

Homework 3

Pierpaolo Spaziani - 0316331

Introduzione

Lanciando il programma si nota che è una versione modificata di quello analizzato durante il corso e con stesso funzionamento del precedente homework. Se corretto, il codice inserito permette di effettuare lo shutdown della macchina dopo il tempo specificato.

Analisi statica

Essendo la stessa applicazione già vista ed analizzata, procedendo con l'analisi statica con Ghidra si riconosce, con alcune eccezioni, lo stesso schema nel codice.

Si trova infatti la stessa [WinMain](#) ma con 2 nuove funzioni in aggiunta:

[FUN_004024a0](#) e [FUN_00401560](#).

La prima, rinominata [AntiDebug1](#), contiene [IsDebuggerPresent](#) come condizione per la [ShowWindow](#).

La seconda, rinominata [Mappatura](#), salva la lunghezza della parte alta del file e la sua mapped view; questi dati vengono utilizzati nella funzione [FUN_004016b0](#), presente nel blocco [WM_CREATE](#) della [WindowProcedure](#), per fare un controllo sull'integrità del file nel caso venisse modificato. Infatti avviando l'eseguibile modificato, compare la finestra di errore descritta nella [FUN_004042a0](#), rinominata [AntiDebug3](#), contenuta nella [TimerProcedure](#). Questa funzione è stata modificata dal programmatore inserendo delle istruzioni '9a 42' che impediscono il corretto disassemblaggio del codice.

Sempre nella [WinMain](#) c'è la [FUN_00401830](#), chiamata [InitDS](#) durante il corso e nell'homework 2, che differisce dalla precedente per la presenza della [FUN_004016f0](#), rinominata [AntiDebug2](#). In questa funzione è presente in molte posizioni l'istruzione 'eb ff c0 48', inserita dal programmatore per impedire il corretto disassemblaggio del codice. Per poter quindi procedere con la corretta analisi, è necessario modificare il codice sostituendo con dei [NOP](#) le istruzioni di disturbo. Una volta terminata la procedura è necessario fare il Clear, ripetere il Disassemblaggio e fare il Re-create della funzione per avere la procedura pulita. Nella funzione in chiaro è possibile individuare il caricamento dinamico, eseguito carattere per

```

C:\> Decompile: AntiDebug1 - (hw3.exe)
1
2 void __cdecl AntiDebug1(HWND param_1,int param_2)
3
4 {
5     BOOL BVar1;
6
7     BVar1 = IsDebuggerPresent();
8     if (BVar1 == 0) {
9         ShowWindow(param_1,param_2);
10        return;
11    }
12
13    /* WARNING: Subroutine does not return */
14    ExitProcess(0);
15 }

```

```

C:\> Decompile: AntiDebug3 - (hw3.exe)
1
2 /* WARNING: Globals starting with '_' overlap smaller symbols at the same memory location */
3
4 void __cdecl AntiDebug3(int param_1)
5
6 {
7     int iVar1;
8     bool bVar2;
9     char local_10c [128];
10    char local_8c [132];
11
12    if (*(int *) (param_1 + 192) != _DAT_004050a4) {
13        iVar1 = DAT_00407100 + 1;
14        bVar2 = DAT_00407100 == 0;
15        DAT_00407100 = iVar1;
16        if (bVar2) {
17            FUN_004026a0(local_10c,0x80,"%s=%lu/0x%lx");
18            FUN_004026a0(local_8c,0x80,"DEBUG %s:%d");
19            MessageBoxA((HWND)0x0,local_10c,local_8c,0);
20        }
21
22        /* WARNING: Subroutine does not return */
23        ExitProcess(0);
24    }
25    return;
26 }

```

```

C:\> Decompile: AntiDebug2 - (hw3.exe)
26 local_2d = 'k';
27 local_29 = 'e';
28 local_2c = 'e';
29 local_2b = 'r';
30 local_2a = 'n';
31 local_28 = 'l';
32 local_27 = '3';
33 local_26 = '2';
34 local_25 = '.';
35 local_24 = 'd';
36 local_22 = 'l';
37 local_23 = 'l';
38 local_21 = '\0';
39 hModule = LoadLibraryA(&local_2d);
40 local_2d = '0';
41 local_24 = 'u';
42 local_29 = 'u';
43 local_2c = 'u';
44 local_21 = 't';
45 local_28 = 't';
46 local_2b = 't';
47 local_2a = 'p';
48 local_27 = 'D';
49 local_26 = 'e';
50 local_25 = 'b';

```

carattere, della libreria `kernel32.dll` e della funzione `OutputDebugStringA`. Il controllo della presenza del debugger con questa funzione, viene effettuato nel blocco `WM_SIZE` della `WindowProcedure` con la funzione `FUN_00404000`.

```
51 local_1d = 'g';
52 local_23 = 'g';
53 local_22 = 'S';
54 local_20 = 'r';
55 local_1f = 'i';
56 local_1e = 'n';
57 local_1c = 'A';
58 local_1b = '\0';
59 GetProcAddress(hModule,&local_2d);
60 return;
61 }
```

