_PREFIX_1 "GPGGA"
#define PARSER_PREFIX_2 "GPRMC"
#define PARSER_PREFIX_LENGTH 5
#define MAX_UART_BUFFER 256
#define DATA_MAX_LENGTH 100
#define DATA_MIN_LENGTH 9
#define MAX_SAVE_BUFFER_SIZE 256

uint8_t should_save = FALSE;
uint8_t reset = TRUE;
uint8_t counter = 0;
uint8_t counter_max_val = 75;
uint8_t is_triggered = FALSE;

static uint32_t uart_read(uint8_t *buffer){/* Read data from UART0
until 0x00 terminator */};

static uint8_t parse(uint8_t *in_buffer, uint32_t length, uint8_t
**out_buffers)
{
  uint8_t i = 0;
  uint8_t res = PARSER_INVALID

  if (length < DATA_MIN_LENGTH || length > DATA_MAX_LENGTH ||
*in_buffer != '$' || in_buffer[length - 1] != '\n' || in_buffer[6] !=
',')
    return PARSER_INVALID;

  if (0 == strncmp(in_buffer + 1, PARSER_PREFIX_1,
PARSER_PREFIX_LENGTH))
    res = PARSER_TYPE_1;
  else if (0 == strncmp(in_buffer + 1,
PARSER_PREFIX_2,PARSER_PREFIX_LENGTH))
    res = PARSER_TYPE_2;
  else
      return PARSER_INVALID;

  in_buffer = in_buffer + PARSER_PREFIX_LENGTH + 2;

  out_buffers[i++] = in_buffer;
  while (NULL != (in_buffer = strchr(in_buffer, ',')))
  {
    *in_buffer = '\0';
    out_buffers[i++] = ++in_buffer;
  }
```

```
  return res;
}

static uint32_t format_save(uint8_t *in_buffer, uint8_t a, uint32_t
length, uint8_t *out_buffer)
{
  *out_buffer = a;
  *(uint32_t *)&out_buffer[1] = length;
  memcpy(&out_buffer[5], in_buffer, length);
  return length + 5;
}

static uint32_t format1(uint8_t **in_buffer, uint8_t *out_buffer)
{
  *(int *)out_buffer = atoi(in_buffer[0]);
  out_buffer += sizeof(float);
  *(int *)out_buffer = atoi(in_buffer[8]);
  return 2 * sizeof(float);
}

static uint32_t format2(uint8_t **in_buffers, uint8_t *out_buffer)
{
  char val1_str[10] = { 0 };
  for (uint8_t i = 0; i < 3; i += 2)
  {
    memset(val1_str, 0, sizeof(val1_str));
    uint8_t *pos = strchr(in_buffers[i], '.');
    memcpy(val1_str, in_buffers[i], pos - in_buffers[i] - 2);
    uint16_t val1 = atoi(val1_str);
    float val2 = atof(pos - 2);
    float res =  val1 + (val2 / 60);
    if (*in_buffers[i + 1] == 'W' || *in_buffers[i + 1] == 'S')
      res *= -1;
      *(float *)out_buffer = res;
      out_buffer += sizeof(float);
  }
  return 2 * sizeof(float);
}

static void configure1(void)
{
  cli();
  should_save = FALSE;
  counter = 0;
  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1 = 0;
  OCR1A = 62499;
  TCCR1B |= (1 << WGM12);
  TCCR1B |= (1 << CS12) | (1 << CS10);
  TIMSK1 |= (1 << OCIE1A);
  sei();
}
static void configure2(void)
{
  cli();
  EICRA |= 0x01 << 2;
  EIMSK |= 0x01 << 2;
```

```
  sei();
}

ISR(TIMER1_COMPA_vect)
{
  if (++counter == counter_max_val)
  {
    should_save = TRUE;
    counter = 0;
    counter_max_val = (is_triggered == TRUE) ? 15 : 150;
  }
}

ISR(INT2_vect)
{
  static uint8_t interrupt_buffer[MAX_SAVE_BUFFER_SIZE];
  uint8_t *temp_buffer;
  uint8_t a;
  uint32_t length;
  if (PIND & 0x01)
  {
     is_triggered = TRUE;
     a = 2;
  }
  else
  {
     is_triggered = FALSE;
     a = 3;
  }
  length = format_save(temp_buffer, a, 0, interrupt_buffer);
  save_to_flash(interrupt_buffer, length);
}

static void save_to_flash(uint8_t *buffer, uint32_t length){ /* Save
buffer to flash at next empty address */ }

static void configure_usart0(void) { /* Configure USART 0 */ }

static void configure_spi(void) { /* Configure SPI */ }

static void configure_two_wire_serial_interface(void) { /* Configure
the two wire serial interface */ }

int main()
{
  uint32_t size = 0;
  uint8_t recv_buffer[MAX_UART_BUFFER];

  uint8_t *parsed_buffers[20];
  uint8_t save_buffer[MAX_SAVE_BUFFER_SIZE];
  uint8_t temp_buffer[MAX_SAVE_BUFFER_SIZE];
  uint32_t save_length = 0;
  uint8_t res;
  uint32_t formatted_length;
  uint8_t a = FALSE;

  configure_sysclk(); /* Assume system clock is 16 MHz */
  configure_usart0();
  configure_spi();
```

```
  configure_two_wire_serial_interface();
  configure1();
  configure2();

  while (1)
  {
    formatted_length = 0;
    size = uart_read(recv_buffer);

    if (size > 0)
    {
      res = parse(recv_buffer, size, parsed_buffers);
      if (reset == TRUE)
      {
         if(res != PARSER_TYPE_2) continue;

         formatted_length = format1(parsed_buffers, temp_buffer);

         reset = FALSE;
         a = TRUE;
      }
      else if (should_save == TRUE)
      {
        if(res != PARSER_TYPE_1) continue;

        formatted_length = format2(&parsed_buffers[1], temp_buffer);

        should_save = FALSE;

        a = FALSE;
      }

      if(formatted_length > 0)
      {
        save_length = format_save(temp_buffer, (a == TRUE) ? 0 : 1,
formatted_length, save_buffer);
        save_to_flash(save_buffer, save_length);
      }
    }
  }
}
```

וקובץ בינארי שמכיל Memory Dump של המכשיר:

```
root@kali:/media/sf_CTFs/shabak/Cat_and_Mouse/program# xxd -g1
external_mem_dump.bin
00000000: 00 08 00 00 00 48 e8 01 00 ba 49 04 00 02 00 00  .....H....I.....
00000010: 00 00 01 08 00 00 00 aa 05 00 42 a9 8a 0b 42 01  ..........B...B.
00000020: 08 00 00 00 43 05 00 42 0e 8a 0b 42 01 08 00 00  ....C..B...B....
00000030: 00 7a 05 00 42 18 89 0b 42 01 08 00 00 00 27 06  .z..B...B.....'.
00000040: 00 42 0a 88 0b 42 01 08 00 00 00 9b 06 00 42 40  .B...B........B@
...
000057f0: 42 ac 28 0b 42 01 08 00 00 00 96 4c 00 42 60 28  B.(.B......L.B`(
00005800: 0b 42 01 08 00 00 00 c8 4c 00 42 21 28 0b 42 01  .B......L.B!(.B.
00005810: 08 00 00 00 9e 4c 00 42 00 29 0b 42 9e 4c 00 42  .....L.B.).B.L.B
00005820: 00 29 0b 42                                      .).B
```

מעיון בקוד עצמו נראה שפונקציית main מחכה לקלט סריאלי, ואז מבצעת עיבוד ראשוני באמצעות פונקציית parse. פונקציה זו מצפה לקבל קלט לפי אחד משני פורמטים: פורמט GPGGA ופורמט GPRMC. מחיפוש מהיר נראה שהפורמטים הללו מייצגים מידע הקשור ל-GPS.

התוכנה מקבלת את הקלט בלולאה, ראשית רשומה אחת של GPGGA ולאחריה רצף אינסופי של רשומות GPRMC. את הרשומות התוכנה מייצגת באמצעות ייצוג פנימי שלה (הפורמט של הודעת GPGGA נבנה בפונקציית format1 והפורמט של הודעת GPRMC נבנה בפונקציית format2). לאחר מכן, הייצוג נכתב אל זיכרון הפלאש של המכשיר במידה ומורם דגל לעשות כן. הדגל עבור רשומת GPGGA מורם פעם אחת בתחילת כל תוכנית, בעוד הדגל עבור רשומת GPRMC נשלט על ידי טיימר.

הטיימר קופץ בכל x יחידות זמן, מעלה משתנה-מונה באחת ומחליט לכתוב את הרשומה לפלאש במידה והמונה הגיע ל-counter_max_val.

מעבר למידע שמתקבל דרך הממשק הסריאלי, התוכנה מקבלת קלט גם דרך Pin D. כאשר ערך הסיגנל משתנה, מתקבל Interrupt והאירוע נכתב לפלאש. כמו כן, הערך של counter_max_val מתעדכן לפי האירוע הנ"ל.

הפורמט של השמירה לפלאש, כפי שניתן לראות מ-format_save, הוא בית אחד עבור סוג הרשומה:

- 0: הודעת GPGGA
- 1: הודעת GPRMC
- 2: סיגנל אשר מעיד על is_triggered = TRUE
- 3: סיגנל אשר מעיד על is_triggered = FALSE

לאחריו גודל המשך הרשומה (4 בתים) ומיד לאחר מכן תוכן באורך מספר הבתים שצוינו בשדה הקודם.

Type	Length	Payload
1 Byte	4 Bytes	<Length> Bytes

הן לרשומות GPGGA והן לרשומות GPRMC ישנו Payload של שמונה בתים: שני משתני Integer עבור GPGGA ושני משתני Float עבור GPRMC. מכאן אפשר לפרש את הרשומה הראשונה ב-dump שראינו קודם:

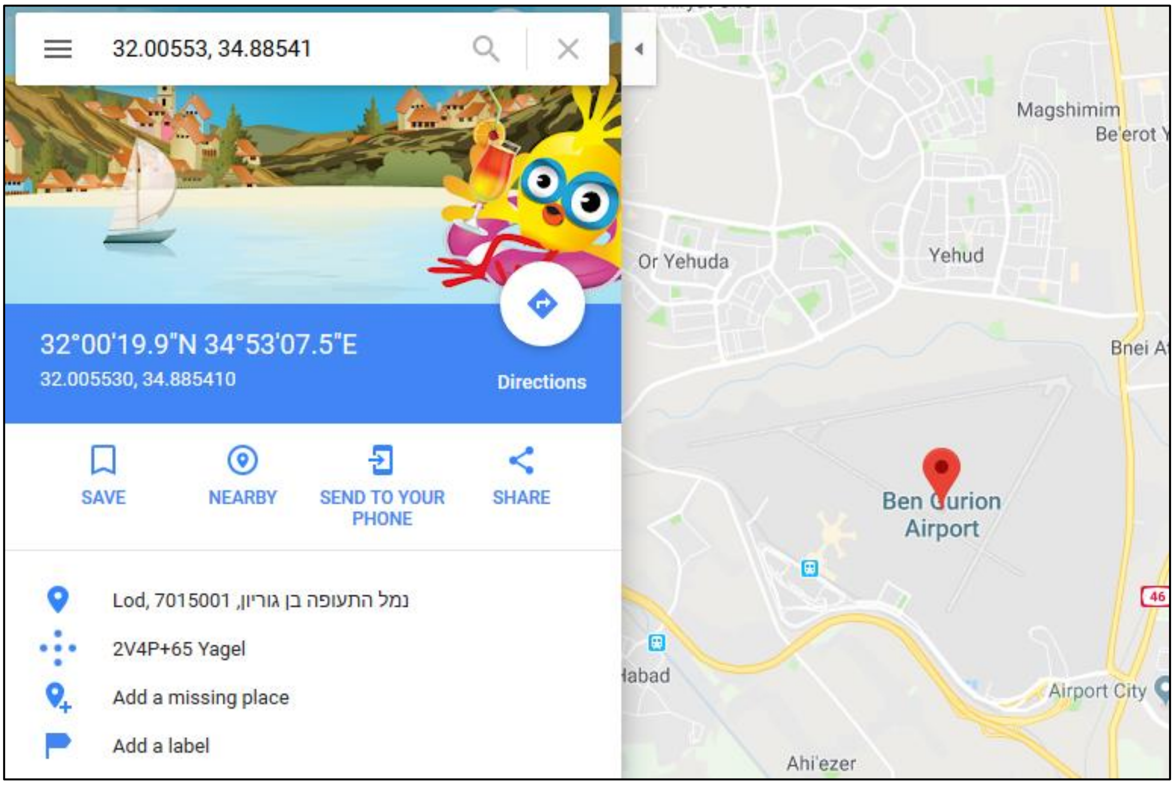
```
00 08 00 00 00 48 e8 01 00 ba 49 04 00
```

נראה שמדובר ברשומה מסוג 0 (GPGGA), באורך 0x00000008 בתים, עם הערכים 0x0001e848 ו-0x000449ba (שימו לב להיפוך שנגרם עקב השימוש בייצוג Little Endian). אם נתבונן בערכים בבסיס עשרוני נקבל 125000 ו-281018, מה שנראה כמו תאריך ושעה.

באותו אופן, נפרש את רשומת ה-GPRMC הראשונה שנפגוש (מתחילה ב-0x12):

```
01 08 00 00 00 aa 05 00 42 a9 8a 0b 42
```

זוהי רשומת GPRMC, באורך 8, עם הערכים 0x420005aa ו-0x420b8aa9. נפרש כ-Float ונקבל: 32.00553, 34.85541. אם נתייחס לנתונים אלו בתור קואורדינטות אורך ורוחב, נקבל מיקום בלב נמל התעופה בן גוריון:



אפשר להוציא כך את כל הקואורדינטות, ולקבל מפה של המסלול המדויק בו טייל המכשיר:

הקואורדינטות נותנות לנו את המסלול, אבל אנחנו צריכים להצליב את המסלול הזה עם הזמן המדויק שבה נשלחה ההודעה המפלילה. לכן אנחנו צריכים להבין מתי בדיוק כל רשומה מסוג GPRMC נכתבה אל הפלאש.

אנחנו יודעים שרשומת GPRMC נכתבת לפלאש ברגע שהטיימר מרים דגל של should_save, וזה קורה ברגע שהוא משלים counter_max_val קריאות. counter_max_val הוא 15 או 150, בהתאם לערך הנוכחי של is_triggered (למעט בתחילת התוכנה, שם הוא מתחיל כ-75). אנחנו יודעים את המצב הנוכחי של is_triggered כי כל שינוי שלו נכתב לפלאש, ולכן אנחנו יכולים להסיק את הערך של counter_max_val. כל מה שנשאר זה להבין כל כמה יחידות זמן קופץ הטיימר, ונוכל לחשב את הזמן שעובר בין כל כתיבה לכתיבה.

הקוד כולל את השורה הבאה:

