The Italian CRaCKiNG Encyclopedia vol. 3
CRaCKiNG with SoftIce

Scritto da

---==: ALoR :==---
---==> Proud member of NEURO ZONE 2 <==---

Maggio 1998

Target = **, ***

(*=Lamer, **=Novizio, ***=Apprendista, ****=Esperto, *****=CrackMaster)

==========

Table of context :

==========

1.0 DISCLAIMER

Every reference to facts, things or persons (virtual, real or esoteric) are purely casual and involuntary. Any trademark nominated here is registered or copyright of their respective owners. The author of this manual does not assume any responsability on the content or the use of informations retriven from here; he makes no guarantee of correctness, accuracy, reliability, safety or performance. This manual is provided "AS IS" without warranty of any kind. You alone are fully responsible for determining if this page is safe for use in your environment and for everything you are doing with it!

Ormai lo saprete a memoria.....ma mi è necessario per pararmi il culo !!!

2.0 I TOOLS NECESSARI

I tools necessari per crackare li abbiamo già visti nel primo volume. Comunque è meglio ricordarli.

* WDasm 8.9 or higher
* Un editor esadecimale (Es.: UltraEdit)
* Un manuale di Assembly
* Un manuale delle API. (importantissimo !!)

In oltre è necessario anche un debugger in realtime, che svolge le funzioni che mancano a WDASM quindi è necessario

* Soft-Ice 3.2 or higher

3.0 ICECRACKING

Questo capitolo è dedicato al cracking dei programmi shareware per Windows. Il metodo utilizzato qui è diverso e notevolmente migliorato rispetto a quello usato nel primo volume dell'enciclopedia. Vedremo tra poco come utilizzare le innumerevoli funzioni che ci offre Soft-Ice (vedi Appendice -A-).
In particolare analizzeremo come utilizzare al meglio il breakpoint sulle API GetdlgitemTextA e GetWindowTextA.
Il nostro obbiettivo è trovare il codice di registrazione abbinato al nome che gli forniamo.
Procederò con un approccio diverso rispetto agli altri manuali. Procederò ora ad illustrare, con un esempio pratico, come registrare uno shareware. Per il nostro scopo, utilizzerò uno dei più famosi shareware per Windows: Winzip 6.3.

Per prima cosa testiamo il programma per dedurre tutte le varie "scocciature" che presenta lo shareware, per winzip possiamo trovare:

 - Un nag screen iniziale
 - La scritta "Unregistered" nella barra del titolo del programma
 - Possiamo registrarci attraverso il menu Help - About - Register

Secondo, dobbiamo decidere come crackare il programma, ci si presentano varie ipotesi:

 - Trovare la giusta Key in memoria
 - Jumpare la registration check all'avvio
 - Rimuovere tutte le "scocciature" dello shareware

Il terzo tipo è molto poco raffinato ed è tipico dei crakers stupidi. In questo modo si camuffa il programma, non lo si cracka !! :-)
Il secondo l'abbiamo già affrontato nei precedenti manuali. Ma talvolta non funziona o porta al crash di sistema.
La nostra scelta ricade quindi sul primo tipo. E' molto più semplice.
E' raffinato. NON modifica il codice del programma, ottima cosa !! E lo trovo più da cracker :-)

OK. Martini... Sigaretta... e si parte... :-)

** Carichiamo Soft-ice e Winzip 6.3.

Andiamo al menu Help - About - Register. Qui ti verrà chiesto il tuo nome e il codice di registrazione. Io inserirò name: ALoR e code: 123456789. Ok.
Apparirà un messaggio di errore. Il nostro obbiettivo è trovare il punto del codice che richiama questa procedura. In questo ci viene in aiuto Soft-ice.
Una cosa che non si poteva fare con WDasm era mettere breakpoint sulle API (vedi Appendice -A-). Invece con Soft-ice sfrutteremo proprio le API di Windows che sono utilizzate per leggere il testo da un Textbox come quelli del name e del code. Le API uste comunemente sono :

  GetWindowTextA
  GetDlgItemTextA

Mettiamo i BreakPoint sulle API (BPX GetDlgItemTextA e BPX GetWindowTextA). Continuando con l'esecuzione del programma (CTRL+D). Soft-ice prenderà il controllo e si fermerà sulla nostra API. Siccome i campi da leggere sono 2, Soft-ice si bloccherà un'altra volta. In tutto 4, 2 per ogni campo. Ctrl+D, e noteremo che prima del messaggio non legge più dai textbox. Ora, dopo la quarta interruzione, tracciamo la call con F8. Ora ci troviamo nel codice dell'API. Possiamo tracciare fino al RET oppure F12 e... siamo all'istruzione appena successiva alla CALL GetDlgItemText. Togliamo tutti i Breakpoint (BC *) e ne mettiamo uno su questa istruzione. Che, se avete fatto bene, dovrebbe essere :

xxxx:00409D73 MOVZ EAX, BYTE PTR [00471258]

