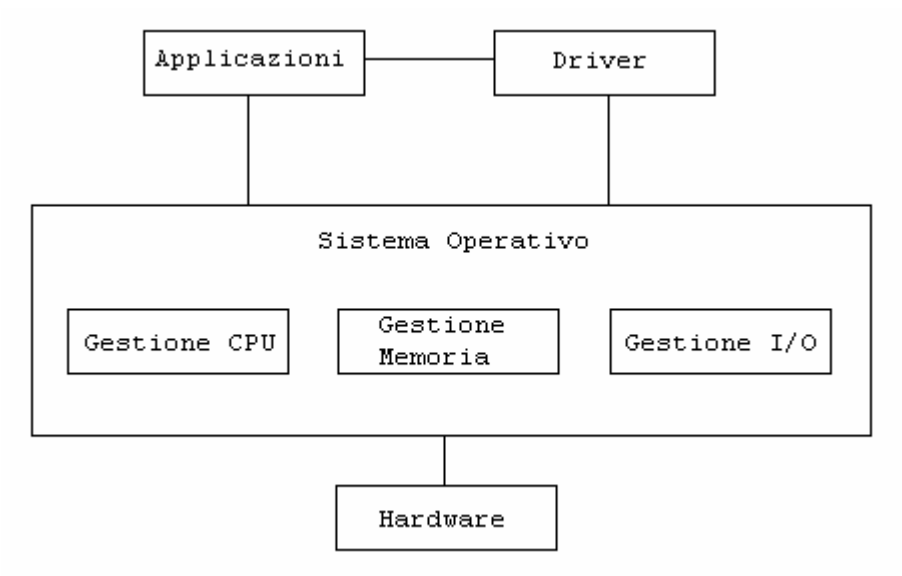
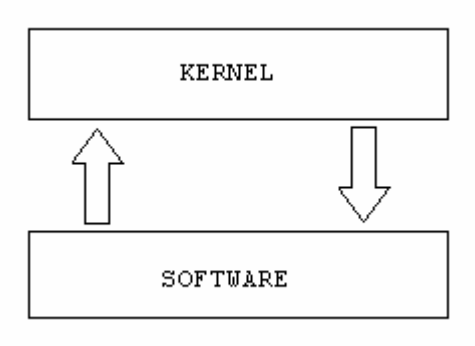
Analysis and Working of a Rootkit in the Operative System

0x100 - Il Sistema Operativo

Per comprendere il funzionamento di un rootkit è necessario partire dalle basi, infatti bisogna avere la piena coscienza di come lavora un Sistema Operativo. Farò quindi una brevissima e basilare introduzione. Un SO od OS (Operative System) controlla le varie parti di un computer: disco fisso, memoria, strumenti di input (tastiera, mouse ecc) e di output. Un sistema operativo è diviso in diverse componenti come il kernel, il gestore di file di sistema, un gestore di memoria virtuale, uno scheduler uno spooler e un' interfaccia utente (ad esempio Gui o Shell). Un sistema operativo quindi deve determinare i programmi che utilizzano la CPU, quelli che sono caricati in memoria, e controlla i vari input e manda i segnali di output. I più vecchi sistemi operativi come MS-DOS o CP/M-80 potevano lanciare solo un programma per volta mentre gli OS più moderni come Winzozz, Mac o Linux possono eseguire contemporaneamente molte applicazioni.



Esistono poi vari tipi di applicazioni. Ad esempio i Device Drivers sono programmi che indicano all'OS come gestire i componenti di hardware esterno. Mentre altri programmi come giochi, database, applicazioni vengono isolati dal sistema operativo che salva la informazioni sull'hard disk. Una componente fondamentale di un OS come abbiamo citato prima è il Kernel.



Un kernel tradizionale, detto monolitico (differente da un microkernel e dai kernel modulari) integra dentro di se molte componenti come la gestione della memoria, lo scheduler e i gestori di file system, oltre ai driver necessari per il controllo di tutte le periferiche collegate. Questo tipo di kernel ha come contro sicuramente la complessità sia di progettazione sia di manutenzione, ma di sicuro come pro la velocità, la funzionalità e l'efficienza. Questa velocissima spiegazione per sommi capi serviva appunto per introdurvi l'argomento.

