

AGGIRARE L' IsDebuggerPresent

Introduzione

In questo breve paper, voglio illustrare uno dei modi che adotto per aggirare una semplice tecnica di anti-debugging, che sfrutta semplicemente un API di Windows per verificare se è attivo il debugging dell'applicazione.

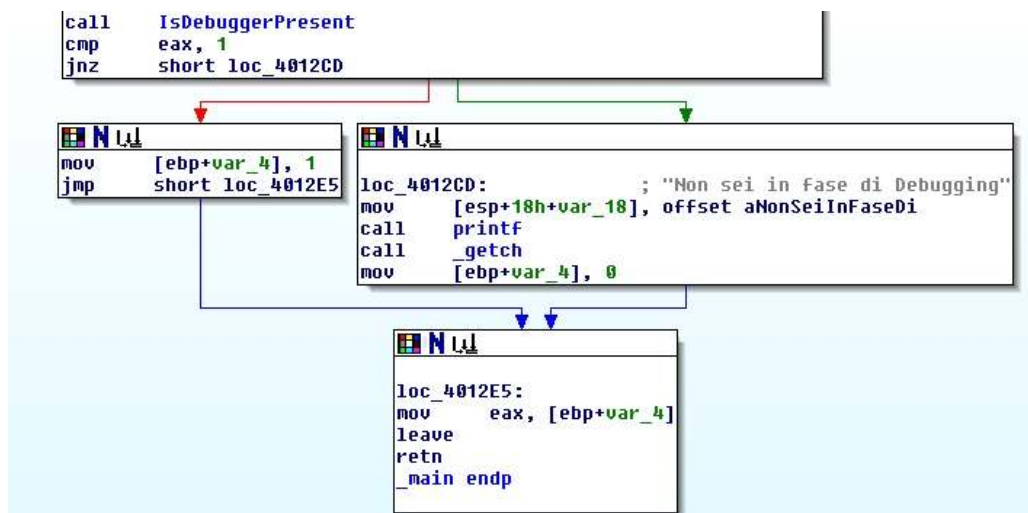
Anche se non è una funzione complessa di anti-debugging, è comunque una funzione che ancora oggi in alcuni software viene utilizzata.

IsDebuggerPresent

E' questo il nome dell'api che consente di stabilire se è attiva l'azione di debugging dell'applicazione, questa funzione restituisce un risultato booleano(0=False,1=True).

Mettiamoci al Lavoro...

Ho pensato che il sorgente del semplice/banale programmino che ho creato per questo paper, lo mostro alla fine, adesso vi mostro la situazione che ci troviamo davanti dopo aver disassemblato l'eseguibile con IDA.



Il codice è abbastanza comprensibile, riassumendo a livello di programmazione, non è altro che un'istruzione IF che compara il risultato dell'api `IsDebuggerPresent` con il valore 1 e cioè TRUE, poi esegue un `JNZ` (Jump if Not Zero), in questo modo se il Flag Zero (ZF) vale 0 allora salta alla `loc_4012CD`, altrimenti esegue il pezzo di codice che viene indicato con la freccia rossa.

Questo significa che se salta alla `loc_4012CD`, la funzione `IsDebuggerPresent` restituisce un valore FALSE, e quindi segnala che non è in corso nessuna azione di Debugging, e come si nota richiama la funzione `printf` e `getch`, per poi fare un `return 1`.

E' ora di aggirare l'anti-debugger

Come detto in precedenza, esistono diversi modi per poter aggirare questa funzione, il modo che utilizzo io, non fa altro che sostituire l'istruzione `JNZ` (Jump if Not Zero) con l'istruzione `JZ` (Jump if Zero), in questo modo ribalto la situazione del controllo, cioè la funzione `IsDebuggerPresent` restituisce sempre un risultato simile al precedente, ma quando controlla il Flag Zero (ZF), se il Flag vale 1 allora salta alla `loc_4012CD`.

In questo modo, non vado a ritoccare il risultato dell'api, ma vado solamente a invertire il controllo sul Flag Zero.

