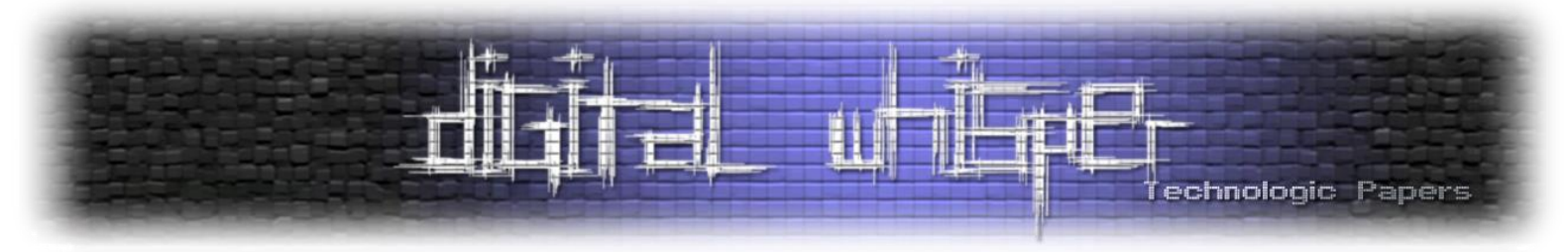


כל מה שרציתם לדעת על PsExec ומעולם לא העזתם לשאול

מאת עומר כחלון

הקדמה

Psexec הוא אחד הכלים החזקים והידועים מבית היוצר של Sysinternals. כעקרון, מטרתו הלגיטימית היא לאפשר ניהול של עמדות Windows ללא צורך בגישה פיזית לעמדה. הכלי מאפשר התחברות מרוחקת והרצת קוד על מכונת היעד. תוקפי סייבר לעיתים רבות מנצלים את הכלי על מנת לבצע Lateral Movement ובכך להתפשט לעמדות נוספות ברשת. כחוקרים, חשוב להבין איך עובד הכלי על מנת שנוכל לזהות שימוש שלו ברשתות. במאמר זה אסביר איך פועל הכלי מאחורי הקלעים.

הדגמה בסיסית

לצורך הנוחות, אגדיר את עמדת המקור (שמריצה את ה-PsExec) כ-Client ואת העמדה המרוחקת אליה אנחנו פונים באמצעות PsExec על מנת להריץ פקודת כ-Server. הרצת הכלי היא די פשוטה ומתבצעת באופן הבא:

```
Psexec.exe \\remote server <executable/action>
```

נראה דוגמה:

```
C:\Users\omer\Desktop\DW - psexec>whoami
desktop-ufhfqmj\omer

C:\Users\omer\Desktop\DW - psexec>PsExec64.exe \\192.168.17.134 cmd

PsExec v2.43 - Execute processes remotely
Copyright (C) 2001-2023 Mark Russinovich
Sysinternals - www.sysinternals.com

Microsoft Windows [Version 10.0.17763.1935]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
msedgewin10\ieuser

C:\Windows\system32>
```

ובכן, אפשר לראות בתמונה שהרצתי את PsExec על הכתובת 192.168.17.134 ובחרתי לפתוח את האפליקציה cmd. לאחר מכן הרצתי את הפקודה whoami וניתן לראות שכעת קיבלנו את שם המשתמש על ה-Server.

ל-PsExec יש המון פרמטרים שיכולים להשפיע על הריצה. הסבר על כל הפרמטרים השונים שיש ל-PsExec תוכלו למצוא ב-MSDN (קישור לדף הרלוונטי מצורף בסוף).

איך קורה הקסם?

אז בואו נתחיל, בחלק זה אחקור את הקובץ ב-Ida. כמובן שלא אעבור פונקציה פונקציה, אלא אתמקד רק בפונקציות הרלוונטיות לפעילות הכלי.

בדיקת גרסת מערכת ההפעלה

תחילה מתבצעת בדיקה אם גרסת מערכת הפעלה מתאימה לריצה של PsExec. גרסת מערכת ההפעלה צריכה להיות Windows XP ומעלה. הבדיקה מתבצעת באמצעות שימוש בפונקצית API בשם VerifyVersionInfo:

```
loc_4083D7:                 ; lpMem
push    edi
call    sub_4139EA
add     esp, 4
mov     [ebp+nSize], 10h
lea     eax, [ebp+nSize]
push    eax                 ; nSize
push    offset word_4956A0 ; lpBuffer
call    ds:GetComputerNameW
call    call_VerifyVersionInfo ; check the os version
test    eax, eax
jnz     short loc_408431
```

במידה וגרסת מערכת ההפעלה לא מתאימה, תודפס הודעה למסך:

```
.text:00007FF75F139A2E lea     ecx, [rax+2]     ; Ix
.text:00007FF75F139A31 call    __acrt_iob_func
.text:00007FF75F139A36 mov     rcx, rax         ; Format
.text:00007FF75F139A39 lea     rdx, aPsexecRequires ; "PsExec requires Windows XP or higher.\n"...
.text:00007FF75F139A40 call    printf
.text:00007FF75F139A45 mov     eax, 0FFFFFFFFh
.text:00007FF75F139A4A jmp     loc_7FF75F139DA2
```

