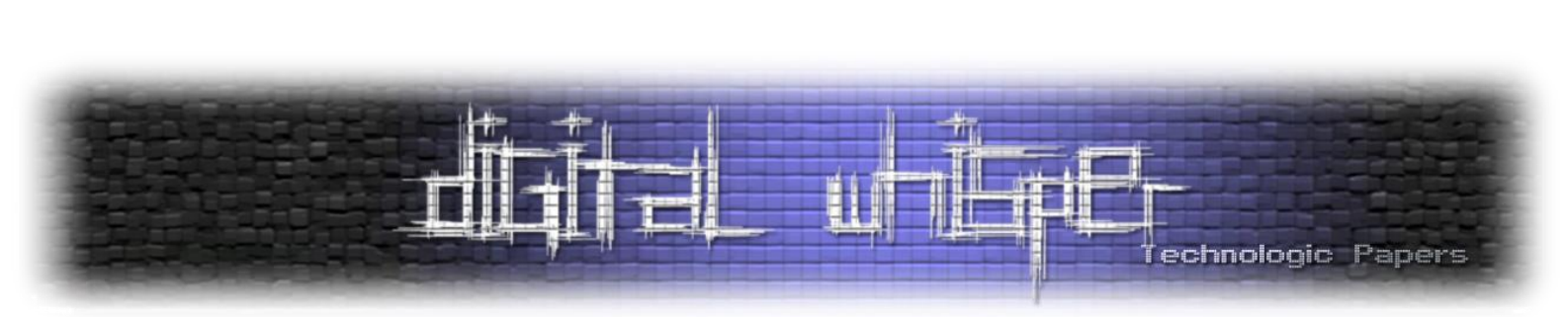


ניתוח ה-CryptoWall3 - פיסת קוד ששווה 300 מליון דולר

מאת d4d

הקדשה

לפני שנתחיל במאמר עצמו, ברצוני להקדיש את המאמר לחבר - בינימין יעקובוביץ' ז"ל אשר נהרג בפיגוע דריסה בצומת חלחול בחודש נובמבר של שנת 2015, בעת שירותו הצבאי במג"ב. יהי זכרו ברוך.

הקדמה

בחודשים של אוקטובר ונובמבר של שנת 2015 היה לא מעט רעש סביב ה-CryptoWall3, זאת בעקבות גל הדבקה נרחב שכלל לא מעט קמפיינים שמטרתם הייתה להפיץ וירוס זה. מטרתו של הוירוס הינה להצפין את הקבצים במחשב הנפגע, ולדרוש כופר על מנת לפענח אותם בחזרה.

אני מכיר שני אנשים באופן אישי אשר נדבקו ב-Malware הנ"ל ובחור נוסף אשר פרסם בפייסבוק על כך שנדבק. ופורסמו אודותיו לא מעט כתבות באתרי החדשות בעולם.

וירוס זה משתמש בלא מעט טכניקות הגנה שונות. מטרת המאמר הינה להבין כיצד לנתח את ההגנות של הוירוס, כיצד להסירן על מנת שיהיה קל לנתח את הקוד, וכן ניתוח כללי של הוירוס עצמו.

במאמר הזה אציג את השלבים הבאים:

- באילו שיטות השתמשו ב-CryptoWall3 על מנת להקשות על אנשים לנתח אותו
- איך להוריד את ההגנות שיש ב-CryptoWall3
- סקירה כללית על מבנה הוירוס וכיצד הוא פועל.

סקירה כללית ל-Packer של ה-CryptoWall3

CryptoWall3 משתמש במספר Packer-ים שונים על מנת להסוות את דרך הפעולה שלו ולנסות למנוע מהאנטי וירוסים לזהות אותו.

Packers הם למעשה מוצר שכל מטרתם הוא לארוז את הקובץ המקורי כך שיהיה קשה לאנטי-וירוסים לזהות את הקובץ המקורי ולהקשות על חוקרים לפתוח את הקובץ בצורה פשוטה בעזרת Debugger. בדרך כלל, התוצר הוא קובץ חדש שפותח את הקובץ המקורי בזיכרון או חלקים ממנו ישנם סוגים שונים של Packers.

במקרים שבהם ה-Packers מצפינים את הקבצים כאשר הם על הדיסק ומפענחים אותם רק כאשר אותם קבצים בזיכרון - יהיו אנשים אשר יכנו אותם "Crypters". כך למעשה מגן ה-Packer על הוירוס מפני רברסינג ואנטי וירוסים.

ב-CryptoWall3 יוצרי הוירוס השתמשו בכמה טריקים מעניינים כדי לנסות למנוע לבצע רברסינג לקובץ. על מנת למנוע מ-sandboxes לרוץ, יוצרי הוירוס הכניסו מספר קטעי קוד אשר מבצעים "דברים לא חשובים" בלולאות כמה פעמים על מנת להקשות על איתור הקוד המעניין, ובנוסף הכניסו פעולות אשר ביצעו כל מיני חישובים שונים בלולאות שונות מספר פעמים על מנת לעכב את זמן ריצת הוירוס (יש לא מעט sandboxes שמגבילים את זמן ניתוח הדגימה למספר שניות ולאחר מכן נסגרים).

לדוגמא, ניתן לראות בתמונה הבאה את אחת מלולאות אלו:

```
003E0CCF   6A 00            PUSH 0
003E0CD1   6A 02            PUSH 2
003E0CD3   FF95 20FFFFFF    CALL DWORD PTR SS:[EBP-0E0]          CreateToolHelp32Snapshot
003E0CD9   8BF0             MOV ESI,EAX
003E0CDB   83FE FF          CMP ESI,-1
003E0CDE   74 0A            JE SHORT 003E0CEA
003E0CE0   C785 3CF2FFFF 2  MOV DWORD PTR SS:[EBP-0DC4],128
003E0CEA   8D85 3CF2FFFF    LEA EAX,[EBP-0DC4]
003E0CF0   50               PUSH EAX
003E0CF1   56               PUSH ESI
003E0CF2   FF95 1CFFFFFF    CALL DWORD PTR SS:[EBP-0E4]          Process32First
003E0CF8   85C0             TEST EAX,EAX
003E0CFA   74 46            JE SHORT 003E0D42
003E0CFC   8B85 44F2FFFF    MOV EAX,DWORD PTR SS:[EBP-0DBC]
003E0D02   3B85 0CFFFFFF    CMP EAX,DWORD PTR SS:[EBP-0F4]
003E0D08   75 26            JNE SHORT 003E0D30
003E0D0A   B9 00010000      MOV ECX,100
003E0D0F   8D95 60F2FFFF    LEA EDX,[EBP-0DA0]
003E0D15   8D85 EDECFFFF    LEA EAX,[EBP-1313]
003E0D1B   8A1A             MOV BL,BYTE PTR DS:[EDX]
003E0D1D   8818             MOV BYTE PTR DS:[EAX],BL
003E0D1F   40               INC EAX
003E0D20   42               INC EDX
003E0D21   49               DEC ECX
003E0D22   75 F7            JNE SHORT 003E0D1B
003E0D24   8B85 54F2FFFF    MOV EAX,DWORD PTR SS:[EBP-0DAC]
003E0D2A   8985 0CFFFFFF    MOV DWORD PTR SS:[EBP-0F4],EAX
003E0D30   8D85 3CF2FFFF    LEA EAX,[EBP-0DC4]
003E0D36   50               PUSH EAX
003E0D37   56               PUSH ESI
003E0D38   FF95 18FFFFFF    CALL DWORD PTR SS:[EBP-0E8]          Process32Next
003E0D3E   85C0             TEST EAX,EAX
003E0D40   75 BA            JNE SHORT 003E0CFC
003E0D42   56               PUSH ESI
```

טכניקה נוספת הינה הכנסת קוד אשר מנצל באג ב-ollydbg ו-2 ollydbg על מנת להקריס את ה-Debugger, ברגע שרברסר ינסה להריץ את הוירוס עם Debugger את קטע הקוד הבא:

```
001224BA   895D DC      MOV DWORD PTR SS:[EBP-24],EBX
001224BD   1E           PUSH DS
001224BE   8D45 DC      LEA EAX,[EBP-24]
001224C1   0FA9         POP GS
001224C3   FF75 08      PUSH DWORD PTR SS:[EBP+8]
001224C6   65:FF10      CALL DWORD PTR GS:[EAX]
```

הוא יקבל exception (בגלל שיש קריאה למה שנמצא ב-GS ולא ל-DS עצמו), עם זאת, בעת ריצת הוירוס אין הבדל - מפני ששניהם מכילים את אותו התוכן.