0x200 – Rootkit?

Un rootkit (o ghostware) è un software o più precisamente una tecnologia capace di rendersi completamente invisibile su un sistema operativo. Bisogna specificare certamente che la tecnologia rootkit in se non è malevola poichè anzi aiuta spesso a rendere più efficenti applicazioni e sistemi operativi stessi. Ma i problemi occorrono quando questa tecnologia viene usata per altri scopi come per occultare trojans, backdoor, virus, e compiere azioni all'insaputa dell'utente. Un rootkit infatti non solo è invisibile all'occhio, alle ricerche e al task manager ma è completamente invisibile anche a qualsiasi tipo di antivirus e software in grado di rilevare i problemi in un sistema operativo. Ed è proprio questa invisibilità che è seriamente preoccupante perché una volta che un rootkit è stato inserito diventa praticamente impossibile eliminarlo se non formattando il disco e reinstallando l'OS. Altrettanto preoccupante è l'utilizzo di questa tecnologia ai fini di programmi commerciali. Infatti anche se utilizzati per rendere migliore e più efficiente un programma un rootkit indebolisce il sistema e lo rende molto più vulnerabile ad attacchi esterni di altri worms e trojans. Famoso di questo genere è sicuramente il caso Sony. Di seguito un esempio di rootkit lanciato dal dos. Come potete osservare è una vera e propria console che permette ogni tipo di azione sul computer colpito dal nascondere files e cartelle, a loggare la tastiera fino a far crashare la macchina.

```
Win2K Rootkit by the team rootkit.com
Version 0.4 Alpha
-------------------------------------------------
command           description

ps                show proclist
help              this data
buffertest        debug output
hidedir           hide prefixed file/dir
hideproc          hide prefixed processes
debugint          <BSOD>fire int3
```

```
sniff keys          toggle keyboard sniffer
echo <string>       echo the given string

*<BSOD> means Blue Screen of Death if a kernel
debugger is not present!
*'prefixed' means the process or filename starts
with letters '_root_'.

"sniffkeys
sniffkeys
keyboard sniffing now OFF
------------------------------------------------
--letmein--dir--
```

Entrando a questo punto maggiormente nei dettagli andiamo ad esaminare come si compone un rootkit. Esistono due tipi fondamentali di rootkit: Userspace rootkits che consistono in versioni modificate di file eseguibili, librerie o file di configurazione e Kernelspace rootkits che agiscono direttamente nella memoria del kernel, modificandone il funzionamento. Parlando in modo generico bisogna evidenziare naturalmente che un rootkit ha bisogno di una base di lancio per essere eseguito su un sistema all'insaputa. Questa è chiamata "payload" ed è quella situazione che consente di eseguire il codice: un bug, un esegubile, un'allegato, insomma qualunque cosa che possa lanciare il rootkit. A questo punto esaminiamo il rootkit vero e proprio: un insieme di strumenti (come abbiamo anche visto sopra) che consentono di monitorare e ed eseguira azioni di ogni tipo; ad esempio solitamente contiene le versioni trojan delle utility di monitoraggio del sistema, uno sniffer e sistemi per la pulizia dei logs. Quando un rootkit infetta un sistema (specialmente su Unix) spesso procede alla sostituzione di alcuni comandi molto utilizzati dagli utenti root, come comandi di monitoraggio, cosicchè quando un utente esegue quel comando il rootkit fa credere di averlo eseguito ma nel mentre svolge attività invisibili all'ignaro user; questa funzione sarà approfondita più avanti.

0x300 – Un po' di storia

I rootkit sono presenti da anni, forse, dicono alcuni, addirittura da più tempo dei worms e dei virus. Il termine "rootkit" è difatti una dizione vecchiotta che risale agli anni 90 quando i rootkit si iniziarono a diffondere e i sistemi utilizzati erano quelli Unix. Il termine è l'unione di due parole:

- Root - in ambiente Unix indica l'utente amministratore con i maggiori privilegi. In Windows è appunto l'amministratore.
- Kit – letteralmente "equipaggiamento" o "attrezzatura" ovvero quell'insieme di strumenti utili appunto al root.