תנאי השימוש ב-PsExec

הקובץ מחפש את המפתח "HKEY_CURRENT_USER\Software\Sysinternals\PsExec" ב-Registry ובודק האם קיים value בשם "EulaAccepted". ה-Eula מתייחס לתנאי השימוש שמוצגים למשתמש במידה ולא אושרו. המחרוזות המכילות את מפתח וערך ה-Registry נוצרות באופן דינאמי (רק בעת ריצה):

```
.text:00007FF70EFA1CD1 xor     eax, eax
.text:00007FF70EFA1CD3 lea     rdx, [rsp+268h+SubKey] ; lpSubKey
.text:00007FF70EFA1CD8 mov     [rsp+268h+hKey], rax
.text:00007FF70EFA1CDD mov     r9d, 101h
.text:00007FF70EFA1CE3 mov     dword ptr [rsp+268h
.text:00007FF70EFA1CE7 xor     r8d, r8d
.text:00007FF70EFA1CEA lea     rax, [rsp+268h+hKey
.text:00007FF70EFA1CEF mov     rcx, 0FFFFFFFF80000001h ; hKey
.text:00007FF70EFA1CF6 mov     [rsp+268h+phkResult], rax ; phkResult
.text:00007FF70EFA1CFB call    cs:RegOpenKeyExW
.text:00007FF70EFA1D01 test    eax, eax
.text:00007FF70EFA1D03 jnz     short loc_7FF70EFA1D61

[rsp+268h+SubKey]=[Stack[000015AC]:aSoofftwwaarre]
aSoofftwwaarre db 'S',0,'o',0,'f',0,'t',0,'w',0,'a',0,'r',0,'e',0,'\',0,'S',0,'y',0,'s'
               db 0,'i',0,'n',0,'t',0,'e',0,'r',0,'n',0,'a',0,'l',0,'s',0,'\',0,'P',0
               db 's',0,'E',0,'x',0,'e',0,'c'

.text:00007FF70EFA1D05 mov     rcx, [rsp+268h+hKey] ; hKey
.text:00007FF70EFA1D0A lea     rax, [rsp+268h+cbData]
.text:00007FF70EFA1D0F mov     [rsp+268h+lpcbData], rax ; lpcbData
.text:00007FF70EFA1D14 lea     rdx, ValueName  ; "EulaAccepted"
.text:00007FF70EFA1D1B lea     rax, [rsp+268h+Data]

RAX 0000004CA6BCF208
RBX 0000000000000001
RCX FFFFFFFF80000001
RDX 0000004CA6BCF210
RIP 00007FF70EFA1CFB
R8  0000000000000000
R9  0000000000000101
Modules
Path
C:\Users\omer\Desktop\D
C:\Windows\WinSxS\amd
C:\Windows\SYSTEM32\M
```

במידה וה-Data ב-EulaAccepted שווה ל-1 זה אומר שהמשתמש כבר אישר את תנאי השימוש ואין צורך להציג אותם שוב, 0 משמעו ההפך. לאחר אישור המשתמש מתבצע שינוי הערך באותו האופן:

```
.text:00007FF70EFA2766
.text:00007FF70EFA2766 loc_7FF70EFA2766:       ; phkResult
.text:00007FF70EFA2766 lea     r8, [rsp+4C0h+var_470]
.text:00007FF70EFA276B mov     rcx, 0FFFFFFFF80000001h ; hKey
.text:00007FF70EFA2772 lea     rdx, [rbp+3C0h+SubKey] ; lpSubKey
.text:00007FF70EFA2779 call    cs:RegCreateKeyW
.text:00007FF70EFA277F test    eax, eax
.text:00007FF70EFA2781 jnz     short loc_7FF70EFA27BB

[rbp+3C0h+SubKey]=[Stack[000024A8]:aSoofftwwaarre]
aSoofftwwaarre db 'S',0,'o',0,'f',0,'t',0,'w',0,'a',0,'r',0,'e',0,'\',0,'S',0,'y',0,'s'
               db 0,'i',0,'n',0,'t',0,'e',0,'r',0,'n',0,'a',0,'l',0,'s',0,'\',0,'P',0
               db 's',0,'E',0,'x',0,'e',0,'c'

.text:00007FF70EFA2783 mov     rcx, [rsp+4C0h+var_470] ; hKey
.text:00007FF70EFA2788 lea     rax, [rsp+4C0h+var_490]
.text:00007FF70EFA278D mov     dword ptr [rsp+4C0h+lpcbData], 4 ; cbData
.text:00007FF70EFA2795 lea     rdx, ValueName  ; "EulaAccepted"
.text:00007FF70EFA279C mov     r9d, 4          ; dwType
.text:00007FF70EFA27A2 mov     [rsp+4C0h+lpData], rax ; lpData
.text:00007FF70EFA27A7 xor     r8d, r8d        ; Reserved
.text:00007FF70EFA27AA call    cs:RegSetValueExW
.text:00007FF70EFA27B0 mov     rcx, [rsp+4C0h+var_470] ; hKey

RAX 0000000000000001
RBX 0000000000000001
RCX FFFFFFFF80000001
RDX 0000007747D6F9A0 Stack[000024A8]:
RSI 0000000000000000
ID  0
VIP 0
VIF 0
AC  0
VM  0
RF  0
NT  0
C:\Windows\SYSTEM32\MPR.dll
Threads
Decimal Hex   State   Name
9384    24A8  Ready   PsExec64.exe
19184   4AF0  Ready   7FF87E1BD110
3244    CAC   Ready   7FF87E1BD110
```

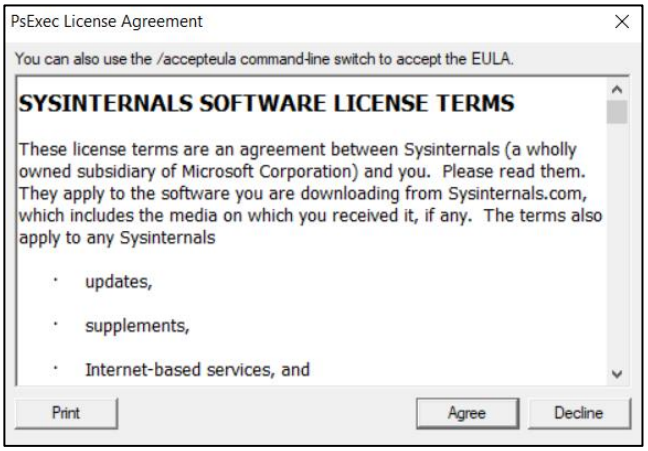
אם זו הריצה הראשונה של הכלי על העמדה הערך ב-Registry לא יהיה קיים, לאחר הצגת תנאי השימוש הוא יווצר עם הערך 1 או 0 תלוי אם המשתמש אישר את תנאי השימוש.

