HELLO

***Paper Title: In-Depth Malware Analysis of TLauncher.exe***

*Scan and Analysis performed by EDOT*

# Index

[Entrypoints]

vaddr=0x00402ce1 paddr=0x000020e1 haddr=0x00000100 type=program

1 entrypoints

```
[0x00402ce1]> pdf @ main
          ;-- section..text:
          ; CALL XREF from entry0 @ 0x402c89(x)
┌ 1012: int main (int argc);
|          ; arg int32_t argc @ ebp+0x10
|          ; var int32_t var_4h @ ebp-0x4
|          ; var int32_t var_108h @ ebp-0x108
|          ; var int32_t var_209h @ ebp-0x209
|          ; var int32_t var_20ah @ ebp-0x20a
|          ; var int32_t var_20bh @ ebp-0x20b
|          ; var int32_t var_20ch @ ebp-0x20c
|          ; var int32_t var_310h @ ebp-0x310
|          ; var int32_t var_414h @ ebp-0x414
|          ; var int32_t var_518h @ ebp-0x518
|          ; var int32_t var_61ch @ ebp-0x61c
|          ; var int32_t var_a5ch @ ebp-0xa5c
|          ; var int32_t var_a60h @ ebp-0xa60
|          ; var int32_t var_b64h @ ebp-0xb64
|          ; var int32_t var_1364h @ ebp-0x1364
```

```
|       ; var int32_t var_1b64h @ ebp-0x1b64
|       ; var int32_t var_1b6ch @ ebp-0x1b6c
|       ; var int32_t var_1b70h @ ebp-0x1b70
|       ; var int32_t var_1b74h @ ebp-0x1b74
|       ; var int32_t var_1b78h @ ebp-0x1b78
|       ; var int32_t var_1b7ch @ ebp-0x1b7c
|       ; var int32_t var_1b80h @ ebp-0x1b80
|       ; var int32_t var_1b84h @ ebp-0x1b84
|       0x00401000   55           push ebp              ; [00] -r-x section size 24576 named .text
|       0x00401001   8bec          mov ebp, esp
|       0x00401003   b8841b0000    mov eax, 0x1b84
|       0x00401008   e8a3160000    call 0x4026b0
|       0x0040100d   a120a04000    mov eax, dword [0x40a020]   ; [0x40a020:4]=0xbb40e64e
|       0x00401012   33c5          xor eax, ebp
|       0x00401014   8945fc        mov dword [var_4h], eax
|       0x00401017   8b4510        mov eax, dword [argc]
|       0x0040101a   53            push ebx
|       0x0040101b   56            push esi
|       0x0040101c   8b3524704000   mov esi, dword
[sym.imp.KERNEL32.dll_GetModuleHandleA] ; [0x407024:4]=0x9944
reloc.KERNEL32.dll_GetModuleHandleA ; "D\x99"
|       0x00401022   57            push edi
|       0x00401023   89857ce4ffff   mov dword [var_1b84h], eax
|       0x00401029   33c0          xor eax, eax
```

```
|        0x0040102b     bf4c724000     mov edi, str.kernel32.dll   ; 0x40724c ; "kernel32.dll"
|        0x00401030     57             push edi                    ; 0x40724c ; "kernel32.dll"
|        0x00401031     898580e4ffff   mov dword [var_1b80h], eax
|        0x00401037     898590e4ffff   mov dword [var_1b70h], eax
|        0x0040103d     898584e4ffff   mov dword [var_1b7ch], eax
|        0x00401043     898588e4ffff   mov dword [var_1b78h], eax
|        0x00401049     89858ce4ffff   mov dword [var_1b74h], eax
|        0x0040104f     ffd6           call esi
|        0x00401051     bb04010000     mov ebx, 0x104              ; 260
|        0x00401056     85c0           test eax, eax
|      ┌─< 0x00401058     742a           je 0x401084
|      |  0x0040105a     6830724000     push str.SetDefaultDllDirectories ; 0x407230 ;
"SetDefaultDllDirectories"