Ora il programma ha letto il nome e il codice. Se steppiamo, probabilmente troveremo un jump condizionale che segue una call di controllo del codice. Troviamolo insieme steppando (F10) fino a che non compare il messaggio di errore. Dopo poco, il messaggio appare, annotiamo l'indirizzo della call che richiama il messaggio (00409DA8).
Così abbiamo scoperto che il controllo sul codice viene effettuato da qualche parte tra 00409D73 e 00409DA8.
Per trovare la funzione e per capire a fondo il codice, ci viene in aiuto il buon vecchio WDasm. Disassambliamo... ed ecco il codice...

```
* Reference To: USER32.GetDlgItemTextA, Ord:00F5h
                                       |
:00409D6D  Call dword ptr [00476AC8]
:00409D73  movzx eax, byte ptr [00471258]  <- memorizza il nome
```

```
:00409D7A  test eax, eax
:00409D7C  je 00409D92                     <- salta se il campo è vuoto
:00409D7E  movzx eax, byte ptr [0046F578]  <- memorizza il codice
:00409D85  test eax, eax
:00409D87  je 00409D92                     <- salta se il campo è vuoto
:00409D89  call 004096EA                   <- controllo sul codice
:00409D8E  test eax, eax
:00409D90  jne 00409DD3                    <- Jmp se è sbagliato


* Referenced by a (U)nconditional or (C)onditional Jump at Addresses:
|:00409D7C(C), :00409D87(C)
|

:00409D92  call 00409F9C                   <- funzione "codice errato"

* Possible Reference to String Resource ID=00654: "Incomplete or incorrect
                                                  information"

:00409D97  push 0000028E
:00409D9C  call 00424ECF
:00409DA1  pop ecx
:00409DA2  push eax
:00409DA3  push [ebp+08]
:00409DA6  push 0000003D
:00409DA8  call 004230FD                   <- messaggio di errore
:00409DAD  add esp, 0000000C
:00409DB0  mov eax, dword ptr [0046C860]
:00409DB5  inc eax
:00409DB6  mov dword ptr [0046C860], eax
:00409DBB  cmp dword ptr [0046C860], 00000003
:00409DC2  jne 00409DCF
:00409DC4  push 00000000
:00409DC6  push [ebp+08]

* Reference To: USER32.EndDialog, Ord:00B4h

:00409DC9  Call dword ptr [00476AE4]

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00409DC2(C)
|
:00409DCF  xor eax, eax
:00409DD1  jmp 00409E45                    <- jmp alla fine della funzione



* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00409D90(C)
|
:00409DD3  push 00471258

* Possible StringData Ref from Data Obj ->"Name"

:00409DD8  push 0046396C

* Possible StringData Ref from Data Obj ->"WinZip"

:00409DDD  push 00466F00

* Reference To: KERNEL32.WriteProfileStringA, Ord:0288h

..........
..........      In queste righe il programma usa le API
..........      WriteProfileStringA
..........      WritePrivateProfileStringA
..........      rispettivamente per scrivere la registrazione
..........      nel file Win.ini e nel Winzip32.ini

:00409E16  Call dword ptr [00476AE4]
:00409E1C  xor eax, eax
:00409E1E  jmp 00409E45                    <- jmp alla fine della funzione

..........
..........


* Referenced by a (U)nconditional or (C)onditional Jump at Addresses:
|:00409C45(U), :00409CF9(U), :00409D44(U), :00409DD1(U), :00409E1E(U)
```

```
|:00409E2D(U), :00409E41(U)
|
:00409E45  leave
:00409E46  ret 0010                          <- FINE PROCEDURA
```

Possiamo facilmente verificare che la call giusta è quella all'indirizzo 00409d89, modificando il flag "Z" (vedi Appendice -A-) prima di eseguire il jne 00409DD3.

Ora che sappiamo quale è la procedura che controlla il codice... sarà un gioco da ragazzi trovare il codice giusto in memoria... :-) Ricordando che winzip aveva salvato in [00471258] il nome e in [0046F578] il nostro codice... sarà ancora più facile...
Il metodo che utilizzo è controllare i parametri passati alle call attraverso i PUSH. Analizzandoli prima e dopo la call... si noterà la differenza :-)

Ecco cosa ho trovato :

```
:004097E4  push eax
:004097E5  push 00471258                        <-- indirizzo del nome
:004097EA  call 004098C3                        <-- calcolo codice
:004097EF  pop ecx
:004097F0  pop ecx                              <-- provate "d ECX" :-)
:004097F1  push 0046F578                        <-- vi ricordate cos'era ?...
:004097F6  lea eax, dword ptr [ebp+FFFFFDF8]    <-- in questo istante EBP
                                                 contiene 0012F950. +FFFFFDF8
                                                 in complemento a 16 è -208.
                                                 Quindi stà mettendo in EAX
                                                 qualcosa che si trova all'
                                                 indirizzo 0012F748. Cosa ci
                                                 sarà mai a quell'indirizzo.:)
:004097FC  push eax
:004097FD  call 004465D0                        <-- parametri passati:
                                                     0046F578  il nostro codice
                                                     EAX       il codice giusto
:00409802  pop ecx
:00409803  pop ecx
```

OK. Facendo "d ECX" oppure "d 0012F748" abbiamo trovato il codi giusto, nel mio caso 11AA0220.
Tolgo tutti i breakpoint (bc *) e scrivo il codice giusto.
WOW... abbiamo registrato winzip, e senza modificare il codice... :-)

Trovata la procedura di calcolo del codice, è facile ricavarne un key generator... Probabilmente affronterò questo argomento nel prossimo volume dell'enciclopedia.