וכך זה נראה ב-Registry:

```
Registry Editor
File  Edit  View  Favorites  Help
Computer\HKEY_CURRENT_USER\SOFTWARE\Sysinternals\PsExec
Sysinternals
  AccessChk
  Hex2Dec
  LoadOrder
  Process Explorer
  PsExec

Name          Type      Data
(Default)     REG_SZ    (value not set)
EulaAccepted  REG_DWORD 0x00000001 (1)
```

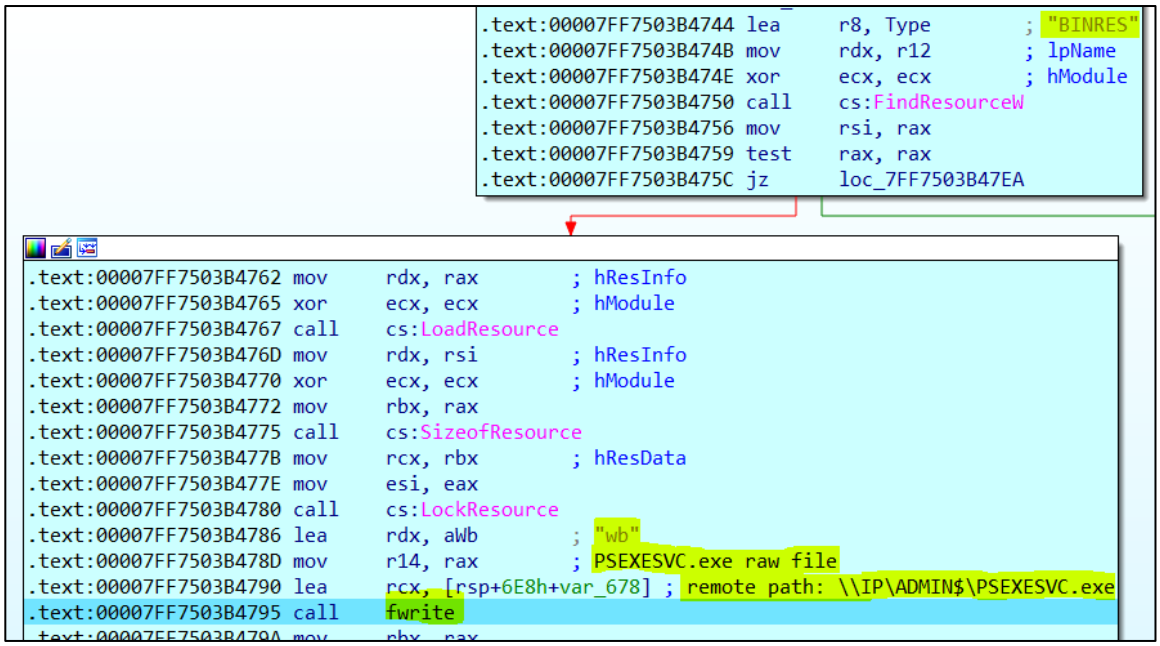
אם המשתמש עדיין לא אישר את תנאי השימוש יוקפץ למסך החלון הבא:



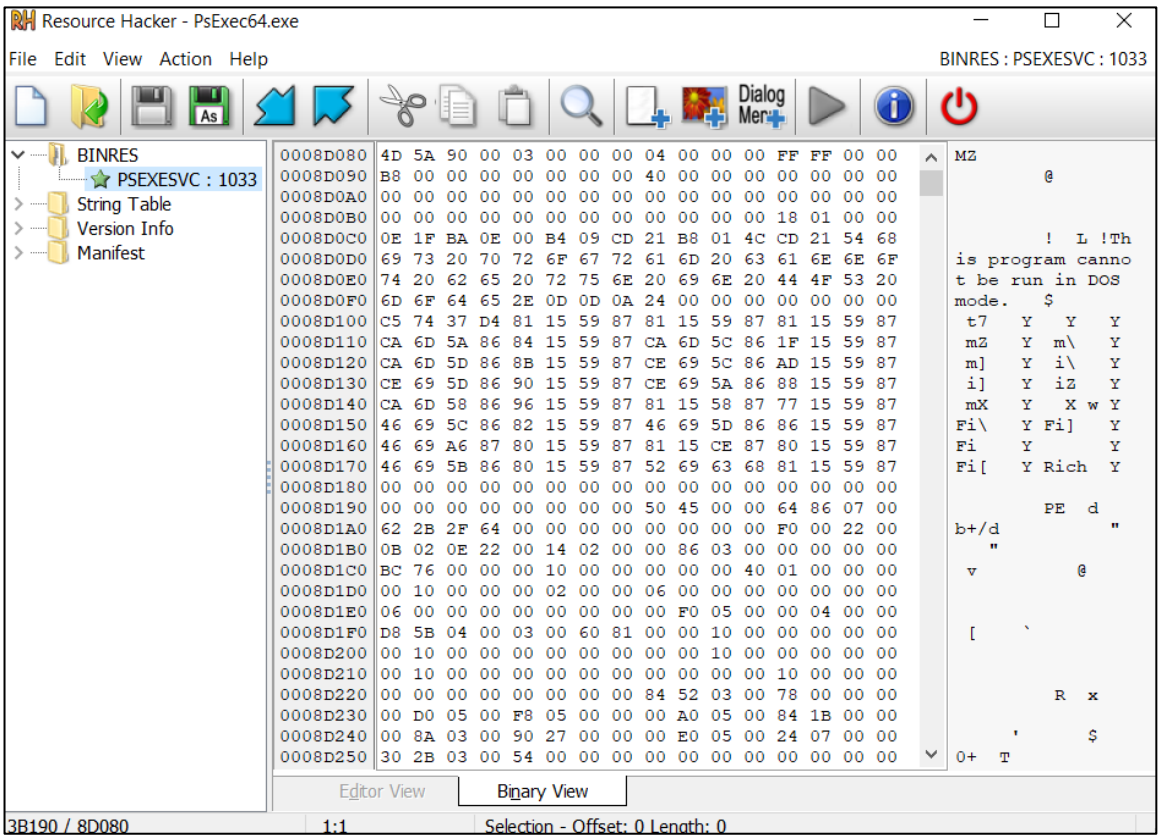
חילוץ ה-Service

הינה ספוילר: PsExec מתקין Service על מכונת היעד (ה-Server). ה-Service אחראי על התקשורת בין העמדות, הרצת הפקודות והעברת הפלט חזרה ל-Client. ה-Service נשמר כמשאב בתוך הקובץ. חילוץ המשאב מתבצע בצורה קלאסית, מתבצע חיפוש של שם המשאב (במקרה הזה BINRES) וחילוצו לזיכרון באמצעות הפונקציות: FindResource, LoadResource, SizeOfResource ו-LockResource.

