Cos’è e come si crea

Un Hack

-

> Guida teorica alla creazione

di un Hack

Autore: SIDat
sidat@live.it

Combattiamo l’ignoranza sulla creazione di hack per giochi online e non, spiegando prima tutti i concetti base e poi procedendo alla creazione teorica di un classico hack per il più famoso e giocato MMORPG online: Metin2.

Indice:

Introduzione

Proprio qualche giorno fa mi ha scritto un ragazzo che voleva effettivamente capire come si crea un hack e mi ha fatto venire in mente quando anche io non sapevo proprio dove sbattere la testa.

In questa guida, quindi, vi cercherò di spiegare, come prima cosa, cos' è un hack e successivamente, cosa si deve fare per crearlo. Il tutto sarà in linea teorica, cioè io non andrò a fornire un codice per la creazione di un hack. Andremo inoltre ad affrontare le principali problematiche di questa pratica, cioè l'unpacking del gioco. Vi fornirò infine molto materiale esterno da studiare dove potrete affrontare nello specifico alcuni degli argomenti che tratterò blandamente.

Quindi, cercherò di non dilungarmi troppo negli argomenti, per rendere di facile lettura e comprensione questa guida, buona lettura.

Disclaimer: Ci tengo a precisare che tutte le informazioni contenute in questa guida mirano a insegnare, con le più nobili intenzioni, quella che è un pratica che si può rilevare illegale. Quindi invito il lettore a non prendere le parole dette come motivo di incitazione a svolgere pratiche illegali.

IL REVERSE ENGINEERING

Vorrei iniziare questa guida, introducendovi in quella che è la sovracategoria della creazione di hacks, cioè il reverse engineering.

Questa pratica, meglio nota come, reversing, è la pratica di rovesciamento del codice, ed è anche la filosofia di un cracker, nonché un mondo stupendo dove il cracker cerca di rovesciare il codice di un software (programma/gioco che sia).

Per fare un esempio pratico di reversing in ambito informatico basta pensare a tutte le crack di programmi commerciali quali Photoshop e company; queste crack sono appunto fornite dai cracker, che analizzano il codice e cercano di raggirarlo tramite tecniche come, code injection, dll injection ecc…

Andiamo per esempio ad analizzare il procedimento di creazione di un keygen per un programma Adobe, come Dreamweaver (il keygen lo potete trovare sul mio blog).

Come prima cosa il cracker va a controllare che il programma non sia protetto da un eventuale packer. Successivamente andrà analizzare il codice, (ovviamente il codice sarà di basso livello, cosa che capiremo successivamente) per capire l'algoritmo di generazione del serial number. Una volta capito quello, si può creare un keygen, però l'ultimo problema che incombe è il fatto che i seriali sono tutti memorizzati nei database Adobe al momento in cui sono forniti ai clienti, quindi il programma prima lo accetterà, ma dopo poco tempo si renderà conto del fatto che il seriale non è presente nel database e quindi chiuderà il programma. Ovviamente per ovviare a questo, non c'è niente di più semplice: togliere la possibilità al programma di connettersi ad internet, tramite il firewall.

Se non avete capito bene, non importa, anzi sono contento perchè che vuol dire che vi state avvicinando a questo campo, e visto che in Italia non siamo in tanti a cavarcela con il reversing, fa piacere (io mi escluderei, solo che per la scorrevolezza della frase non lo faccio LOL).

Spiegheremo tutto quel che ho detto nei paragrafi successivi, passo per passo, ma senza approfondire troppo. Per approfondire le varie parti che affronteremo in questa guida fornirò link esterni molto utili.

CONOSCENZE BASILARI

Andiamo adesso ad analizzare le conoscenze basilari che dovete avere per poter capire effettivamente cos'è l'hack e come lo si crea, andando a capire prima i componenti hardware sulla quale ci concentreremo, o meglio, sulla quale si concentra tutto ciò che vediamo sul computer, quali RAM & CPU (lasciando appunto stare la memoria permanente e le altri componenti "meno importanti").

Dopodiché ci concentreremo sulla parte software, cioè l'analisi del codice, nella quale vi indicherò qualche guida e magari qualche programma indispensabile per un analisi attenta di un codice closed-source (cioè, il quale source non è accessibile perché non rilasciato dai produttori).

RAM & CPU

In questo paragrafo andremo a conoscere 2 delle principali componenti del nostro computer, per aiutare a capire a fondo questa guida.

Partiamo dalla RAM:

La **RAM**, sigla di, **R**andom **A**ccess **M**emory (Memoria ad accesso casuale), è un tipo di memoria dove i dati vengono riposti in modo assolutamente casuale.