|      |  0x0040105f     50             push eax
|      |  0x00401060     ff1520704000   call dword [sym.imp.KERNEL32.dll_GetProcAddress] ;
0x407020 ; "2\x99" ; FARPROC GetProcAddress(HMODULE hModule, LPCSTR lpProcName)
|      |  0x00401066     85c0           test eax, eax
|     ┌──< 0x00401068     741a           je 0x401084
|     ||  0x0040106a     6800080000     push 0x800                  ; 2048
|     ||  0x0040106f     ffd0           call eax
|     ||  0x00401071     85c0           test eax, eax
|    ┌───< 0x00401073     740f           je 0x401084
|    |||  0x00401075     c78580e4ff..   mov dword [var_1b80h], 1
|   ┌────< 0x0040107f     e968010000     jmp 0x4011ec
```

```
|    |└└└> 0x00401084     57           push edi
|    |     0x00401085     ffd6         call esi
|    |     0x00401087     85c0         test eax, eax
|    | ┌─< 0x00401089     7417         je 0x4010a2
|    | |  0x0040108b     681c724000     push str.SetDllDirectoryA   ; 0x40721c ;
"SetDllDirectoryA"
|    | |  0x00401090     50           push eax
|    | |  0x00401091     ff1520704000   call dword [sym.imp.KERNEL32.dll_GetProcAddress] ;
0x407020 ; "2\x99" ; FARPROC GetProcAddress(HMODULE hModule, LPCSTR lpProcName)
|    | |  0x00401097     85c0         test eax, eax
|    |┌──< 0x00401099     7407         je 0x4010a2
|    | ||  0x0040109b     681a724000     push 0x40721a            ; '\x1ar@'
|    | ||  0x004010a0     ffd0         call eax
|    | └└> 0x004010a2     53           push ebx
|    |     0x004010a3     8d85f8feffff   lea eax, [var_108h]
|    |     0x004010a9     50           push eax
|    |     0x004010aa     ff151c704000   call dword [sym.imp.KERNEL32.dll_GetSystemDirectoryA]
; 0x40701c ; UINT GetSystemDirectoryA(LPSTR lpBuffer, UINT uSize)
|    |     0x004010b0     8d85f8feffff   lea eax, [var_108h]
|    |     0x004010b6     50           push eax
|    |     0x004010b7     ff1518704000   call dword [sym.imp.KERNEL32.dll_lstrlenA] ; 0x407018 ;
int lstrlenA(LPCSTR lpString)
|    |     0x004010bd     8b3514704000   mov esi, dword [sym.imp.KERNEL32.dll_lstrcatA] ;
[0x407014:4]=0x9904 reloc.KERNEL32.dll_lstrcatA
```

```
|   |    0x004010c3     8d8df8feffff   lea ecx, [var_108h]
|   |    0x004010c9     807c08ff5c     cmp byte [eax + ecx - 1], 0x5c ; '\\'
|   |  ┌─< 0x004010ce     740a         je 0x4010da
|   |  |  0x004010d0     6818724000     push 0x407218           ; '\x18r@' ; "\"
|   |  |  0x004010d5     8bc1          mov eax, ecx
|   |  |  0x004010d7     50            push eax
|   |  |  0x004010d8     ffd6          call esi
|   |  └─> 0x004010da     53            push ebx
|   |    0x004010db     8d85f0fcffff   lea eax, [var_310h]
|   |    0x004010e1     6a00          push 0
|   |    0x004010e3     50            push eax
|   |    0x004010e4     e837150000     call 0x402620
|   |    0x004010e9     83c40c        add esp, 0xc
|   |    0x004010ec     8d85f8feffff   lea eax, [var_108h]
|   |    0x004010f2     50            push eax
|   |    0x004010f3     8d85f0fcffff   lea eax, [var_310h]
|   |    0x004010f9     50            push eax
|   |    0x004010fa     ff1510704000   call dword [sym.imp.KERNEL32.dll_lstrcpyA] ; 0x407010 ;
LPSTR lstrcpyA(LPSTR lpString1, LPCSTR lpString2)