Inizialmente gli unici tipi conosciuti erano rootkit destinati all'unico scopo di mascherare intrusioni compiute da remoto sui vecchi sistemi Unix, rendendo invisibili applicazioni come trojans o backdoor che l'intruso utilizzava. Alla fine degli anni '90 si inizia a intuire la reale pericolosità di questa tecnologia, si iniziarono a pubblicare articoli e l'informazione iniziò a girare. Ciò che infatti rendeva e rende ancora i rootkit particolarmente pericolosi è il modo con cui sono stati creati per essere in continua evoluzione, per essere invisibili a

qualsiasi antivirus e per essere così complessi da evitare ogni rilevamento.

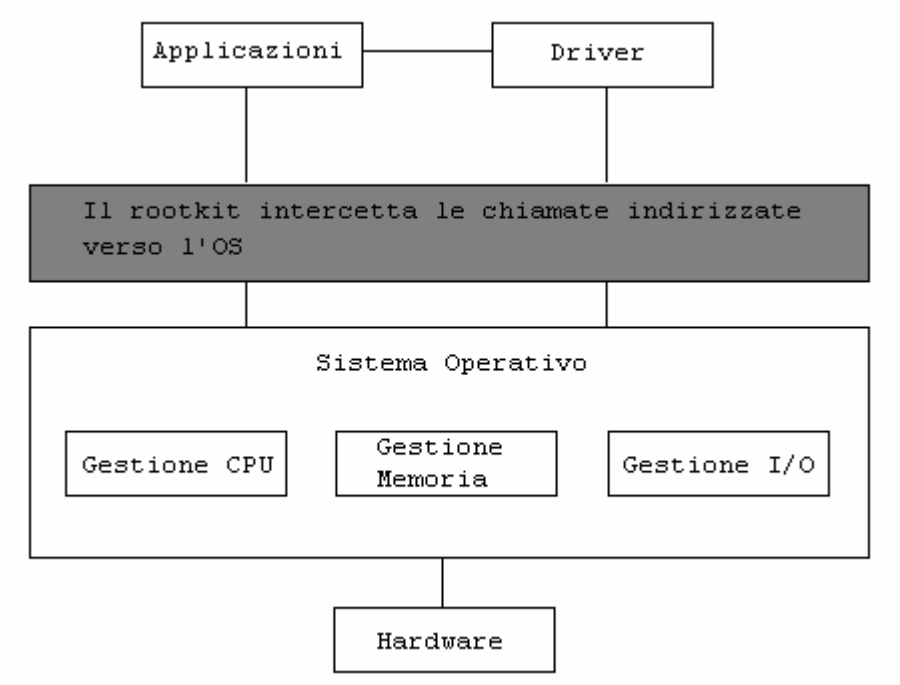
0x400 – Funzionamento Generale
In questo paragrafo tenterò di spiegare e mostrare le varie modalità di azione e tecniche che utilizzano di solito i rootkit userspace. Infatti come spiegato già in precedenza i rootkit non agiscono come virus a senso unico atti a compiere una sola e unica azione bensì il loro potenziale è per certi aspetti sicuramente molto più elevato. Le tecniche saranno spiegate solo teoricamente.