Chi ha studiato decentemente, il linguaggio C, dovrebbe ben conoscere questo componente per l'uso dei puntatori (di cui parlerò bene dopo, in quanto indispensabili), chi non lo conosce, oltre al fatto che dovrebbe impararlo ☺, gli spiegherò all'incirca come funziona.

Iniziamo col dire che la RAM è una memoria provvisoria, cioè allo spegnimento del computer, viene formattata; quindi è la memoria di cui usufruiscono i programmi per immagazzinare informazioni (QUESTA COSA DEVE ENTRARVI IN TESTA).

La RAM è divisa in tante "locazioni di memoria" espresse ciascuna tramite un indirizzo virtuale in memoria, detto anche address. Ogni VA (virtual address), corrisponde ad un byte in memoria; quindi stabilito un byte con indirizzo 0, gli altri sono in ordine crescente, ma ricordiamoci però che è sbagliato dire che il primo

programma che viene aperto sul pc occupa gli indirizzi da 0 a 600 e il secondo da 600 a 1200, perché ripeto che la memoria è occupata in modo casuale.

Per quanto riguarda la RAM ci sarebbero altre cose da dire, ma penso di avervi dato un infarinatura adeguata all'argomento di questa guida e se mi è sfuggito qualcosa lo spiegherò successivamente.

Passiamo quindi alla CPU:

La CPU è sicuramente un componente indispensabile per il funzionamento del computer perché si occupa di portare i dati, nella memoria.

Funziona quindi da regista del computer, smistando le informazioni alle locazioni di memoria in modo casuale per quanto riguarda la RAM e in modo sequenziale per quanto riguarda la memoria permanente (gli hard disk). La CPU è formata da varie componenti interne, ma per questa guida e più in generale, nel reversing, si ci concentra sui **registri**, sullo **stack** e sui **flags**.

I registri e lo stack sono due tipi differenti di memoria, molto utili al pc per immagazzinare un certo tipo di informazioni che variano da registro a registro; i registri variano da 8 a 32 bit e spesso, sono usati dai programmi per fare piccoli calcoli oppure immagazzinare degli offset (indirizzi) di memoria, lo stack invece è un altro tipo di memoria più ampio costruito secondo un sistema detto LIFO, The **L**ast **I**s **F**irst **O**ut (mi meraviglio del fatto che me lo sono ricordato senza andare a leggere documentazione LOL), cioè l'ultimo dato che viene immagazzinato è il primo ad uscire; questo perché questo è un tipo di memoria sequenziale, cioè non posso estrarre dei dati dallo stack in base al mio volere, posso solamente leggerli perché sono anch'essi accompagnati da un indirizzo. L'estrazione e l'immissione dei dati, vedrete che sarà effettuabile esclusivamente a basso livello tramite l'ausilio di istruzioni come PUSH e POP.

COMPILAZIONE & DEBUGGING

Il compilatore, il software che compila appunto il codice sorgente del nostro programma, è un programma che serve a tradurre il codice da “alto livello” a “basso livello”, ciò vale a dire che il codice sorgente diventa codice assembly.

Quindi l’unico modo per analizzare il codice di un programma compilato è andare a visualizzare questo codice assembly, tramite l’ausilio di un software chiamato debugger(tirati uno schiaffo se pensavi che il debugger servisse per togliere i bug LOL). Chiarito questo concetto, dovete sapere che il linguaggio assembly si dice “di basso livello” perché è il linguaggio di programmazione che più si avvicina al linguaggio macchina, quindi chi si era abituato a leggere robe del tipo: …

```
if (miocazzo < tuocazzo)  {
printf (“cazzino”);
} else {
printf (“cazzone”);
}
```

… se lo dimentichi, perché un ciclo if, ad esempio, sarà dettato da un istruzione di salto condizionato, ed otterremo un codice tipo:

```
CMP EAX, EBX
JNZ SHORT 00401231
JMP 0040123F
```

Purtroppo, però non posso spiegarvi l’asm, perché questa guida diventerebbe un libro di 300 pagine, quindi vi rimando a qualche link interessante dove potrete trovare tutte le informazioni che cercate al riguardo.