```
configure_sysclk(); /* Assume system clock is 16 MHz */
```

שמעידה על שעון של 16 MHz. עובדה זו מסתדרת גם עם ה-spec של השבב שצוין בתרשים המצורף.

בנוסף, מחיפוש בגוגל, נראה שההגדרות הבאות קשורות גם הן לקביעת זמן ההתעוררות של הטיימר:

```
    OCR1A = 62499;
    TCCR1B |= (1 << WGM12);
    TCCR1B |= (1 << CS12) | (1 << CS10);
    TIMSK1 |= (1 << OCIE1A);
```

מצאנו את ההסבר המצוין הבא באתר הזה:

```
Timer1 is set to interrupt on an overflow, so if you are using an
ATmega328 with a 16MHz clock. Since Timer1 is 16 bits, it can hold a
maximum value of (2^16 - 1), or 65535. At 16MHz, we'll go through one
clock cycle every 1/(16*10^6) seconds, or 6.25*10^-8 s. That means 65535
timer counts will pass in (65535 * 6.25*10^-8s) and the ISR will trigger
in about 0.0041 seconds.
To control this you can also set the timer to use a prescaler, which
allows you to divide your clock signal by various powers of two, thereby
increasing your timer period.  For example, if you want the LED blink at
one second intervals. In the TCCR1B register, there are three CS bits to
set a better timer resolution.  If you set CS10 and CS12 using:
TCCR1B |= (1 << CS10); and TCCR1B |= (1 << CS12);, the clock source is
divided by 1024. This gives a timer resolution of 1/(16*10^6 / 1024), or
0.000064 seconds (15625 Hz). Now the timer will overflow every (65535 *
6.4*10^-5s), or 4.194s.
```

במקרה שלנו, הטיימר סופר עד 62499, ולכן הוא יקפוץ כל 3.999936 = 0.000064 * 62499 שניות, או בקירוב כל 4 שניות. ולכן, אם counter_max_val מוגדר ל-15, הטיימר יקפוץ כל דקה, ואם הוא מוגדר ל-150, הטיימר יקפוץ כל עשר דקות.

בעזרת המידע הזה נוכל לייצר את הסקריפט הבא:

```python
import os
import struct
from datetime import datetime, timedelta

ticks = 0
add_ticks = [75]

with open("external_mem_dump.bin", "rb") as f:
    while True:
        type = struct.unpack('B', f.read(1))[0]
        if type == "":
            break

        length = struct.unpack('I', f.read(4))[0]

        if type == 0:
            assert(length == 8)
            value1, value2 = struct.unpack('ii', f.read(8))
            print ("0\t{}\t{} - reset".format(value1, value2))
            ticks = 0
            add_ticks = [75]
            the_time = datetime.strptime("{} {}".format(value1, value2),
"%H%M%S %d%m%y")
        elif type == 1:
            assert(length == 8)
            if len(add_ticks) == 1:
                ticks_to_add = add_ticks[0]
            else:
                ticks_to_add = add_ticks.pop(0)
            ticks += ticks_to_add
            the_time += timedelta(seconds=ticks_to_add*4)
            value1, value2 = struct.unpack('ff', f.read(8))
```

```
            print ("1\t{:.5f}\t{:.5f}\t{}\t{}".format(value1, value2,
ticks, the_time))
        elif type == 2:
            assert(length == 0)
            print ("2\tis_triggered = TRUE (Log every 15 ticks)")
            add_ticks = [add_ticks[0], 15]
        elif type == 3:
            assert(length == 0)
            print ("3\tis_triggered = FALSE (Log every 150 ticks)")
            add_ticks = [add_ticks[0], 150]
        else:
            assert(False)
```

הסקריפט מממש את כל מה שהזכרנו, יחד עם נקודה אחת אחרונה שצריך לשים לב אליה: אם הערך של is_triggered משתנה, מה שיגרום לשינוי של counter_max_val, הטיימר הבא עדיין יקפוץ לפי הערך הישן, והערך של counter_max_val יתעדכן רק לאחר מכן לערך החדש.

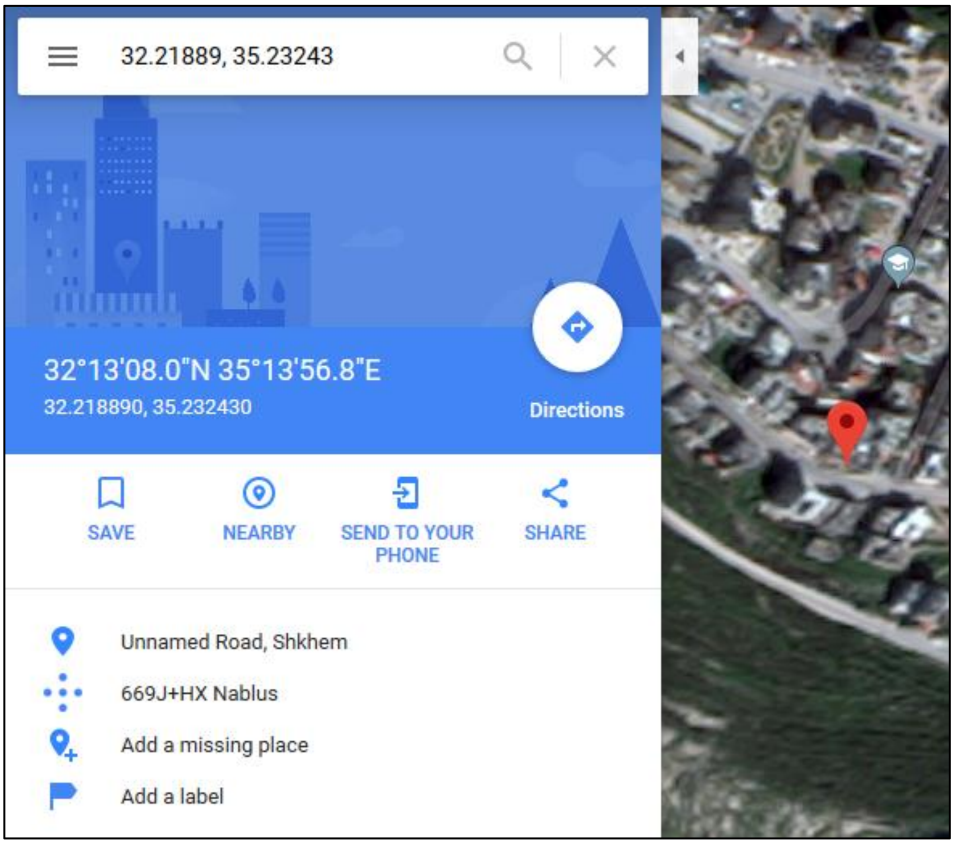
נריץ ונקבל:

```
0     125000      281018 - reset
2     is_triggered = TRUE (Log every 15 ticks)
1     32.00553          34.88541          75    2018-10-28 12:55:00
1     32.00514          34.88482          90    2018-10-28 12:56:00
1     32.00535          34.88388          105   2018-10-28 12:57:00
1     32.00601          34.88285          120   2018-10-28 12:58:00
1     32.00645          34.88208          135   2018-10-28 12:59:00
1     32.00696          34.88113          150   2018-10-28 13:00:00
1     32.00732          34.88019          165   2018-10-28 13:01:00
1     32.00666          34.87967          180   2018-10-28 13:02:00
…
1     32.21888          35.23234          29085 2018-10-30 00:36:00
1     32.21895          35.23225          29235 2018-10-30 00:46:00
1     32.21903          35.23241          29385 2018-10-30 00:56:00
1     32.21892          35.23242          29535 2018-10-30 01:06:00
1     32.21889          35.23243          29685 2018-10-30 01:16:00
1     32.21893          35.23226          29835 2018-10-30 01:26:00
1     32.21908          35.23232          29985 2018-10-30 01:36:00
1     32.21906          35.23253          30135 2018-10-30 01:46:00
1     32.21887          35.23255          30285 2018-10-30 01:56:00
…
```

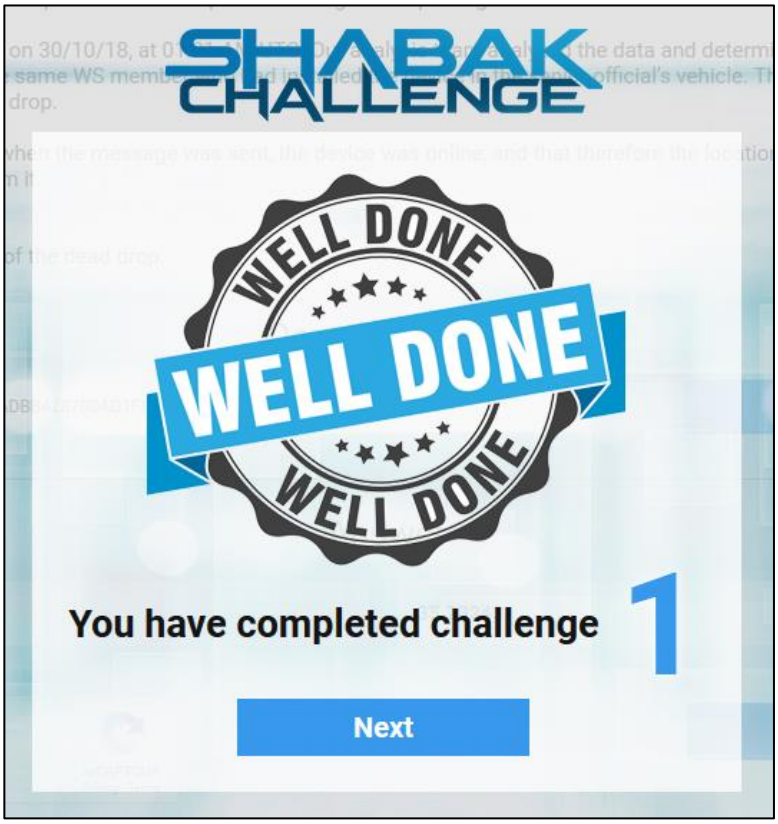
ההודעה נשלחה ב-30.10.2018 01:21, מה שאומר שהרשומה הקרובה ביותר היא:

```
1     32.21889          35.23243          29685 2018-10-30 01:16:00
```

הקואורדינטות הן: 32.21889, 35.23242. אי שם במרכז שכם:



והתוצאה:



שאלה #2: Code Red

תיאור האתגר:

We have uncovered some suspicious online traffic and identified communication between two IP addresses. Both these addresses come from computers we believe belong to senior members of the White September organization (see attached file chat.pcap).

The Technological Department believes that the WS members are using an application designed for concealing messages.

Based previously gathered intel, our analysts team assesses that WS is planning a terror attack against Israel in the near future. They believe the exact date was disclosed in the aforementioned communication.

Your Mission:
Reveal the exact date and time of the planned attack.

לאתגר צורף קובץ תעבורת רשת בשם chat.pcap.

אחד הדברים הראשונים שכדאי לעשות כאשר מקבלים קובץ כזה הוא לעקוב אחרי ה-TCP Streams השונים שלו. ניתן לעשות זאת באמצעות הממשק הגרפי של WireShark, או באמצעות הסקריפט הבא שמייצא כל Stream לקובץ טקסט:

```
END=$(tshark -r chat.pcap -T fields -e tcp.stream | sort -n | tail -1);
for ((i=0;i<=END;i++)); do echo $i; tshark -r chat.pcap -qz
follow,tcp,ascii,$i > follow-stream-$i.txt;done
```

במקרה שלנו, הפקודה ייצאה 39 סטרימים שונים. באמצעות מעבר ידני עליהם, זיהינו שהסטרימים המעניינים הם 3 ו-11, אשר בניגוד לשאר הסטרימים לא כללו גישה לאתרי אינטרנט שגרתיים אלא מעין דו-שיח בין שתי נקודות ברשת המקומית.

תחילה, בוצעה בקשת POST ל-sessions/ שהניבה את התשובה הבאה:


```
{"Id":"8d671e57-4f6a-4a7c-a22b-
724578cecbfa","Urls":["https://static.wixstatic.com/media/57cf4c_afea1f0
bb82348d9bdc24653ea3208f9~mv2.png","https://static.wixstatic.com/media/5
7cf4c_9fa5cba479a24e73a24fb52163d9209b~mv2.png","https://static.wixstati
c.com/media/57cf4c_4080f95bf84349e5887042c3a06f7114~mv2.png","https://st
atic.wixstatic.com/media/57cf4c_0bf3bbad5f74409bad0a3a10b1dbd537~mv2.png
","https://static.wixstatic.com/media/57cf4c_0aa6e7ffcc024f7ba2b6611f72f
2432d~mv2.png","https://static.wixstatic.com/media/57cf4c_d9f88c5ddc9348
8d91ac03c56cc901ae~mv2.png","https://static.wixstatic.com/media/57cf4c_5
6a9ed0fd9c84c98935307aebb4783f7~mv2.png"]}
```


לאחר מכן, מספר בקשות POST ל-messages/ שהניבו תוצאות מהסוג הבא:


```
{"SessionId":"8d671e57-4f6a-4a7c-a22b-724578cecbfa","Content":"<base64
encoded data>","Counter":0}
```

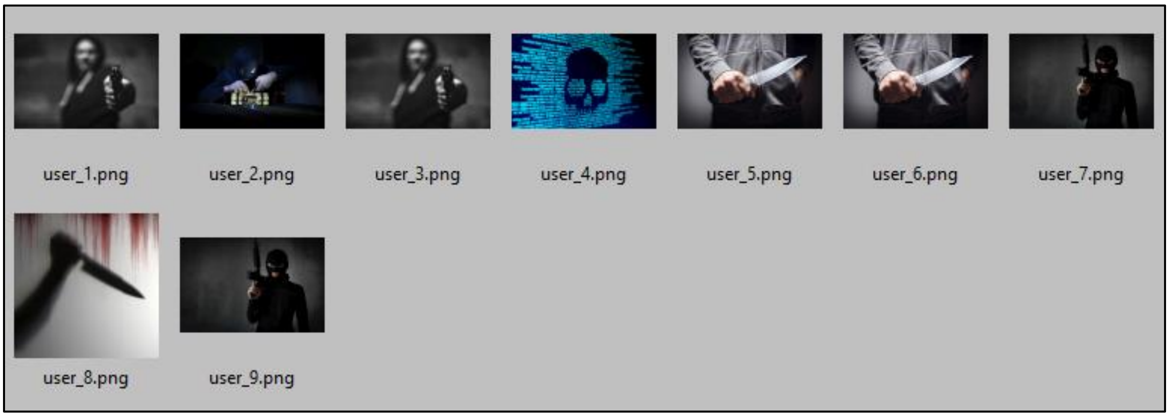

בכל תוצאה, Content הכיל מחרוזת Base64, ו-Counter הכיל מספר כלשהו.

עוד כדאי לציין שתגובת ה-HTTP הייתה chunked, כלומר, הפורמט של התשובה היה מספר שמציין את אורך ה-chunk, ולאחריו מספר בתים באורך זה. למשל:

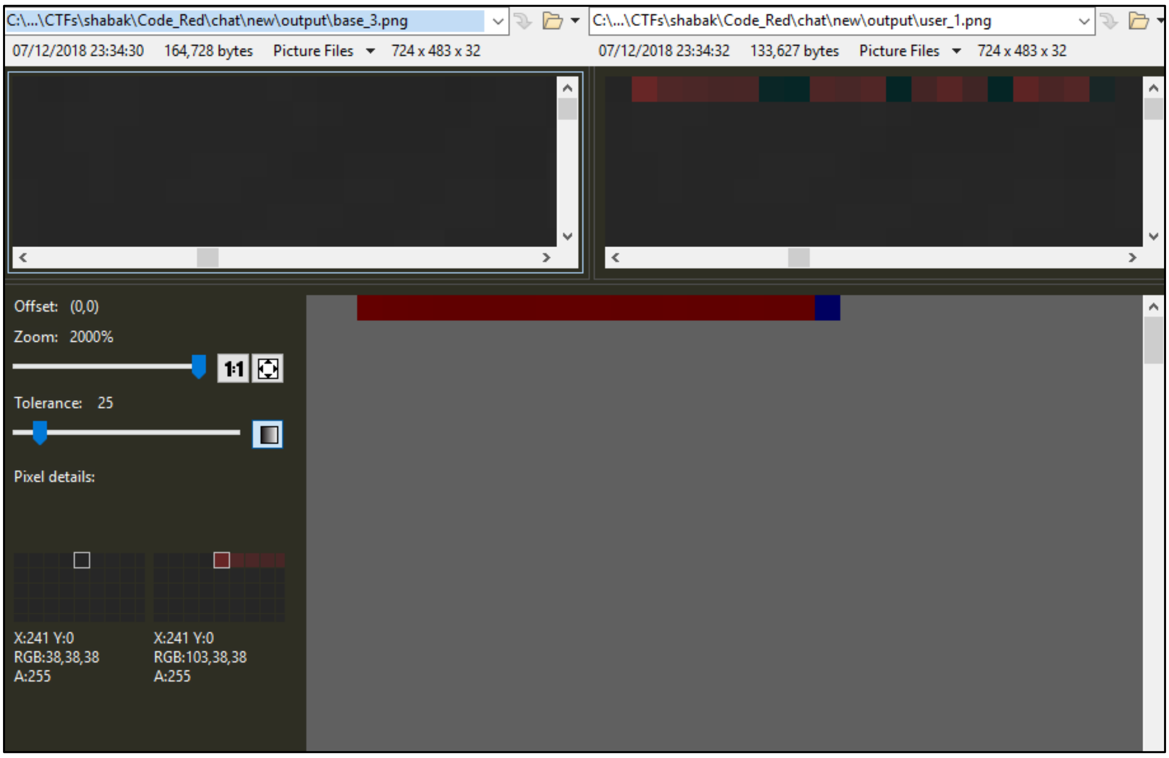
```
POST /messages HTTP/1.1
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8
Host: 192.168.202.128

400
{"SessionId":"8d671e57-4f6a-4a7c-a22b-724578cecbfa","Content":"<bas64
encoded chunk...>
400
<bas64 encoded chunk of length 0x400...>
400
<bas64 encoded chunk of length 0x400...>
400
<bas64 encoded chunk of length 0x400...>
```

כלומר, כדי לקבל אובייקט JSON נקי, אפשר לומר שהיה צריך לסנן החוצה כל שורה שנייה (אשר כוללת את אורך ה-chunk). נתחיל מאובייקט ה-JSON הראשון, שכולל רשימת תמונות, ונוריד את כל התמונות:

אם נתרגם את קידוד ה-base64 של האובייקטים הבאים ונייצא לקבצים, נקבל גם כן תמונות דומות (הקוד שעושה זאת יצורף בסוף):



התמונות שהורדנו והתמונות שחילצנו נראות דומות, אבל אם נשווה אותן פיקסל-פיקסל נגלה שישנו אזור מסוים שבו יש הבדל בערכי הצבע האדום. כך זה נראה ויזואלית:



אפשר לראות משמאל-למעלה את תמונת הבסיס (שהורדה מהאינטרנט) לצד התמונה שחולצה מאובייקט ה-JSON מימין, ולמטה סימון של הפיקסלים השונים. שימו לב שהתמונה המקורית רציפה, בעוד שנראה שהשינוי נעשה על התמונה המחולצת.

בפינה השמאלית התחתונה אפשר לראות את הערכים עבור פיקסל מסוים, כאשר ערך ה-R של התמונה המקורית הינו 38 והערך עבור התמונה המחולצת הינו 103. נבצע XOR ביניהם ונקבל 0x41, או בתרגום ל-ASCII, האות A. נמשיך כך ונקבל:

```
Ahlan, how are you?
```

נראה טוב! אבל אם נבצע את הפעולה הזאת על התמונות האחרות (כמובן, תוך הקפדה שאנחנו תמיד משווים בין תמונה מחולצת אשר דומה ויזואלית לתמונת בסיס), נקבל במקרים האחרים ג'יבריש! פה נכנס לפעולה ערך ה-Counter שראינו שצורך לאובייקט ה-JSON. מסתבר שיש לבצע XOR גם איתו. במקרה הראשון, הערך היה 0 כך שהפעולה לא הייתה הכרחית, אך במקרים האחרים הפעולה הזו נדרשת על מנת לקבל פלט הגיוני.

הסקריפט הבא:

- יוריד את התמונות הרלוונטיות מהאינטרנט
- יחלץ את התמונות הרלוונטיות מאובייקטי ה-JSON
- יזהה עבור כל תמונה מחולצת מהי תמונת הבסיס המתאימה באמצעות אלגוריתם בסיסי להשוואת תמונות

- ישווה בין ערכי ה-R של כל פיקסל ופיקסל בתמונת הבסיס (base) ובתמונה המחולצת (user), ויחשב את Counter ^ base[x,y][R] ^ user[x,y][R] במידה ו-base[x,y][R] != user[x,y][R]

```
from PIL import Image, ImageFont, ImageChops
import functools, requests, operator, logging
import base64, pickle, heapq, json, glob, math
import re, os

re_line_with_single_number = re.compile(r'^[0-9A-F]+$')
OUT_DIR = "output"
USER_IMAGE_PREFIX = "user_"
BASE_IMAGE_PREFIX = "base_"
CACHE_FILE = "cache.db"

#http://www.guguncube.com/1656/python-image-similarity-comparison-using-several-
techniques
def image_similarity_histogram_via_pil(filepath1, filepath2):

    image1 = Image.open(filepath1)
    image2 = Image.open(filepath2)

    h1 = image1.histogram()
    h2 = image2.histogram()

    rms = math.sqrt(functools.reduce(operator.add,list(map(lambda a,b: (a-b)**2,
h1, h2)))/len(h1) )
    return rms

def clean_json_str(s):
    lines = s.split("\n")
    return "".join(lines[::2]) # Return every other line

def extract_json_objects_from_stream(stream_id):
    res = []
    stream_file = "follow-stream-{}.txt".format(stream_id)
    logging.info("Extracting JSON objects from stream file:
'{}'".format(stream_file))
    with open(stream_file) as f:
        for raw_json_str in re.findall(r"(\{[^}]+\})", f.read()):
            j = json.loads(clean_json_str(raw_json_str))
            res.append(j)
    logging.info("Extracted {} JSON objects".format(len(res)))
    return res

def find_base_img(img):
    logging.info("Finding base image for '{}'".format(img))
    similarity = []
    for base_img in
glob.glob(os.path.join(OUT_DIR,"{}*.png".format(BASE_IMAGE_PREFIX)) ):
        score = image_similarity_histogram_via_pil(base_img, img)
        logging.debug("\tScore for '{}' is '{}'".format(base_img, score))
        heapq.heappush(similarity, (score, base_img))
    best_match = heapq.heappop(similarity)[1]
    logging.info("Best match for '{}': '{}'".format(img, base_img))
    return best_match

def decode_message(img1, img2, xor):
    image1 = Image.open(img1)
```

```
    image2 = Image.open(img2)

    assert(image1.size == image2.size)

    xSize = image1.size[0]
    ySize = image1.size[1]

    diff = []
    for i in range(xSize):
        for j in range(ySize):
            if image1.getpixel((i, j)) != image2.getpixel((i, j)):
                v1 = image1.getpixel((i, 0))[0]
                v2 = image2.getpixel((i, 0))[0]
                diff.append(chr(v1 ^ v2 ^ xor))
    return ("".join(diff))

logging.basicConfig(level=logging.INFO, format='Log:\t%(message)s')

json_objects = []
for stream_id in ["3", "11"]:
    json_objects += extract_json_objects_from_stream(stream_id)

if not os.path.exists(OUT_DIR):
    logging.info("Output directory missing, loading from scratch")
    os.mkdir(OUT_DIR)

    image_to_counter = dict()
    for i, obj in enumerate(json_objects):
        if "Urls" in obj:
            for url_index, url in enumerate(obj["Urls"]):
                logging.info("Fetching URL: {}".format(url))
                response = requests.get(url, stream=True)
                out_path =
os.path.join(OUT_DIR,"{}{}.png".format(BASE_IMAGE_PREFIX,url_index))
                logging.info("Saving to: {}".format(out_path))
                with open(out_path, "wb") as o:
                    o.write(response.content)
        else:
            out_image_name = "{}{}.png".format(USER_IMAGE_PREFIX, i)
            out_path = os.path.join(OUT_DIR, out_image_name)
            logging.info(
                "Saving image from object #{} (Counter: {}) to: {}".format(i,
obj["Counter"], out_path))
            with open(out_path, "wb") as o:
                o.write(base64.b64decode(obj["Content"]))
                image_to_counter[out_image_name] = int(obj["Counter"])

    pickle.dump( image_to_counter, open( CACHE_FILE, "wb" ) )
else:
    logging.info("Output directory missing, loading from cache")
    image_to_counter = pickle.load( open( CACHE_FILE, "rb" ) )

for img in glob.glob( os.path.join(OUT_DIR,"{}*.png".format(USER_IMAGE_PREFIX)) ):
    logging.info("Extracting message from '{}'".format(img))
    base_img = find_base_img(img)
    print ("\n", decode_message(img,
base_img,image_to_counter[os.path.basename(img)]),"\n")
```

הפלט:

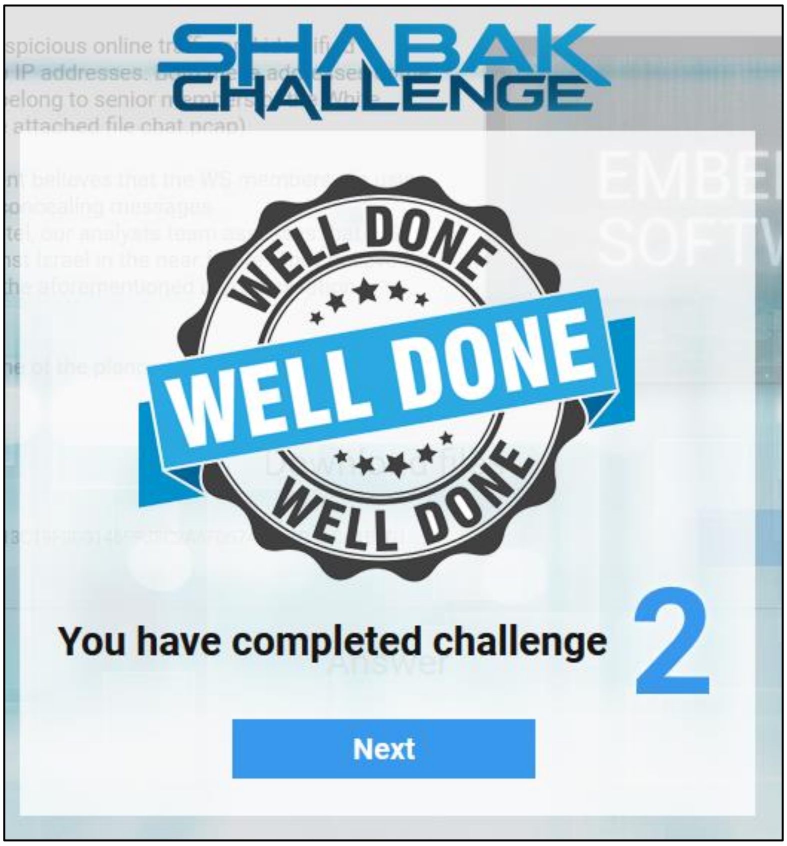
```
Log:    Extracting JSON objects from stream file: 'follow-stream-3.txt'
Log:    Extracted 6 JSON objects
Log:    Extracting JSON objects from stream file: 'follow-stream-11.txt'
Log:    Extracted 4 JSON objects
Log:    Output directory missing, loading from scratch
Log:    Fetching URL: https://static.wixstatic.com/media/57cf4c_afea1f0bb82348d9bdc24653ea3208f9~mv2.png
Log:    Saving to: output\base_0.png
Log:    Fetching URL: https://static.wixstatic.com/media/57cf4c_9fa5cba479a24e73a24fb52163d9209b~mv2.png
Log:    Saving to: output\base_1.png
Log:    Fetching URL: https://static.wixstatic.com/media/57cf4c_4080f95bf84349e5887042c3a06f7114~mv2.png
Log:    Saving to: output\base_2.png
Log:    Fetching URL: https://static.wixstatic.com/media/57cf4c_0bf3bbad5f74409bad0a3a10b1dbd537~mv2.png
Log:    Saving to: output\base_3.png
Log:    Fetching URL: https://static.wixstatic.com/media/57cf4c_0aa6e7ffcc024f7ba2b6611f72f2432d~mv2.png
Log:    Saving to: output\base_4.png
Log:    Fetching URL: https://static.wixstatic.com/media/57cf4c_d9f88c5ddc93488d91ac03c56cc901ae~mv2.png
Log:    Saving to: output\base_5.png
Log:    Fetching URL: https://static.wixstatic.com/media/57cf4c_56a9ed0fd9c84c98935307aebb4783f7~mv2.png
Log:    Saving to: output\base_6.png
Log:    Saving image from object #1 (Counter: 0) to: output\user_1.png
Log:    Saving image from object #2 (Counter: 3) to: output\user_2.png
Log:    Saving image from object #3 (Counter: 5) to: output\user_3.png
Log:    Saving image from object #4 (Counter: 6) to: output\user_4.png
Log:    Saving image from object #5 (Counter: 8) to: output\user_5.png
Log:    Saving image from object #6 (Counter: 1) to: output\user_6.png
Log:    Saving image from object #7 (Counter: 2) to: output\user_7.png
Log:    Saving image from object #8 (Counter: 4) to: output\user_8.png
Log:    Saving image from object #9 (Counter: 7) to: output\user_9.png
Log:    Extracting message from 'output\user_1.png'
Log:    Finding base image for 'output\user_1.png'
Log:    Best match for 'output\user_1.png': 'output\base_6.png'

 Ahlan, how are you?

Log:    Extracting message from 'output\user_2.png'
Log:    Finding base image for 'output\user_2.png'
Log:    Best match for 'output\user_2.png': 'output\base_6.png'

 Allahumdulillah! How are all the brothers?