0x410 – Registry DLL Injection
Questa tecnica è una tecnica utilizzabile solo ed esclusivamente sui sistemi Windows. Infatti questi ultimi sono sistemi composti da un'insieme di files chiamati DLL (Dynamic Link Library) il cui scopo è interagire tra di loro per far funzionare il sistema operativo. Quando infatti per esempio un programma viene installato su un sistema Windows viene automaticamente aggiornato il Registro di Sistema che funge da database per questo programma che ogni volta che viene avviato controlla il Registro alla ricerca di ulteriori DLL utili al suo funzionamento. La tecnica del Registry DLL Injection dunque, che ho illustrato brevemente sopra, non fa altro che aggiungere un file binario ad un file DLL legittimo, senza toccare l'orginale. A questo punto viene modificato anche il Registro in modo che i tentativi di utilizzo delle DLL originali vengono redirettati su quelli infetti. Ma poiché il file DLL originale non è stato modificato la presenza del rootkit non viene scoperta. In questo modo quindi un eventuale rootkit può fare eseguire a suo piacimento azioni alla completa insaputa dell'utente root. Se quindi l'admin non si accorge che ci sono file DLL infetti o che la chiave del Registro è stata modificata per puntare su quei files il rootkit potrà continuare lavorare senza alcun problema.

0x420 – Sostituzione Files
Una cosa molto simile alla DLL Injection viene effettuata sempre dai rootkit in entrambi i sistemi operativi Unix o Windows. Infatti un'altra tecnica utilizzata dai rootkit è quella di sostituire i file binari di alcuni programmi dell'untente root con file binari appositi infetti. L'effetto provocato è che quando l'amministratore prova ad utilizzare questi programmi il tutto sembrerà funzionare al meglio ma in realtà sotto sotto un rootkit starà lavorando e starà eseguendo azioni nella più totale invisibilità. Il reale problema di questa tecnica che non sussiste invece nella DLL Injection è che ogni file contiene due elementi che sono la data e l'ora di creazione che vengono quindi modificati quando un rootkit edita i file binari. Infatti esistono programmi di controllo denominati "checksum" che monitorano questi cambiamenti e li registrano. Se l'amministratore di sistema non ha conservato i vecchi valori dei chacksum il programma di controllo dell'integrità dei files non noterà alcuna differenza. Con un po' di fatica poi, si può tentare di fare in modo che le versioni infette abbiano la stessa dimensione dei files originali, così riuscendo a modificare anche la data e l'ora i file di checksum non registreranno alcun cambiamento. Ancora più problematica è questa tecnica nel caso in cui ci siano checksum crittografati in algoritmi come md5 o sha-1, in questo caso infatti modificare i valori di checksum per un rootkit diverebbe davvero complicato se non quasi impossibile. Ultimamente sono diffuse però varie tecniche sulla creazione

di checksum uguali, ma con contenuto differente. Consigliabili sono sicuramente la consultazione di www.doxpara.com/md5_someday.pdf
Più dettagliatamente i comandi che di solito vengono sostituiti (specialmente su Unix) sono: find (ricerca gruppi di files), ls (elenca i contenuti della directory), netstat (mostra lo stato di rete), ps (i processi in esecuzione), who (utenti connessi), e w (utilizzo del sistema, logs azioni compiute). Dato che quasi tutti i rootkit imitano i valori di checksum dei file che rimpiazzano, tenere traccia unicamente di queste informazioni non è sufficiente a determinare la presenza di un rootkit.

0x430 – Hooking
Per svolgere certe operazioni che coinvolgono il sistema operativo, i programmi, hanno bisogno di un modo per comunicare con l'OS: esistono dunque delle librerie di funzioni che sono chiamate API (Application Programming Interface) che tutti i programmi possono utilizzare per eseguire i propri comandi, in particolare esistono alcune di queste funzioni specialmente utili ai programmatori che si chiamano "hook" che consentono di conoscere il funzionamento interno di un sistema operativo. Infatti queste hook monitorando e conoscendo il sistema consentono ad esempio di scrivere programmi di diagnostica o scansione, ma alla stessa maniera possono essere per l'appunto utilizzati anche in modo opposto dai rootkit ovvero per modificare il funzionamento del sistema operativo. Questa tecnica utilizzata dai rootkit è denominata "hooking".