Mettiamo in pratica il concetto

Ora che vi ho spiegato in teoria come funziona il metodo, bisogna metterlo in pratica:

Per fare ciò abbiamo bisogno di un Hex Editor (io utilizzo Hex Workshop), e di una calcolatrice per calcolare l'Offset che ci servirà per andare al punto in cui dovremo modificare l'istruzione.

1) Troviamo l'Offset:

Indirizzo Istruzione – ImageBase – Virtual Address + Offset Raw Data

Indirizzo Istruzione: e' l'indirizzo dell'istruzione che si vuole modificare(cioè la JNZ) nel mio caso con IDA PRO è (4012C2 Hex)

ImageBase,Virtual Address e Offset Raw Data, si trovano in testa al file disassemblato

```
.text:00401000 ; Input MD5 : 50E0FD91F19411DFB3FA08BEF60B769AF
.text:00401000
.text:00401000 ; File Name : C:\Documents and Settings\All Users\Documenti
.text:00401000 ; Format : Portable executable for 80386 (PE)
.text:00401000 ; Imagebase : 400000
.text:00401000 ; Section 1. (virtual address 00001000)
.text:00401000 ; Virtual size : 00000914 ( 2324.)
.text:00401000 ; Section size in file : 00000A00 ( 2560.)
.text:00401000 ; Offset to raw data for section: 00000400
.text:00401000 ; Flags 60000060: Text Data Executable Readable
.text:00401000 ; Alignment : default
.text:00401000 ; OS type : MS Windows
.text:00401000 ; Application type: Executable 32bit
.text:00401000
```

Nel mio caso l'Offset trovato è : 6C2 (Hex)

2) Apriamo Hex Workshop, andiamo su Edit->Goto e inseriamo l'Offset trovato, poi selezioniamo Hex e impostiamo l'opzione "From Where" dall'inizio del file.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	
000006A9	04	89	45	F8	8B	45	F8	E8	7B	04	00	00	E8	16	01	00	00	E8	F1	05	00	00	83	F8	01	75	09	C7	45	FC	01	
000006C8	00	00	00	EB	18	C7	04	24	00	30	40	00	E8	57	05	00	00	E8	D2	04	00	00	C7	45	FC	00	00	00	00	8B	45	
000006E7	FC	C9	C3	90	90	90	90	90	90	55	B9	00	31	40	00	89	E5	EB	14	8D	B6	00	00	00	00	8B	51	04	8B	01	83	
00000706	C1	08	01	82	00	00	40	00	81	F9	00	31	40	00	72	EA	5D	C3	90	90	90	90	90	90	90	90	55	89	E5	DB	E3	