Nella `WindowProcedure` c'è inoltre la funzione `FUN_00401dc0`, rinominata `AntiDebug3`, nella quale, sempre per impedire il debugging dell'applicazione, viene cambiato il flag `BeingDebugged` nel segmento `FS:[30h]` ad offset 2.

```
C:\Decompile: AntiDebug4 - (hw3.exe)
1 void __cdecl AntiDebug4(int *param_1)
2 {
3     int in_FS_OFFSET;
4     *param_1 = *param_1 + (uint)((*(uint *)*(int *)in_FS_OFFSET + 0x30) + 2) & 7) != 0);
5     return;
6 }
```

Come nello scorso homework, la funzione relativa allo shutdown del sistema viene settata in `InitDS`, chiamata nella `TimerProcedure` e fa riferimento a `DAT_004040e0`. Una volta modificato il dato in funzione, disassemblato e rimossi i '9a 42' presenti che confondono il disassemblatore, viene fuori la funzione di gestione una volta scaduto il timer. È possibile osservare infatti la funzione `FUN_00404040` nella quale, una volta rimosse le istruzioni '9a 42', è presente l'`ExitWindowsEx`.

La cosa che risalta nella funzione è un `do/while` di 52 cicli, nel quale vengono fatti delle manipolazioni sulla funzione `FUN_004050c0`, la quale risulta impossibile da analizzare essendo incomprensibile.

Avendo individuato tutti i sistemi anti-debugging e non potendo più proseguire con l'analisi statica, è stato deciso di passare all'analisi dinamica dopo aver patchato una copia dell'eseguibile per rimuovere i vari riferimenti ai sistemi anti-debug appena analizzati.

Analisi dinamica

Passando ad OllyDbg sul codice patchato, è possibile procedere mettendo un breakpoint all'indirizzo `004019ed`, ovvero il punto nella `TimerProc` dove si esegue il primo dei due controlli che permettono di entrare nella funzione di gestione allo scadere del timer.

```
004019e3 a1 30 70 MOV EAX,[AppDS_check_Go_Stop]
004019e8 83 ec 10 SUB ESP,0x10
004019eb 85 c0 TEST EAX,EAX
004019ed 74 08 JZ LAB_004019f7
004019ef 3b 35 2c CMP ESI,dword ptr [AppDS_shutdown_time]
004019f5 73 11 JNC LAB_00401a08
```

```
C:\Decompile: TimerProc - (hw3.exe)
1 void TimerProc(HWND param_1)
2 {
3     uint atmTime;
4     uint time;
5     atmTime = AppDS_time.time;
6     time = AppDS_time.time + 1;
7     AppDS_time.time = time;
8     if (AppDS_time.check_Go_Stop != 0) {
9         secondsLabel();
10    }
11    RedrawWindow(param_1,(RECT *)0x0,(HRGN)0x0,5);
12    if ((AppDS_time.check_Go_Stop != 0) && (AppDS_time.shutdown_time <= time)) {
13        (*(code *)AppDS_time.shutdown_function)(&AppDS_time);
14    }
15    if ((atmTime & 7) == 0) {
16        AntiDebug3((int)&AppDS_time);
17        return;
18    }
19    return;
20 }
```

Il jump nell'if, che non dovrebbe essere eseguito perchè nell'applicazione non è stato avviato il timer, può essere effettuato modificando manualmente il flag Z da 1 a 0 prima di effettuare l'istruzione all'indirizzo `004019ed` e poi modificando il flag C da 1 a 0 per l'istruzione ad indirizzo `004019f5`. In questo modo è possibile andare avanti e portare l'esecuzione alla funzione `FUN_004040e0`, ovvero la `ShutdownFunction`.

Andando avanti si procede fino ad entrare nel `do/while` visto precedentemente, nel quale viene effettuato lo shift del codice. Questa tecnica è stata utilizzata per rendere il codice

statico illeggibile. Terminato, si entra nella funzione `FUN_004050c0`, completamente differente da quella presente su Ghidra.

Analizzando il codice si può vedere che vengono caricati man mano in EAX dei valori esadecimali con cui vengono eseguiti degli XOR e dei confronti con dei valori esadecimali prestabiliti.

Seguendo il flusso e prendendo nota dei valori trovati, è possibile eseguire gli XOR e convertire il risultato ottenuto da esadecimale in ASCII e arrivare al seguente risultato:

3F	28	2F	A5	5D	47	3D	4F	3F
XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR
0C	5A	61	C0	2E	13	0D	70	1E
=	=	=	=	=	=	=	=	=
33	72	4e	65	73	54	30	3f	21
↓	↓	↓	↓	↓	↓	↓	↓	↓
3	r	N	e	s	T	0	?	!