Log:    Extracting message from 'output\user_3.png'
Log:    Finding base image for 'output\user_3.png'
Log:    Best match for 'output\user_3.png': 'output\base_6.png'

 We are all very proud of you.

Log:    Extracting message from 'output\user_4.png'
Log:    Finding base image for 'output\user_4.png'
Log:    Best match for 'output\user_4.png': 'output\base_6.png'

 We want the party to start at 20:10 exactly.

Log:    Extracting message from 'output\user_5.png'
Log:    Finding base image for 'output\user_5.png'
Log:    Best match for 'output\user_5.png': 'output\base_6.png'

 Inshallah, Allahu Akbar.

Log:    Extracting message from 'output\user_6.png'
Log:    Finding base image for 'output\user_6.png'
Log:    Best match for 'output\user_6.png': 'output\base_6.png'

 Ahlan habibi, I'm fine.

Log:    Extracting message from 'output\user_7.png'
Log:    Finding base image for 'output\user_7.png'
Log:    Best match for 'output\user_7.png': 'output\base_6.png'

 How are you?

Log:    Extracting message from 'output\user_8.png'
Log:    Finding base image for 'output\user_8.png'
```

```
Log:    Best match for 'output\user_8.png': 'output\base_6.png'

 They are all excited for new year's eve.

Log:    Extracting message from 'output\user_9.png'
Log:    Finding base image for 'output\user_9.png'
Log:    Best match for 'output\user_9.png': 'output\base_6.png'

 Inshallah, hopefully a lot of people will come.
```

השיחה עצמה:

```
Ahlan, how are you?
Allahumdulillah! How are all the brothers?
We are all very proud of you.
We want the party to start at 20:10 exactly.
Inshallah, Allahu Akbar.
Ahlan habibi, I'm fine.
How are you?
They are all excited for new year's eve.
Inshallah, hopefully a lot of people will come.
```

כלומר, המתקפה מתוכננת ל-31/12/18 בשעה 20:10:



אתגר #1: Find the Code

הוראות האתגר:

We have received intel that White September is trying to recruit Israeli residents to operate on its behalf within Israel. Our SIGINT unit discovered a seemingly innocent website where remote technological jobs are advertised. They believe this site is WS's recruiting front.

Based on the advertised jobs, the unit infers that WS is attempting to recruit tech agents from within key positions in Israeli security and government organizations.

After submitting a fake resume, the SIGINT agents received a password-protected ZIP file and an instruction document. The instructions informed them that the ZIP password is a series of digits and that the ZIP file contains Python code and two images. To advance to the next stage of recruitment, the unit's agents need to find a secret code in the image files.

Your Mission:

Unlock the ZIP file, extract its contents, and crack the code implanted in the images.

ראשית, נפצח את הסיסמא של קובץ ה-zip באמצעות John the Ripper:

```
root@kali:/media/sf_CTFs/shabak/Find_the_Code# ~/utils/john/run/zip2john clues.zip >
zip.hashes

ver 2.0 efh 5455 efh 7875 clues.zip/clue.png PKZIP Encr: 2b chk, TS_chk, cmplen=2424517,
decmplen=2428065, crc=4E28B3FE ver 2.0 efh 5455 efh 7875 clues.zip/clueTwo.jpg PKZIP Encr:
2b chk, TS_chk, cmplen=522, decmplen=12427, crc=B30EDBEE ver 2.0 efh 5455 efh 7875
clues.zip/something.txt PKZIP Encr: 2b chk, TS_chk, cmplen=299, decmplen=532, crc=729ED871
NOTE: It is assumed that all files in each archive have the same password. If that is not
the case, the hash may be uncrackable. To avoid this, use option -o to pick a file at a
time.

root@kali:/media/sf_CTFs/shabak/Find_the_Code# ~/utils/john/run/john --incremental=digits
zip.hashes
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Warning: OpenMP is disabled; a non-OpenMP build may be faster
Press 'q' or Ctrl-C to abort, almost any other key for status
262626 (clues.zip)
1g 0:00:00:00 DONE (2018-12-06 22:53) 1.851g/s 48474p/s 48474c/s 48474C/s 262507..262005
Use the "--show" option to display all of the cracked passwords reliably
Session completed

root@kali:/media/sf_CTFs/shabak/Find_the_Code# unzip -P 262626 clues.zip
Archive:  clues.zip
  inflating: clue.png
  inflating: clueTwo.jpg
  inflating: something.txt
```

מכיוון שידענו שהסיסמא מורכבת מספרות בלבד, הפיצוח לקח שניות בודדות. הסיסמה היא 262626.

נבחן את something.txt:

```
#env 3.7 from PIL import Image, ImageFont import textwrap from
pathlib import Path def find_text_in_image(imgPath): image =
Image.open(imgPath) red_band = image.split()[0] xSize =
image.size[0] ySize = image.size[1] newImage = Image.new("RGB",
image.size) imagePixels = newImage.load() for f in range(xSize):
for j in range(zSize) if bin(red_band.getpixel((i, j)))[-1] ==
'0': imagePixels[i, j] = (255, 255, 255) else: imagePixels[i, j]
= (0,0,0) newImgPath=str(Path(imgPath).parent.absolute())
newImage.save(newImgPath+'/text.png')
```

נראה כמו סקריפט פייתון. נסדר את השורות ואת הריווח, נתקן כמה שגיאות קלות ונסיר תלויות מיותרות:

```
#env 3.7
from PIL import Image, ImageFont
#import textwrap from pathlib
#import Path

def find_text_in_image(imgPath):
    image = Image.open(imgPath)
    red_band = image.split()[0]
    xSize = image.size[0]
    ySize = image.size[1]
    newImage = Image.new("RGB", image.size)
    imagePixels = newImage.load()
    for i in range(xSize):
        for j in range(ySize):
            if bin(red_band.getpixel((i, j)))[-1] == '0':
                imagePixels[i, j] = (255, 255, 255)
            else:
                imagePixels[i, j] = (0,0,0)
    #newImgPath=str(Path(imgPath).parent.absolute())
    newImage.save('./text.png')
```

הסקריפט מחפש תמונה נסתרת באמצעות התמקדות בפיקסלים האדומים של התמונה בלבד. כעת אפשר להריץ את הסקריפט על התמונה clue.png שמצאנו בתוך קובץ ה-zip.

התוצאה היא התמונה הבאה:

```
Binary, Start 10,000 place, Fibonacci
```

שימו לב שקיימת עוד תמונה, clueTwo.jpg, אך הסקריפט לא עוזר במקרה של התמונה הזו. כנראה שהמשפט שהוצאנו מ-clue.png אמור לעזור לנו לפענח רמז שמסתתר ב-clueTwo.jpg. clueTwo.jpg היא בסך הכל תמונה לבנה גדולה עם מלבן אפור קטן בפינה הימנית התחתונה (המסגרת השחורה אינה חלק מהתמונה ונועדה להדגשת הגבולות בלבד):

הפרט המעניין היחיד בתמונה הוא אזור מסוים בלב הייצוג הבינארי של התמונה, שבו קיים הטקסט הבא:

```
000004B0  00 A2 8A 28 00 A2 8A 28 00 A2 8A 28 00 A2 8A 28  .¢.(.¢.(.¢.(.¢.(
000004C0  00 A2 8A 28 00 A2 8A 28 00 A2 8A 28 00 A2 8A 28  .¢.(.¢.(.¢.(.¢.(
000004D0  00 A2 8A 28 00 A2 8A 28 00 A2 8A 28 00 A2 8A 28  .¢.(.¢.(.¢.(.¢.(
000004E0  00 62 79 6F 75 62 67 64 62 6F 78 7A 6D 6B 74 78  .byoubgdboxzmktx
000004F0  00 A2 8A 28 00 A2 69 71 72 A2 8A 28 00 A2 8A 28  .¢.(.¢iqr¢.(.¢.(
00000500  00 A2 8A 74 00 A2 8A 28 00 A2 8A 28 00 A2 8A 28  .¢.t.¢.(.¢.(.¢.(
00000510  00 A2 8A 28 00 A2 8A 28 00 A2 8A 28 00 A2 8A 28  .¢.(.¢.(.¢.(.¢.(
00000520  00 A2 8A 28 00 A2 8A 28 00 A2 8A 28 00 A2 8A 28  .¢.(.¢.(.¢.(.¢.(
```

ניסינו אינספור כלים וכיוונים על מנת לפצח את הרמז, עד שהגענו אל הפירוש הבא:

- יש להתייחס אל התמונה כבינארי
- יש לקפוץ אל הביט ה-10,000 של הקובץ
- משם, יש לחלץ את הבתים בהתבסס על קפיצות בהתאם לסדרת פיבונאצ'י

הסקריפט הבא מבצע זאת:

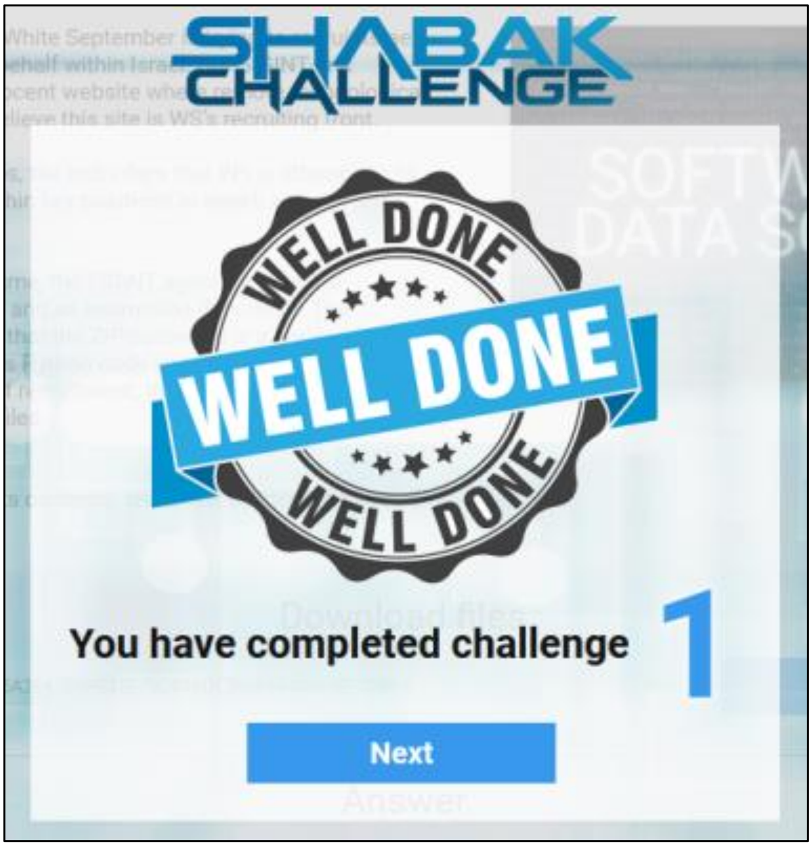
```
import mmap, os, string

def bits_to_bytes(bits):
    return bits // 8

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

def get_next_fib():
    a = 0
    b = 1
```

```
    while True:
        yield a
        s = a + b
        a = b
        b = s

with memory_map("clueTwo.jpg", access=mmap.ACCESS_READ) as f:
    index = bits_to_bytes(10000)

    for offset in get_next_fib():
        index += offset
        c = chr(f[index])
        if c not in string.printable:
            break
        print(c, end='')
```

התוצאה:

```
yougotit
```

באופן מפתיע ביותר, yougotit לא התקבל כתשובה, אבל you got it כן התקבל:



אתגר The Persian :#2

הוראות האתגר:

The code you discovered enabled our SIGINT unit to proceed in the recruitment process. The unit received a mysterious file. Finding the code in the file will enable us to further advance in the WS recruiting process.

Your Mission:

Analyze the file's content and discover the code

קובץ ה-zip המצורף הכיל קובץ אשר היה קרוי WhoAmI.jpg, אך תוכנו למעשה היה טקסטואלי. התוכן הכיל טקסט רב אשר במבט ראשון התאים לתבנית הבאה:

```
{"0x1": [{"text": "z9u05d3su05e9u05d7u05e4gz p2P <more...>", "value":
69349}, {...}, {...}]}
```

מה שבלט לעין היה החזרות הרבות של מחרוזות מסוג u05**XX** - ייצוג של טווח יוניקוד שמתאים לעברית.

הפעולה הטבעית הבאה הייתה לתרגם את התווים לעברית (שימו לב שרק לתווים מהתבנית u05XX קיים תרגום טריוויאלי לעברית, מה שמשאיר לא מעט ספרות ותווים באנגלית). התוצאה הייתה ג'יבריש. כמו כן, מחכה לנו נתון נוסף של value שלא השתמשנו בו.

אפשרות אחת, שאמנם נראתה קלושה במבט ראשון אך בדיעבד התגלתה בתור הכיוון הנכון, הייתה לחשב את הסכום הגימטרי של כל האותיות בעברית. הסכום שחושב עבור האובייקט הראשון התאים לערך הנקוב!

מכאן, הדבר הטבעי הבא היה לוודא את התוצאה עבור כל שאר האובייקטים. אולם, תוך כדי ריצה התגלו הבעיות הבאות:

1. לא לכל האובייקטים קיים ערך בשדה value, לעיתים במקום ערך היה כתוב פשוט "?".
2. לא כל האובייקטים הכילו שדה "text".

בפעם הראשונה שהשדה "text" היה חסר, הטקסט נכלל בשדה שנקרא "return" במקום. בפעם הבאה שהשדה היה חסר, הטקסט נכלל בשדה "in". בפעם השלישית, בשדה "base64", ולכולם כבר אמור להיות ברור שמדובר פה במסר נסתר עם הוראות נוספות.

לאחר שליפת כל השמות של השדות שהכילו טקסט ואשר לא נקראו "text", המסר שהתקבל היה:

```
return in base64 sum of values below median
```

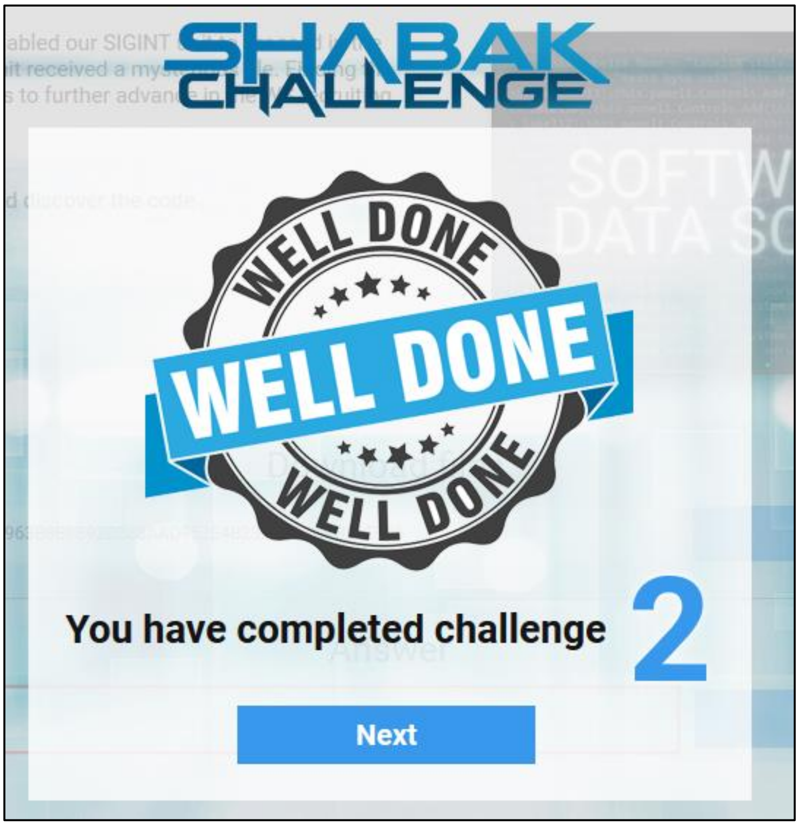
אם כך, כל שנותר הוא לעקוב אחרי ההוראות:

```
import statistics, logging, base64, json, re

heb_utf_regex = "u(05[a-f0-9]{2})"

gematria_map = {
    0x05D0: 1, # ℘span>
    0x05D1: 2, # Ȟspan>
    0x05D2: 3, # 工span>
    0x05D3: 4, # 甒span>
    0x05D4: 5, # 伯span>
    0x05D5: 6, # 弯span>
    0x05D6: 7, # 潔span>
    0x05D7: 8, # 缯span>
    0x05D8: 9, # 輯span>
    0x05D9: 10, # 鼯span>
    0x05DA: 20, # 꾻span>
    0x05DB: 20, # 뾈span>
    0x05DC: 30, # 켯span>
    0x05DD: 40, # span>
    0x05DE: 40, # □span>
    0x05DF: 50, # Ｏspan>
    0x05E0: 50, # □pan>
    0x05E1: 60, # □pan>
    0x05E2: 70, # □pan>
    0x05E3: 80, # □pan>
    0x05E4: 80, # pan>
    0x05E5: 90, # pan>
    0x05E6: 90, # pan>
    0x05E7: 100, # pan>
    0x05E8: 200, #
    0x05E9: 300, #
    0x05EA: 400, #

}

def get_gematria_sum(text):
    sum = 0
    for res in re.findall(heb_utf_regex, text):
        v = int(res, 16)
        sum += gematria_map[int(res, 16)]
    return sum

logging.basicConfig(level=logging.INFO, format='Log:\t%(message)s')

with open ("WhoAmI.jpg") as f:
    message = ""
    values = []

    all_text = f.read()
    all_text = all_text.replace("?", '"""')
    j = json.loads(all_text)

    for section_id, text_val_arr in j.items():
        logging.info("Section: " + section_id)
        for text_val in text_val_arr:
            text_label = "text"
            for x in text_val:
                if x == "value":
                    pass
                elif x != text_label:
                    message += x + " "
                    text_label = x
```

```python
            text = text_val[text_label]
            value = text_val["value"]
            calculated_value = get_gematria_sum(text)
            if (value != ""):
                logging.info("Gematria value from source: {}, calculated value:
{}".format(value, calculated_value))
                assert(int(value) == calculated_value)
            else:
                logging.info("Gematria value not found in source, calculated
value: {}".format(calculated_value))

            values.append(calculated_value)

    print("Secret message: {}".format(message))

    med = statistics.median(values)
    print("Median: {}".format(med))

    s = sum([v for v in values if v < med])
    print("Sum of values under median: {}".format(s))

    print("Base64: ", base64.b64encode(str(s).encode("utf-8")))
```

התוצאה:

```
Secret message: return in base64 sum of values below median
Median: 63664.5
Sum of values under median: 2501577
Base64:  b'MjUwMTU3Nw=='
```



The Usual Suspect :#3 אתגר

הוראות האתגר:

An ISA cyber-attack against White September's servers procured a set of data files. The files contained WS members' browsing activity. An analysis of the servers' logs revealed the conspiratorial activity of ten apparent suspects. We know there are more suspects but have been unable to identify them as of yet. The number of additional suspects is unknown.

Your Mission:

Identify the remaining suspects based on the activity of the provided list of suspects.

Find the most frequently used IP address for each of the additional suspects.

***Your solution should be submitted as a comma-separated list of IP addresses in ascending order, without any spaces or additional characters.**

For example: 11.1.11.1,2.22.22.2 (assuming two new suspects were found).

לאתגר צורף קובץ CSV של 10,000,000 רשומות (כ-600 מגה). הרשומות נראו כך:

```
$ head log.csv
uid, uip, date, url
8771,186.189.5.113,01/08/2014 17:24,http://jlt.edu/
7175,241.64.83.103,01/11/2014 12:54,http://sds-german.cc/
7903,224.4.135.115,01/03/2014 10:52,http://fdd.co.uk/
2690,211.105.56.228,01/01/2014 19:13,http://epc.org/
1518,198.77.195.195,01/11/2014 19:33,http://www.popshop.dk/
6589,225.108.49.121,01/07/2014 00:32,http://mend.co.uk/
6756,116.225.186.227,01/01/2014 20:44,http://simi.org/
7207,73.151.121.164,01/30/2014 07:19,http://mnj.cc/
1199,114.9.216.164,01/25/2014 14:21,http://lra.org/
```

כלומר, אוסף רשומות שכל אחת כוללת מזהה משתמש, כתובת IP, תאריך, וכתובת אתר. כמו כן, צורפה רשימה של חשודים:

```
2449, 6796, 9237, 4024, 3538, 3608, 7239, 435, 5206, 2211
```

בסך הכל, הרשימה כללה 10,000 רשומות אשר התייחסו לחשודים:

```
$ cat log.csv | egrep
"^(2449|6796|9237|4024|3538|3608|7239|435|5206|2211)," | wc -l
10000
$ cat log.csv | egrep
"^(2449|6796|9237|4024|3538|3608|7239|435|5206|2211)," | head
3538,67.141.120.237,01/16/2014 07:31,http://isa.edu/
5206,10.192.20.173,01/23/2014 17:51,http://fdd.co.uk/
4024,230.167.210.226,01/12/2014 23:26,http://apcls.info/
3608,127.95.83.100,01/24/2014
18:02,http://www.math.rutgers.edu/~sontag/fermi-eng.html
4024,143.204.212.207,01/16/2014 18:48,http://www.realestatebook.com/
7239,143.204.212.207,01/12/2014 20:26,http://emetic.edu/
```

```
435,42.74.74.110,01/02/2014 21:40,http://professional-edu.blogspot.com/
2211,109.242.247.39,01/06/2014
12:08,http://www.noodletools.com/noodlebib/
3538,51.70.255.188,01/10/2014 14:50,http://alansmith17.tumblr.com/
4024,230.167.210.226,01/13/2014 07:52,http://kdp.edu/
```

המחשבה הראשונית הייתה לחפש באילו אתרים ביקרו כל החשודים, ולראות אילו עוד משתמשים ביקרו באתרים הללו:

```
suspects = [2449, 6796, 9237, 4024, 3538, 3608, 7239, 435, 5206, 2211]
suspect_history = defaultdict(set)
non_suspect_history = defaultdict(set)
with open('log.csv') as f:
    csv_reader = csv.reader(f, delimiter=',')
    next(csv_reader, None)  # skip the headers
    for row in csv_reader:
        if int(row[UID]) in suspects:
            suspect_history[row[UID]].add(row[URL])
        else:
            non_suspect_history[row[UID]].add(row[URL])

    common_sites_for_suspects =
set.intersection(*list(suspect_history.values()))
    print ("Common sites for suspects: ", common_sites_for_suspects)
    counter = 0
    for user, history in non_suspect_history.items():
        if common_sites_for_suspects.issubset(history):
            counter += 1
    #          print (user)
    print ("Number of non-suspects who visited all common sites: ",
counter)
```

התוצאה: ישנם שני אתרים שכל החשודים ביקרו בהם ('http://cgsb.net', 'http://anla.gr/'), אבל 3128 משתמשים אחרים ביקרו בשניהם. לא מספיק טוב בשביל לצמצם את הרשימה. הרעיון הבא היה לבדוק אילו משתמשים השתמשו בכתובות IP שבהן השתמש חשוד כלשהו:

```
suspect_ips = set()
with open('log.csv') as f:
    csv_reader = csv.reader(f, delimiter=',')
    next(csv_reader, None)  # skip the headers
    for row in csv_reader:
        if int(row[UID]) in suspects:
            suspect_ips.add(row[IP])

    print("IPs used by suspects: ", suspect_ips)
    f.seek(0)
    next(csv_reader, None)  # skip the headers
    counter = 0
    for row in csv_reader:
        if int(row[UID]) not in suspects and row[IP] in suspect_ips:
            #print(row)
            counter += 1
    print ("Number of times which users used suspects' IPs: ", counter)
```

התוצאה: משתמשים רגילים השתמשו בכתובות של חשודים 9659 פעמים. יותר מדי. כדי לחשב מסלול מחדש, הבטנו בנתונים הגולמיים:

```
user_to_ips = defaultdict(set) # user to all the ips he used
with open('log.csv') as f:
    csv_reader = csv.reader(f, delimiter=',')
    next(csv_reader, None)  # skip the headers
    for row in csv_reader:
        user_to_ips[int(row[UID])].add(row[IP])

for user, ip_set in user_to_ips.items():
    print ("!" if user in suspects else " ", user, len(ip_set), ip_set)
```

הסקריפט מדפיס מיפוי של כל משתמש אל כל כתובות ה-IP שלו. אם המשתמש חשוד, בתחילת השורה יודפס סימן קריאה.

ראשית, נבחן את נתוני החשודים:

```
$ cat user_to_ips.txt | grep !
! 3538 10 {'127.95.83.100', '124.9.188.155', '162.219.33.114', … , '109.242.247.39'}
! 5206 10 {'127.95.83.100', '162.219.33.114', '10.192.20.173', … , '41.239.144.6'}
! 4024 10 {'230.167.210.226', '162.219.33.114', '10.192.20.173', … , '143.204.212.207'}
! 3608 10 {'127.95.83.100', '139.210.78.22', '124.9.188.155', … , '109.242.247.39'}
! 7239 10 {'127.95.83.100', '162.219.33.114', '10.192.20.173', … , '143.204.212.207'}
! 435 10  {'127.95.83.100', '162.219.33.114', '10.192.20.173', … , '41.239.144.6'}
! 2211 10 {'127.95.83.100', '162.219.33.114', '51.70.255.188', … , '41.239.144.6'}
! 2449 10 {'127.95.83.100', '124.9.188.155', '162.219.33.114', … , '41.239.144.6'}
! 6796 10 {'230.167.210.226', '10.192.20.173', '68.17.81.83', … , '143.204.212.207'}
! 9237 10 {'230.167.210.226', '10.192.20.173', '68.17.81.83', … '143.204.212.207'}
```

מה שקופץ פה לעין הוא החזרות הרבות של כתובות IP בין חשודים. למשל, הכתובת הראשונה של החשוד הראשון (127.95.83.100) היא גם הכתובת הראשונה של החשוד השני, הרביעי, החמישי, השישי, השביעי והשמיני. התופעה חוזרת עם כתובות אחרות. מכאן הגיע הרעיון "לצבוע" את הכתובות הללו ולראות אילו משתמשים השתמשו בכתובות אלו:

```
all_suspect_ips = set()
for suspect in suspects:
    all_suspect_ips.update(user_to_ips[suspect])

with open("user_to_ips.txt") as f, open("user_to_ips_colored.txt", "w")
as o:
    user_to_ips_content = f.read()
    for ip in all_suspect_ips:
        user_to_ips_content = user_to_ips_content.replace(ip,
"SUSPECT_IP")
    o.write(user_to_ips_content)
```

התוצאה, כאשר מסננים רק משתמשים שאינם חשודים (כלומר, ללא סימן קריאה):

```
$ cat user_to_ips_colored.txt | grep SUSPECT_IP | grep -v !
  7159 10 {'SUSPECT_IP', 'SUSPECT_IP', '219.62.226.13', 'SUSPECT_IP',
'126.127.244.219', 'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP',
'SUSPECT_IP'}
  5630 10 {'53.101.7.178', '176.143.187.81', 'SUSPECT_IP', '219.62.226.13',
'82.148.13.233', '126.127.244.219', '114.14.209.5', '58.176.126.71',
'7.199.192.98', '100.115.183.61'}
  9091 10 {'53.101.7.178', '176.143.187.81', 'SUSPECT_IP', '219.62.226.13',
'82.148.13.233', 'SUSPECT_IP', '126.127.244.219', '7.199.192.98', 'SUSPECT_IP',
'100.115.183.61'}
```

```
  1808 10 {'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP',
'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP'}
  5772 10 {'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP', '219.62.226.13',
'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP',
'SUSPECT_IP'}
  5590 10 {'53.101.7.178', '176.143.187.81', 'SUSPECT_IP', 'SUSPECT_IP',
'219.62.226.13', 'SUSPECT_IP', '126.127.244.219', '7.199.192.98', 'SUSPECT_IP',
'100.115.183.61'}
  7051 10 {'176.143.187.81', 'SUSPECT_IP', 'SUSPECT_IP', '219.62.226.13',
'SUSPECT_IP', '126.127.244.219', 'SUSPECT_IP', 'SUSPECT_IP', '100.115.183.61',
'SUSPECT_IP'}
  6529 10 {'176.143.187.81', 'SUSPECT_IP', 'SUSPECT_IP', '219.62.226.13',
'SUSPECT_IP', '126.127.244.219', '7.199.192.98', 'SUSPECT_IP', '100.115.183.61',
'SUSPECT_IP'}
  4918 10 {'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP',
'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP'}
  87 10 {'53.101.7.178', '176.143.187.81', 'SUSPECT_IP', '82.148.13.233',
'219.62.226.13', 'SUSPECT_IP', '126.127.244.219', '114.14.209.5',
'7.199.192.98', '100.115.183.61'}
  9482 10 {'176.143.187.81', 'SUSPECT_IP', 'SUSPECT_IP', '219.62.226.13',
'SUSPECT_IP', '126.127.244.219', 'SUSPECT_IP', 'SUSPECT_IP', 'SUSPECT_IP',
'SUSPECT_IP'}
```

זה נראה כמו הכיוון הנכון - ישנם משתמשים (כמו למשל 4918) שכל כתובות ה-IP שלהם הן כתובות של חשודים!

נייצר סקריפט שיבצע את הזיהוי כולו מההתחלה ועד הסוף:

```
from collections import defaultdict, Counter
import logging, pickle, socket, time, csv, os

def most_common(lst):
    data = Counter(lst)
    return data.most_common(1)[0][0]

