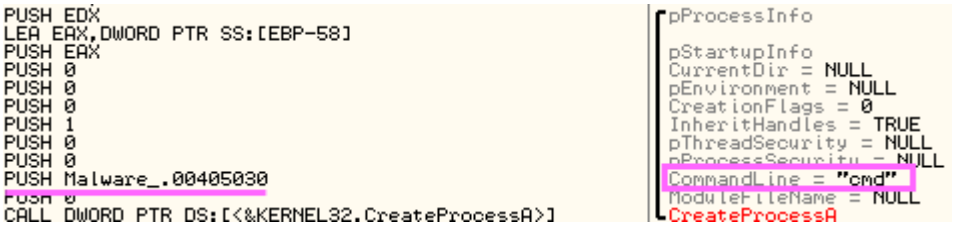
Report UNIT 3 WEEK 11.3

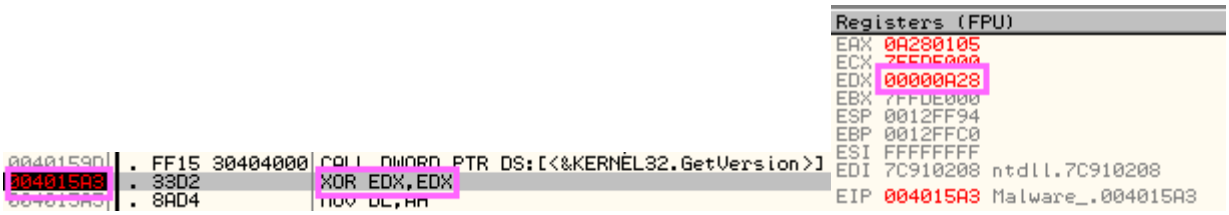
Olly

1. All'indirizzo 0040106E il malware effettua una chiamata di funzione alla funzione CreateProcess. Qual è il valore del parametro CommandLine che viene passato sullo stack?



Il valore del parametro CommandLine è "**cmd**", passato sullo stack utilizzando l'indirizzo Malware_.00405030.

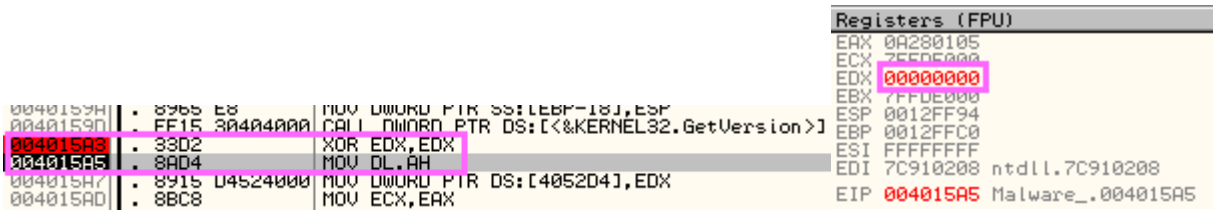
2. Inserite un breakpoint software all'indirizzo 004015A3. Qual è il valor del registro EDX?



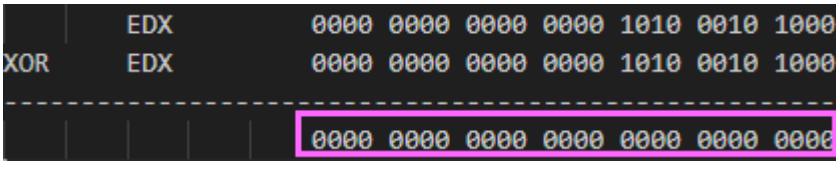
Il valore del registro EDX è 00000A28, in binario base2 0000 0000 0000 0000 0000 1010 0010 1000.

3. Eseguite a questo punto uno step-into. Indicate qual è ora il valore del registro EDX motivando la risposta.

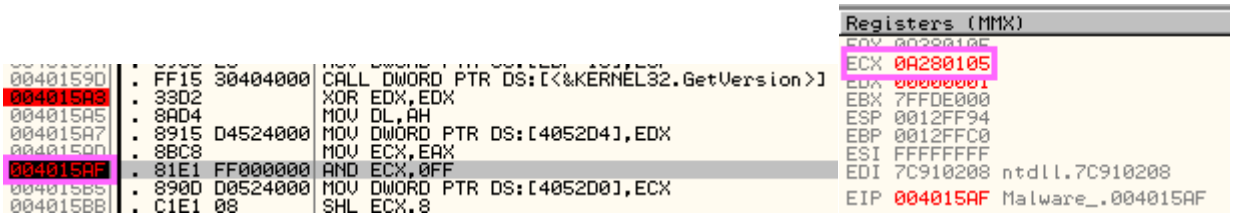
Che istruzione è stata eseguita?