Come funziona un'attività di hooking? I rootkit intercettano le chiamate alle funzioni eseguite dalle applicazioni sul sistema operativo. Il concetto di base è molto semplice. Un esempio banale, come fa un rootkit tramite l'hooking ad essere invisibile? Quando viene richiamata la funzione per mostrare l'elenco dei processi in esecuzione il rootkit si aggancia alla chiamata e sostituisce la chiamata con un'altra che mostra i processi senza il rootkit. Chiaramente detto così è semplice, poi non è così immediata la cosa. In sostanza recapita la stessa chiamata ma

leggermente modificata, così ne il mittente (il programma) ne il destinario (il sistema operativo) non si accorgono di nulla. Esistono due tipi fondamentali di hooking: locale e globale. Il primo intercetta chiamate mandate da un determinato programma verso una funzione. L'altra intercetta le chiamate a funzione provenienti da qualsiasi programma in esecuzione al momento. Anche in questo caso come per le DLL injections e le sostituzioni di files il rilevamento è molto difficile. Consigliabili sono però due tools come AntiHook (www.infoprocess.com.au) o ProcessGuard (www.diamondcs.com.au/processguard) che però devono essere installati prima di un eventuale infezione di rootkit altrimenti questo potrebbe modificare le chiamate anche di questi programmi.

0x440 – Lib_Call Hijacking

Invece di intervenire sui singoli binari dei vari comandi e sui vari eseguibili, alcuni rootkit utilizzano una tecnica chiamata appunto Lib_Call Hijacking che come dice già il nome interviene sulla LibC (la libreria C standard dei sistemi unix e linux). Questa infatti contiene i riferimenti alle chiamate delle funzioni di sistema. Il rootkit è in grado di modificare il funzionamento della libreria in modo che esegua prima il codice malevolo della chiamata alla funzione reale. Tutto ciò si può fare grazie a ld.so che è una componente della libreria ovvero il "loader delle librerie dinamiche". Tramite questa infatti è possibile far caricare una libreria propria che ridefinisce i vari comandi e le varie chiamate. Per farlo può essere usata una sintassi simile:

```
#export LD_PRELOAD=
```

Oppure si può editare in questo punto

```
/etc/ld.so.preload
```

0x450 – Sniffing

Nei rootkit è spesso presente uno sniffer. Uno sniffer non fa altro che sniffare (intercettare) i dati che transitano in una rete: questo significa anche ogni genere di password e dato sensibile. Solitamente un computer preleva solamente il traffico che è indirizzato direttamente a lui, proprio per arginare questo problema e far funzionare a dovere uno sniffer la scheda di rete del computer infettato viene modificata e impostata in modalità "promiscua" questo consente di ascoltare tutti i dati in transito nella rete. Naturalmente si è tentato di fermare anche quest'attività degli sniffer, infatti possono essere create delle reti dette "commutate". In una rete commutata i dati non passano direttamente da un computer ad un altro che verifica da solo se è il destinatario dei dati, ma bensì il traffico fa prima tappa da uno switch che poi smista i vari dati verso i computer destinatari in questo modo lo sniffer non potrebbe agire perché riceverebbe solo i dati del computer infettato e nient'altro. Ma anche contro questa rete è stato trovato un rimedio ovvero lo "spoofing". Ma per tutte queste tecniche ci sono decine di migliaia di altre guide sul web che è inutile stare a trattarle qui.

0x450 – Pulire i Logs

Nei rootkit è inoltre presente solitamente un'utility che consente di sistemare a dovere i logs. I file di log infatti loggano (registrano) tutto ciò che è stato fatto sul computer come ad esempio chi l'ha utilizzato, per quanto tempo, le azioni eseguite. Grazie a questi files un amministratore potrebbe dunque notare facilmente la presenza di un

rootkit o di un' intruso. Un rootkit di conseguenza ricerca i file di logs che hanno registrato la loro presenza su un computer. Alcuni rootkit cancellano totalmente i logs e bloccano la funzione di logging. Questo di sicuro permette di non essere rintracciati, ma mostra chiaramente la presenza di qualcuno sul computer quindi è utile solo per una botta e fuga, e non se l'intruso ha intenzione di utilizzare il pc infetto a lungo. Altri rootkit invece consentono proprio l'editing dei logs, che non è una cosa semplice perché anche una sola imperfezione sarebbe una prova. Tra le informazioni che compaiono in un log e che solitamente i rootkit vanno ad editare sono queste:

- IP della macchina che ha eseguito una richiesta sul pc.
- Nome utente dell'account utilizzato.
- Data e ora.
- Comandi e richieste eseguite.
- Codice di stato HTTP inviato dal pc vittima.
- Numero di byte trasferiti.