Consiglio questi link:

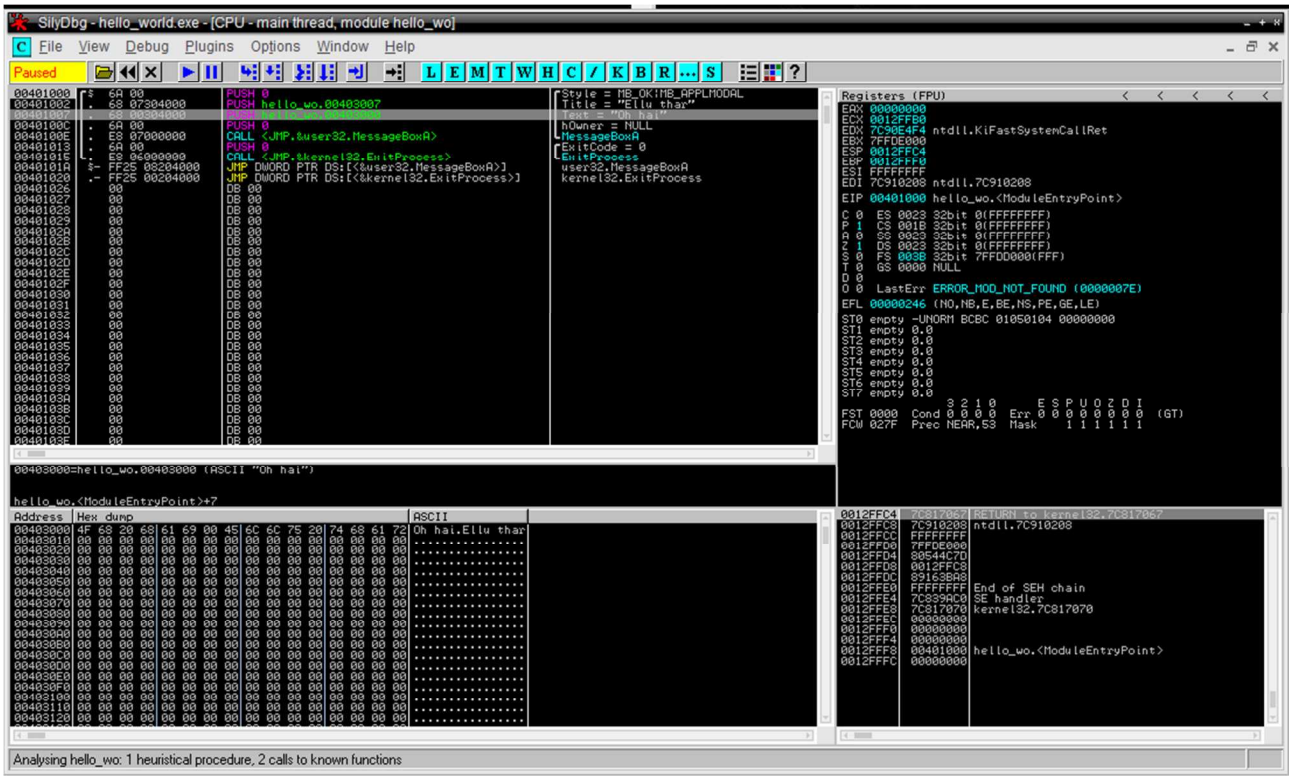
- UIC - Quequero

- Ra.M software una guida meravigliosa

PROGRAMMI INDISPENSABILI

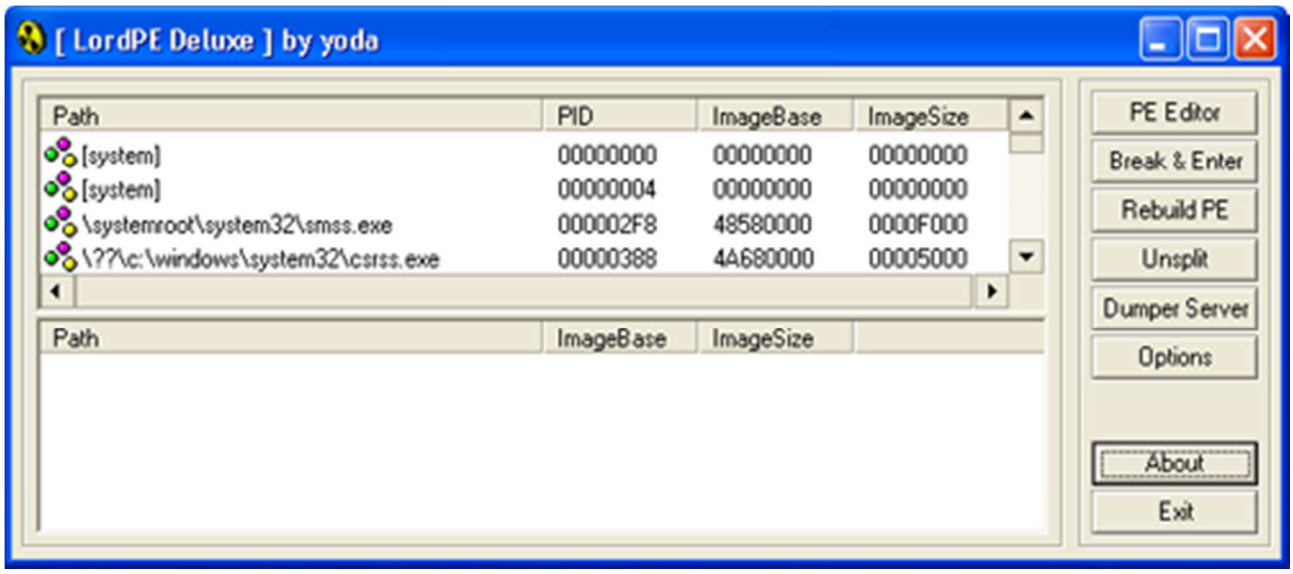
Ci sono, anche in questo ambito informatico, dei tools che sono indispensabili.