|   |    0x00401100     680c724000     push str.ntmarta.dll       ; 0x40720c ; "ntmarta.dll"
|   |    0x00401105     8d85f0fcffff   lea eax, [var_310h]
|   |    0x0040110b     50            push eax
|   |    0x0040110c     ffd6          call esi
```

```
|   |   0x0040110e    8b3d0c704000   mov edi, dword [sym.imp.KERNEL32.dll_LoadLibraryA] ;
[0x40700c:4]=0x98e8 reloc.KERNEL32.dll_LoadLibraryA
|   |   0x00401114    8d85f0fcffff   lea eax, [var_310h]
|   |   0x0040111a    50             push eax
|   |   0x0040111b    ffd7           call edi
|   |   0x0040111d    53             push ebx
|   |   0x0040111e    898590e4ffff   mov dword [var_1b70h], eax
|   |   0x00401124    8d85ecfbffff   lea eax, [var_414h]
|   |   0x0040112a    6a00           push 0
|   |   0x0040112c    50             push eax
|   |   0x0040112d    e8ee140000     call 0x402620
|   |   0x00401132    83c40c         add esp, 0xc
|   |   0x00401135    8d85f8feffff   lea eax, [var_108h]
|   |   0x0040113b    50             push eax
|   |   0x0040113c    8d85ecfbffff   lea eax, [var_414h]
|   |   0x00401142    50             push eax
|   |   0x00401143    ff1510704000   call dword [sym.imp.KERNEL32.dll_lstrcpyA] ; 0x407010 ;
LPSTR lstrcpyA(LPSTR lpString1, LPCSTR lpString2)
|   |   0x00401149    6800724000     push str.PROPSYS.dll       ; 0x407200 ; "PROPSYS.dll"
|   |   0x0040114e    8d85ecfbffff   lea eax, [var_414h]
|   |   0x00401154    50             push eax
|   |   0x00401155    ffd6           call esi
|   |   0x00401157    8d85ecfbffff   lea eax, [var_414h]
|   |   0x0040115d    50             push eax
```

```
| | 0x0040115e ffd7 call edi
| | 0x00401160 53 push ebx
| | 0x00401161 898584e4ffff mov dword [var_1b7ch], eax
| | 0x00401167 8d85e8faffff lea eax, [var_518h]
| | 0x0040116d 6a00 push 0
| | 0x0040116f 50 push eax
| | 0x00401170 e8ab140000 call 0x402620
| | 0x00401175 83c40c add esp, 0xc
| | 0x00401178 8d85f8feffff lea eax, [var_108h]
| | 0x0040117e 50 push eax
| | 0x0040117f 8d85e8faffff lea eax, [var_518h]
| | 0x00401185 50 push eax
| | 0x00401186 ff1510704000 call dword [sym.imp.KERNEL32.dll_lstrcpyA] ; 0x407010 ;
LPSTR lstrcpyA(LPSTR lpString1, LPCSTR lpString2)
| | 0x0040118c 68f4714000 push str.Secur32.dll ; 0x4071f4 ; "Secur32.dll"
| | 0x00401191 8d85e8faffff lea eax, [var_518h]
| | 0x00401197 50 push eax
| | 0x00401198 ffd6 call esi
| | 0x0040119a 8d85e8faffff lea eax, [var_518h]
| | 0x004011a0 50 push eax
| | 0x004011a1 ffd7 call edi
| | 0x004011a3 53 push ebx
| | 0x004011a4 898588e4ffff mov dword [var_1b78h], eax
| | 0x004011aa 8d85e4f9ffff lea eax, [var_61ch]
```

```
| |  0x004011b0  6a00  push 0
| |  0x004011b2  50  push eax
| |  0x004011b3  e868140000  call 0x402620
| |  0x004011b8  83c40c  add esp, 0xc
| |  0x004011bb  8d85f8feffff  lea eax, [var_108h]
| |  0x004011c1  50  push eax
| |  0x004011c2  8d85e4f9ffff  lea eax, [var_61ch]
| |  0x004011c8  50  push eax
| |  0x004011c9  ff1510704000  call dword [sym.imp.KERNEL32.dll_lstrcpyA] ; 0x407010 ;
LPSTR lstrcpyA(LPSTR lpString1, LPCSTR lpString2)
| |  0x004011cf  68c8714000  push str.api_ms_win_downlevel_advapi32_l2_1_0.dll ;
0x4071c8 ; "api-ms-win-downlevel-advapi32-l2-1-0.dll"