ניתן לראות בתמונה שלאחר חילוץ המשאב לזיכרון מתבצעת העתקה שלו לשיתוף ה-ADMIN$ של ה-Server. שיתוף ה-ADMIN$ מוביל בסופו של דבר לנתיב: C:\Windows\System32. אם נסתכל ב-Resource Hacker נוכל לראות את התוכן של המשאב BINRES הוא קובץ הרצה בשם Psexesvc.exe שהוא כמובן ה-Service המדובר:



וב-Resource Hacker:



התקנת והפעלת ה-Service

לאחר שהקובץ של ה-Service הועתק לשרת, השלב הבא הוא להגדיר את הקובץ כ-Service ולהריץ אותו:

```
.text:00084466 loc_84466:                ; dwDesiredAccess
.text:00084466 push    0F003Fh
.text:0008446B push    0                 ; lpDatabaseName
.text:0008446D push    esi               ; lpMachineName
.text:0008446E call    ds:OpenSCManagerW
```

מתבצעת פתיחה של ה-SCManager על ה-Server. ה-SCManager אחראי על ניהול ה-Service-ים במערכת (למשל להתקין, להתחיל/להפסיק Service וכל היוצא בזה). לכן על מנת להתקין Service על עמדה מרוחקת נצטרך גישה ל-SCManager שלה.

הגישה מתבצעת באמצעות הפונקציה OpenSCManagerW (היוצרת את התקשורת באמצעות RPC, יוסבר בהמשך). המקבלת כפרמטר את שם העמדה עליה אנחנו רוצים לפתוח את ה-SCManager.

בהמשך נראה את ההגדרה עצמה של השירות:

```
.text:00084495 push    0
.text:00084497 push    0                ; lpServiceStartName
.text:00084499 push    0                ; lpDependencies
.text:0008449B push    0                ; lpdwTagId
.text:0008449D push    0                ; lpLoadOrderGroup
.text:0008449F lea     ecx, [ebp+BinaryPathName] ; %SystemRoot%\PSEXESVC.exe
.text:000844A5 push    ecx              ; lpBinaryPathName
.text:000844A6 push    0                ; dwErrorControl
.text:000844A8 push    SERVICE_DEMAND_START ; dwStartType
.text:000844AA push    eax              ; dwServiceType
.text:000844AB push    0F01FFh          ; dwDesiredAccess
.text:000844B0 push    [ebp+lpDisplayName] ; lpDisplayName
.text:000844B6 push    [ebp+lpServiceName] ; lpServiceName
.text:000844BC push    edi              ; hSCManager to our server
.text:000844BD call    ds:CreateServiceW
```

בתמונה ניתן לראות שהנתיב של השירות הוא PSEXESVC.exe אותו חלצנו והעברנו ל-Server קודם לכן. ה-StartType שהוגדר לשירות שלנו הוא: SERVICE_DEMEND_START, הכוונה היא שהשירות לא מתחיל באופן אוטומטי.

לאחר מכן ניתן לראות שהשירות אכן מופעל באמצעות הפונקציה StartServiceW. באופן כללי כל התהליך של התקנת ה-Service מתבצע ב-RPC (נתעמק בפן התקשורתי יותר בהרחבה בהמשך).

יצירת Named Pipes

Named Pipes - באופן פשטני, היא צורת תקשורת המאפשרת לתהליכים באותו מחשב, ולתהליכים במחשבים שונים לתקשר ביניהם. העברת המידע מתבצעת באמצעות קריאה או כתיבה ל-Named Pipe ספציפי (ממש כמו כתיבה או קריאה של קובץ). לדוגמה, נניח שתהליך A כתב מידע ל-Named Pipe בשם \\Computer1\pipe\MyPipe תהליך B יוכל לקרוא את המידע הכתוב ב-pipe ולהשתמש בו כרצונו. גם ב-PsExec התקשורת בין ה-Client ל-Server מבוססת על Named Pipes.

ב-windows גישה ל-pipe-ים מתבצעת בידיוק באותו האופן כמו לקבצים. פתיחה, קריאה וכתיבה ל-Named Pipe תהיה באמצעות הפונקציות: ReadFile, CreateFile ו-WriteFile עם הנתיב ל-Named Pipe עליו תתבצע הפעולה.

על מנת ליצור pipe ה-Client צריך להתחבר לשיתוף ה-IPC$ של השרת (בקצרה, שיתוף ה-IPC$, ראשי תיבות של Inter-Process Communication, הוא שיתוף מיוחד שמאפשר למחשבים לתקשר זה עם זה ברשת.