- GS הינו extra segment שלא נמצא בשימוש במערכות 32 ביט אך ב-64 ביט הוא מחליף את FS ומצביע ל-TEB/TIB (TEB: Thread Environment Block, TIB: Thread Information Block).

על מנת לעקוף טכניקה זו, ניתן לשנות את הקוד מ-GS ל-DS או לקפוץ ידנית לקטע קוד שמצביע EAX, וכמובן - ישנן מספר גרסאות של ollydbg בהם הבאג הזה לא יקרה. אגב, אם לא מבצעים tracing אלא ריצה רגילה התוכנית לא תקרוס. בנוסף, הקובץ לא נבדק ב-x32dbg/x64dbg/windbg, כנראה ששם הבאג לא ישתחזר.

לאחר מכן ה-Packer מפענח את הקוד הזדוני המוצפן בתוכו, מעלה אותו לזיכרון ומריצו בעזרת טכניקה המכונה Process Hollowing, או במילים פשוטות: ליצור תהליך במצב Create Suspended, להחליף לו את הזיכרון בקוד שלנו ולהמשיך את ריצת התהליך.

השלב הראשון, מתבצע Process Hollowing על התהליך של הוירוס עצמו. בשלב השני תהליך זה מתבצע על explorer.exe, ובשלב השלישי התהליך מתבצע על svchost.exe.

בצורה כזאת, ה-Packer מוודא שאין העתק של הוירוס מפוענח כקובץ על הדיסק, ובכך בעצם מקשה על החוקר לחקור את הוירוס.

בחלק הבא יוסבר צעד צעד איך לעקוף כל חלק ולקחת את הקוד שאנו רוצים בכדי לנתח את הוירוס עצמו.

איך להוריד את ההגנה של הוירוס

רוב ה-Packers משתמשים ב-()VirtualAlloc ו-()VirtualProtect כדי להקצות זיכרון בקוד ולהעתיק אליו את הקובץ באיזשהו שלב לאחר פעולת הפענוח.

- VirtualAlloc הינה פונקציית API של Windows איתה מקצים זיכרון באותו תהליך.
- VirtualProtect הינה פונקציית API איתה משנים הרשאות של דף מסוים בזיכרון כדי לקבל הרשאות כתיבה למקום שאין.

הדרך הכי קלה היא לשים BreakPoint (בקיצור: BP) על ()VirtualAlloc ולהריץ את הקוד ולחכות שהתוכנית תיעצר עליו:

```
7C809AF1  8BFF            MOV EDI,EDI                          VOIDPTR kernel32.VirtualAlloc(Address,Size,AllocType,Protect)
7C809AF3  55              PUSH EBP
7C809AF4  8BEC            MOV EBP,ESP
7C809AF6  FF75 14         PUSH DWORD PTR SS:[EBP+14]
7C809AF9  FF75 10         PUSH DWORD PTR SS:[EBP+10]
7C809AFC  FF75 0C         PUSH DWORD PTR SS:[EBP+0C]
7C809AFF  FF75 08         PUSH DWORD PTR SS:[EBP+8]
7C809B02  6A FF           PUSH -1
7C809B04  E8 09000000     CALL VirtualAllocEx
7C809B09  5D              POP EBP
7C809B0A  C2 1000         RETN 10
```

לאחר מכן, נסתכל על הכתובת שמכילה EAX - הזיכרון הוקצה בכתובת 0x003e0000:

```
003E0000  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
003E0010  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
003E0020  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
003E0030  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
003E0040  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
003E0050  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
```

לאחר מכן שמים Memory BP (או HW BP) על הקטע קוד הזה ונראה איפה נכתב המידע הזה. נשים HW BP על "Access" (כאשר שמים BP על Access, אנו מורים ל-Debugger להחזיר את השליטה למשתמש כאשר התוכנית תנסה **לגשת** לתא הזיכרון הספציפי עליו מורה ה-BP).

בתמונה הבאה, אנו מגיעים ללולאה אשר אחראית לכתוב קטע קוד לזיכרון שהוקצה:

```
0012247F  880C03          MOV BYTE PTR DS:[EAX+EBX],CL
00122482  8D48 01         LEA ECX,[EAX+1]
00122485  8BB5 70FFFFFF   MOV ESI,DWORD PTR SS:[EBP-90]
0012248B  334E 04         XOR ECX,DWORD PTR DS:[ESI+4]
0012248E  51              PUSH ECX
0012248F  33C9            XOR ECX,ECX
00122491  8A0C03          MOV CL,BYTE PTR DS:[EAX+EBX]
00122494  5E              POP ESI
00122495  2BCE            SUB ECX,ESI
00122497  880C03          MOV BYTE PTR DS:[EAX+EBX],CL
0012249A  8D48 01         LEA ECX,[EAX+1]
0012249D  8BB5 70FFFFFF   MOV ESI,DWORD PTR SS:[EBP-90]
001224A3  330E            XOR ECX,DWORD PTR DS:[ESI]
001224A5  51              PUSH ECX
001224A6  33C9            XOR ECX,ECX
001224A8  8A0C03          MOV CL,BYTE PTR DS:[EAX+EBX]
001224AB  5E              POP ESI
001224AC  2BCE            SUB ECX,ESI
001224AE  880C03          MOV BYTE PTR DS:[EAX+EBX],CL
001224B1  40              INC EAX
001224B2  FF8D F4FDFFFF   DEC DWORD PTR SS:[EBP-20C]
001224B8  75 A7           JNE SHORT 00122461
001224BA  895D DC         MOV DWORD PTR SS:[EBP-24],EBX
001224BD  1E              PUSH DS
001224BE  8D45 DC         LEA EAX,[EBP-24]
001224C1  0FA9            POP GS                               Modification of segment register
001224C3  FF75 08         PUSH DWORD PTR SS:[EBP+8]
001224C6  65:FF10         CALL DWORD PTR GS:[EAX]
```

בתמונה הבאה, ניתן לראות את השימוש בטריק שיוצרי הוירוס השתמשו בו על מנת להקריס את ה-Debugger:

```
001224BA   895D DC     MOV DWORD PTR SS:[EBP-24],EBX
001224BD   1E          PUSH DS
001224BE   8D45 DC     LEA EAX,[EBP-24]
001224C1   0FA9        POP GS
001224C3   FF75 08     PUSH DWORD PTR SS:[EBP+8]
001224C6   65:FF10     CALL DWORD PTR GS:[EAX]
```