00000725	5D	C3	90	90	90	90	90	90	90	90	55	89	E5	83	EC	08	A1	20	20	40	00	8B	08	85	C9	74	26	EB	0D	90		
00000744	90	90	90	90	90	90	90	90	90	90	FF	10	8B	0D	20	20	40	00	8B	51	04	8D	41	04	A3	20	20	40	00	90		
00000763	85	D2	75	E9	C9	C3	8D	B4	26	00	00	00	55	89	E5	53	83	EC	04	A1	00	19	40	00	83	F8	FF	74	29	85		
00000782	C0	89	C3	74	13	89	F6	8D	BC	27	00	00	00	FF	14	9D	00	19	40	00	4B	75	F6	C7	04	24	30	13	40	00		
000007A1	E8	BA	FE	FF	FF	5B	5B	5D	C3	8B	0D	04	19	40	00	31	C0	85	C9	EB	0A	40	8B	14	85	04	19	40	00	85	D2	
000007C0	75	F4	EB	BD	8D	B6	00	00	00	00	8D	BF	00	00	00	55	89	E5	53	83	EC	04	A1	20	40	40	00	85	C0	75		
000007DF	36	A1	00	19	40	00	BB	01	00	00	00	89	1D	20	40	40	00	83	F8	FF	74	25	85	C0	89	C3	74	0F	90	8D	74	
000007FE	26	00	FF	14	9D	00	19	40	00	4B	75	F6	C7	04	24	30	13	40	00	E8	4A	FE	FF	FF	5B	5B	5D	C3	8B	0D	04	
0000081D	19	40	00	31	C0	85	C9	EB	0A	40	8B	14	85	04	19	40	00	85	D2	75	F4	EB	C1	90	90	90	90	90	90	90	90	
0000083C	90	90	90	90	55	A1	70	40	40	00	89	E5	5D	8B	48	04	FF	E1	89	F6	55	BA	42	00	00	00	89	E5	53	0F	B7	
0000085B	C0	83	EC	64	89	54	24	08	8D	55	A8	31	DB	89	54	24	04	89	04	24	FF	15	D4	50	40	00	BA	1F	00	00	00	
0000087A	B9	01	00	00	00	83	EC	0C	85	C0	75	07	EB	46	01	C9	4A	78	0E	80	7C	2A	A8	41	75	F4	09	CB	01	C9	4A	
00000899	79	F2	83	3B	3C	75	07	89	D8	8B	5D	FC	C9	C3	B9	44	30	40	00	BA	EA	00	00	00	89	4C	24	0C	89	54	24	
000008B8	08	C7	04	24	71	30	40	00	B8	90	30	40	00	89	44	24	04	E8	92	02	00	00	B8	BC	30	40	00	BB	E4	00	00	
000008D7	00	89	44	24	0C	89	5C	24	08	EB	D7	8D	B4	26	00	00	00	8D	BC	27	00	00	00	00	55	89	E5	57	56	53		
000008F6	81	EC	CC	00	00	00	8B	0D	70	40	40	00	85	C9	74	08	8D	65	F4	5B	5E	5F	5D	C3	C7	45	98	41	41	41	41	
00000915	A1	20	30	40	00	8D	75	98	C7	45	9C	41	41	41	41	C7	45	A0	41	41	41	41	41	89	45	B8	A1	24	30	40	00	C7
00000934	45	A4	41	41	41	C7	45	A8	41	41	41	41	41	89	45	BC	A1	28	30	40	00	C7	45	AC	41	41	41	41	C7	45	B0	
00000953	41	41	41	41	89	45	C0	A1	2C	30	40	00	C7	45	B4	41	41	41	41	89	45	C4	A1	30	30	40	00	89	45	C8	A1	