על מנת ליצור pipe-ים נצטרך גישה לשיתוף זה). ההתחברות לשיתוף מתבצעת באמצעות הפונקציה WNetConnection2W:

```
mov     r8, r15
lea     rdx, aSIpc        ; "\\\\%s\\IPC$"
lea     rcx, [rsp+6E8h+var_258]
call    sub_7FF70EFA4C10
mov     rdx, [rsp+6E8h+lpPassword] ; lpPassword
lea     rax, [rsp+6E8h+var_6C8]
mov     [rsp+6E8h+NetResource.lpLocalName], rax
lea     rcx, [rsp+6E8h+NetResource] ; lpNetResource
lea     rax, [rsp+6E8h+var_258]
mov     qword ptr [rsp+6E8h+NetResource.dwScope], rbx
xor     r9d, r9d          ; dwFlags
mov     [rsp+6E8h+NetResource.lpRemoteName], rax
mov     r8, rsi           ; lpUserName
mov     [rsp+6E8h+NetResource.dwDisplayType], ebx
mov     [rsp+6E8h+NetResource.lpComment], rbx
mov     [rsp+6E8h+NetResource.dwUsage], 3
mov     [rsp+6E8h+NetResource.lpProvider], rbx
call    WNetAddConnection2W
movzx   ecx, cs:byte_7FF70F02B0F8
test    eax, eax
mov     eax, 1
cmovz   ecx, eax
mov     cs:byte_7FF70F02B0F8, cl
```

לאחר שה-Client הצליח להתחבר לשיתוף ה-IPC$, השלב הבא הוא ליצור את ה-Named Pipes ולהשתמש בהם כדי להעביר מידע בין ה-Client ל-Server. ה-Named Pipe הראשון שנוצר הוא pipe בשם \\SERVER_IP\pipe\PSEXESVC:

```
.text:00007FF6EA2B87A0
.text:00007FF6EA2B87A0 loc_7FF6EA2B87A0:        ; hTemplateFile
.text:00007FF6EA2B87A0 mov     [rsp+0A470h+hTemplateFile], r12
.text:00007FF6EA2B87A5 lea     rcx, [rbp+0A370h+Name] ; lpFileName
.text:00007FF6EA2B87AC mov     [rsp+0A470h+dwFlagsAndAttributes], r12d ; dwFlagsAndAttributes
.text:00007FF6EA2B87B1 xor     r9d, r9d         ; lpSecurityAttributes
.text:00007FF6EA2B87B4 xor     r8d, r8d         ; dwShareMode
.text:00007FF6EA2B87B7 mov     [rsp+0A470h+dwCreationDisposition], 3 ; dwCreationDisposition
.text:00007FF6EA2B87BF mov     edx, 0C0000000h ; dwDesiredAccess
.text:00007FF6EA2B87C4 call    cs:CreateFileW
.text:00007FF6EA2B87CA mov     cs:hObject, rax
.text:00007FF6EA2B87D1 cmp     rax, 0FFFFFFFFFFFFFFFFh
.text:00007FF6EA2B87D5 jnz     short loc_7FF6EA2B87F5
07BC4 00007FF6EA2B87C4: sub_7FF6EA2B8410+3B4 (Synchronized with RIP)

00 00 00 00  00 00 00 00 00 00 00 00  ................
31 00 39 00  32 00 2E 00 31 00 36 00  \.\.1.9.2...1.6.
31 00 37 00  2E 00 31 00 33 00 34 00  8...1.7...1.3.4.
69 00 70 00  65 00 5C 00 50 00 53 00  \.p.i.p.e.\.P.S.
45 00 53 00  56 00 43 00 00 00 00 00  E.X.E.S.V.C.....
```

תפקידו של ה-pipe הנ"ל הוא לנהל את השירות שנוצר על ה-Server, לדאוג להרצת הפקודות וסביבת הריצה על ה-Server.

מלבד ה-pipe הנ"ל נוצרים עוד 3 pipe-ים נוספים עבור stdin,stdout ו-stderr:

```
.text:00007FF6EA2B8BF4 lea     rax, asc_7FF6EA31F3C8 ; "."
.text:00007FF6EA2B8BFB test    r15b, r15b
.text:00007FF6EA2B8BFE mov     rdi, rsi
.text:00007FF6EA2B8C01 mov     ebx, r12d
.text:00007FF6EA2B8C04 cmovnz  rdi, rax
.text:00007FF6EA2B8C08 call    cs:GetCurrentProcessId
.text:00007FF6EA2B8C0E mov     [rsp+0A470h+dwFlagsAndAttributes], eax
.text:00007FF6EA2B8C12 lea     r9, aPsexesvc    ; "PSEXESVC"
.text:00007FF6EA2B8C19 mov     r8, rdi
.text:00007FF6EA2B8C1C mov     qword ptr [rsp+0A470h+dwCreationDisposition], r13
.text:00007FF6EA2B8C21 lea     rdx, aSPipeSSDStdin ; "\\\\%s\\pipe\\%s-%s-%d-stdin"
.text:00007FF6EA2B8C28 lea     rcx, [rbp+0A370h+Name] ; Buffer
.text:00007FF6EA2B8C2F call    _strcat
.text:00007FF6EA2B8C34 mov     edx, 40000000h  ; dwDesiredAccess
.text:00007FF6EA2B8C39 lea     rcx, [rbp+0A370h+Name] ; lpFileName
.text:00007FF6EA2B8C40 call    create_the_pipe
.text:00007FF6EA2B8C45 mov     cs:qword_7FF6EA3361F0, rax
.text:00007FF6EA2B8C4C cmp     rax, 0FFFFFFFFFFFFFFFFh
.text:00007FF6EA2B8C50 jnz     short loc_7FF6EA2B8C5A
0 00007FF6EA2B8C40: sub_7FF6EA2B8410+830 (Synchronized with RIP)

00 39 00  32 00 2E 00 31 00 36 00  \.\.1.9.2...1.6.
00 37 00  2E 00 31 00 33 00 34 00  8...1.7...1.3.4.
00 70 00  65 00 5C 00 50 00 53 00  \.p.i.p.e.\.P.S.
00 53 00  56 00 43 00 2D 00 44 00  E.X.E.S.V.C.-.D.
00 54 00  4F 00 50 00 2D 00 55 00  E.S.K.T.O.P.-.U.
00 51 00  4D 00 4A 00 2D 00 31 00  F.H.F.Q.M.J.-.1.
00 38 00  2D 00 73 00 74 00 64 00  1.3.6.8.-.s.t.d.
00 00 00  00 00 00 00 00 00 00 00  i.n............
```

כפי שכבר הספקתם לנחש, ה-pipe-ים האלו יהיו אחראים על קבלת הקלט מה-Client והחזרת הפלט והשגיאות אל ה-console שנמצא גם הוא ב-Client. בתמונה מעלה נוכל לראות דוגמה ליצירת pipe עבור stdin. ל-pipe-ים אלו יש פורמט קבוע והוא:

```
\\<server_ip>\pipe\PSEXESVC - <client_hostname> - <pid> - stdin/out/err
```

התבנית הזו יכולה לעזור לנו בזיהוי הרצת PsExec ברשת. נדבר על כך בהמשך.