| |  0x004011d4  8d85e4f9ffff  lea eax, [var_61ch]
| |  0x004011da  50  push eax
| |  0x004011db  ffd6  call esi
| |  0x004011dd  8d85e4f9ffff  lea eax, [var_61ch]
| |  0x004011e3  50  push eax
| |  0x004011e4  ffd7  call edi
| |  0x004011e6  89858ce4ffff  mov dword [var_1b74h], eax
| |  ; CODE XREF from main @ 0x40107f(x)
| └──> 0x004011ec  33ff  xor edi, edi
|    0x004011ee  68027f0000  push 0x7f02
|    0x004011f3  57  push edi
|    0x004011f4  893d80ab4000  mov dword [0x40ab80], edi  ; [0x40ab80:4]=0
```

```
|       0x004011fa      ff156c714000   call dword [sym.imp.USER32.dll_LoadCursorA] ; 0x40716c
; HCURSOR LoadCursorA(HINSTANCE hInstance, LPCSTR lpCursorName)
|       0x00401200      50             push eax
|       0x00401201      ff1570714000   call dword [sym.imp.USER32.dll_SetCursor] ; 0x407170 ;
HCURSOR SetCursor(HCURSOR hCursor)
|       0x00401207      8d8d94e4ffff   lea ecx, [var_1b6ch]
|       0x0040120d      e8e2010000     call 0x4013f4
|       0x00401212      be00080000     mov esi, 0x800          ; 2048
|       0x00401217      56             push esi
|       0x00401218      8d859cecffff   lea eax, [var_1364h]
|       0x0040121e      57             push edi
|       0x0040121f      50             push eax
|       0x00401220      e8fb130000     call 0x402620
|       0x00401225      8b3d18704000   mov edi, dword [sym.imp.KERNEL32.dll_lstrlenA] ;
[0x407018:4]=0x9910 reloc.KERNEL32.dll_lstrlenA
|       0x0040122b      83c40c         add esp, 0xc
|       0x0040122e      ffb57ce4ffff   push dword [var_1b84h]
|       0x00401234      ffd7           call edi
|       0x00401236      3bc6           cmp eax, esi
|     ┌─< 0x00401238      7d13           jge 0x40124d
|     |  0x0040123a      ffb57ce4ffff   push dword [var_1b84h]
|     |  0x00401240      8d859cecffff   lea eax, [var_1364h]
|     |  0x00401246      50             push eax
```

```
|     |  0x00401247      ff1510704000   call dword [sym.imp.KERNEL32.dll_lstrcpyA] ; 0x407010 ;
LPSTR lstrcpyA(LPSTR lpString1, LPCSTR lpString2)
|     └─> 0x0040124d      33f6          xor esi, esi
|       0x0040124f      53            push ebx
|       0x00401250      8d859cf4ffff   lea eax, [var_b64h]
|       0x00401256      56            push esi
|       0x00401257      50            push eax
|       0x00401258      89b5a0f5ffff   mov dword [var_a60h], esi
|       0x0040125e      e8bd130000     call 0x402620
|       0x00401263      83c40c         add esp, 0xc
|       0x00401266      33db          xor ebx, ebx
|       0x00401268      393588ab4000   cmp dword [0x40ab88], esi   ; [0x40ab88:4]=0
|     ┌─< 0x0040126e      0f8eba000000   jle 0x40132e
|    ┌──> 0x00401274      a18cab4000     mov eax, dword [0x40ab8c]   ; [0x40ab8c:4]=0
|    ╎│  0x00401279      ff3498         push dword [eax + ebx*4]
|    ╎│  0x0040127c      8d85f4fdffff   lea eax, [var_20ch]
|    ╎│  0x00401282      50            push eax
|    ╎│  0x00401283      ff1510704000   call dword [sym.imp.KERNEL32.dll_lstrcpyA] ; 0x407010
; LPSTR lstrcpyA(LPSTR lpString1, LPCSTR lpString2)
|    ╎│  0x00401289      80bdf4fdff..   cmp byte [var_20ch], 0x2f   ; '/'
|   ┌───< 0x00401290      7566          jne 0x4012f8