Abbiamo già parlato del debugger, cioè il programma che ci permette di visualizzare il codice assembly di un programma; chi è un po' più vecchiotto sicuramente sarà attaccato a SoftIce, ma per voi che siete newbies io consiglio il più usato e moderno dei debugger: OllyDBG



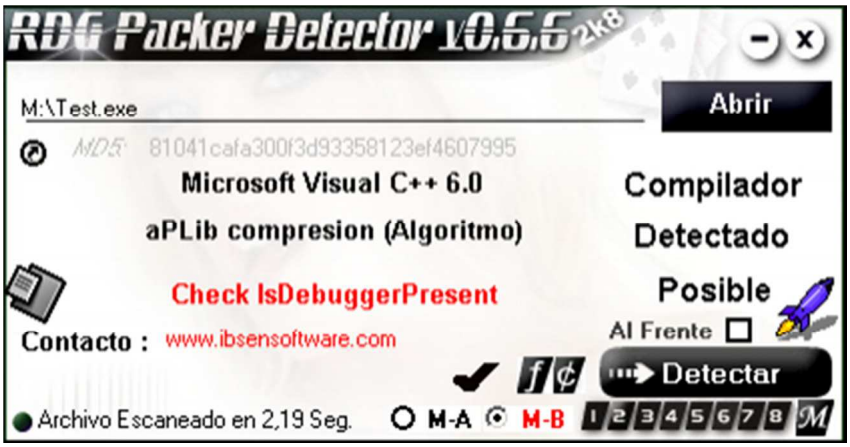
OllyDBG (che da adesso chiameremo Olly), è il programma che useremo più spesso di tutti, ed è per questo che dobbiamo conoscerlo bene. Per non appesantire troppo la guida, vi riporto una buonissima guida su Olly, QUA . Invece QUA trovate il download della versione 1.10, perché sono convinto sia la migliore visto che la 2.0 è ancora una beta e a me non piace ha creato qualche problema.

-LordPE, è un altro programma fondamentale, e serve per analizzare il PE (**P**ortable **E**xutable) del programma in modo da capire se l'entrypoint è spoofato, camuffato, virtualizzato e cazzi e mazzi…



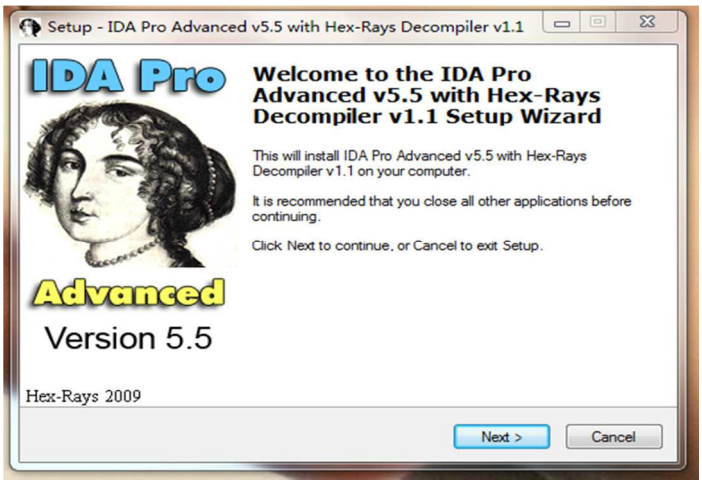
Downlaod QUA. Invece una buonissima guida che io vi consiglio vivamente per capire bene, sia come sono fatti i programmi e sia cos'è il PE e il manual unpacking, QUA.