באופן דומה נוצרים שני ה-pipe-ים הנוספים, stdout ו-stderr. לאחר שה-pipe-ים מוכנים לשימוש, נוצרים שלושה thread-ים המריצים שלוש פונקציות שונות. כל פונקציה אחראית להתמודד עם stdin,stdout ו-stderr. (נראה דוגמה לפונקציה כזו בחלק הבא שידבר על ההצפנה):

```
.text:00007FF70F028DB8 lea     rcx, handle_stdout ; StartAddress
.text:00007FF70F028DBF mov     rdi, rax
.text:00007FF70F028DC2 call    _beginthread
```

```
.text:00007FF70F028E13 lea     rcx, handle_stdin ; StartAddress
.text:00007FF70F028E1A mov     rdi, rax
.text:00007FF70F028E1D call    _beginthread
```

```
.text:00007FF70F028D5D lea     rcx, handle_stderr ; StartAddress
.text:00007FF70F028D64 mov     rdi, rax
.text:00007FF70F028D67 call    _beginthread
```

לאחר שכל הפונקציות מוכנות לפעולה, ישתנה לנו התיאור של ה-console וכך נוכל לדעת שהתחברנו לשרת בהצלחה:

```
.text:00007FF7CDA58E7C lea     rcx, [rbp+0A370h+ConsoleTitle] ; lpConsoleTitle
.text:00007FF7CDA58E83 call    cs:SetConsoleTitleW
```

וכך זה יראה:

```
\\192.168.17.134: cmd
Copyright (C) 2001-2021 Mark Russinovich
Sysinternals - www.sysinternals.com
```

הצפנה

כל הפקודות שמועברות ב-session מוצפנות באמצעות AES. כשאנחנו מקלידים פקודה מתבצעת קריאה של תו תו של המידע שהכנסנו. עבור כל תו נשלח לפונקציה send_and_encrypt_data_to_server:

```
.text:00007FF70F0293F8 call    cs:ReadConsoleW
.text:00007FF70F0293FE test    eax, eax
.text:00007FF70F029400 jz      short loc_7FF70F02945B

.text:00007FF70F029402 mov     edx, cs:dword_7FF70F0AC26C
.text:00007FF70F029408 movzx   eax, [rsp+48h+Buffer]
.text:00007FF70F02940D mov     [r14+rdx*2], ax
.text:00007FF70F029412 inc     edx
.text:00007FF70F029414 mov     eax, edx
.text:00007FF70F029416 add     rax, rax
.text:00007FF70F029419 mov     cs:dword_7FF70F0AC26C, edx
.text:00007FF70F02941F cmp     rax, 20000h
.text:00007FF70F029425 jnb     short loc_7FF70F029482

0007FF70F029482
0007FF70F029482 loc_7FF70F029482:
0007FF70F029482 call    __report_rangecheckfailure
0007FF70F029482 sub_7FF70F029390 endp
0007FF70F029482

.text:00007FF70F029427 mov     r9d, [rsp+48h+NumberOfCharsRead]
.text:00007FF70F02942C lea     r8, [rsp+48h+Buffer]
.text:00007FF70F029431 mov     [rax+r14], bp
.text:00007FF70F029436 mov     rdx, rbx
.text:00007FF70F029439 mov     rcx, [rsi]
.text:00007FF70F02943C call    send_and_encrypt_data_to_server
```

אם נכנס לתוך הפונקציה send_and_encrypt_data_to_server נוכל לראות את ההצפנה שמתבצעת באמצעות CryptEncrypt ולאחר מכן כתיבה של המידע המוצפן ל-pipe ה-stdin:

```
.text:00007FF70F026502 mov     [rsp+68h+lpOverlapped], rsi ; pbData (current char)
.text:00007FF70F026507 call    cs:CryptEncrypt
.text:00007FF70F02650D test    eax, eax
.text:00007FF70F02650F jz      short loc_7FF70F02655C

.text:00007FF70F026511 lea     r9, [rsp+68h+NumberOfBytesWritten] ; lpNumberOfBytesWritten
.text:00007FF70F026516 mov     [rsp+68h+lpOverlapped], rbx ; lpOverlapped
.text:00007FF70F02651B mov     r8d, 4          ; nNumberOfBytesToWrite
.text:00007FF70F026521 lea     rdx, [rsp+68h+Buffer] ; lpBuffer
.text:00007FF70F026526 mov     rcx, rdi        ; hFile (stdin pipe handle)
.text:00007FF70F026529 call    cs:WriteFile
.text:00007FF70F02652F test    eax, eax
.text:00007FF70F026531 jz      short loc_7FF70F02655C
```