Per proteggersi dalla modifica dei logs si possono percorrere varie strade: stampare i logs come vengono generati ad esempio permette di confrontare se qualcosa è stato modificato. Ancora è possibile creare copie diverse dei files di logs così un rootkit modificherà quella nella posizione nota, ma le altre resteranno protette e utili per un confronto. Segnalo di seguito alcuni siti molto utili all'approfondimento dell' argomento "logs" perché io l'ho trattato molto brevemente e superficialmente: www.analog.cx, www.mrunix.net/webalizer.

0x460– Tecniche Avanzate di Occultamento

Tre sono le recenti tecniche utilizzate dai rootkit specialmente per occultarsi degne di nota.

a) Utilizzo di file_ops e inode_ops del filesystem /proc invece delle sys_call.
b) Creazione un un array di puntatori alle sys_call senza modificare il simbolo esportato.
c) Patch della sys_call ma non del puntatore.

Vediamole ora più nei dettagli utilizzando uno scritto di FuSyS.

a) Tramite questo metodo le chiamate di sistema risultano inalterate. Il controllo è comunque banale, dovendo portare il confronto degli indirizzi di memoria dalle sys_call alle file_ops e inode_ops.

```
old_readdir_root = proc_root.FILE_OPS->readdir;
old_lookup_root = proc_root.INODE_OPS->lookup;
proc_root.FILE_OPS->readdir = &new_readdir_root;
proc_root.INODE_OPS->lookup = &new_lookup_root;
```

b) Con questo metodo il semplice controllo dei puntatori non è più sufficiente. Infatti la maggior parte dei tool di controllo riceve il simbolo dell'array delle chiamate attraverso la funzione query_module(). Tale simbolo non viene modificato: viene sostituito invece l'indirizzo dell'array nella funzione system_call() del kernel.

```
void *hacked_sys_call_table;
```

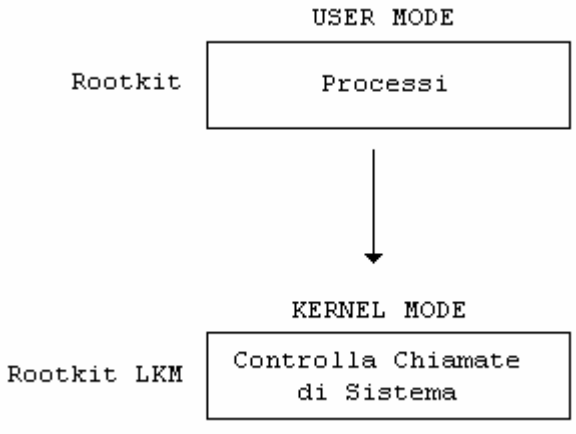
```
hacked_sys_call_table=kmalloc(256*sizeof(long int), GFP_KERNEL);
memcpy(hacked_sys_call_table, sys_call_table, 256*sizeof(long int));

(int)*((int*)ptr) =(int) hacked_sys_call_table;
```

In questo esempio “ptr” punta all’indirizzo originale della tabella delle chiamate nella funzione system_call().

c) Con questo metodo è necessario controllare il codice della system_call e confrontarlo con un hash o con una porzione della chiamata. Così facendo, chiamate modificate, ma con indirizzo normale, possono essere identificate