אפשר לשנות את השורה כפי שמוצג בתמונה הבאה:

```
001224BD   1E          PUSH DS
001224BE   8D45 DC     LEA EAX,[EBP-24]
001224C1   0FA9        POP GS
001224C3   FF75 08     PUSH DWORD PTR SS:[EBP+8]
001224C6   FF10        CALL DWORD PTR DS:[EAX]
001224C8   90          NOP
```

על מנת להגיע ללולאת הפענוח, אנו יכולים להמשיך קדימה בקוד או פשוט שנחכה שה-HW BP שהצבנו קודם לכן יגרום לתוכנית לעצור, זאת ניתן לראות בתמונה הבאה:

```
0012246A   8D48 01     LEA ECX,[EAX+1]
0012246D   8BB5 70FFFFFF MOV ESI,DWORD PTR SS:[EBP-90]
00122473   334E 08     XOR ECX,DWORD PTR DS:[ESI+8]
00122476   51          PUSH ECX
00122477   33C9        XOR ECX,ECX
00122479   8A0C03      MOV CL,BYTE PTR DS:[EAX+EBX]
0012247C   5E          POP ESI
0012247D   2BCE        SUB ECX,ESI
0012247F   880C03      MOV BYTE PTR DS:[EAX+EBX],CL
00122482   8D48 01     LEA ECX,[EAX+1]
00122485   8BB5 70FFFFFF MOV ESI,DWORD PTR SS:[EBP-90]
0012248B   334E 04     XOR ECX,DWORD PTR DS:[ESI+4]
0012248E   51          PUSH ECX
0012248F   33C9        XOR ECX,ECX
00122491   8A0C03      MOV CL,BYTE PTR DS:[EAX+EBX]
00122494   5E          POP ESI
00122495   2BCE        SUB ECX,ESI
00122497   880C03      MOV BYTE PTR DS:[EAX+EBX],CL
0012249A   8D48 01     LEA ECX,[EAX+1]
0012249D   8BB5 70FFFFFF MOV ESI,DWORD PTR SS:[EBP-90]
001224A3   330E        XOR ECX,DWORD PTR DS:[ESI]
001224A5   51          PUSH ECX
Address=00000001
ECX=037D009B

Address   Hex dump                                          ASCII (ANSI - He
003E0000  9B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ¢
003E0010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

השלב הבא הוא לחכות לעוד עצירה ב-VirtualAlloc() ומגיעים לקטע קוד הבא שמזכיר את UPX (UPX הינו Packer מאוד ידוע, מבוסס קוד פתוח והוא מתחיל ב-pushad ונגמר ב-popad ומטרתו רק להקטין את גודל הקובץ ולא למנוע מרברסר לנתח אותו), ישנם כותבי וירוסים שאוהבים לקחת את הקוד של UPX מכיוון שזה קוד פתוח והם יכולים להוסיף לו עוד הגנות:

```
003E0F40   8945 E8     MOV DWORD PTR SS:[EBP-18],EAX
003E0F43   60          PUSHAD
003E0F44   8B75 EC     MOV ESI,DWORD PTR SS:[EBP-14]
003E0F47   8B7D E8     MOV EDI,DWORD PTR SS:[EBP-18]
003E0F4A   FC          CLD
003E0F4B   B2 80       MOV DL,80
003E0F4D   31DB        XOR EBX,EBX
003E0F4F   A4          MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
003E0F50   B3 02       MOV BL,2
003E0F52   E8 6D000000 CALL 003E0FC4
003E0F57 ^ 73 F6       JAE SHORT 003E0F4F
003E0F59   31C9        XOR ECX,ECX
003E0F5B   E8 64000000 CALL 003E0FC4
```

נשים BP על popad שם נגמר העתקה לזיכרון של קטע הקוד כאשר הוא עדיין מוצפן:

```
003E0FE0   61              POPAD
003E0FE1   C685 8BFBFFFF   MOV BYTE PTR SS:[EBP-475],47
003E0FE8   C685 8CFBFFFF   MOV BYTE PTR SS:[EBP-474],65
003E0FEF   C685 8DFBFFFF   MOV BYTE PTR SS:[EBP-473],74
003E0FF6   C685 8EFBFFFF   MOV BYTE PTR SS:[EBP-472],43
003E0FFD   C685 8FFBFFFF   MOV BYTE PTR SS:[EBP-471],6F
003E1004   C685 90FBFFFF   MOV BYTE PTR SS:[EBP-470],6D
Stack [0011F218]

Address   Hex dump                                          ASCII (ANSI - He
03E00000  A6 B1 7B EB E8 EB EB EB EF EB EB EB 14 14 EB EB
03E00010  53 EB EB EB EB EB EB EB AB EB EB EB EB EB EB EB
03E00020  EB EB EB EB EB EB EB EB EB EB EB EB EB EB EB EB
03E00030  EB EB EB EB EB EB EB EB EB EB EB EB 53 EB EB EB
03E00040  E5 F4 51 E5 EB 5F E2 26 CA 53 EA A7 26 CA BF 83
03E00050  82 98 CB 9B 99 84 8C 99 8A 86 CB 88 8A 85 85 84
03E00060  9F CB 89 8E CB 99 9E 85 CB 82 85 CB AF A4 B8 CB
03E00070  86 84 8F 8E C5 E6 E6 E1 CF EB EB EB EB EB EB EB
03E00080  58 28 BC E0 1C 49 D2 B3 1C 49 D2 B3 1C 49 D2 B3
03E00090  11 1B 0D B3 18 49 D2 B3 1F 31 33 B3 02 49 D2 B3
03E000A0  1F 31 0C B3 1D 49 D2 B3 B9 82 88 83 1C 49 D2 B3
03E000B0  EB EB EB EB EB EB EB EB BB AE EB EB A7 EA EF EB
03E000C0  D5 C3 5B BF EB EB EB EB EB EB EB EB 0B EB E9 EA
03E000D0  E0 EA E7 EB EB 8F EA EB EB 45 EB EB EB EB EB EB
```

אחרי שממשיכים עוד קצת בקוד (או כאמור - שמחכים שה-HW BP ייתפס), מגיעים ללולאה שמפענחת את ההצפנה ומקבלים למעשה קובץ בינארי שעוד נמצא בזיכרון:

```
003E106C   8B55 E8         MOV EDX,DWORD PTR SS:[EBP-18]
003E106F   0FB61402        MOVZX EDX,BYTE PTR DS:[EAX+EDX]
003E1073   3356 04         XOR EDX,DWORD PTR DS:[ESI+4]
003E1076   8B4D E8         MOV ECX,DWORD PTR SS:[EBP-18]
003E1079   881401          MOV BYTE PTR DS:[EAX+ECX],DL
003E107C   40              INC EAX
003E107D   4B              DEC EBX
003E107E   75 EC           JNE SHORT 003E106C
003E1080   8B45 E8         MOV EAX,DWORD PTR SS:[EBP-18]
003E1083   8985 78FFFFFF   MOV DWORD PTR SS:[EBP-88],EAX
003E1089   8B85 78FFFFFF   MOV EAX,DWORD PTR SS:[EBP-88]
003E108F   66:8138 4D5A    CMP WORD PTR DS:[EAX],5A4D
003E1094   0F85 20020000   JNE 003E12BA
003E109A   8B85 78FFFFFF   MOV EAX,DWORD PTR SS:[EBP-88]
003E10A0   8B40 3C         MOV EAX,DWORD PTR DS:[EAX+3C]
003E10A3   8B55 E8         MOV EDX,DWORD PTR SS:[EBP-18]
003E10A6   8D3C02          LEA EDI,[EAX+EDX]
003E10A9   813F 50450000   CMP DWORD PTR DS:[EDI],4550
Jump is taken
Dest=003E106C