באופן זהה קורה גם התהליך ההפוך על מנת לקבל את הפלט מה-Server נקרא את המידע מ-pipe ה-stdout ופענוחו באמצעות CrypDecrypt. בתמונה מטה ניתן לראות את הפלט המפוענח לאחר הרצת הפקודה: "whoami":

```
.text:00007FF70F0263B2 call    cs:ReadFile
.text:00007FF70F0263B8 test    eax, eax
.text:00007FF70F0263BA jnz     short loc_7FF70F0263DC

.text:00007FF70F0263DC
.text:00007FF70F0263DC loc_7FF70F0263DC:        ; hKey
.text:00007FF70F0263DC mov     rcx, [r14]
.text:00007FF70F0263DF xor     r9d, r9d         ; dwFlags
.text:00007FF70F0263E2 mov     [rsp+48h+pdwDataLen], rsi ; pdwDataLen
.text:00007FF70F0263E7 xor     edx, edx         ; hHash
.text:00007FF70F0263E9 mov     [rsp+48h+lpOverlapped], rbx ; pbData
.text:00007FF70F0263EE lea     r8d, [r9+1]      ; Final
.text:00007FF70F0263F2 call    cs:CryptDecrypt
.text:00007FF70F0263F8 mov     edi, eax
.text:00007FF70F0263FA test    eax, eax
.text:00007FF70F0263FC jz      short loc_7FF70F02640E

.text:00007FF70F0263FE mov     r8d, [rsi]       ; Size
.text:00007FF70F02640E

000057F2 00007FF70F0263F2: sub_7FF70F026310+E2 (Synchronized with RIP)

00 00 00 00 00 80  AB AB AB AB AB AB AB AB  .......€«««««««««
AB AB AB AB AB AB  EE FE EE FE EE FE EE FE  «««««««««ª.ª.ª.ª.
00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
EE FE EE FE EE FE  7D E6 53 B8 03 2B 00 3F  ª{.ª.ª.ª.ªS¸.+.?
65 64 67 65 77 69  6E 31 30 5C 69 65 75 73  msedgewin10\ieus
39 3D 02 1B D9 46  25 65 F2 F2 61 B8 E0 4B  er9=...F%eªªa¸àK
```

זווית תקשורתית

אז יכול להיות שבמהלך המאמר זרקנו מילה פה ושם על איך הדברים עובדים מבחינה תקשורתית. כעת נריץ את PsExec שוב והפעם נסניף את התקשורת על ה-Server ונסתכל על התקשורת בין שתי העמדות. לצורך ההקלטה השתמשתי ב-Wireshark. אתמקד רק בתקשורת שרלוונטית ל-PsExec ולא בכל ה-session.

תחילת התקשורת:

192.168.17.1	192.168.17.134	SMB2	176	Tree Connect Request Tree: \\192.168.17.134\ADMIN$
192.168.17.134	192.168.17.1	SMB2	138	Tree Connect Response
192.168.17.1	192.168.17.134	SMB2	234	Create Request File:
192.168.17.134	192.168.17.1	SMB2	298	Create Response File:
192.168.17.1	192.168.17.134	SMB2	146	Close Request File:
192.168.17.134	192.168.17.1	SMB2	182	Close Response
192.168.17.1	192.168.17.134	SMB2	382	Create Request File: PSEXESVC.exe
192.168.17.134	192.168.17.1	SMB2	410	Create Response File: PSEXESVC.exe

אפשר לראות את הפניה לשיתוף ה-ADMIN$ וכמו כן את ההעתקה של ה-Service שלנו PSEXECSVC.exe. העברת ה-Service מתבצע על גבי פרוטוקול העברת הקבצים המוכר והידוע SMB.

התקנת ה-Service

התקנת ה-Service שראינו קודם לכן, מתבצעת בעזרת הפונקציות: OpenSCManagerW, StartServiceW, CreateServiceW וכדומה. כשקראנו לפונקציה OpenSCManagerW העברנו לה כפרמטר את שם המחשב עליו אנחנו רוצים לרוץ ובכך קיבלנו גישה ל-Service Control Manager שעל העמדה המרוחקת.

192.168.17.1	192.168.17.134	DCERPC	214	Bind: call_id: 2, Fragment: Single, 3 context items
192.168.17.134	192.168.17.1	DCERPC	162	Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5
192.168.17.1	192.168.17.134	EPM	222	Map request, SVCCTL, 32bit NDR
192.168.17.134	192.168.17.1	EPM	226	Map response, SVCCTL, 32bit NDR
192.168.17.1	192.168.17.134	TCP	66	65333 → 49673 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
192.168.17.134	192.168.17.1	TCP	66	49673 → 65333 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=
192.168.17.1	192.168.17.134	TCP	60	65333 → 49673 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
192.168.17.1	192.168.17.134	DCERPC	218	Bind: call_id: 2, Fragment: Single, 2 context items
192.168.17.134	192.168.17.1	DCERPC	344	Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5
192.168.17.1	192.168.17.134	DCERPC	568	AUTH3: call_id: 2, Fragment: Single, NTLMSSP_AUTH,
192.168.17.1	192.168.17.134	SVCCTL	166	Unknown operation 64 request
192.168.17.134	192.168.17.1	TCP	54	49673 → 65333 [ACK] Seq=291 Ack=791 Win=2101760 Len
192.168.17.134	192.168.17.1	DCERPC	86	Fault: call_id: 2, Fragment: Single, Ctx: 0, status

כל תהליך התקנת ה-Service והתחלתו, מתבצע באמצעות RPC. בגדול ומבלי להכנס לפרטים עמוקים, RPC מאפשר לתקשר ולבצע פעולות על מחשב אחר ברשת. בתמונה ניתן לראות שהשירות המבוקש הוא SVCCTL.