-RDG Packer Detector, è appunto un packer detector che aiuta a capire se il programma da crackare o meglio il gioco da analizzare è protetto da qualche packer commerciale e non.



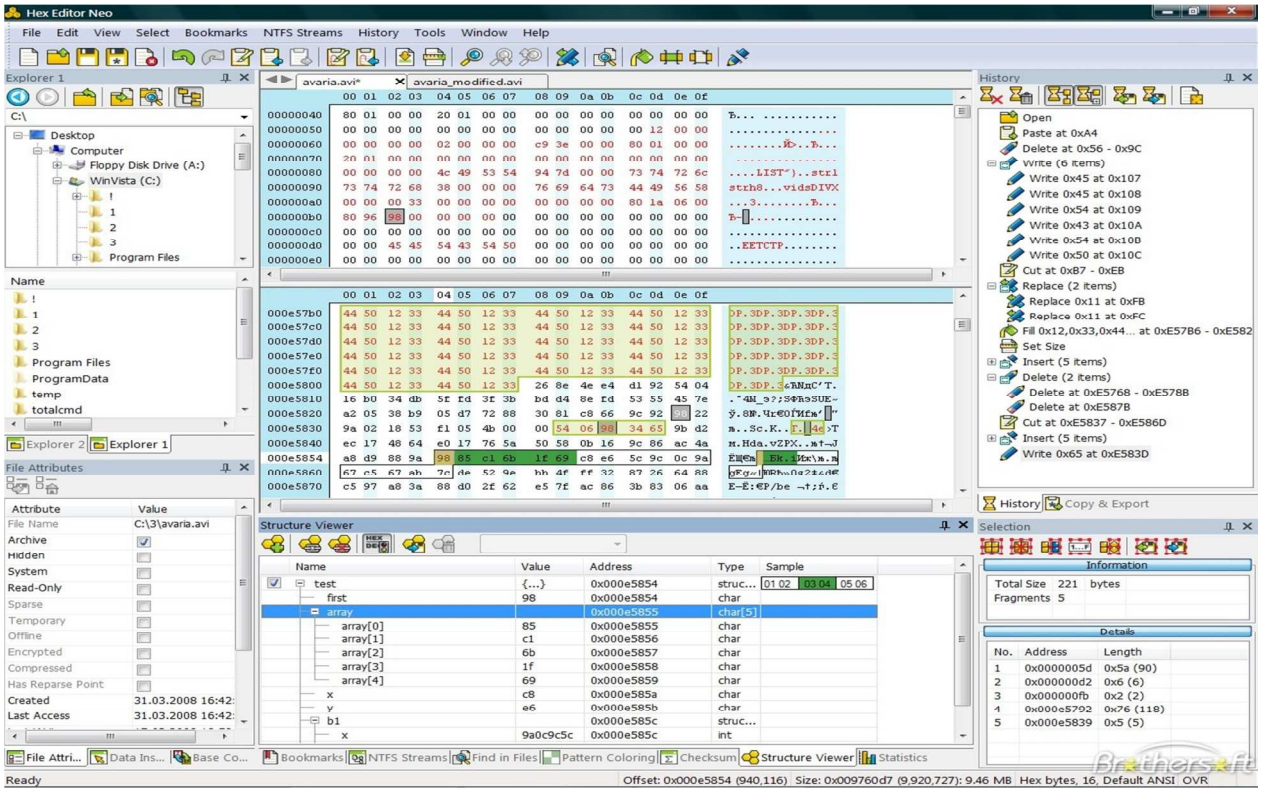
Download QUA.

-IDA Pro, il miglior decompilatore attualmente trovabile sul web. Molto utile per l'analisi del codice, ma a pagamento; c'è però la versione lite gratuita di cui vi posterò il download



Download IDA lite : QUA

-Hex Editor, programma che serve a visualizzare ed editare il codice esadecimale del programma, utilissimo ed indispensabile in alcuni casi; ce ne sono tantissimi gratuiti e belli, io personalmente uso hex editor neo