Provando ad inserire la stringa trovata nel programma originale è stato accertato che **3rNesT0?!** è effettivamente la stringa cercata.

```

004050c0 8BEC 1C      SUB ESP,1C
004050c3 A1 E0A04000 MOV EAX,DWORD PTR DS:[40A0E0]
004050c8 8B5424 20    MOV EDI,DWORD PTR SS:[ESP+20]
004050cc 85C0        TEST EAX,EAX
004050ce 74 02       JE SHORT hw3_copi.004050d2
004050d0 E8 63C70424 CALL 24451838
004050d5 3F         AAS
004050d6 282F       SUB BYTE PTR DS:[EDI],CH
004050d8 A5         MOVSDI,DWORD PTR DS:[ESI]
004050d9 C74424 04 5D473 MOV DWORD PTR SS:[ESP+4],4F3D475D
004050e1 C74424 08 3F000 MOV DWORD PTR SS:[ESP+8],3F
004050e9 A1 E0A04000 MOV EAX,DWORD PTR DS:[40A0E0]
004050ee 85C0        TEST EAX,EAX
004050f0 74 02       JE SHORT hw3_copi.004050f4
004050f2 E8 63837C24 CALL 24BCD45A
004050f7 24 09       AND AL,9
004050f9 74 04       JE SHORT hw3_copi.004050ff
004050fb 83C4 1C     ADD ESP,1C
004050fe C3         RETN
004050ff 0FB60424    MOVZX EAX, BYTE PTR SS:[ESP]
00405103 3202       XOR AL, BYTE PTR DS:[EDI]
00405105 3C 0C      CMP AL,0C
00405107 75 F2      JNZ SHORT hw3_copi.004050fb
00405109 0FB64424 01 MOVZX EAX, BYTE PTR SS:[ESP+1]
0040510e 3242 01    XOR AL, BYTE PTR DS:[EDI+1]
00405111 3C 5A      CMP AL,5A
00405113 75 E6      JNZ SHORT hw3_copi.004050fb
00405115 0FB64424 02 MOVZX EAX, BYTE PTR SS:[ESP+2]
0040511a 3242 02    XOR AL, BYTE PTR DS:[EDI+2]
0040511d 3C 61      CMP AL,61
0040511f 75 DA      JNZ SHORT hw3_copi.004050fb
00405121 0FB64424 03 MOVZX EAX, BYTE PTR SS:[ESP+3]
00405126 3242 03    XOR AL, BYTE PTR DS:[EDI+3]
00405129 3C C0      CMP AL,C0
0040512b 75 CE      JNZ SHORT hw3_copi.004050fb
0040512d 0FB64424 04 MOVZX EAX, BYTE PTR SS:[ESP+4]
00405132 3242 04    XOR AL, BYTE PTR DS:[EDI+4]
00405135 3C 2E      CMP AL,2E
00405137 75 C2      JNZ SHORT hw3_copi.004050fb
00405139 0FB64424 05 MOVZX EAX, BYTE PTR SS:[ESP+5]
0040513e 3242 05    XOR AL, BYTE PTR DS:[EDI+5]
00405141 3C 13      CMP AL,13
00405143 75 B6      JNZ SHORT hw3_copi.004050fb
00405145 0FB64424 06 MOVZX EAX, BYTE PTR SS:[ESP+6]
0040514a 3242 06    XOR AL, BYTE PTR DS:[EDI+6]
0040514d 3C 0D      CMP AL,0D
0040514f 75 AA      JNZ SHORT hw3_copi.004050fb
00405151 0FB64424 07 MOVZX EAX, BYTE PTR SS:[ESP+7]
00405156 3242 07    XOR AL, BYTE PTR DS:[EDI+7]
00405159 3C 70      CMP AL,70
0040515b 75 9E      JNZ SHORT hw3_copi.004050fb
0040515d 0FB64424 08 MOVZX EAX, BYTE PTR SS:[ESP+8]
00405162 3242 08    XOR AL, BYTE PTR DS:[EDI+8]
00405165 3C 1E      CMP AL,1E
00405167 75 92      JNZ SHORT hw3_copi.004050fb
00405169 A1 E0A04000 MOV EAX,DWORD PTR DS:[40A0E0]
0040516e 85C0        TEST EAX,EAX
00405170 74 02       JE SHORT hw3_copi.00405174
00405172 E8 63FF5424 CALL 249550DA
00405177 28A1 E0A04000 SUBI BYTE PTR DS:[ECX+40A0E0],AH
0040517d 85C0        TEST EAX,EAX
0040517f 0F84 76FFFFFF JE hw3_copi.004050fb
00405185 E8 6383C41C CALL 1D04D4ED
0040518a C3         RETN
0040518b 8BEC 1C     SUB ESP,1C
0040518e A1 E0000000 MOV EAX,DWORD PTR DS:[E0]
00405193 0000       ADD BYTE PTR DS:[EAX],AL
00405195 0000       ADD BYTE PTR DS:[EAX],AL
00405197 0000       ADD BYTE PTR DS:[EAX],AL
00405199 0000       ADD BYTE PTR DS:[EAX],AL
0040519b 0000       ADD BYTE PTR DS:[EAX],AL

```