SVCCTL הוא interface של RPC, הוא מאפשר ניהול מרוחק של Service-ים. ומעתה והלאה העמדות יתקשרו ביניהם ב-interface זה על מנת לבצע את כל מה שנוגע להתקנת ה-Service. נוכל לראות זאת בפירוט בתמונה הבאה:

Protocol	Length	Info
SVCCTL	310	CreateServiceW request
SVCCTL	134	CreateServiceW response
SVCCTL	134	CloseServiceHandle request
SVCCTL	134	CloseServiceHandle response
SVCCTL	166	OpenServiceW request
SVCCTL	134	OpenServiceW response
SVCCTL	134	StartServiceW request

לאחר שה-Service מופעל נוכל לראות את היצירה של ה-pipe-ים:

SMB2	170	Ioctl Response FSCTL_PIPE_WAIT
SMB2	248	Create Request File: PSEXESVC-DESKTOP-UFHFQMJ-9300-stdin
SMB2	210	Create Response File: PSEXESVC-DESKTOP-UFHFQMJ-9300-stdin
SMB2	162	GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO File: PSEXESVC-DESKTOP-UFHFQMJ-9300-stdin
SMB2	154	GetInfo Response
SMB2	264	Ioctl Request FSCTL_PIPE_WAIT Pipe: PSEXESVC-DESKTOP-UFHFQMJ-9300-stdout
SMB2	170	Ioctl Response FSCTL_PIPE_WAIT
SMB2	250	Create Request File: PSEXESVC-DESKTOP-UFHFQMJ-9300-stdout
SMB2	210	Create Response File: PSEXESVC-DESKTOP-UFHFQMJ-9300-stdout
SMB2	162	GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO File: PSEXESVC-DESKTOP-UFHFQMJ-9300-stdout
SMB2	154	GetInfo Response
SMB2	264	Ioctl Request FSCTL_PIPE_WAIT Pipe: PSEXESVC-DESKTOP-UFHFQMJ-9300-stderr

שימו לב ששמות ה-pipe-ים בנויים לפי התבנית שהוצגה קודם לכן במאמר.

לסיום נראה מחיקה של ה-Service, שוב בעזרת RPC:

```
SVCCTL   166 OpenServiceW request
SVCCTL   134 OpenServiceW response
SVCCTL   134 DeleteService request
SVCCTL   118 DeleteService response
SVCCTL   134 CloseServiceHandle request
SVCCTL   134 CloseServiceHandle response
```

זיהוי פעילות של PsExec ברשת

ניטור יצירת ה-Service של PsExec

כאמור, PsExec יוצר שירות. ניתן לנטר זאת באמצעות ה-Eventlog. נוכל לחפש בלוג ה-System את Event ID מספר 7045 המתעד יצירה של Service. נחפש לוג שמכיל את השם PSEXECSVC ונוכל לקבל אינדיקציה להרצה של PsExec. (השם PSEXECSVC הוא דיפולטי, ניתן לשנותו באמצעות שימוש בפרמטר r- ולאחר מכן לציין את ה-Service Name בו אנו רוצים להשתמש):

Level	Date and Time	Source	Event ID	Task Category
Information	9/23/2024 11:36:22 AM	Service Control Manager	7045	None

Event 7045, Service Control Manager

General Details

A service was installed in the system.
Service Name: PSEXESVC
Service File Name: %SystemRoot%\PSEXESVC.exe
Service Type: user mode service
Service Start Type: demand start
Service Account: LocalSystem

ניטור יצירת ערך ה-Registry

באמצעות sysmon נוכל להשתמש ב-Event IDs מספר 12 ו-13 המתעדים יצירה של ערך Registry חדש ושינוי ערך קיים בהתאמה ועל כן נוכל לנטר את היצירה של ערך ה-EulaAccepted:

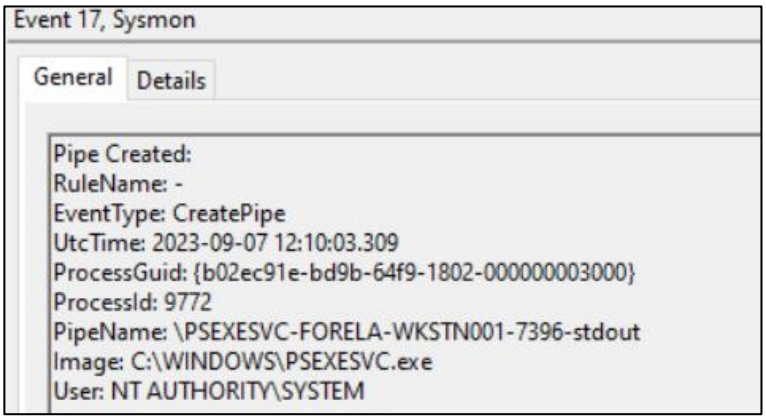
Event 13, Sysmon

General Details

Registry value set:
RuleName: Alert,Sysinternals Tool Used
EventType: SetValue
UtcTime: 2024-09-23 18:36:22.001
ProcessGuid: {6c17bd0f-b525-66f1-970c-000000003100}
ProcessId: 4772
Image: C:\Users\omer\Desktop\DW - psexec\psexec\PsExec64.exe
TargetObject: HKU\S-1-5-21-1430080282-325649845-1509870264-1001\SOFTWARE\Sysinternals\PsExec\EulaAccepted
Details: DWORD (0x00000001)

ניטור יצירת ה-pipe-ים

אם כבר ב-sysmon עסקינן, בעזרת ה-Event IDs מספר 17 ו-18 המתעדים יצירה של pipe וחיבור ל-pipe, נוכל לנטר יצירה וגישה ל-pipe-ים על פי התבנית שהצגנו במאמר:



[תמונה מתוך הבלוג של HackTheBox]

סיכום

PsExec הוא כלי עוצמת המאפשר ניהול מערכות מרחוק. על אף יעודו הלגיטימי, ניתן לראות בו שימוש נרחב גם לתקיפות סייבר, החל מתוקפים מתחילים ועד קבוצות תקיפה ותיקות ומיומנות. לכן, עלינו כחוקרים להיות מודעים לשימוש ב-PsExec, להכיר ולהתחקות אחר העקבות שהוא מותיר, על מנת לזהות שימוש זדוני בו ביעילות.

מקורות מידע

את הגסה הספציפית של ה-PsExec הנחקר (לא אמורים להיות שינויים מהותיים בין הגרסאות בהיבט אופן הפעולה) ניתן למצוא כאן.

על PsExec ב-MSDN:

https://learn.microsoft.com/en-us/sysinternals/downloads/psexec

זיהוי PsExec:

https://www.hackthebox.com/blog/how-to-detect-psexec-and-lateral-movements

PsExec Reverse Engineering:

https://cybergeeks.tech/reverse-engineering-psexec-for-fun-and-knowledge/