Address   Hex dump                                          ASCII (ANSI - He
03E00000  4D B1 7B EB E8 EB EB EB EF EB EB EB 14 14 EB EB
```

קטע הקוד הנ"ל הולך לרוץ בעזרת Process Hollowing על התהליך עצמו (התהליך הראשון של הוירוס). בשלב זה במקום לעבוד יותר מדי קשה אפשר פשוט להעתיק את כל קטע הקוד שנמצא בזיכרון לתוך קובץ בעזרת 010 editor ונקבל קובץ ריצה תקין שניתן פשוט להריץ:

```
Startup   test.bad
Edit As: Hex   Run Script   Run Template

          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h:   4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00   MZ..........ÿÿ..
0010h:   B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ¸.......@.......
0020h:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0030h:   00 00 00 00 00 00 00 00 00 00 00 00 B8 00 00 00   ............¸...
0040h:   0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68   ..º..´.Í!¸.LÍ!Th
0050h:   69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F   is program canno
0060h:   74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20   t be run in DOS 
0070h:   6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00   mode....$.......
0080h:   B3 C3 57 0B F7 A2 39 58 F7 A2 39 58 F7 A2 39 58   ³ÃW.÷¢9X÷¢9X÷¢9X
0090h:   FA F0 E6 58 F3 A2 39 58 F4 DA D8 58 E9 A2 39 58   úðæXó¢9XôÚØXé¢9X
00A0h:   F4 DA E7 58 F6 A2 39 58 52 69 63 68 F7 A2 39 58   ôÚçXö¢9XRich÷¢9X
00B0h:   00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00   ........PE..L...
```

בשלב זה כבר אפשר לנתח את הקוד ב-IDA PRO.

אפשר ללכת בדרך אחרת ולהמשיך בקוד ולחכות להגיע לפונקציה WriteProcessMemory() שם הולך להיכתב הקוד ולשנות את הפקודות של ה-entrypoint ל-jmp entrypoint מה שיכניס את הקוד ללולאה אינסופית ותהיה אפשרות לעשות לו attach עם דיבאגר.

- WriteProcessMemory - זו פונקציית API של ווינדוס בעזרתה אפשר לשנות קוד בתהליכים שונים היא נמצאת בשימוש גם על ידי הדיבאגר כשרוצים לבצע שינוי בקוד של התהליך.

כדי למצוא את ה-entrypoint אפשר לפתוח את הקובץ ששמרנו לדיסק בעזרת ה-hex editor בדיבאגר. ואז נדע היכן לשנות. ישנם הבדלים בין הקוד שנמצא בזיכרון לזה שעל הדיסק. הסיבה לכך היא שבדיסק המידע נשמר כ-raw address ולא כ-virtual address (ומעוד סיבות נוספות), על מנת לקבל את ה-entrypoint שאנו רוצים אנו צריכים קודם למצוא את ה-entrypoint שאמור להיות כשהקוד עולה לזיכרון.

כדי למצוא את ה-entrypoint ניגש ל-base address של הזיכרון שהוקצה שמתחיל ב-MZ נבצע קצת חישוב מתמטי על מנת להגיע ל-entrypoint offset:

- [baseaddr+3c] - מביא את ה-offset בו מתחיל ה-PE format
- [baseaddr + B8] - מביא את ה-PEHeader
- [PEHeader+28] - יביא את ה-offset של ה-entrypoint

הדרך הכי קלה לחישוב של ה-offset הנכון היא להשתמש ב-cff explorer (מפני שמדובר בכלי אשר עושה זאת באופן אוטומטי).

ברגע שנכתוב 170b0 (שזה ה-RVA) נקבל לפי התמונה הבאה:

VA	004170B0
RVA	000170B0
File Offset	000164B0

- relative virtual address - RVA

ה-file offset זה ה-raw address. יש גם חישוב מתמטי ששווה להכיר: Virtual Address - צריך להיות בתחום של ה-RVA

File offset = RVA - virtual address + raw address

אז אחרי שיש ברשותנו את ה-entrypoint ניתן לגשת לקטע קוד שעוד לא הועתק עם WriteProcessMemory() ולשכתב בפקודות כמו בתמונה למטה:

```
03DE64B0   EB FE          JMP SHORT 03DE64B0
03DE64B2   90             NOP
03DE64B3   83EC 08        SUB ESP,8
03DE64B6   E8 05AAFEFF    CALL 03DD0EC0
```

אחרי ה-attach נשכתב את הקוד שוב:

```
Paused
004170B0   55             PUSH EBP
004170B1   89E5           MOV EBP,ESP
004170B3   83EC 08        SUB ESP,8
004170B6   E8 05AAFEFF    CALL 00401AC0
004170BB   85C0           TEST EAX,EAX
```

במקום לנסות לשחק עם ה-attach במקרה הספציפי הזה אפשר לפתוח את הקובץ שנשמר בדיסק ישר לדיבאגר ומשם להמשיך לנתח את הוירוס.

הסיבה היא שה-Packer הראשוני פתח את הקובץ של הוירוס עצמו ואיננו צריכים אותו עכשיו כדי שהוירוס יתפקד.

קטע הקוד הבא מזריק ל-explorer את אותו קובץ בעזרת אותה הטכניקה בדיוק:

```
mov     ecx, [ebp+var_4]
push    ecx
push    offset currentFilename
call    d4d_getArrOfAPIFunctions
mov     edx, [eax+arrOfApiFunctions.wcscpy]
call    edx


call    d4d_getLinearTibAddr
mov     eax, [eax+20h]
mov     processId, eax
mov     dword_B34FC, 0
push    0
push    offset d4d_startAddressInExplorerProcess ; this is a thread function injected inside explorer process
call    sub_9B110
add     esp, 8
```

הקובץ אשר מוזרק ל-explorer מתחיל במקום אחר על ידי שימוש בפונקציה RtlCreateUserThread(). זו פונקציה לא מתועדת של מיקרוסופט שעושה בדיוק מה שעושה CreateRemoteThread().

ב-Thread שמוזרק ל-explorer מתבצעות הפעולות הבאות:

- מתבצע Process Hollowing נוסף אשר פותח את svchost.
- מוחקים את כל הנקודות שיחזור שיש במחשב ומבטלים את ה-Services הבאים:
  - Wscsvc
  - WinDefend
  - wuauserv
  - BITS
  - ERSvc
  - WerSvc

בתמונה הבאה ניתן לראות את קטע הקוד שעוצר את ה-Services במערכת:

```
.text:000A55A6                          ; ---------------------------------------------
.text:000A55A6
.text:000A55A6                          ; // list of services to stop
.text:000A55A6                          ; // wscsvc
.text:000A55A6                          ; // WinDefend
.text:000A55A6                          ; // wuauserv
.text:000A55A6                          ; // BITS
.text:000A55A6                          ; // ERSvc
.text:000A55A6                          ; // WerSvc
.text:000A55A6
.text:000A55A6
.text:000A55A6                          loc_A55A6:                              ; CODE XRE
.text:000A55A6 8B 55 FC                                 mov     edx, [ebp+var_4]
.text:000A55A9 83 C2 01                                 add     edx, 1
.text:000A55AC 89 55 FC                                 mov     [ebp+var_4], edx
.text:000A55AF
.text:000A55AF                          loc_A55AF:                              ; CODE XRE
.text:000A55AF 8B 45 FC                                 mov     eax, [ebp+var_4]
.text:000A55B2 83 BC 85 00 FD FF FF 00                  cmp     [ebp+eax*4+var_300], 0
.text:000A55BA 74 15                                    jz      short loc_A55D1
.text:000A55BC
.text:000A55BC
.text:000A55BC 8B 4D FC                                 mov     ecx, [ebp+var_4]
.text:000A55BF 8B 94 8D 00 FD FF FF                     mov     edx, [ebp+ecx*4+var_300]
.text:000A55C6 52                                       push    edx
.text:000A55C7 E8 84 54 FF FF                           call    d4d_stopService
.text:000A55CC 83 C4 04                                 add     esp, 4
.text:000A55CF EB D5                                    jmp     short loc_A55A6
```

על מנת לבטל את כלל נקודות השחזור במערכת, משנים ל-1 את DisableSR ברגיסטרי שמבטל את הנקודות שיחזור:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\SystemRestore]
"DisableSR"=dword:00000001
```

בתמונה הבאה ניתן לראות את קטע הקוד אשר מבטל את הנקודות שיחזור במערכת:

```
.text:000A6186
.text:000A6186                          ; // disableSR value
.text:000A6186
.text:000A6186 C7 85 1C FD FF FF 01 00+                 mov     [ebp+Data], 1
.text:000A6190 6A 04                                    push    4               ; Type
.text:000A6192 6A 04                                    push    4               ; DataSize
.text:000A6194 8D 8D 1C FD FF FF                        lea     ecx, [ebp+Data]
.text:000A619A 51                                       push    ecx             ; Data
.text:000A619B 8D 95 F4 FE FF FF                        lea     edx, [ebp+disableSR]
.text:000A61A1 52                                       push    edx             ; int
.text:000A61A2 8B 85 68 FF FF FF                        mov     eax, [ebp+KeyHandle]
.text:000A61A8 50                                       push    eax             ; KeyHandle
.text:000A61A9 E8 32 64 FF FF                           call    d4d_ZwSetValueKey
.text:000A61AE 83 C4 14                                 add     esp, 14h
```

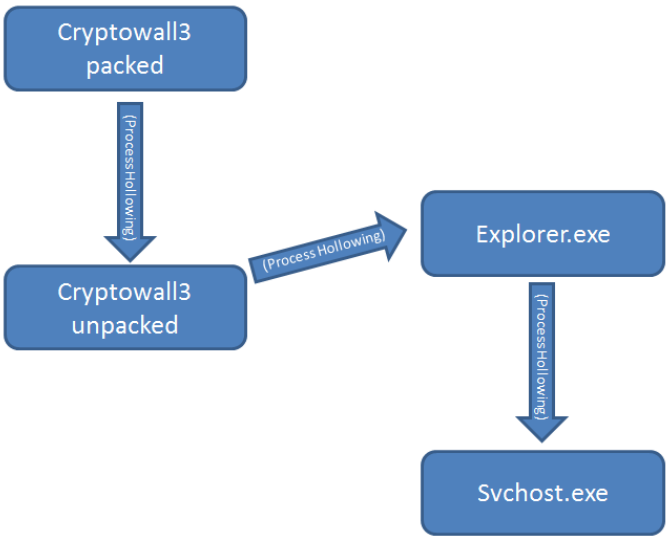
ב-thread שמוזרק ל-svchost מתבצעים הדברים הבאים:

- מתבצע חיבור לשרת C&C של הוירוס לקבלת מפתח RSA
- הצפנת הקבצים

בחלק הבא יהיה פירוט מורחב על הפקודות של ה-C&C ועל איך מוצפנים הקבצים.

סקירה כללית על CryptoWall3

כאמור, המטרה העיקרית של CryptoWall3 היא להצפין את כל הקבצים במחשב ולדרוש כופר עבור אותם קבצים על מנת לפענח את הקבצים בחזרה. התרשים הבא מראה את ההתנהגות הכללית של הוירוס:



בתהליך של explorer יש thread אשר רץ ומטרתו העיקרית היא למחוק את כל הנקודות שיחזור במחשב ולבטל ברגיסטרי את האפשרות לבצע אחת חדשה ולבטל את כל ה-Services אשר יכולים להפריע לו כפי שהוסבר קודם.

התהליך של ה-svchost מתקשר עם השרת C&C כדי לבדוק אם הכל תקין לפני שממשיכים להצפין את הקבצים. הפקודה הראשונה ששולח הוירוס הינה בנויה באופן הבא:

```
{1|crypt1|MD5OfComputerInfo|32/64bit|Cpu Architecture|isAdmin|IP Address of Infected Computer}
```

להלן הסבר:

שם שדה	תיאור
cmdNum	יש **3** אפשרויות: **1,3 או 7**
MalName	Crypt1
Md5OfComputerInfo	פרטים מזהים על המחשב
Is64Bit	**1** אם **32** ביט **2** אם **64**
CPU Architecture	לא ידוע לפי מה נקבע (בניתוח שלי **הופיע 1**)
IsAdmin	**1 -** אם **משתמש מוגבל**, **2 -** אם **מנהל מערכת**
IpAddress	כתובת **ה-**IP של המחשב

הפקודה הנ"ל בודקת כי השרת תקין וניתן לתקשר עימו, במידה והוא לא תקין תהיה לולאה אינסופית שתנסה להתחבר לאחד השרתים אשר הוגדר לו.

כל המידע אשר נשלח מוצפן עם מפתח RC4 אשר נוצר בזמן ריצה באופן אקראי. מה שנוצר זה מחרוזת אקראית שהיא למעשה התיקיה אליה ניגש הוירוס כדי לתקשר עם השרת. המפתח הינו נגזרת משם התיקיה. הוא בנוי באופן כזה שהתווים בו ממוינים ממספרים קודם ובסוף אותיות מהקטן לגדול. לדוגמא, אם לשם התיקיה נקבע: 33i50tglylbv, אז המפתח יהיה: 0335bgilltvy.

הוירוס ניגש לאחת הכתובות שהן hardcoded בקוד, וניגש ל-url/folder שנקבע. בשלב כתיבת מאמר זה, שרת ה-C&C כבר נסגר ולכן לא היה ניתן לדעת בדיוק מה היה אמור להתקבל מהשרת לאחר שליחת מידע זה.

בשלב זה עברתי על חלק הקוד אשר היה אחראי לפענח את התשובה שהגיעה מהשרת, ולפי ניחושים / לוגיקה שהתאימה לזרימת התוכנית, מתברר שבפקודה הראשונה מקבלים תשובה אשר מורכבת מ-2 ערכים:

```
{x,1}
```

במידה ולא מקבלים 1 בתשובה, הוירוס נכנס ללולאה אינסופית ומנסה להתחבר לשרתים אחרים. במידה ויש חיבור תקין עם ה-C&C נוצר thread חדש שתפקידו לבקש מהשרת מפתח RSA.

פרטי הבקשה:

שם שדה	תיאור
cmdNum	7
malName	crypt1
Md5OfComputerInfo	האש של הפרטים המזהים של המחשב
Const?	1

פקודה זו שולחת בקשה לקבל מפתח מסוג RSA על מנת להצפין את הקבצים. על מנת להבין את מה שחוזר מהשרת היה עלי לנחש את הפרמטרים אשר מוחזרים.