3) A questo punto dovremmo trovarci davanti il Byte "75" che non è altro che il Byte che identifica l'istruzione JNZ, ora non dobbiamo fare altro che modificare quel singolo Byte e impostarlo a "74" che identifica l'istruzione JZ.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	
000006A9	04	89	45	F8	8B	45	F8	E8	7B	04	00	00	E8	16	01	00	00	E8	F1	05	00	00	83	F8	01	74	09	C7	45	FC	01	
000006C8	00	00	00	EB	18	C7	04	24	00	30	40	00	E8	57	05	00	00	E8	D2	04	00	00	C7	45	FC	00	00	00	00	8B	45	
000006E7	FC	C9	C3	90	90	90	90	90	90	55	B9	00	31	40	00	89	E5	EB	14	8D	B6	00	00	00	00	8B	51	04	8B	01	83	
00000706	C1	08	01	82	00	00	40	00	81	F9	00	31	40	00	72	EA	5D	C3	90	90	90	90	90	90	90	90	55	89	E5	DB	E3	
00000725	5D	C3	90	90	90	90	90	90	90	90	55	89	E5	83	EC	08	A1	20	20	40	00	8B	08	85	C9	74	26	EB	0D	90		
00000744	90	90	90	90	90	90	90	90	90	90	FF	10	8B	0D	20	20	40	00	8B	51	04	8D	41	04	A3	20	20	40	00	90		
00000763	85	D2	75	E9	C9	C3	8D	B4	26	00	00	00	55	89	E5	53	83	EC	04	A1	00	19	40	00	83	F8	FF	74	29	85		
00000782	C0	89	C3	74	13	89	F6	8D	BC	27	00	00	00	FF	14	9D	00	19	40	00	4B	75	F6	C7	04	24	30	13	40	00		
000007A1	E8	BA	FE	FF	FF	5B	5B	5D	C3	8B	0D	04	19	40	00	31	C0	85	C9	EB	0A	40	8B	14	85	04	19	40	00	85	D2	
000007C0	75	F4	EB	BD	8D	B6	00	00	00	00	8D	BF	00	00	00	55	89	E5	53	83	EC	04	A1	20	40	40	00	85	C0	75		
000007DF	36	A1	00	19	40	00	BB	01	00	00	00	89	1D	20	40	40	00	83	F8	FF	74	25	85	C0	89	C3	74	0F	90	8D	74	
000007FE	26	00	FF	14	9D	00	19	40	00	4B	75	F6	C7	04	24	30	13	40	00	E8	4A	FE	FF	FF	5B	5B	5D	C3	8B	0D	04	
0000081D	19	40	00	31	C0	85	C9	EB	0A	40	8B	14	85	04	19	40	00	85	D2	75	F4	EB	C1	90	90	90	90	90	90	90	90	
0000083C	90	90	90	90	55	A1	70	40	40	00	89	E5	5D	8B	48	04	FF	E1	89	F6	55	BA	42	00	00	00	89	E5	53	0F	B7	
0000085B	C0	83	EC	64	89	54	24	08	8D	55	A8	31	DB	89	54	24	04	89	04	24	FF	15	D4	50	40	00	BA	1F	00	00	00	
0000087A	B9	01	00	00	00	83	EC	0C	85	C0	75	07	EB	46	01	C9	4A	78	0E	80	7C	2A	A8	41	75	F4	09	CB	01	C9	4A	
00000899	79	F2	83	3B	3C	75	07	89	D8	8B	5D	FC	C9	C3	B9	44	30	40	00	BA	EA	00	00	00	89	4C	24	0C	89	54	24	
000008B8	08	C7	04	24	71	30	40	00	B8	90	30	40	00	89	44	24	04	E8	92	02	00	00	B8	BC	30	40	00	BB	E4	00	00	
000008D7	00	89	44	24	0C	89	5C	24	08	EB	D7	8D	B4	26	00	00	00	8D	BC	27	00	00	00	00	55	89	E5	57	56	53		
000008F6	81	EC	CC	00	00	00	8B	0D	70	40	40	00	85	C9	74	08	8D	65	F4	5B	5E	5F	5D	C3	C7	45	98	41	41	41	41	
00000915	A1	20	30	40	00	8D	75	98	C7	45	9C	41	41	41	41	C7	45	A0	41	41	41	41	41	89	45	B8	A1	24	30	40	00	C7
00000934	45	A4	41	41	41	C7	45	A8	41	41	41	41	41	89	45	BC	A1	28	30	40	00	C7	45	AC	41	41	41	41	C7	45	B0	
00000953	41	41	41	41	89	45	C0	A1	2C	30	40	00	C7	45	B4	41	41	41	41	89	45	C4	A1	30	30	40	00	89	45	C8	A1	

4) Salviamo la nostra semplice modifica.

Alla fine se tutto è andato perfettamente, si può compiere un'analisi dinamica dell'eseguibile.

Conclusioni

Questo metodo consente di aggirare questa semplice soluzione di anti-debugging, eseguendo una rapida modifica (patching) all'eseguibile, così da poter poi analizzarlo in modalità dinamica.

RIEPILOGO:

- + Metodo semplice ed efficace per poter aggirare l'Api IsDebuggerPresent
- Modifica effettiva dell'eseguibile (la modifica non avviene in memoria principale, ma viene scritta in memoria di massa).

Codice Sorgente del programma analizzato:

```
#include <windows.h>

int main()
{
    if( IsDebuggerPresent() == TRUE )
        return 1;

    printf("Non sei in fase di Debugging");
    getch();
    return 0;
}
```

Ho realizzato questo paper con lo scopo principale di far comprendere la tecnica utilizzata, e inoltre ho adottato un linguaggio semplice e passo-passo, per cercare di semplificare la comprensione anche agli utenti che per la prima volta si affacciano al Reverse Engineering.

Autore: Mn90 a.k.a. Hacker Virus_90

E-Mail & Msn: mn90@msn.com