|   │╎│  0x00401292      8a85f5fdffff   mov al, byte [var_20bh]
|   │╎│  0x00401298      3c54          cmp al, 0x54              ; 'T' ; 84
|  ┌────< 0x0040129a      7426          je 0x4012c2
```

```
|   ||¦|  0x0040129c    3c57        cmp al, 0x57           ; 'W' ; 87
|  ┌────< 0x0040129e    7408        je 0x4012a8
|  |||¦|  0x004012a0    3c74        cmp al, 0x74           ; 't' ; 116
| ┌─────< 0x004012a2    741e        je 0x4012c2
| ||||¦|  0x004012a4    3c77        cmp al, 0x77           ; 'w' ; 119
|┌──────< 0x004012a6    7550        jne 0x4012f8
|||└────> 0x004012a8    8d85f4fdffff  lea eax, [var_20ch]
||| ||¦|  0x004012ae    50          push eax
||| ||¦|  0x004012af    ffd7        call edi
||| ||¦|  0x004012b1    83f802      cmp eax, 2             ; 2
|||┌────< 0x004012b4    7542        jne 0x4012f8
||||||¦|  0x004012b6    c785a0f5ff..  mov dword [var_a60h], 1
|───────< 0x004012c0    eb36        jmp 0x4012f8
||└─└───> 0x004012c2    8d85f4fdffff  lea eax, [var_20ch]
||||¦|    0x004012c8    50          push eax
||||¦|    0x004012c9    ffd7        call edi
||||¦|    0x004012cb    83f803      cmp eax, 3             ; 3
||||┌───< 0x004012ce    7e28        jle 0x4012f8
|| |||¦|  0x004012d0    80bdf6fdff..  cmp byte [var_20ah], 0x3a   ; ':'
||┌─────< 0x004012d7    751f        jne 0x4012f8
||||||¦|  0x004012d9    83f803      cmp eax, 3             ; 3
|───────< 0x004012dc    7e1a        jle 0x4012f8
||||||¦|  0x004012de    83c0fd      add eax, 0xfffffffd
```

```
| |||||¦|  0x004012e1     50           push eax
| |||||¦|  0x004012e2     8d85f7fdffff   lea eax, [var_209h]
| |||||¦|  0x004012e8     50           push eax
| |||||¦|  0x004012e9     8d859cf4ffff   lea eax, [var_b64h]
| |||||¦|  0x004012ef     50           push eax
| |||||¦|  0x004012f0     e8eb130000     call 0x4026e0
| |||||¦|  0x004012f5     83c40c        add esp, 0xc
| |||||¦|  ; CODE XREF from main @ 0x4012c0(x)
| └└└└└──> 0x004012f8     6aff         push 0xffffffffffffffff
|      ¦|  0x004012fa     68c0714000     push str._DBG            ; 0x4071c0 ; "/~DBG"
|      ¦|  0x004012ff     6aff         push 0xffffffffffffffff
|      ¦|  0x00401301     8d85f4fdffff   lea eax, [var_20ch]
|      ¦|  0x00401307     50           push eax
|      ¦|  0x00401308     6a01         push 1                   ; 1
|      ¦|  0x0040130a     6a7f         push 0x7f                ; '\x7f' ; 127
|      ¦|  0x0040130c     ff1554704000   call dword [sym.imp.KERNEL32.dll_CompareStringA] ;
0x407054 ; int CompareStringA(LCID Locale, DWORD dwCmpFlags, PCNZCH lpString1, int cchCount1,
PCNZCH lpString2, int cchCount2)
|      ¦|  0x00401312     83f802        cmp eax, 2               ; 2
|     ┌──< 0x00401315     750a         jne 0x401321
|     |¦|  0x00401317     c70580ab40..   mov dword [0x40ab80], 1     ; [0x40ab80:4]=0
|     └──> 0x00401321     43           inc ebx
|      ¦|  0x00401322     3b1d88ab4000   cmp ebx, dword [0x40ab88]   ; [0x40ab88:4]=0
```

```
|     └──< 0x00401328     0f8c46ffffff   jl 0x401274
|     ┌└─> 0x0040132e     8d8d94e4ffff   lea ecx, [var_1b6ch]
|     ¦   0x00401334     e83e0f0000     call 0x402277
|     ¦   0x00401339     8bf8          mov edi, eax
|     ¦   0x0040133b     8d47ce        lea eax, [edi - 0x32]