בתחילה, לא היה לי מושג כמה פרמטרים חוזרים מהשרת אך לאחר הסתכלות בקוד ראיתי שהוא מפריד את הפקודה ל-5 חלקים. (החלק בקוד: cmp countOfTokens, 5) בתמונה הבאה:

```
.text:000A6B18 E8 E3 DF FE FF                          call    d4d_explodeBuf
.text:000A6B1D 83 C4 10                                add     esp, 10h
.text:000A6B20
.text:000A6B20
.text:000A6B20 85 C0                                   test    eax, eax
.text:000A6B22 0F 84 FE 00 00 00                       jz      loc_A6C26
.text:000A6B28
.text:000A6B28
.text:000A6B28 83 7D F4 00                             cmp     [ebp+arrOfDataTokens], 0
.text:000A6B2C 0F 84 E4 00 00 00                       jz      loc_A6C16
.text:000A6B32
.text:000A6B32
.text:000A6B32 83 7D EC 05                             cmp     [ebp+countOfTokens], 5
.text:000A6B36 0F 85 DA 00 00 00                       jnz     loc_A6C16
.text:000A6B3C
```

לאחר מכן ניחשתי את הפרמטרים האחרים שצריך לקבל בתשובה גיליתי שהפקודה נראית בסגנון הבא:

`{x,y,z,f,g}`

- ב-x שמתי מספר.
- ב-y שמתי מחרוזת.
- ב-z שמתי מספר.
- ב-f שמתי מספר.
- ב-g שמתי מספר.

המשכתי לעקוב אחרי הקוד ב-Debugger וראיתי לאט לאט איך להשלים כל פרמטר. כשהגעתי לפונקציות שמטפלות בהצפנה ראיתי באיזה חלק מהתשובה הן מנסות להשתמש לטובת הרכבת מפתח ה-RSA.

לאחר מכן הגעתי לקטע קוד שמחשב כל מיני CRC32 ולא הבנתי למה, חיפשתי בגוגל את הערכים והבנתי ש-מדובר ב-country codes, להלן תמונה:

```
.text:000A2A29                         loc_A2A29:                          ; CODE XREF: sub_A2970+A8↑j
.text:000A2A29 8B 45 E4                                mov     eax, [ebp+var_1C]
.text:000A2A2C 8B 48 1C                                mov     ecx, [eax+1Ch]
.text:000A2A2F 51                                      push    ecx
.text:000A2A30 E8 EB 48 FF FF                          call    d4d_calcCRC32_1
.text:000A2A35 83 C4 04                                add     esp, 4
.text:000A2A38
.text:000A2A38
.text:000A2A38 50                                      push    eax
.text:000A2A39 E8 E2 FE FF FF                          call    d4d_checkCountryCodeByCrc32
.text:000A2A3E 83 C4 04                                add     esp, 4
.text:000A2A41
.text:000A2A41
.text:000A2A41 85 C0                                   test    eax, eax
.text:000A2A43 74 18                                   jz      short loc_A2A5D
.text:000A2A45
.text:000A2A45
.text:000A2A45 E8 06 26 00 00                          call    d4d_removeRegKeys
```

לאחר הסתכלות בקוד הבנתי שהוירוס בודק מה ה-Code Country לפי הכתובת IP שנשלחה בפקודה הראשונה ששלחנו לשרת (בה קיבלנו את התשובה {x,1}), מבצע עליו CRC32, במידה והוא ברשימה של CRC32 של מדינות אשר נמצאות ברשימה שהופיעה, ההצפנה לא הייתה מתבצעת ויתרה מזאת - הוירוס היה מסיר עצמו מהמחשב, את רשימת המדינות ניתן לראות בתמונה הבאה:

```
countryCodes    dd BY                          ; Belarus
                dd UA                          ; Ukraine
                dd RU                          ; Russia
                dd KZ                          ; Kazakhstan
                dd AM                          ; Armenia
```

במידה ולא שפר עלינו המזל (אנחנו לא גרים באחת מהמדינות הנ"ל), מתקדמים הלאה - לקטע קוד הבא:

```
.text:000A2A5D C7 45 A0 00 00 00 00    mov     [ebp+var_60], 0
.text:000A2A64 C7 45 A4 00 00 00 00    mov     [ebp+var_5C], 0
.text:000A2A6B C7 45 A8 00 00 00 00    mov     [ebp+var_58], 0
.text:000A2A72 C7 45 9C 00 00 00 00    mov     [ebp+var_64], 0
.text:000A2A79 C7 45 C8 00 00 00 00    mov     [ebp+md5Hash], 0
.text:000A2A80 C7 45 AC 00 00 00 00    mov     [ebp+md5Len], 0
.text:000A2A87 8D 45 AC                lea     eax, [ebp+md5Len]
.text:000A2A8A 50                      push    eax
.text:000A2A8B 8D 4D C8                lea     ecx, [ebp+md5Hash]
.text:000A2A8E 51                      push    ecx
.text:000A2A8F 8D 55 9C                lea     edx, [ebp+var_64]
.text:000A2A92 52                      push    edx
.text:000A2A93 8D 45 A8                lea     eax, [ebp+var_58]
.text:000A2A96 50                      push    eax
.text:000A2A97 8B 4D F0                mov     ecx, [ebp+lengthOfKey]
.text:000A2A9A 51                      push    ecx
.text:000A2A9B 8B 55 EC                mov     edx, [ebp+rsaKey]
.text:000A2A9E 52                      push    edx
.text:000A2A9F 8B 45 D0                mov     eax, [ebp+phProv]
.text:000A2AA2 50                      push    eax
.text:000A2AA3 E8 98 F2 FF FF          call    d4d_importPublicKeyInfo
.text:000A2AA8 83 C4 1C                add     esp, 1Ch
```