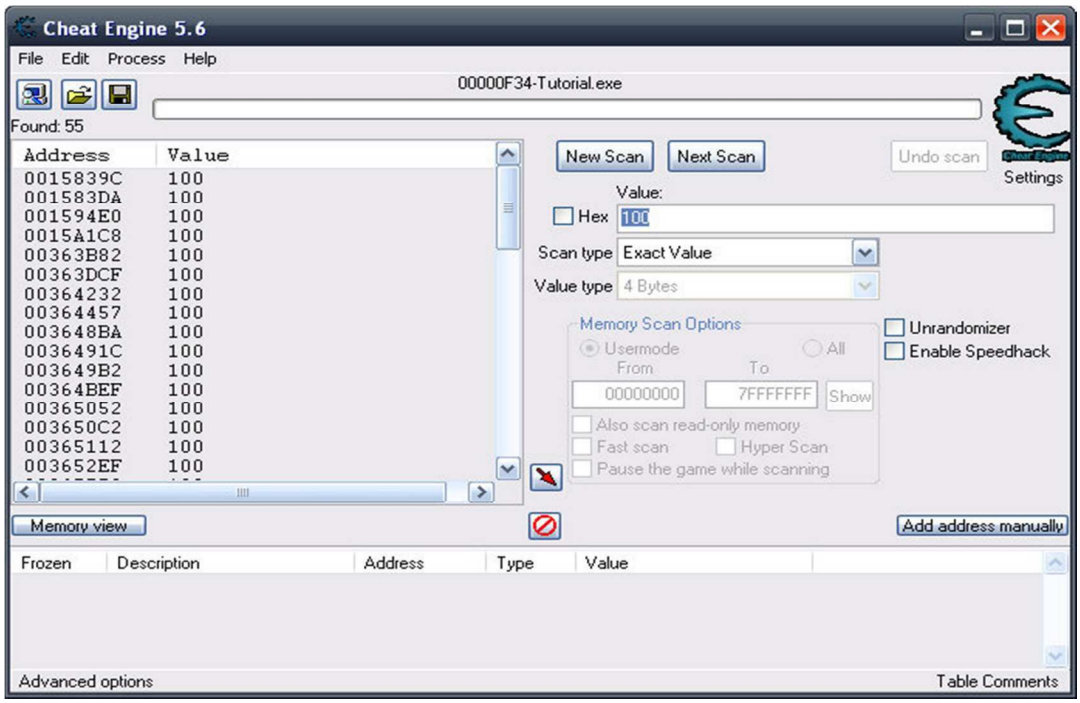
Se volete questo il download QUA , se no cercate su google.

CHEAT ENGINE, SCANNER DI MEMORIA

Sicuramente, in questa guida, questo programma merita un capitolo tutto suo, perché è sicuramente il software che vi troverete ad usare più di tutti nella pratica di creazione di un hack, perché questo strumento ci permette di scansionare e modificare la memoria a nostro piacimento.

Non so se voi avete mai cercato di fare lo speed hack per Metin2 con questo software, perché cercando semplicemente “speed hack metin2” trovate molte guide dettagliate sulla creazione di quest’hack con cheat engine. Io, nella mia ignoranza, quando leggevo queste guide eseguivo quel che mi veniva detto di fare e basta, senza capire realmente il perché (ma dopotutto avevo 10 anni).

Senza questo strumento, o un altro memory scanner, non andrete proprio da nessuna parte, quindi vi consiglio vivamente di studiare con attenzione questo strumento, e vi consiglio QUESTA guida molto interezzante



Il download lo trovate QUA.

PROTEZIONE DEL CODICE

Prima di passare alla vera e propria creazione di un hack dovete sapere che il più delle volte, purtroppo, non è per niente facile accedere al codice assembly del programma per colpa di software di protezione contro i crackers [

] chiamati packer. I packer più usati sono i packer commerciali anche perché gli unici che hanno queste manie di protezionismo verso i propri software, preferiscono comprare una protezione decente più tosto che usare protezioni gratuite e ridicole come UPX. I packer hanno varie tecniche di protezione, tra cui la più semplice anche da capire è quella di code injection. Cioè il packer va ad iniettare nel programma una porzione di codice che gli permette di criptare i dati del programma. Questo nella più banale delle ipotesi, ma ci sono quelli che vanno a virtualizzare intere sezioni del programma dando al reverser cracker per focaccia (il mio umorismo non ha limite.. se umorismo lo si può chiamare LOL).

MANUAL UNPACKING

Dopo questi sprazzi di ironia, arriviamo al metodo di risoluzione del problema del packing, che incombe spesso nella creazione di hacks particolari che richiedono l'accesso al codice asm del programma.