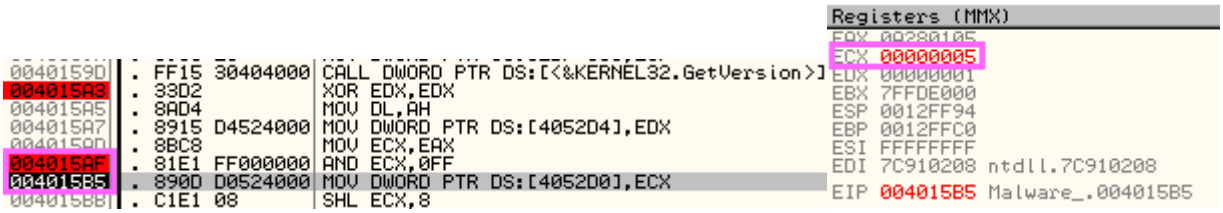
Il valore del registro EDX è 00000000, che corrisponde a 0 in decimale. È stata eseguita l'istruzione XOR (Exclusive OR) tra il registro EDX e sé stesso, impostandolo a 0.



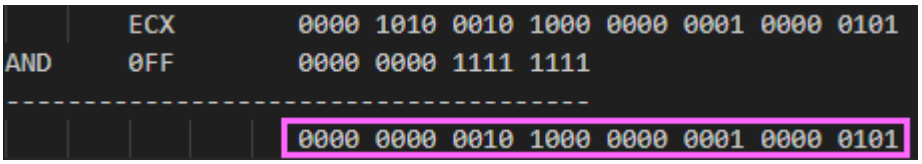
4. Inserite un secondo breakpoint all'indirizzo di memoria 004015AF. Qual è il valore del registro ECX? Eseguite uno step-into. Qual è ora il valore di ECX? Spiegate quale istruzione è stata eseguita.



Il valore del registro ECX è 0A280105, in binario base2 0000 1010 0010 1000 0000 0001 0000 0101.

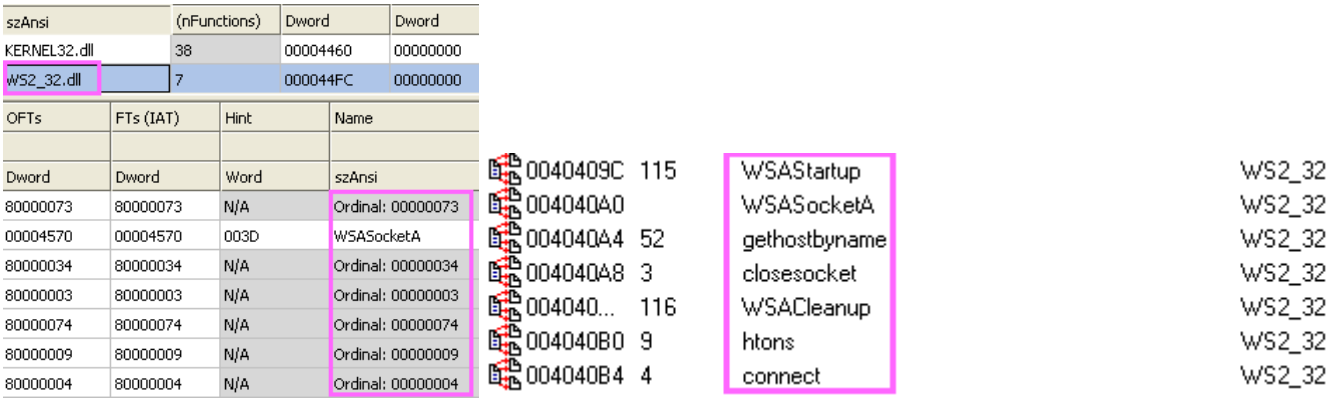


Dopo aver eseguito lo step-into il valore di ECX è 00000005, che corrisponde a 5 in decimale (0000 0000 0010 1000 0000 0001 0000 0101 binario base2). L'istruzione che è stata eseguita è AND confrontando i bit dei due operandi. Se entrambi i bit corrispondenti sono impostati a 1, il risultato sarà 1. Altrimenti, il risultato sarà 0.