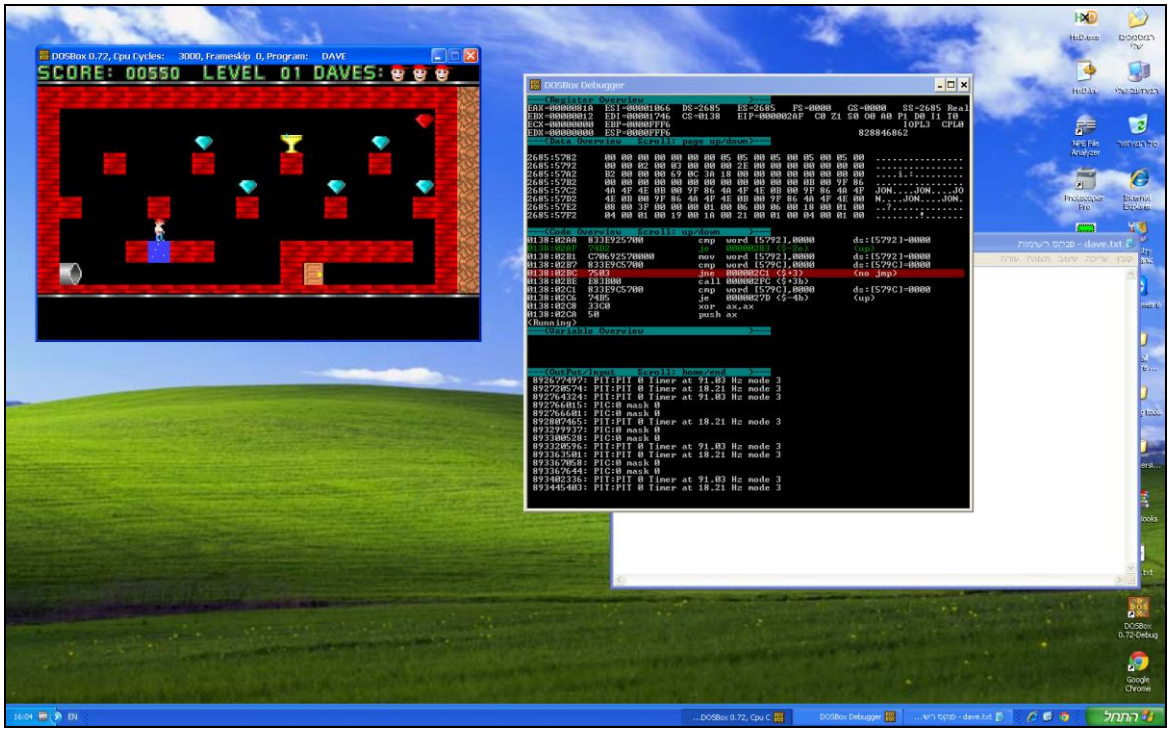
הקטע קוד הזה מסדר את המפתח בייצוג הבינארית שלו כמו בתמונה למטה:

```
0000  30 82 01 0a 02 82 01 01  00 ac ed c3 1d 11 7f 63
0010  db 25 50 2e 9a c6 c1 f5  b7 23 c8 a0 71 a4 6e d6
0020  c8 29 17 8f 76 b6 8c 88  33 bf c9 0e 3d c8 0d 87
0030  11 60 e4 f0 77 ae e5 b4  47 6f b1 35 98 d3 44 d0
0040  52 c7 60 2e 7f e9 6c 3c  61 c2 36 3d a7 f5 32 88
0050  de 3c c4 79 62 91 b0 4b  24 78 a2 2e 6a 29 a9 ee
0060  0e 7a d8 0d 9e 12 7b b2  53 d1 17 8c 01 dc eb fb
0070  18 4d c0 ae df 61 7e 2b  dd 15 b5 65 b3 bc b9 25
0080  58 c9 ed 9e ef 9f 26 9b  79 c3 8e 13 92 9e 62 f3
0090  fe 8d ab 33 b4 40 a1 7b  0e b1 71 56 b4 9d 7b cb
00a0  61 9d 70 1d 9d b4 49 c9  46 42 fc 64 44 67 eb 8b
00b0  ea 7c 29 31 cb 4c 32 12  91 6c dd 04 59 07 51 6a
00c0  e6 40 fa ea 4e b2 ae 64  21 2e 6b 00 99 f0 7c 26
00d0  6e ad 6c 15 18 36 dc 81  61 e9 ce 28 7f f8 89 82
00e0  ee ed c5 ee 54 ee aa cd  01 72 75 71 59 fd fc cd
00f0  4d 53 3e 22 71 47 7f 24  e5 51 28 36 12 09 6b 0d
0100  af c9 37 9b e0 d1 00 67  11 02 03 01 00 01
```

לאחר מכן מחשבים את ה-MD5 של מפתח, לטובת שימוש עתידי. השלב הבא הינו להוריד תמונה מהאינטרנט של הוראות לאיך לפענח את הקבצים:

```
text:000A2AD0                                 loc_A2AD0:                                  ; CODE XREF: sub_A2970+185↓j
text:000A2AD0 8D 4D A4                                          lea     ecx, [ebp+var_5C]
text:000A2AD3 51                                                push    ecx
text:000A2AD4 8D 55 A0                                          lea     edx, [ebp+var_60]
text:000A2AD7 52                                                push    edx
text:000A2AD8 8B 45 CC                                          mov     eax, [ebp+md5Str]
text:000A2ADB 50                                                push    eax
text:000A2ADC E8 5F 41 00 00                                    call    d4d_getPngPictureFromCnC
text:000A2AE1 83 C4 0C                                          add     esp, 0Ch
```

בגלל שהשרת נפל אין לי ממש את ה-png שיוצרי הוירוס הכינו, אז על מנת שהוירוס לא יגלה שהוא רץ במעבדה - העלתי תמונה של מחקר אחר שאני עורך במקביל ☺



בשלב הבא הוירוס שומר את המפתח תחת המיקום הבא ב-Registry:

```
HKCU\software\md5OfComputerInfo
```

שם גם נשמר קובץ ה-HTML וקובץ הטקסט שמסבירים בו מה קרה לקבצים.

לאחר מכן, בודקים אם הכתובת של TOR תקינה. בשלב זה הבנתי איזה עוד חלק בפקודה חסר לי:

```
.text:000A2B30 8B 4D A4                                         mov     ecx, [ebp+var_5C]
.text:000A2B33 51                                               push    ecx
.text:000A2B34 8B 55 A0                                         mov     edx, [ebp+var_60]
.text:000A2B37 52                                               push    edx
.text:000A2B38 8B 45 E4                                         mov     eax, [ebp+var_1C]
.text:000A2B3B 50                                               push    eax
.text:000A2B3C E8 1F 17 00 00                                   call    d4d_CheckforTorUrl
.text:000A2B41 83 C4 0C                                         add     esp, 0Ch
.text:000A2B44 89 45 C4                                         mov     [ebp+isValidURL], eax
```

בתמונה הזו ראיתי באיזה חלק הוא משתמש מהקלט שהתקבל מהשרת (במקרה שלי: במחרוזת "abc"),

זה חלק מהקובץ HTML שמוצג למשתמש אחרי שהצפינו את הקבצים:

For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below:

1.http://www.torforall.com/
2.http://www.torman2.com/
3.http://www.torwoman.com/
4.http://www.torroadsters.com/

If for some reasons the addresses are not available, follow these steps:

1. Download and install tor-browser: http://www.torproject.org/projects/torbrowser.html.en
2. After a successful installation, run the browser and wait for initialization.
3. Type in the address bar: crypt1/abc
4. Follow the instructions on the site.

IMPORTANT INFORMATION:

Your Personal PAGE: http://www.torforall.com/
Your Personal PAGE(using TOR): crypt1/abc
Your personal code (if you open the site (or TOR 's) directly): abc

ולאחר ניתוח מלא, הבנתי כיצד בנויה התשובה מהשרת:

שם שדה	תיאור
unk	כל מספר
torUrl	כתובת של TOR
Victim uri	שם ייחודי
Civtim country code	הקוד של המדינה
RSA public key	מפתח בגודל 2048 ביט נשלח

[למי שמעוניין לראות פרטים מלאים על התקשורת ושאר הדברים מוזמן להסתכל בסוף המאמר על לינק לניתוח מלא של CryptoWall3]

השלב הבא הינו הצפנת הקבצים - ישנו קטע קוד אשר בו סורקים את כל הכוננים שיש במחשב, במידה והכונן הינו CD_ROM מדלגים לכונן הבא:

```
text:000A2C0F
text:000A2C0F 8B 4D DC                    mov     ecx, [ebp+var_24]
text:000A2C12 8B 55 D8                    mov     edx, [ebp+var_28]
text:000A2C15 8D 04 4A                    lea     eax, [edx+ecx*2]
text:000A2C18 89 45 C0                    mov     [ebp+var_40], eax
text:000A2C1B 8B 4D C0                    mov     ecx, [ebp+var_40]
text:000A2C1E 51                          push    ecx
text:000A2C1F E8 8C EE FE FF              call    d4d_getArrOfAPIFunctions
text:000A2C24 8B 90 F0 01 00 00           mov     edx, [eax+arrOfApiFunctions.GetDriveTypeW]
text:000A2C2A FF D2                       call    edx
text:000A2C2C
text:000A2C2C
text:000A2C2C 83 F8 05                    cmp     eax, DRIVE_CDROM
text:000A2C2F 0F 84 F5 00 00 00           jz      loc_A2D2A
text:000A2C35
```