Per accedere quindi alla memoria dovremo **unpackare** il nostro programma/gioco, e non esistono programmi che ce lo fanno per noi, quindi dovremo farlo manualmente. Ovviamente non posso spiegarvi qui il manual unpacking, ma vi ho già dato un link dove viene spiegato molto bene, il come procedere con protezioni di medio/basso livello. Per quelle molto complicate, spesso, una volta riconosciuto il packer possiamo cercare una guida sul web che ovviamente ci darà informazioni molto utili ma per niente leggibili da un newbie. Quindi senza quella guida ed un po' di pratica non andrete da nessuna parte. La guida l'ho già postata, ma lo rifaccio per chi non avesse avuto il tempo/voglia di andare a visitare tutti i link.

Potete trovarla QUA.

IN COSA CONSISTE CREARE UN HACK

Sicuramente, la prima domanda che uno si deve fare è: “In che cosa consiste realmente la creazione di un hack?”

Detto nel gergo informatico, creare un hack è l’equivalente di trovare un puntatore statico in un programma. Questa definizione la capirete più avanti, ma vi posso anticipare che la difficoltà nella ricerca di questo puntatore dipende dalla protezione che ha il programma.

Detto in parole più povere creare un hack vuol dire creare un software in grado di modificare un determinato valore che è stato memorizzato dal programma nella memoria RAM. Come abbiamo già visto, tutte le informazioni, le scritte, i numeri ecc.. sono memorizzate nella RAM e quindi sono accessibili grazie al fatto che questa gli affida un numero univoco chiamato address; quindi ogni address ha un valore, che se modificato può dare effetti molto vantaggiosi, quali speed hack, combo hack…

Adesso, che avete compreso bene cos’è un hack, possiamo procedere con la sua creazione, e l’esposizione dei problemi legati a quest’ultima.

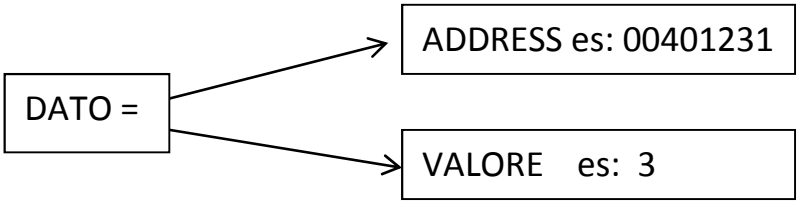
CREAZIONE TEORICA DI UN HACK (Metin2)

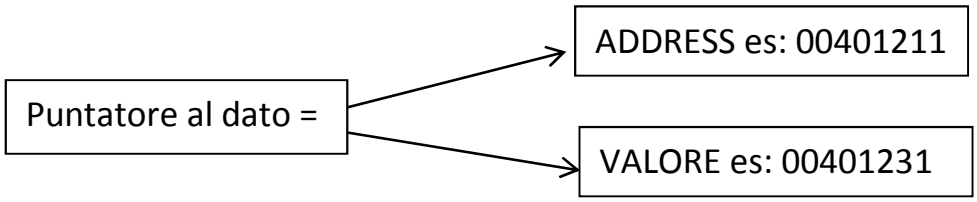
Passiamo adesso alla creazione teorica di un hack, prendendo come riferimento un gioco molto famoso e giocato: Metin2.

Una volta unpackato il gioco procedura che, oltre ad essere molto complessa, è anche molto lunga, ma potete trovare qualche cosa sul web che, per non rendere troppo "illegale" la mia guida, non vi linkerò. Vi posso dire che per quanto mi ricorda io, Metin2 è packato da HackShield, uno dei più buoni packer esistenti, quindi non sarà un'impresa facile.

Passato questo grande ostacolo, si può passare all'analisi del codice e quindi allo sviluppo di un hack. Ipotizziamo di aver bisogno di creare un hack per il teletrasporto. Io personalmente non ho mai creato un hack per Metin2, ma vi posso dire, in linea teorica, come bisognerebbe procedere.

Ipotizziamo di essere nelle coordinate (x,y) 3 , 2. Se non sbaglio in alto a destra nella schermata di Metin2 possiamo vedere le nostre coordinate, quindi visualizzeremo appunto 3 e 2. Sicuramente il metodo migliore per analizzare la memoria è usare un memory scanner bello e potente come cheat engine. Questo buonissimo programma ci permette di fare un scan della memoria a runtime, quindi ci permetterà di trovare l'address delle coordinate x e y. C'è però un grandissimo problema chiamato **DMA** (**D**ynamic **M**ovement **A**ddress) che consiste nel fatto che la RAM, come ho già detto, è una memoria ad accesso casuale, quindi l'address che avremo trovato, non ci servirà proprio ad un bel niente, perché quando riavvieremo il gioco questo sarà cambiato. C'è però un punto a nostro favore, cioè i puntatori. I puntatori sono delle variabili che hanno un indirizzo ed un valore, come tutte le altre variabili, ma come valore hanno un altro indirizzo. Quindi se l'address di x (che ha valore 3) è 00401231, ci sarà un puntatore con address (per esempio) 00401211 che avrà come valore 00401231. Abbastanza chiaro? Spero di sì, anche se so che non è una cosa semplice, ma chi ha studiato linguaggi come il C, dovrebbe aver capito; chi non l'ha fatto, deve schematizzarsi il fatto che:





Se mi avete seguito fin qui, possiamo introdurre il problema “alla seconda”, cioè i multi puntatori, che com’è facile intuire sono i puntatori ai puntatori.

Vi starete adesso chiedendo: “Bene ma che me ne faccio dei puntatori?” Esistono 2 tipi di puntatori: quelli dinamici e quelli statici ovviamente. Quindi, noi dovremo cercare i puntatori statici, e spesso quando si tratta di puntatori multi livello, è dura trovarli, e questo cambia da programma a programma (se non vado errato il massimo di livelli di puntatori è 4).

Per fortuna, il santo cheat engine ha un sistema di rilevamento dei puntatori che funziona tramite una sorta di bruteforce, cioè controlla tutti gli indirizzi possibili e vede se contengono il valore del nostro address. Il santo cheat engine però ogni tanto non funziona, infatti il “pointer scanner” oltre a trovare un numero infinito di puntatori, ogni tanto ci da anche dei puntatori sbagliati, per questo bisogna passare al metodo tramite debugging, che è ben illustrato nella guida che ho postato nel capito su cheat engine. Chiarito il come superare il problema del DMA, possiamo passare alla creazione dell’hack, la quale può venir fatta tramite il linguaggio di programmazione che preferite, basti sapere il come si modifica la memoria. I linguaggi più utilizzati sono il visual basic, l’autoIT e il C++. I primi 2 perché facili da usare e dalla possibilità di creare interfacce grafiche intuitive in poco tempo, il terzo per la sua professionalità e ampiezza di possibilità.

GLI HACKS IMPOSSIBILI, PERCHE’?

Sicuramente vi sarete chiesti il perché non si trovano gli hacks degli yang o degli oggetti, quindi faccio questa piccola parentesi per chiarirvi questo dubbio.

Ovviamente per quanto riguarda la velocità del personaggio, la velocità di combo, il teletrasporto e co, sono modificabili perché hanno valori presenti sulla nostra RAM, e quindi visibili sul nostro schermo, direttamente o indirettamente. Gli yang, gli

oggetti e tutti gli hacks impossibili invece, sono gestiti solo apparentemente dalla nostra RAM, perché ovviamente anche gli yang, essendo caratteri ASCII, sono convertibili in binario e a loro volta devono essere memorizzati in memoria, quindi l'unico risultato che potremmo ottenere sarebbe quello di riuscire a vedere sul nostro schermo 999.999.999.999 yang. Solamente però sul nostro schermo, perché i nostri yang reali, non sono gestibili tramite il nostro client, perché ricordiamo che tutte le nostre informazioni sono proprio sul server di Metin2. Quindi questa regola degli hacks impossibili è valida solo ed esclusivamente sui giochi che usurfuiscono di un server per memorizzare le informazioni e che quindi utilizzano la RAM come memoria momentanea.

Se vi state chiedendo:" ma allora io dovrei perdere tutti i dati al riavvio del computer quando gioco a Skyrim!", vi picchio, perché ovviamente quei genietti dei programmatori hanno inventato il salvataggio LOL.

Benissimo, adesso che abbiamo chiarito anche questo ultimo punto, dovete solo pensare a creare il vostro hack tramite l'aiuto di un qualcosa chiamato cervello.

CONCLUSIONI

Spero siate tutti consapevoli del fatto che usciti da questa guida, avete ancora molta strada da fare per arrivare a creare un hack distribuibile, ma sicuramente avete fatto un grande passo avanti.

Questa guida è stata principalmente pensata allo scopo, non solo di insegnarvi alcune cose, ma di funzionare come fonte di indirizzamento verso la strada che vi porterà poi alla vera e propria creazione di un hack, tramite una spiegazione prettamente teorica della sua creazione, preceduta da tutte le conoscenze basilari ed il reindirizzamento ad esse.

Spero di essere stato il più chiaro possibile, e sarò felice di rispondere, tramite email, a chiunque voglia capire meglio oppure abbia delle domande.