0x500 – Rootkit LKM

Il normale lavoro dei rootkit di sostituzione e infezione dei files però è possibile da arginare ad esempio memorizzando copie dei programmi. In questo caso si dispone anche di copie pulite e quindi il rootkit diventa inutile. Ma l’evoluzione continua di queste temibili minacce ha portato alla creazione di un tipo pericolosissimo di rootkit di tipo kernelspace: Rootkit Loadable Kernel Module. Questo tipo di rootkit sfrutta i moduli caricabili del kernel solitamente presenti nei sistemi Unix e Linux. Ecco cosa è successo: una volta per modificare il kernel di linux bisognava modificare e ricompilare nuovamente tutto il codice sorgente dell’OS. Sono stati però integrati i LKM ovvero appunto del moduli caricabile del Kernel che consentono di aggiungere comandi senza stare a ricompilare tutto il sorgente che è un’operazione complicata nel quale possono sorgere errori. Cosa fanno dunque i Rootkit LKM? Semplicemente intervengono direttamente sul kernel e invece di sostituire i programmi ne alterano il funzionamento dal kernel. Così analizzando gli eseguibili sembrano perfettamente in regola, ma quando vengono eseguiti il modulo LKM lo intercetterà ed eseguirà il proprio. In questo stesso modo infatti si possono dirottare anche la chiamate si sistema alle varie funzioni. Tra i rootkit LKM più noti ci sono sicuramente SuckIT, Knark, Adore, Rial, Rtkit e Tuxkit.



0x510 – Composizione del Rootkit LKM
Un rootkit LKM si può sostanzialmente dire che è composto da tre parti.

- Il blocco contenente i define e gli include dell'header del kernel.
- Il main contenente il codice.
- Il blocco che gestisce il comportamento del rootkit nel kernel contenente le funzioni init_module() per l'inserimento e cleanup_module() per la sua rimozione.

Qui sotto un esempio di come potrebbero presentarsi nel codice delle funzioni di init_module() e cleanup_module().

```
Int init_module(void)                              /*Eseguita al caricamento*/
{
old_chdir = sys_call_table[SYS_chdir]              /*L'indirizzo della
                                                     chiamata di sistema dalla
                                                     sys_call_table è salvata
                                                     al puntatore*/

sys_call_table[SYS_chdir]= (void *) my_chdir;/*La sys_call_table punta
                                               alla mia implementazione
                                               di chiamata*/
```

```
}
voidcleanup_module(void)                           /*Eseguita alla rimozione*/
{
sys_call_table[SYS_chdir] = old_chdir;             /*Ripristinato l'indirizzo
                                                     della sys_call_table*/
```

0x600 – Rootkit recenti
Qui di sotto presento una lista dei più recenti rootkit presentati su rootkit.com corredati da link per raggiungere la loro pagina di download. Fate molta attenzione, specialmente chi non ha una minima consoscenza in questo campo perché autoinfettarvi con un rootkit non credo sia la cosa migliore da fare.

http://www.rootkit.com/board_project_fused.php?did=proj5	Hacker Defender
http://www.rootkit.com/board_project_fused.php?did=proj6	HE4Hook
http://www.rootkit.com/board_project_fused.php?did=proj7	BASIC CLASS
http://www.rootkit.com/board_project_fused.php?did=proj9	Vanquish
http://www.rootkit.com/board_project_fused.php?did=proj11	NT Rootkit
http://www.rootkit.com/board_project_fused.php?did=proj12	FU
http://www.rootkit.com/board_project_fused.php?did=proj13	WinlogonHijack
http://www.rootkit.com/board_project_fused.php?did=proj14	klister
http://www.rootkit.com/board_project_fused.php?did=proj15	Patchfinder2
http://www.rootkit.com/board_project_fused.php?did=proj16	MyNetwork
http://www.rootkit.com/board_project_fused.php?did=proj17	MTDWin
http://www.rootkit.com/board_project_fused.php?did=proj18	NTFSHider
http://www.rootkit.com/board_project_fused.php?did=proj19	VideoCardKit
http://www.rootkit.com/board_project_fused.php?did=proj20	VICE
http://www.rootkit.com/board_project_fused.php?did=proj21	Klog
http://www.rootkit.com/board_project_fused.php?did=proj22	NtIllusion
http://www.rootkit.com/board_project_fused.php?did=proj23	AFX Rootkit
http://www.rootkit.com/board_project_fused.php?did=proj24	SInAR
http://www.rootkit.com/board_project_fused.php?did=proj27	Shadow Walker
http://www.rootkit.com/board_project_fused.php?did=proj28	BootRootkit
http://www.rootkit.com/board_project_fused.php?did=proj29	CHAZ – Nima Bagheri
http://www.rootkit.com/board_project_fused.php?did=proj30	Clandestin File System Driver
http://www.rootkit.com/board_project_fused.php?did=proj31	FUTo
http://www.rootkit.com/board_project_fused.php?did=proj32	Windows Memory Forensic Toolkit
http://www.rootkit.com/board_project_fused.php?did=proj33	RAIDE
http://www.rootkit.com/board_project_fused.php?did=proj34	BOOT KIT
http://www.rootkit.com/board_project_fused.php?did=proj36	BluePill
http://www.rootkit.com/board_project_fused.php?did=proj37	DEFRAG
http://www.rootkit.com/board_project_fused.php?did=proj38	Keyboard Hook
http://www.rootkit.com/board_project_fused.php?did=proj39	CheatEngine