לאחר מכן, כל כונן נסרק ב-thread נפרד שבו מצפינים את כל הקבצים:

```
.text:000A2CD2
.text:000A2CD2 8B 55 F8                   mov     edx, [ebp+var_8]
.text:000A2CD5 8B 45 E8                   mov     eax, [ebp+var_18]
.text:000A2CD8 8D 0C 90                   lea     ecx, [eax+edx*4]
.text:000A2CDB 51                         push    ecx
.text:000A2CDC 6A 00                      push    0
.text:000A2CDE 8B 55 FC                   mov     edx, [ebp+var_4]
.text:000A2CE1 52                         push    edx
.text:000A2CE2 68 E0 69 0A 00             push    offset d4d_encryptFiles
.text:000A2CE7 6A FF                      push    0FFFFFFFFh
.text:000A2CE9 E8 52 76 FF FF             call    d4d_resumeThread
.text:000A2CEE 83 C4 14                   add     esp, 14h
.text:000A2CF1 89 45 BC                   mov     [ebp+var_44], eax
```

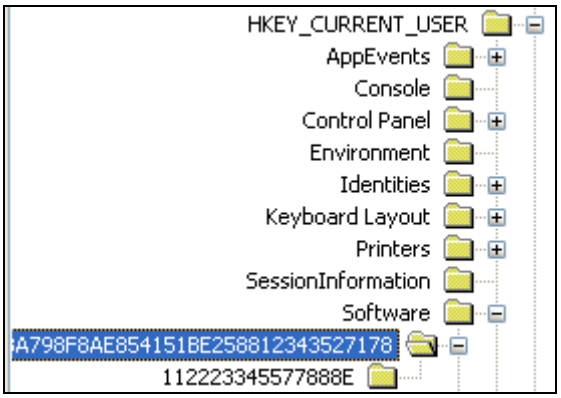
כל קובץ מוצפן בעזרת מפתח AES בגודל 256 ביט שנוצר באופן אקראי ומוצפן יחד עם ה-RSA שהוא ה-public. בכדי לבדוק האם הקובץ מוצפן, הוירוס בודק אם ה-0x10 בתים ראשונים בקובץ זהים ל-Hash של מפתח RSA שהוא public ולאחריו יש את המפתח AES שמוצפן ב-RSA ובסוף את המידע עצמו שמוצפן ב-AES.

הוירוס מוסיף את עצמו לרגיסטרי ל-autorun במקומות הבאים:

```
HKCU/Software/Microsoft/Windows/CurrentVersion/run
HKCU/Software/Microsoft/Windows/CurrentVersion/RunOnce
```

השם של המפתח הוא CRC32 של ה-MD5 שנוצר מהפרטים של המחשב, החישוב מתבצע על ה-Hash כשהוא באותיות קטנות.

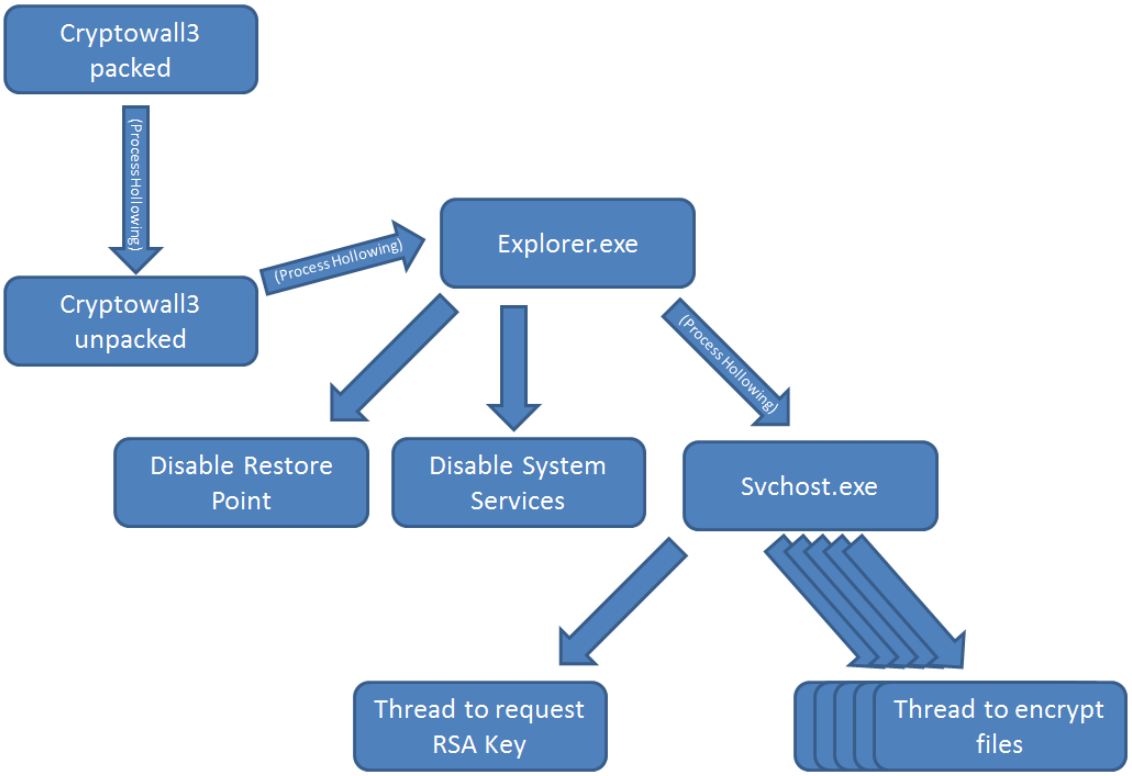
בנוסף, מתבצעת רשימת כלל הקבצים אשר הוצפנו, באותו מקום אך במפתח פנימי יותר שמתחיל ב-112223...:



אחרי שהוירוס סיים להצפין את כל הקבצים הוא מוחק את עצמו מהמחשב והקורבן נשאר רק עם הוראות לאיך לפענח את הקבצים.

פיענוח הקבצים מתבצע רק לאחר שהקורבן משלם כסף ליוצרי הוירוס. כחלק מתהליך זה מקבלים קישור לתוכנה להורדה, בה הקורבן מזין את המפתח שקיבל והתוכנה דואגת לפענח את הקבצים בחזרה. כדי לקבל את המפתח הנכון אתם משתמשים במזהה שיוצרי הוירוס יצרו.

להלן סקיצה כללית של פעילות הוירוס:



סיכום

במאמר זה הצגתי את השלבים שבהם יש לפעול על מנת להוריד את ההגנות שיש ל-CryptoWall3, כולל התגברות על המכשולים ששמו על מנת להקשות על הרברסר.

כמו כן הצגנו סקירה כללית על איך עובד הוירוס בגדול. ונקודה שמראה לנו כמה "לא מבזבזים זמן" בנושא הזה - בזמן שאנו מדברים כבר יצא CryptoWall4...

מקורות נוספים:

- CryptoWall3 report

על מחבר המאמר (d4d)

מחבר המאמר עוסק בתחום ה-Reverse Engineering בחברת איירון סורס במחלקת ה-Security ואוהב לחקור משחקי מחשב והגנות, לכל שאלה שיש או ייעוץ ניתן לפנות אלי בשרת ה-IRC של Nix, בערוץ: #reversing

או באתר:

www.cheats4gamer.com

או בכתובת האימייל:

llcashall@gmail.com