|     ¦   0x0040133e     83f831        cmp eax, 0x31            ; '1' ; 49
|     ¦┌─< 0x00401341     771a          ja 0x40135d
|     ¦|  0x00401343     83bda4f5ff..   cmp dword [var_a5ch], 0
|    ┌───< 0x0040134a     7511          jne 0x40135d
|    |¦|  0x0040134c     68fa000000     push 0xfa                ; 250
|    |¦|  0x00401351     46            inc esi
|    |¦|  0x00401352     ff1594704000   call dword [sym.imp.KERNEL32.dll_Sleep] ; 0x407094 ;
VOID Sleep(DWORD dwMilliseconds)
|    |¦|  0x00401358     83fe05        cmp esi, 5               ; 5
|    |└──< 0x0040135b     72d1          jb 0x40132e
|    └└─> 0x0040135d     83bd80e4ff..   cmp dword [var_1b80h], 0
|     ┌─< 0x00401364     7549          jne 0x4013af
|     |  0x00401366     83bd90e4ff..   cmp dword [var_1b70h], 0
|     |  0x0040136d     8b359c704000   mov esi, dword [sym.imp.KERNEL32.dll_FreeLibrary] ;
[0x40709c:4]=0x98c0 reloc.KERNEL32.dll_FreeLibrary
|     ┌──< 0x00401373     7408          je 0x40137d
|     ||  0x00401375     ffb590e4ffff   push dword [var_1b70h]
|     ||  0x0040137b     ffd6          call esi
|     └──> 0x0040137d     33db          xor ebx, ebx
```

```
|     |  0x0040137f     399d84e4ffff   cmp dword [var_1b7ch], ebx
|    ┌─< 0x00401385     7408           je 0x40138f
|    ||  0x00401387     ffb584e4ffff   push dword [var_1b7ch]
|    ||  0x0040138d     ffd6           call esi
|    └─> 0x0040138f     399d88e4ffff   cmp dword [var_1b78h], ebx
|    ┌─< 0x00401395     7408           je 0x40139f
|    ||  0x00401397     ffb588e4ffff   push dword [var_1b78h]
|    ||  0x0040139d     ffd6           call esi
|    └─> 0x0040139f     399d8ce4ffff   cmp dword [var_1b74h], ebx
|    ┌─< 0x004013a5     7408           je 0x4013af
|    ||  0x004013a7     ffb58ce4ffff   push dword [var_1b74h]
|    ||  0x004013ad     ffd6           call esi
|    └└─> 0x004013af    8d47ce         lea eax, [edi - 0x32]
|        0x004013b2     83f831         cmp eax, 0x31             ; '1' ; 49
|    ┌─< 0x004013b5     771f           ja 0x4013d6
|     |  0x004013b7     83bda4f5ff..   cmp dword [var_a5ch], 0
|    ┌─< 0x004013be     7516           jne 0x4013d6
|    ||  0x004013c0     6a10           push 0x10                 ; 16
|    ||  0x004013c2     68b0714000     push str.Launcher_Error   ; 0x4071b0 ; "Launcher Error"
|    ||  0x004013c7     8d859ce4ffff   lea eax, [var_1b64h]
|    ||  0x004013cd     50             push eax
|    ||  0x004013ce     6a00           push 0
|    ||  0x004013d0     ff1574714000   call dword [sym.imp.USER32.dll_MessageBoxA] ;
0x407174 ; int MessageBoxA(HWND hWnd, LPCSTR lpText, LPCSTR lpCaption, UINT uType)
```

```
|   └└> 0x004013d6   8d8d94e4ffff   lea ecx, [var_1b6ch]
|       0x004013dc   e875000000     call 0x401456
|       0x004013e1   8b4dfc         mov ecx, dword [var_4h]
|       0x004013e4   8bc7           mov eax, edi
|       0x004013e6   5f             pop edi
|       0x004013e7   5e             pop esi
|       0x004013e8   33cd           xor ecx, ebp
|       0x004013ea   5b             pop ebx
|       0x004013eb   e8aa120000     call 0x40269a
|       0x004013f0   c9             leave
└       0x004013f1   c21000         ret 0x10
```