Appendice -A- USARE SOFTICE

Visto che con Soft-ice non è fornito un manuale dei comandi, cercherò ora di spiegarvi le sue principali funzionalità. Iniziamo quindi dall'interfaccia. La finestra principale di Soft-ice è divisa in settori, ognuno di questi può essere abilitato oppure no. Vediamo quali sono, e i comandi per richiamarle:

```
Code Window - Qui si può leggere il codice del programma. Comando WC
Data Window - Segmento Data.                              Comando WD
Floating Point Window - Registri in floating point.       Comando WF
Register Window - Tutti i registri e i Flag.              Comando WR
Watch Window - Per seguire i cambiamenti di un registro.  Comando WW
```

Io consiglio di tenere attivate solo WC e WR.
Per impostare di default queste finestre, in modo da avere sempre la stessa configurazione ad ogni avvio di Soft-ice. Lanciate il Symbol Loader e selezionate dal menu edit la voce Soft-Ice inizialization setting. Nella sezione general scrivete nella textbox Inizialization String i comandi che più desiderate. Fate attenzione a dividerli con un ; e a terminare la stringa sempre con X; Appena installato Soft-ice visualizza wc e wl quindi nel nostro caso la stringa sarà "wr;wl;X;" per avere la configurazione WR WC.
Nella versione per Windows 95 questa configurazione si trova nel file "winice.dat" e può essere modificato come un file di testo.

Vediamo ora i comandi che useremo più di frequente:

*** Breakpoint

BPX indirizzo - Breakpoint su indirizzo
BPX nome api - Breakpoint sull'API
BL - Visualizza la lista dei breakpoint
BC numero breakpoint - Cancella il breakpoint relativo al numero di lista

*** Stepping

F8 - Step Into
F10 - Step Over
F12 - RET ritorna all'istruzione dopo la call

*** Memoria

D indirizzo - Visualizza il contenuto di quell'indirizzo
D registro - Visualizza il contenuto del registro

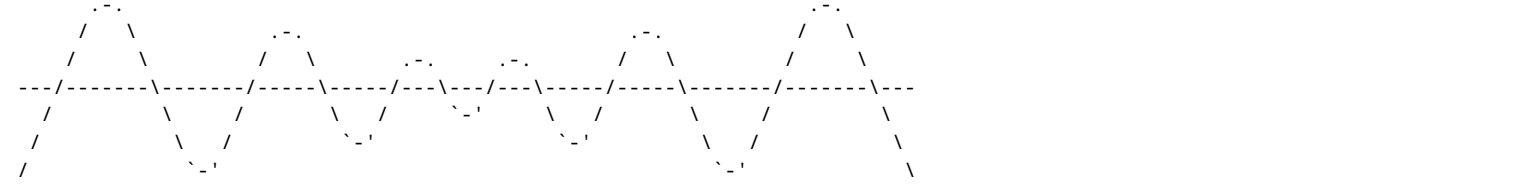
*** Flag

Nella finestra dei registri (WR) basta selezionare un Flag (es. Z) e pigiare
il tasto "ins", il suo valore cambierà da 0 a 1 e viceversa.

 4.0 CONCLUSIONI

Il prossimo numero è già in cantiere... argomento: Key Generator.
Per avere eventuali future release di questo manuale scrivete a: nz2@usa.net
oppure visitate il sito http://Alor.home.ml.org

"If you give a man a crack he'll be hungry again
 tomorrow, but if you teach him how to crack, he'll
 never be hungry again"
                                        +ORC

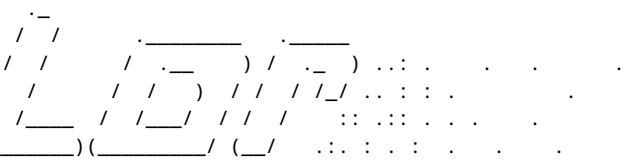


Greetings to: LordKasKo (my cracking teacher...)

The other NEURO ZONE 2 members (10t8or, DK2DEnd, LordKasKo,
                                MaPHas, Ob1, TurboSnail, XXXX)

All the Cracker on the NET from whom I have learned something.

==========



--==> ALoR <==---------------------- - -   -
ICQ: 10666678   In IRC: WhiteFly

e-mail: ALoR@thepentagon.com   www: http://ALoR.home.ml.org/

==========