UID      = 0
IP       = 1
DATE     = 2
URL      = 3

LOG_FILE     = "log.csv"
CACHE_FILE   = "cache.db"

suspects = []

logging.basicConfig(level=logging.INFO, format='Log:\t%(message)s')

with open("hint.txt") as hint_f:
    for suspect in hint_f.readlines():
        suspect = suspect.rstrip()
        suspects.append(suspect)

logging.info("Suspects: {}".format(suspects))

if not os.path.exists(CACHE_FILE):
    logging.info("Cache file missing, loading from scratch")
    user_to_ips              = defaultdict(set)   # user to all the ips he used
    ips_to_user              = defaultdict(set)   # ip to all the users which used it
    user_to_nonunique_ips    = defaultdict(list) # user to list of IPs he used
    user_to_most_common_ip   = dict()          # user to most common IP he used
    all_suspect_ips          = set()           # Set of all suspect IPs
```

אתגר השב"כ 2018

www.DigitalWhisper.co.il

```
    with open(LOG_FILE) as f:
        time1 = time.time()
        csv_reader = csv.reader(f, delimiter=',')
        logging.info("Parsing log file...")
        for i, row in enumerate(csv_reader):
            user_to_ips[row[UID]].add(row[IP])
            ips_to_user[row[IP]].add(row[UID])
            user_to_nonunique_ips[row[UID]].append(row[IP])
        time2 = time.time()

    logging.info("Parsed {} rows in {} seconds".format(i, time2-time1))

    logging.info("Number of unique users:
{}".format(len(user_to_ips.keys())))
    logging.info("Number of unique IPs:
{}".format(len(ips_to_user.keys())))

    logging.info("Calculating most common IP per user...")
    for user, ip_list in user_to_nonunique_ips.items():
        mc = most_common(ip_list)
        user_to_most_common_ip[user] = mc

    logging.info("Searching for all suspect IPs...")
    for suspect in suspects:
        all_suspect_ips.update(user_to_ips[suspect])

    logging.info("All suspect IPs: {}".format(all_suspect_ips))

    pickle.dump( (user_to_ips, ips_to_user,
                  user_to_most_common_ip,
                  all_suspect_ips), open( CACHE_FILE, "wb" ) )

else:
    logging.info("Loading from cache")
    user_to_ips, ips_to_user, user_to_most_common_ip, all_suspect_ips \
        = pickle.load( open( CACHE_FILE, "rb" ) )

non_suspects_using_suspect_ips = defaultdict(int)
for user, user_ips in user_to_ips.items():
    if user in suspects:
        continue
    for ip in user_ips:
        if ip in all_suspect_ips:
            non_suspects_using_suspect_ips[user] += 1

logging.info("Non-suspects using suspect IPs: {}"
                .format(non_suspects_using_suspect_ips))

top_n = 3
logging.info("Fetching top {} new suspects:".format(top_n))
d = Counter(non_suspects_using_suspect_ips)
ips_for_new_suspects = []
for user, num_common_ips in d.most_common(top_n):
    mc_ip = user_to_most_common_ip[user]
    print ("New suspect: {}, # suspicious IPs: {}, Most common IP: {}"
            .format(user, num_common_ips, mc_ip))
    ips_for_new_suspects.append(mc_ip)
```

```python
print (",".join(sorted(ips_for_new_suspects,
        key=lambda ip: socket.inet_aton(ip))))
```

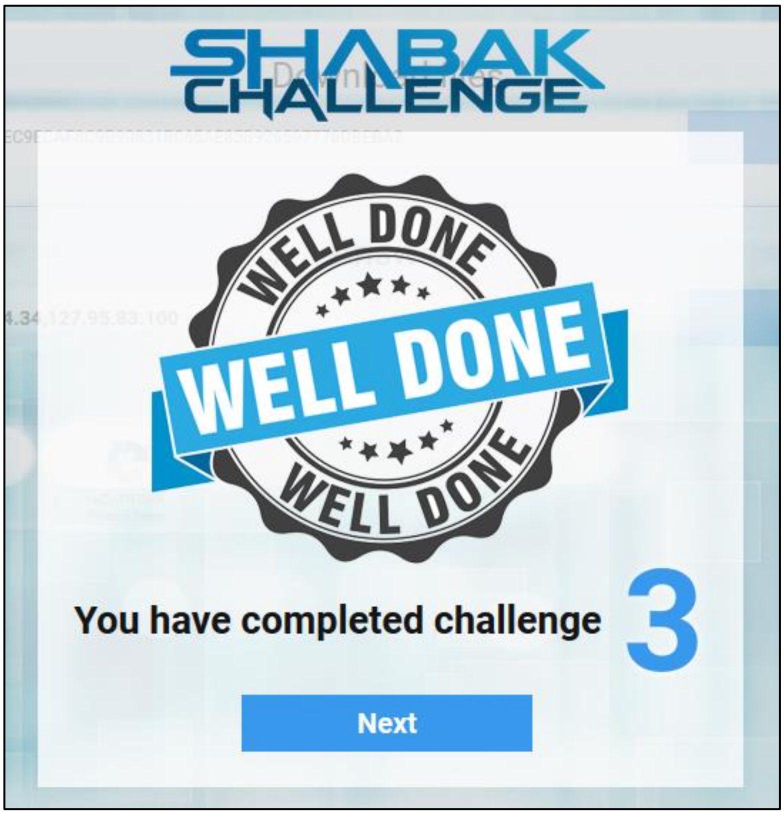
הסקריפט מייצר רשימה של משתמשים רגילים שהשתמשו בכתובות IP של חשודים, ומתוכם שולף את שלושת המשתמשים שהשתמשו בכמות הגדולה ביותר של כתובות חשודות.

הוא מייצג את התוצאה בצורה ממויינת לפי כתובת ה-IP הנפוצה ביותר של כל משתמש, כפי שהאתגר דרש.

התוצאה:

```
Log:    Suspects: ['2449', '6796', '9237', '4024', '3538', '3608', '7239', '435', '5206', '2211']
Log:    Cache file missing, loading from scratch
Log:    Parsing log file...
Log:    Parsed 10000000 rows in 32.40831255912781 seconds
Log:    Number of unique users: 10001
Log:    Number of unique IPs: 10010
Log:    Calculating most common IP per user...
Log:    Searching for all suspect IPs...
Log:    All suspect IPs: {'104.45.191.227', '103.205.114.34', '139.210.78.22', '127.95.83.100', '130.76.88.3', '51.70.255.188', '124.9.188.155', '114.79.247.223', '41.239.144.6', '109.242.247.39', '78.12.11.167', '211.104.116.62', '58.149.209.97', '138.27.249.121', '67.141.120.237', '10.192.20.173', '162.219.33.114', '42.74.74.110', '143.204.212.207', '68.17.81.83', '230.167.210.226'}
Log:    Non-suspects using suspect IPs: defaultdict(<class 'int'>, {'7159': 8, '5630': 1, '9091': 3, '1808': 10, '5772': 9, '5590': 4, '7051': 6, '6529': 5, '4918': 10, '87': 2, '9482': 7})
Log:    Fetching top 3 new suspects:
New suspect: 1808, # suspicious IPs: 10, Most common IP: 41.239.144.6
New suspect: 4918, # suspicious IPs: 10, Most common IP: 103.205.114.34
New suspect: 5772, # suspicious IPs: 9, Most common IP: 127.95.83.100
41.239.144.6,103.205.114.34,127.95.83.100
```

ואכן, אלו החשודים שהשב"כ חיפש:



והתוצר הסופי:

מילות סיכום

באופן כללי, מדובר ביוזמה מבורכת שתופסת תאוצה בשנים האחרונות, ואנחנו מקווים להמשיך ולראות עוד אתגרים בשנים הקרובות, כחלק מקמפיין גיוס או מכל סיבה אחרת.

בסך הכל, נהנינו לפתור את שני המסלולים של סדרת האתגרים. שני המסלולים האחרים: Hardware ו-Signal Processing, הצריכו מומחיות אחרת לחלוטין ולמעשה הפנייה הזו למספר קהלים היא נקודה נוספת לחיוב.

מהצד השני, היו מספר עניינים אשר פגמו מעט בחוויה - למשל, העובדה שהיה צריך להוסיף רווחים ב-Find the Code, או ההתעקשות על רמת דיוק של 5 ספרות אחרי הנקודה ב-Cat and Mouse (כנראה שבשלב מסוים נוספה הבהרה בשדה התשובה בתרגיל). ועדיין, חלק מהתרגילים הצריכו חשיבה מחוץ לקופסא, וחלק דימו בצורה יחסית מדויקת (כנראה) עבודת מודיעין אמיתית.

התחלנו עם מסלול ה-Embedded כי בכך אנחנו עוסקים ביומיום, והצלחנו לפתור את שתי השאלות ביום רביעי בלילה. ביום חמישי נתקלנו בכתבה הזו במאקו:

> שירות הביטחון הכללי (שב"כ) השיק אתמול (רביעי) קמפיין גיוס חדש, שנועד למצוא את אנשי הטכנולוגיה המוכשרים ביותר להצטרף לשורותיו. 150,000 גולשים כבר נכנסו לאתר של האתגר: גולשים רבים ממדינות שונות ניסו להתמודד עם האתגר - ונכון לשעה 10:00 הבוקר (חמישי) רק שניים הצליחו לפתור את כל שלבי האתגר.
>
> לסיפורים הכי מעניינים והכי חמים – הצטרפו לפייסבוק שלנו
>
> גולשים מארה"ב, אנגליה, קנדה, צרפת ורוסיה ניסו לפתור את האתגר - וכך גם משתמשים מעזה, אפגניסטן, טורקיה, איחוד האמירויות, מלזיה, עומאן, קירגיזסטן, מצרים, ערב הסעודית, מרוקו, אינדונזיה, פקיסטן ועירק.
>
> השניים שפתרו את כל השלבים של האתגר הצליחו באחד המסלולים, שנקרא תוכנה משובצת. באתר יש עוד שלושה מסלולים של האתגר שאף אחד עדיין לא הצליח לפתור. כמה אלפים בודדים הצליחו לפתור רק את השלבים הראשונים של האתגר בקטגוריות השונות, ומאות שלחו קורות חיים והגישו מועמדות למשרות הטכנולוגיות השונות.
>
> באתר www.israelneedsu.com ("ישראל צריכה אותך") נמצא האתגר, שנכתב על-ידי מומחים ומהנדסים של השב"כ. מדובר בחידה מורכבת הבנויה ממספר שלבים ומשלבת כמה אלמנטים טכנולוגיים מתקדמים, הנוגעים לתחומי הפעילות של גוף הביטחון.

היה נחמד מאוד לראות שהיינו הראשונים לפתור מסלול כלשהו, ולזכות ברבע שעה של תהילה אנונימית.

Windows Notification Facility

מאת טל בלום

הקדמה

החל מ-Windows 8, התווסף לקרנל מנגנון חדש בשם Windows Notification Facility, או בקיצור - WNF. WNF הוא מנגנון פנימי של הקרנל בשיטת Subscriber\Publisher להעברת מידע והתראות על אירועים במערכת ההפעלה בין רכיבים שונים למטרות סנכרון. המידע נגיש גם מ-Kernel Mode וגם מ-User Mode, וגם בין תהליכים שונים. המנגנון הלך וגדל בין מערכות ההפעלה והיום יש לו שימושים רבים בפעילות הפנימית של מערכת ההפעלה.

ב-Black Hat האחרון (2018) Alex Ionescu, מכותבי הספר Windows Internals, ביחד עם Gabrielle Viala, הציגו את ה-WNF וסוגיות אבטחה שהוא מעלה [1]. עד ההרצאה כמעט ולא היה בכלל מידע באינטרנט על המנגנון הזה, שהוא לחלוטין Undocumented.

במאמר זה אסביר מה הוא אותו WNF, איך המנגנון הזה עובד, את המבנה שלו ומה השימושים שלו במערכת ההפעלה. בחלק השני של המאמר אני אציג רעיונות כיצד ניתן לנצל אותו לרעה כדי לעקוף כלי EDR, ואציג ווקטורי תקיפה ואפשרויות מעניינות אחרות שהוא מספק ב-Windows.

מה זה Windows Notification Facility

WNF פועל בשיטה של Publisher ו-Subscriber. כלומר, רכיב (יכול להיות או דרייבר או תהליך ב-User Space) יכול לעשות Subscribe לאירוע מסוים (נקרא StateName), על מנת לקבל התראה בכל פעם שרכיב אחר עשה "Publish" אל אותו State Name. כלומר, ביצע שינוי במידע שעליו (נקרא StateData).

לדוגמא, כשמערכת ההפעלה עושה Boot, ה-HD קודם כל עולה כ-Read-Only כדי ש-AutoChk יבדוק את תקינות ה-HD. ברגע שהוא מסיים את הבדיקה, הדיסק עובר Mount שוב עם הרשאות של RW ונשלח WNF Event שמיידע את שאר הרכיבים שעשו Subscribe לאותו State Name שהדיסק תקין וניתן לרשום כעת אל ה-HD.

נכון לגרסא החדשה ביותר של Windows10 (1809) קיימים כ-4,000 אירועי WNF במערכת ההפעלה. WNF משמש כדי לתקשר בין תהליכים, בין Sessions, ובין User Mode ל-Kernel Mode. המגבלה היחידה שלו היא שכל State Name יכול להכיל State Data רק עד ל-4KB.

הרצה של תהליך גורמת להתראת WNF, סגירת תהליך גורם להתראת WNF, חיבור של רכיב שמע גורם להתראת WNF, חיבור לרשת Wi-Fi גורם להתראת WNF, כשהמערכת זיהתה שהמשתמש לא פעיל (המחשב לא קיבל קלט בפרק זמן מסוים) הדבר גורם להתראת WNF, ועוד רבים.

ה-WNF הוא לא שכבה מעל ETW או ALPC, או כל מנגנון אחר של הקרנל, ולא קשור אליהם. אלא מנגנון נפרד, בעל שימושים פנימיים בלבד של מערכת ההפעלה, ונגיש רק דרך כמה פונקציות לא מתועדות.

העובדה שהוא עדיין יחסית לא מוכר גורמת למנגנון להיות מעניין ביותר. כלי EDR כמו Sysmon לא מכירים אותו ואין להם שום דרך לזהות שימוש בו. תוקפים יכולים לנצל את המנגנון להסתרת הפעילות הזדונית שלהם וככה לעקוף מערכות הגנה. על כך נדבר בהרחבה בהמשך.

WNF הוא מנגנון אשר קיים גם בפלטפורמות אחרות של Microsoft, כמו Windows Phone או XBox. כך שיש גם התראות WNF על יצירת שיחת טלפון ועוד התראות רבות אחרות שרלוונטיות רק לגרסאת המובייל או ל-XBox, אך עדיין קיימות בכל הפלטפורמות.

מעבר לאפשרות שלנו לקריאה וכתיבה של ה-State Data ולהירשם ל-State Names שכבר קיימים במערכת ההפעלה, יש API גם ליצירת State Names חדשים.

WNF State Names מעניינים / רגישים

מתוך המוני ה-State Names הקיימים במערכת ההפעלה (שרובם לא מעניינים אותנו כאנשי אבטחת מידע), יש כמה שכן יכולים לספק לנו מידע מעניין ואפילו לשנות את פעילות המכונה:

- WNF_WIFI_CONNECTION_STATUS - מודיע בכל פעם שהמחשב מתחבר ל-WI-FI
- WNF_AUD_CAPTURE\RENDER - מודיע לנו כל פעם שתהליך משמיע\מקליט סאונד. כולל שם התהליך שהפעיל את זה.
- WNF_SEB_USER_PRESENCE_CHANGED \ WNF_SEB_USER_PRESENT - מודיע כאשר משתמש לא היה פעיל X זמן, לפי ההגדרה של Windows.
- WNF_SHEL_(DESKTOP) APPLICATION_(STARTED \ TERMINATED) - נותן לך את השם של כל תכנית שהתחילה\נסגרה.
- WNF_EDGE_LAST_NAVIGATED_HOST - ה-URL האחרון שהמשתמש כתב לתוך הדפדפן Edge.
- WNF_SYS_SHUTDOWN_IN_PROGRESS - נותן התראה כשהמחשב בתהליך כיבוי.
- WNF_FSRL_OPLOCK_BREAK - מקבל רשימה של PID-ים וסוגר אותם!

כדי להראות עוד כמה מהשימושים של WNF במערכת ההפעלה, שווה להסתכל על פרוייקט מגניב בגיטהאב בשם Mach2:

https://github.com/riverar/mach2/

ב-Windows יש מה שנקרא "Insider Preview Features". כש-Windows רוצים לבדוק פיצ'ר חדש מסוים, הם מחילים אותו רק על חלק מהמשתמשים כדי לראות את התגובה שלהם ולדעת מה אהוד על ידי המשתמשים ומה לא. לדוגמא הפיצ'ר Dark Mode, שמאפשר לשנות את העיצוב של האפליקציות של Windows מלבנות ומוארות לשחורות ואפלות.

יש אלפי פיצ'רים כאלה, שהסוויץ' האם הם יהיו מופעלים או לא נמצא כמו שניחשתם בתוך WNF. הפרויקט הזה משנה את ערכי ה-State Names הרלוונטים, ואתה יכול בעצם להחליט בעצמך איזה פיצ'רים יהיו לך או לא יהיו לך.

ככה בעצם אפשר להרוס ל-Microsoft את כל הבדיקות שלהם, כי כאמור ה-WNF הוא מנגנון לא מתועד שהמשתמשים לא אמורים לשחק בו.

מבנה ה-State Name

ישנם 4 סוגים של State Names:

- Well-Known - State Names שמגיעים מוגדרים עם מערכת ההפעלה, לא ניתן להגדיר חדשים מהסוג הזה. המידע שבהם הוא Persistent, כלומר נשמר גם לאחר שהמחשב נכבה. נמצאים ברג'יסטרי תחת הנתיב:

```
HKLM\SYSTEM\CurrentControlSet\Control\Notifications
```

- Permanent - המידע בהם יכול להיות Persistent ויכול להיות שלא. תלוי ב-IsPermanent flag מצאים גם כן ברג'יסטרי תחת הנתיב:

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Notifications
```

- Persistent - הם Persistent בהקשר שהם נשמרים גם אחרי שהתהליך שרשם אותם נסגר, אך לא נשמרים לאחר שהמחשב נכבה. נמצאים ברג'יסטרי תחת הנתיב:

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\VolatileNotifications
```

- Temporary - פחות נפוצים. כשהתהליך נסגר, ה-State Name הזמני נעלם גם הוא. אין לו רשומה ברג'יסטרי.

כדי ליצור Persistent או Permanent Names צריך הרשאות Administrator.

ה-State Names עצמם מוגדרים במערכת ההפעלה עם ID ייחודי. עבודת Reverse Engineering, שהתחיל אותה redplaint בבלוג שלו [2], הוציאה מתוך perf_nt_c.dll את רשימת ה-ID-ים. שלמרות שהם Undocumented, בתוך ה-dll יש מחרוזת שמציינת את השם שלהם והסבר קצר. רשימה יותר מלאה שמקשרת בין שם ה-State Name ל-ID שלו, נמצאת כאן: [3].

ה-ID של כל State Name הוא בעצם ערך שמורכב מהשדות: Version, Lifetime, Scope, Data permanence flag ו-Unique id. שעליהם עשו XOR עם ערך קבוע: 0x41C64E6DA3BC007:

```
typedef struct _WNF_STATE_NAME_INTERNAL
{
    ULONG64 Version : 4;
    ULONG64 Lifetime : 2;
    ULONG64 DataScope : 4;
    ULONG64 IsPermanent : 1;
    ULONG64 Unique : 53;
} WNF_STATE_NAME_INTERNAL, *PWNF_STATE_NAME_INTERNAL;
```

[https://blog.quarkslab.com/playing-with-the-windows-notification-facility-wnf.html]

Lifetime - מתייחס לארבעת הסוגים של State Names (WellKnown, Permanent, Persistent, Temporary):

```
typedef enum _WNF_STATE_NAME_LIFETIME
{
    WnfWellKnownStateName = 0x0;
    WnfPermanentStateName = 0x1;
    WnfPersistentStateName = 0x2;
    WnfTemporaryStateName = 0x3;
} WNF_STATE_NAME_LIFETIME;
```

Scope - State names יכולים להיות גלובאליים, או ספציפיים ל-Session, או ל-User או ל-Process.

```
typedef enum _WNF_DATA_SCOPE
{
  WnfDataScopeSystem = 0x0,
  WnfDataScopeSession = 0x1,
  WnfDataScopeUser = 0x2,
  WnfDataScopeProcess = 0x3,
  WnfDataScopeMachine = 0x4,
} WNF_DATA_SCOPE;
```

ה-IsPermanent Flag רלוונטי ל-Persistent State Names אשר יכולים להיות Persistent לאחר כיבוי המחשב ויכולים שלא.

כדי לפענח את המידע של השדות האלה, יש קודם כל לבצע XOR של ערך ה-State name עם הערך הקבוע (שלא ברור מה המשמעות שלו):

```
WNF_XOR_KEY = 0x41C64E6DA3BC0074;
```

כך לדוגמא, יש את ה-State Name שנקרא WNF_SYS_SHUTDOWN_IN_PROGRESS אשר מיידע כאשר המחשב נמצא בתהליך כיבוי. וה-ID המייצג אותו במערכת ההפעלה הוא: 0x4195173ea3bc0875.

נעשה את הפעולות הבאות כדי לפענח:

```
#include <iostream>
#define WNF_XOR_KEY 0x41C64E6DA3BC0074
using namespace std;

int main() {
long int StateName = 0x4195173ea3bc0875;
long int ClearStateName = StateName ^ WNF_XOR_KEY;
cout<<"Clear State Name: "<<hex<<ClearStateName<<endl;
long int Version = ClearStateName & 0xf;
cout<<"Version: "<<hex<<Version<<endl;
long int LifeTime = (ClearStateName >> 4) & 0x3;
cout<<"LifeTime: "<<hex<<LifeTime<<endl;
long int DataScope = (ClearStateName >> 6) & 0xf;
cout<<"Scope: "<<hex<<DataScope<<endl;
long int IsPermanent = (ClearStateName >> 0xa) & 0x1;
cout<<"IsPermanent: "<<hex<<IsPermanent<<endl;
long int Unique = ClearStateName >> 0xb;
cout<<"Unique ID: "<<hex<<Unique<<endl;
}
```

כך שמאפייניו של WNF_SYS_SHUTDOWN_IN_PROGRESS הם:

```
Clear State Name: 53595300000801
Version: 1
LifeTime: 0   Well Known
Scope: 0    System
IsPermanent: 0   Nope
Unique ID: a6b2a600001
```

WNFUN

בהרצאה של Alex ו-Gabrielle, הוצגו כלים שנכתבו על ידם לשימוש ב-WNF ולאחר מכן גם הועלו ל-Github:

https://github.com/ionescu007/wnfun/

הכלים כוללים את WNFDump - היום מדובר בכלי עם הכי הרבה יכולות, והכי נח לשימוש ב-WNF. הוא מאפשר לך לחפש State Names מעניינים (גם דרך הרג'יסטרי וגם דרך ברוט-פורס של ה-ID במידה ומדובר ב-Temporary State Name).

הוא מאפשר לך לקרוא את המידע השמור בהם, ונותן לך לכתוב בעצמך אליהם ולראות איך זה משנה את פעילות מערכת ההפעלה. בנוסף הוא מאפשר לך להירשם ל-State Name מסוים ומציג כל פעם שקרה בו שינוי. כדאי לציין שאם מוחקים את כל המידע שבכל ה-State Names, המחשב נהרס - ה-Explorer קורס בלי דרך להחזיר אותו, ואתם בעצם לא יכול יותר לעשות כלום. וגם Reboot לא יתקן אותו.

דוגמא לפלט של הכלי להוצאת כל ה-State Names דרך הרג'יסטרי (כמובן שהרשימה ממשיכה עוד הרבה):

```
Administrator: Command Prompt
C:\Users\Test\Desktop\wnfun-master\wnftools_x86>wnfdump.exe -d

WNF State Name [Well-Known Lifetime]                          | S | L | P | AC | N | CurSize | MaxSize | Changes
-------------------------------------------------------------------------------------------------------------------
WNF_WEBA_CTAP_DEVICE_STATE                                    | S | W | N | RW | I |       0 |      12 |       0
WNF_WEBA_CTAP_DEVICE_CHANGE_NOTIFY                            | S | W | N | RW | I |       0 |       4 |       0
WNF_PNPA_DEVNODES_CHANGED                                     | S | W | N | RO | U |       0 |       0 |      27
WNF_PNPA_DEVNODES_CHANGED_SESSION                             | s | W | N | RO | U |       0 |       0 |       0
WNF_PNPA_VOLUMES_CHANGED                                      | S | W | N | RO | U |       0 |       0 |       0
WNF_PNPA_VOLUMES_CHANGED_SESSION                              | s | W | N | RO | U |       0 |       0 |       0
WNF_PNPA_HARDWAREPROFILES_CHANGED                             | S | W | N | RO | U |       0 |      16 |       0
WNF_PNPA_HARDWAREPROFILES_CHANGED_SESSION                     | s | W | N | RO | U |       0 |      16 |       0
WNF_PNPA_PORTS_CHANGED                                        | S | W | N | RO | U |       0 |      64 |       0
WNF_PNPA_PORTS_CHANGED_SESSION                                | s | W | N | RO | U |       0 |      64 |       0
WNF_AUDC_CPUSET_ID                                            | P | W | N | RO | U |       0 |      16 |       0
WNF_AUDC_PHONECALL_ACTIVE                                     | S | W | N | RO | U |       0 |       4 |       0
WNF_AUDC_TUNER_DEVICE_AVAILABILITY                            | S | W | N | RO | U |       0 |       4 |       0
WNF_AUDC_HEALTH_PROBLEM                                       | S | W | N | RO | U |       0 |      16 |       0
WNF_AUDC_CPUSET_ID_SYSTEM                                     | S | W | N | NA | U |   65535 |       4 |       0
WNF_AUDC_RENDER                                               | S | W | N | RO | U |    4096 |    4096 |      13
WNF_AUDC_VOLUME_CONTEXT                                       | S | W | N | RO | U |       0 |    4096 |       0
WNF_AUDC_CAPTURE                                              | S | W | N | RO | U |    4096 |    4096 |       1
WNF_AUDC_RINGERVIBRATE_STATE_CHANGED                          | S | W | N | RO | U |       0 |       4 |       0
WNF_AUDC_SPATIAL_STATUS                                       | S | W | N | RO | U |    4096 |    4096 |       3
WNF_AUDC_DEFAULT_RENDER_ENDPOINT_PROPERTIES                   | S | W | N | RO | U |      68 |     256 |       1
WNF_AUDC_CHAT_APP_CONTEXT                                     | S | W | N | RO | U |       0 |    4096 |       0
WNF_EXEC_OSTASKCOMPLETION_REVOKED                             | S | W | N | RW | I |       0 |       8 |       0
WNF_EXEC_THERMAL_LIMITER_CLOSE_APPLICATION_VIEWS              | S | W | N | RO | U |       0 |       0 |       0
WNF_EXEC_THERMAL_LIMITER_TERMINATE_BACKGROUND_TASKS           | S | W | N | RO | U |       4 |       4 |       1
WNF_EXEC_THERMAL_LIMITER_DISPLAY_WARNING                      | S | W | N | RO | U |       0 |       4 |       0
```

והנה דוגמא להרשמה ל-WNF_SHEL_APPLICATION_STARTED, ה-State Name הזה מתעדכן כל פעם שנפתחת תוכנה במחשב (עם GUI). פתחתי כמה תוכנות לבדיקה וראיתי איך זה מתעדכן:

```
Administrator: Command Prompt - wnfdump.exe -n WNF_SHEL_APPLICATION_STARTED
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\Test\Desktop\wnfun-master\wnftools_x86

C:\Users\Test\Desktop\wnfun-master\wnftools_x86>wnfdump.exe -n WNF_SHEL_APPLICATION_STARTED
Change #29      98 bytes were written
a:{d65231b0-b2f1-4857-a4ce-a8e7c6ea7d27}\cmd.exe
Change #30      108 bytes were written
p:microsoft.microsoftedge_8wekyb3d8bbwe!microsoftedge
Change #31      96 bytes were written
p:microsoft.windowscalculator_8wekyb3d8bbwe!app
Change #32      50 bytes were written
a:c:\python27\python.exe
```

בנוסף הם סיפקו כלים כדי ליישם Client ו-Server - מאפשר לך ליצור State Name משלך ולהירשם אליו. הכלים פורסם ללא ה-Source code.

יש בפרוייקט ב-Github גם סקריפטים של Python, שמיישמים את אותו דבר. אך עם תיעוד טוב של המבנים בזיכרון, הפונקציות הלא מתועדות שבהם נעשה שימוש וכוללים רשימה מועילה של ה-ID-ים של ה-Well Known Name States.

WNF API (איך WNFUN עובד)

אף אחת מהפונקציות שמתעסקות עם WNF לא מתועדת, אך יש פרוייקטים באינטרנט שחקרו את הפונקציות הלא מתועדות ב-Windows ופרסמו אותן [5]. כדי ליצור State Name חדש משלנו ניתן להשתמש ב:

```
NTSTATUS ZwCreateWnfStateName (
_Out_    PWNF_STATE_NAME      StateName,
_In_     WNF_STATE_NAME_LIFETIME     NameLifeTime,
_In_     WNF_DATA_SCOPE  DataScope,
_In_     BOOLEAN PersistData,
_In_opt_     PCWNF_TYPE_ID    TypeId,      // This is an optional way to
get type-safety
_In_     ULONG    MaximumStateSize,        // Cannot be above 4KB
_In_     PSECURITY_DESCRIPTOR     SecurityDescriptor  // *MUST* be present
);
```

כדי למחוק את ה-State Name ניתן להשתמש ב-ZwDeleteWnfStateName. כדי להוסיף Data ל-State Name ולשלוח התראה, אפשר לערוך State Name בעזרת ZwUpdateWnfStateData:

```
NTSTATUS ZwUpdateWnfStateData (
_In_     PCWNF_STATE_NAME         StateName,
_In_reads_bytes_opt_(Length)          const VOID* Buffer,
_In_opt_     ULONG    Length,      \\ Must be less than MaximumSize when
registered
_In_opt_     PCWNF_TYPE_ID    TypeId,      \\ Optionally, for type-safety
_In_opt_     const    PVOID       ExplicitScope,   \\ Process handle, User
SID, Session ID
_In_     WNF_CHANGE_STAMP     MatchingChangeStamp, \\ Expected current
change stamp
_In_     LOGICAL          CheckStamp   \\ Enforce the above or silently
ignore it
);
```

כדי למחוק Data ניתן להשתמש ב-ZwDeleteWnfStateData. כדי לקרוא Data State יש את ZwQueryWnfStateData:

```
NTSTATUS     ZwQueryWnfStateData(
_In_     PCWNF_STATE_NAME         StateName,
_In_opt_     PCWNF_TYPE_ID    TypeId,
_In_opt_     const VOID*      ExplicitScope,
_Out_        PWNF_CHANGE_STAMP    ChangeStamp,
_Out_writes_bytes_to_opt_(*BufferSize, *BufferSize) PVOID    Buffer,
_Inout_      PULONG       BufferSize   \\ Can be 0 to receive the current
size
);
```

בהרשמה ל-State Name קיים אתה יכול לקבל התראה כל פעם שמידע מפורסם לתוך ה-State Name, או כשה-State Name נהרס ואפילו כשמישהו אחר עושה לו Subscribe.

כדי להירשם ל-State Name יש את הפונקציה:

```
NTSTATUS RtlSubscribeWnfStateChangeNotification (
_Outptr_     PWNF_USER_SUBSCRIPTION*      Subscription,
_In_         WNF_STATE_NAME       StateName,
_In_         WNF_CHANGE_STAMP     ChangeStamp,
_In_         PWNF_USER_CALLBACK  Callback,
_In_opt_     PVOID           CallbackContext,
_In_opt_     PCWNF_TYPE_ID    TypeId,
_In_opt_     ULONG    SerializationGroup,
_In_opt_     ULONG    Unknown);
```

בנוסף לפונקציות שתיארתי בחלק הזה, שהן כולן User-Modeיות (למרות שהן Undocumented וגם מייקרוסופט לא מפרסמים את הסימבולים שלהם), ישנם גם כמובן פונקציות מקבילות אליהן ב- Kernel Mode:

```
ExSubscribeWnfStateChange
ExQueryWnfStateData
```

שיטות אפשריות לתקיפה

לאחר שסקרנו את המנגנון, והבנו קצת מה השימושים שלו ומה ניתן להשיג באמצעותו, נראה איך ניתן לנצל אותו לפעולות זדוניות. אך לפני שנסקור את כל האפשרויות, יש 2 נקודות שחשוב לציין בנקודה זו בנוגע לשימוש ב-WNF בכללי.

ראשית, חשוב לזכור ש-WNF קיים רק החל מ-WIndows 8, כך שגם אם פוגען יירצה להשתמש ב-WNF, הוא לא יעבוד על Windows 7.

נקודה שנייה היא ש-WNF, כמו כל מה שלא מתועד ב-Windows, יכול להשתנות בין כל עדכון של מערכת ההפעלה. כך שבעתיד אולי הפונקציה שבה השתמשנו מצפה לפרמטרים שונים, המיקום של המבנים בזיכרון השתנה או שבכלל השתנה מבנה ה-State Name, והקוד יפסיק לעבוד.

מעבר לכך שכל Administrator יכול לכתוב על גבי State Names בצורה כזו שתהרוס את מערכת ההפעלה, ישנם עוד בעיות אבטחה רבות במנגנון הזה.

קודם כל המימוש של המנגנון עצמו מלא בחורי אבטחה - במחקר של Alex Ionescu הוא סיפר על כך שהוא מצא בעיות אבטחה רבות כשהוא שיחק עם WNFDump, לדוגמא הרבה מידע שאמור להיות מוגבל בין תהליכים ובין משתמשים, ב-WNF הוא גלובאלי. את בעיות האבטחה האלו Microsoft יתקנו בגרסא הקרובה.

שיטות תקיפה בעזרת יצירת State Names **חדש**

שימוש ב-State Names יכולה להיות דרך טובה לשמירת מידע בדרך שהיא חשאית, ויכולה להחליף כל מיני שיטות שבהם פוגענים משתמשים. אחת התכונות החשובות של State Names היא כאמור האפשרות שלו להיות Persistent לאחר reboot. רישום של State Names חדשים מסוג Permanent או Persistent יכולה להיות דרך מעולה לשמור על Persistency במערכת.

אם ניצור State Names שהם Permanent או Persistent, ניתן יהיה לראות קריאה וכתיבה לרג׳יסטרי אך זה יהיה הקרנל שיבצע אותם. כן יהיו אירועי ETW על הרשמה ל-State Name וביטול הרשמה, וגם על כתיבה ל-State Name. אך גם האירועים האלה נוצרים רק כשמשתמשים בפונקציות של Rtl. אם נשתמש ישירות ב-Zw, נוכל לעקוף את זה ולא יווצרו בכלל אירועי ETW.

מנגד, אם ניצור Temporary Name חדש, נאבד את ה-Persistency, אך לגמרי נמנע מהרג׳יסטרי. כן צריך כאמור לקרוא ל-ZwCreateRegisterName, אך לא נשמר על זה תיעוד במערכת ההפעלה.

זאת בעצם דרך בלתי נראית להעברת מידע בין תהליכים. הדרך היחידה לגלות אותו היא אם יש לך Hook על הפונקציה הזאת.

היתרון הגדול של השימוש ב-WNF הוא בכך שכל הפעילות שלו (כמו הכתיבה לרג׳יסטרי) מתבצעת ע"י עבודה פנימית של מערכת ההפעלה. בנוסף הוא יכול לתת לך Persistency. והכי חשוב - כלי ה-EDR היום לא מכירים את נתיבי הרג׳יסטרי האלה, ובטח שלא יכולים להתחקות אחרי מי ביצע את הפעולה, והאם הפעולה היא לגיטימית או לא. כל אלה, ביחד עם עצם זה שהמנגנון נותן פלטפורמה מאוד נוחה להעברת מידע בין תהליכים ולשלוח מידע ל-Kernel והפוך, גורמים ל-WNF להיות פלטפורמה מאוד מושכת לכותבי פוגענים.

חלק מהשימושים בה יכולים להיות כדי להסתיר מידע שצריך שהוא יהיה Persistent. (אולי לשמור שם את ה-Stage2?) או כדי להעביר מידע בין תהליכים שונים בצורה מאוד חשאית בעזרת Temporary State Name.

שימוש ב-State Names **"ריקים"**

במערכת ההפעלה יש הרבה Well-Known State Names שלא בשימוש, לדוגמא כל אלה שקשורים ל-Xbox או ל-Windows Phone. (השמות שלהם מתחילים ב-WNF_XBOX ו-WNF_CELL בהתאמה)

כך שבמקום ליצור State Names חדשים כדי להסתיר בהם מידע, ניתן לעשות זאת גם דרך פרסום מידע ב-State Names הקיימים, שאנו יודעים שמערכת ההפעלה לא תיגע בהם. כאשר משתמשים ב-State Name- קיים כדאי למצוא אחד שנותן לכולם אפשרויות כתיבה וקריאה, ויכול להכיל עד 4KB.

מציאת חולשה של מערכת ההפעלה בפירסור ה-State Name

מכיוון שכל התראת WNF יכולה להכיל עד 4KB, ובגלל שזה מנגנון פנימי, התהליכים שמשתמשים במידע הזה בדרך כלל יפרסרו אותו בלי לבדוק את לגיטימיות הקלט, כך שייתכן שיש אפשרות שניתן לנצל חולשה ולגרום להזרקה של Shellcode משלנו.

שיטה אחת למצוא חולשה כזאת היא לעשות Fuzzing של הערכים בתוך ה-State Names ולראות מה גורם למערכת להתנהג באופן לא צפוי. (לדוגמא קריסה של Service או אפילו Blue Screen), שיטה אחרת היא דרך בחינה של ה-Subscriptions של תהליך, ולבדוק אותם ספציפית.

כדי למצוא את כל ה-Subscriptions של תהליך, יש להשיג את האובייקט nt!_Eprocess.WnfContext המכיל את האובייקט WNF_PROCESS_CONTEXT, שהוא אובייקט לא מתועד.

ניתן להשיג את האובייקט גם דרך nt!ExpWnfProcessesListHead, אשר מכיל רשימה מקושרת של ה-WNF_PROCESS_CONTEXT של כל התהליכים. האובייקט WNF_PROCESS_CONTEXT לא מתועד, אך אלכס הציג אותו בהרצאה שלו:

```
Typedef struct _WNF_PROCESS_CONTEXT
{
      WNF_CONTEXT_HEADER Header;
      PEPROCESS Process;
      LIST_ENTRY WnfProcessListEntry;
      Struct
      {
            PWNF_SCOPE_INSTANCE WnfScopeTypeSession;
            PWNF_SCOPE_INSTANCE WnfScopeTypeUser;
            PWNF_SCOPE_INSTANCE WnfScopeTypeProcess;
      } ImplicitScopeInstances;
      EX_PUSH_LOCK NameInstanceListLock;
      LIST_ENTRY TemporaryNameslistListHead;
      EX_PUSH_LOCK SubscriptionListLock;
      LIST_ENTRY ProcessSubscriptionListHead;
      EX_PUSH_LOCK PendingQueueLock;
      LIST_ENTRY DeliveryPendingListHead;
      PKEVENT NotificationEvent;
} WNF_PROCESS_CONTEXT, *PWNF_PROCESS_CONTEXT;
```

בתוכו יש כ-LIST_ENTRY את רשימת ה-Subscriptions של התהליך. אם רוצים לדעת מה דרייברים עושים עם המידע שהם מקבלים מה-State Names, ה-Subscriptions של התהליך System מכיל אותם. וכך אתה יכול להשיג את כל ה-Kernel Callbacks.

בדרך הזו אפשר לחפש דרייברים שלא משתמשים נכון במידע שהם מקבלים, ואולי למצוא דרך להרצת קוד בקרנל.

הזרקת קוד עם WNF

במקום השיטה הרגילה להזרקת קוד, שמשתמשת בפונקציות WriteProcessMemory ,VirtualAllocEx, CreateRemoteThread, שכל כלי ה-EDR יודעים היום לזהות כפעילות עוינת, יש צורך למצוא שיטות שונות לעשות את אותו תהליך מבלי שזה יהיה מזוהה.

בשנה שעברה פורסמה שיטה בשם AtomBombing, שהראתה איך ניתן להזריק קוד לתהליך אחר בדרך יצירתית יותר - https://www.digitalwhisper.co.il/files/Zines/0x4E/DW78-3-AtomBombing.pdf.

דבר דומה יכול גם להיות אפשרי דרך WNF. אם ידוע על תהליך מסוים שקורא מתוך State Name מסוים, ע"י שינוי הערך ב-State Name הזה נוכל להכניס מידע משלנו לתוך זיכרון תהליך היעד.

לאחר שהמידע שלנו נמצא בזיכרון של תהליך היעד, צריך לגרום לשינוי הרצת התוכנית כדי שתריץ את הקוד שהזרקנו.

כדי לעשות את זה יש למצוא את פונקציית ה-Callback איתה הוא נרשם לאותו State Name, בדרך שהסברנו מקודם (דרך WNF_PROCESS_CONTEXT).

ברגע שמצאנו את זה, ניתן לשנות את ה-Callback Function, לפונקציה אחרת. למה שאנחנו הזרקנו.

CasperStager

לאחר שההרצאה פורסמה, כצפוי התחילו ליישם את השיטות האלו. כרגע יש רק PoC ראשון שנכתב ב-C#, בשם CasperStager, הממשש חלק מהטכניקות שדיברנו עליהם:
https://github.com/ustayready/CasperStager

השיטה כאן היא שמירת ה-Stage2 של הפוגען בתוך State Name קיים, כמו שהצגתי בתחילת המאמר. (בגלל גודל התוכנית הוא שמר אותה ב-3 State Names שונים שקשורים ל-Xbox)

ה-Dropper בעצם מקבל מהאינטרנט את ה-Stage2 ורושם אותו אל תוך ה-State Name, במקום לשמור אותו על הדיסק. ואז בכל ריצה של התוכנית, קורא אותו משם ומריץ אותו.

זה לא נותן לך הרצה עצמאית של ה-Stage2, אך זאת דרך חמודה להסתיר את הפוגען "האמיתי", ואת הפעולות שלו.

הפעילות של ה-Stage2 בפרוייקט הזה היא למלא את כל ה-WNF State Names באפסים, כדי להשבית את המערכת.

סיכום

WNF הוא עדיין בגדר אדמה לא מיושבת, המידע והמקורות לנושא עדיין דלילים. למרות שעדיין אין מימוש של כל הטכניקות שהצגתי המנצלות את ה-WNF, כבר עכשיו ניתן לראות את ה"פוטנציאל" הטמון בו, ואני מאמין שבעתיד ניראה עוד ועוד פרוייקטים שמצליחים לנצל אותו לשימושים שונים.

פוגענים שישתמשו בשיטות האלה יהנו מכך שהמנגנון הזה גם כל כך גלובאלי ועדיין כל כך מתחת לרדאר של רוב פתרונות האבטחה היום.

למרות שנדמה לנו שאנו מכירים טוב את מערכת ההפעלה Windows, תמיד נוספים אליה פיצ׳רים חדשים שיכולים להוות חור אבטחתי, והמירוץ חתול ועכבר בין כותבי הפוגענים למערכות ההגנה תמיד נמשך.

לכל שאלה או הערה על WNF או סתם בכללי ניתן ליצור קשר בכתובת blum.tal2@gmail.com

מקורות

[1] - ההרצאה מ-BH2018:

https://www.youtube.com/watch?v=MybmgE95weo

[2] הבלוג של RedPlait:

http://redplait.blogspot.com/2017/08/wnf-ids-from-perfntcdll.html

[3] רשימה של Well Known State Names:

https://github.com/ionescu007/wnfun/blob/master/script_python/WellKnownWnfNames.py

[4] דוגמא נהדרת שמסבירה על WNF:

https://blog.quarkslab.com/playing-with-the-windows-notification-facility-wnf.html

[5] https://processhacker.sourceforge.io/doc/ntzwapi_8h.html

דברי סיכום

בזאת אנחנו סוגרים את הגליון ה-102 של Digital Whisper, אנו מאוד מקווים כי נהנתם מהגליון והכי חשוב- למדתם ממנו. כמו בגליונות הקודמים, גם הפעם הושקעו הרבה מחשבה, יצירתיות, עבודה קשה ושעות שינה אבודות כדי להביא לכם את הגליון.

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

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

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

www.DigitalWhisper.co.il

"Talkin' bout a revolution sounds like a whisper"

הגליון הבא ייצא בסוף חודש ינואר 2019.

אפיק קסטיאל,

ניר אדר,

31.12.2018