**1. Executable Overview**

### 1.1 General Information

- **File Type**: Windows PE (Portable Executable)
- **Architecture**: Likely x86 (32-bit) or x64 (64-bit) based on the context (specific analysis would require the actual binary).

### 1.2 Purpose of Analysis

The objective is to assess the assembly code for:

- Malicious behaviors or indicators.
- Abnormal or suspicious function calls.
- Potential data exfiltration or system manipulation attempts.

## 2. Entry Point Analysis

### 2.1 Entry Point Address

- **Address**: 0x00402ce1
- This address signifies where the execution starts. The entry point is crucial for analyzing program flow and understanding how the program initializes.

### 2.2 Initial Setup

The program likely performs:

- Stack setup for function calls.
- Initialization of global variables.
- Loading critical libraries required for further execution.

## 3. Dynamic Library Loading

### 3.1 Libraries Loaded

The dynamic loading of libraries can indicate whether the program is leveraging legitimate Windows functions or attempting to hide malicious behavior:

- **kernel32.dll**:

  - Responsible for core Windows functionalities such as memory management, file I/O, and process/thread creation.
  - Functions from this library are often exploited by malware to interact with the system stealthily.
- **ntmarta.dll**:
  - Contains advanced Windows security functions, which could include user authentication.
  - Its usage may point toward access control manipulation.
- **Secur32.dll**:
  - Provides security support for authentication protocols (e.g., Kerberos).
  - Could indicate potential attempts to compromise secure connections or leverage secure protocols.
- **PROPSYS.dll**:
  - Deals with property system functionality for Windows objects.
  - If used improperly, it can be leveraged for information retrieval.

**3.2 Functions Utilized**

The binary calls several Windows API functions:

- **GetModuleHandleA**: Retrieves a handle for a specified module. This could be used to verify if a DLL is already loaded.
- **GetProcAddress**: Obtains the address of an exported function in a DLL. This dynamic resolution can be utilized to evade static analysis.
- **LoadLibraryA**: Loads a DLL into the address space of the calling process. Malicious software often uses this to load additional payloads.

**3.3 Analysis of API Usage**

The combination of these API calls suggests:

- Possible stealth operations to load DLLs only when needed.
- Use of conditional logic to handle scenarios where loading fails, which could mask malicious behavior under certain circumstances.