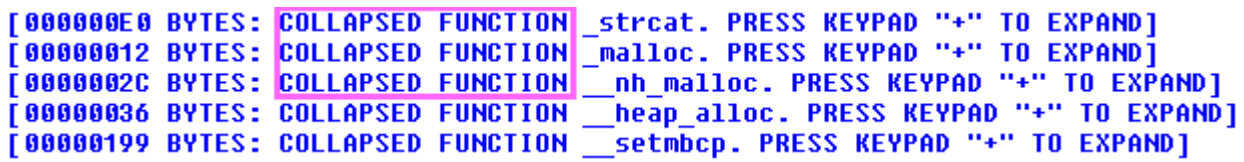


5. Spiegare a grandi linee il funzionamento del malware.

Da CFF le librerie importate dinamicamente sono KERNEL32 e WS2_32.



Anche il codice sembra avere meccanismi di offuscamento/compressione per nascondere il suo comportamento e rendere più difficile l'analisi statica.



Nella main vengono richiamate le funzioni di **WS2_32.dll** per la programmazione del socket e le comunicazioni di rete. Le funzioni richiamate possono far ipotizzare l'implementazione di una backdoor utilizzando **WSASocketA** per creare un socket di rete e **connect** per connettersi al server remoto e inviare o ricevere comandi e dati tramite questa connessione.

```
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

Una grande parte del codice nell’hex-view risulta in byte 00, il contenuto del codice è stato nascosto per rendere più difficile l’analisi. Quando un file viene compresso o criptato, il suo contenuto viene trasformato in una forma che può apparire come sequenze di byte casuali o vuoti nell'hex view.

Considerando l’analisi statica, si può concludere che il funzionamento del malware implementa funzioni che recuperano informazioni sul sistema, ovvero una backdoor.

```
; char aProgramNameUnk[]
aProgramNameUnk db '<program name unknown>',0
                align 4
; char aGetlastactivep[]
aGetlastactivep db 'GetLastActivePopup',0 ; DA
                align 10h
; char aGetactivewindo[]
aGetactivewindo db 'GetActiveWindow',0   ; DATA
; char ProcName[]
ProcName        db 'MessageBoxA',0       ; DATA
; char LibFileName[]
LibFileName     db 'user32.dll',0        ; DATA
```

```
; DWORD __stdcall WaitForSingleObject(HANDLE hHa
                extrn WaitForSingleObject:dword ;
; BOOL __stdcall CreateProcessA(LPCSTR lpApplicat
                extrn CreateProcessA:dword ; DAT
; void __stdcall Sleep(DWORD dwMilliseconds)
                extrn Sleep:dword         ; DATA XR
                                          ; _main+2
; DWORD __stdcall GetModuleFileNameA(HMODULE hMod
                extrn GetModuleFileNameA:dword ;
                                          ; __setar
; BOOL __stdcall GetStringTypeA(LCID Locale,DWORD
                extrn GetStringTypeA:dword
```

```
; int __stdcall LCMapStringW(LCID Locale,DWORD
                extrn LCMapStringW:dword ; DATA
                                         ; ___cr
; int __stdcall LCMapStringA(LCID Locale,DWORD
                extrn LCMapStringA:dword ; DATA
                                         ; ___cr
; int __stdcall MultiByteToWideChar(UINT CodePa
                extrn MultiByteToWideChar:dword
                                         ; DATA
                                         ; ___cr
; HMODULE __stdcall LoadLibraryA(LPCSTR lpLibFi
                extrn LoadLibraryA:dword ; DATA
; FARPROC __stdcall GetProcAddress(HMODULE hMod
                extrn GetProcAddress:dword
```

```
; LPSTR GetCommandLineA(void)
                extrn GetCommandLineA:dword ;
; DWORD GetVersion(void)
                extrn GetVersion:dword   ; DAT
                                         ; Get
                                         ; and
; void __stdcall ExitProcess(UINT uExitCode)
                extrn ExitProcess:dword  ; DAT
                                         ; _do
; BOOL __stdcall TerminateProcess(HANDLE hPro
                extrn TerminateProcess:dword
; HANDLE GetCurrentProcess(void)
                extrn GetCurrentProcess:dword
```