**4. Flow Control and Execution Path**

**4.1 Control Flow**

The assembly likely contains several control flow structures:

- **Conditional Jumps**: Used to handle scenarios based on whether certain libraries or functions were loaded successfully.
- **Loops and Calls**: Indicative of routine operations that may iterate over a set of actions or continuously check for conditions.

**4.2 Analysis of Jumps and Calls**

- **Jump Instructions (je, jne, jmp)**: These are crucial for analyzing how the binary decides its next steps, especially in response to the success or failure of operations.
- This structure could indicate resilience in design but also an effort to obscure malicious intents through conditional execution paths.

**5. String Analysis**

**5.1 Static Strings in Binary**

- Common strings include names of libraries (kernel32.dll, ntmarta.dll, etc.). While not inherently suspicious, if strings containing sensitive keywords (e.g., URLs, IP addresses, or user data) are discovered, this could indicate malicious intent.

**5.2 Potential Indicators**

- Look for obfuscated strings or strings that contain function names that suggest malicious activities (e.g., malware, backdoor, exploit).

- Any presence of encrypted or base64 encoded strings may require further investigation.

**6. Behavioral Analysis**

**6.1 Runtime Behavior**

To assess the actual behavior of the binary:

- **Static Analysis Limitations**: While static analysis provides insights, runtime behavior is essential for understanding true intentions. A controlled environment or sandbox is needed.

**6.2 Sandbox Analysis**

- **Monitor Network Activity**: Observe for any outbound connections that may indicate data exfiltration or command-and-control communications.
- **File System Changes**: Track file modifications or creations, particularly in system directories.
- **Registry Modifications**: Check if the program attempts to modify registry keys that control startup behaviors or persistence mechanisms.

**7. Potential Malware Indicators**

**7.1 Malicious Behavior Patterns**

Several indicators may suggest malware presence:

- **Dynamic Code Execution**: Use of dynamic loading can evade traditional antivirus detection.
- **Unusual API Calls**: Calls to less common APIs could suggest exploitation attempts or malware functionalities (e.g., keylogging, screen capture).

**7.2 Anomaly Detection**

- **Anomalous Activity**: Watch for activity that deviates from normal operational patterns, such as accessing sensitive files or connecting to known malicious IPs.

**8. Malware Detection Techniques**

**8.1 Signature-Based Detection**

- Use established antivirus software to scan for known signatures associated with malware. This approach can quickly identify common threats.

**8.2 Heuristic-Based Detection**

- Analyzing behavior patterns rather than specific signatures can identify unknown or newly developed malware.

**8.3 Behavioral Analysis**

- Run the binary in a sandbox environment to monitor behavior, looking for:
    - Suspicious file operations (e.g., deletion, modification of critical files).
    - Unexpected network connections or data transfer activities.

**8.4 Static Code Analysis Tools**

- Use disassemblers or decompilers (like IDA Pro, Ghidra, or Radare2) to inspect code structure and logic flows for potentially harmful routines.

**9. Conclusion**

The analysis of the provided assembly code reveals:

- **Dynamic Loading**: While not inherently malicious, the use of dynamic loading combined with sensitive library calls suggests a need for further scrutiny.
- **Control Structures**: The control flow indicates that the binary has conditional operations that could mask malicious behavior under certain circumstances.
- **Potential Threats**: While no overt malicious indicators are present in this static analysis, the use of API calls and dynamic loading raises red flags that warrant further dynamic analysis.

**Recommendations**

- Conduct dynamic analysis in a secure environment to observe real-time behavior.
- Implement comprehensive malware detection tools to ensure robust protection against potential threats.

- Investigate deeper into any abnormal behaviors or patterns observed during testing to confirm or rule out malicious intent.

---

This report serves as a guideline for analyzing the provided assembly code with a focus on identifying potential malware indicators. For a complete assessment, consider integrating both static and dynamic analysis methodologies to achieve a well-rounded understanding of the executable's behavior and intents.