0x700 - Difendersi

0x710 – Prevenire
Non esistono metodi fissi o programmi appositi di prevenzione. Certamente bisogna ricordare che la cosa più vulnerabile a questo mondo è la mente umana. Con questa frase voglio dire che bisogna essere furbi e attenti anche nello svolgere semplici attività come il navigare in internet poiché le insidie più pericolose si nascondono dietro delle stupidate. Nonostante tutto ecco qualche precauzione che si può adottare nel reale pericolo di infezione rootkit.

- Account Limitato: l’utilizzo di un’account limitato certamente non previene del tutto l’infezione di un rootkit ma molte volte è in

grado di fermare worms o virus che necessitano di particolari privilegi per annidarsi nel computer.

- Virtual Machine: eseguire i programmi su una virtual machine al massimo limiterà l'infezione alla macchina virtuale stessa ma non al reale sistema operativo installato nel computer vero e proprio.
- Sandbox: utilizzando una sandbox le applicazioni verrano eseguite in una zona protetta che preverrà ogni tipo di infezione al sistema in quanto si potrà decidere di non salvare i cambiamenti effettuati sull'OS.
- Hips: un HIPS (Host Intrusion Prevention System) è un software di protezione che analizza in tempo reale le azioni che un' applicazione sta effettuando sul sistema informando l'user.

0x720 - Correggere

Certamente se un computer è stato infettato da un rootkit non è semplice da sapere ma soprattutto da pulire. Il metodo più semplice è quello di formattare il pc e reinstallare il sistema operativo. Nonostante tutto molte volte si cerca di evitare questa drastica soluzione poiché porta alla perdita di parecchi dati, e anche copiare i dati dopo aver formattato comporta il rischio di trasportare anche il rootkit nel nuovo sistema operativo. Per evitare tutto ciò sono stati messi a punto alcuni software che tentano di scovare i rootkit nel caso questi ci siano: non esiste ancora un software davvero perfetto. Mentre tra gli antivirus comunque se un virus è conosciuto viene quasi sempre rilevato dagli antivirus e il vero problema sta nella continua creazione di nuovi virus, con un rootkit è diverso, e non esistono infatti software infallibili al 100%. Sicuramente sono stati fatti grossi passi in avanti, vediamo un po' il meglio della rete in questo campo.

- Strider GhostBuster (Microsoft)ancora in sviluppo.
  http://research.microsoft.com/rootkit/
- BlackLight (F-Secure) buon rilevatore di rootkit.
  http://www.f-secure.com/blacklight
- IceSword (Xfocus) ottimo rilevatore in cinese o giapponese.
  http://xfocus.net/tools/200505/1032.html
- Toolkit AntiRootkit vari.
  www.invisiblethings.org
  www.rootkit.nl
  www.chkrootkit.org
  www.sysinternals.com
- Torna utile visitare anche il sito www.rootkit.com

0x800 – Fonti
Scrivendo questa guida ho attinto da alcune fonti informazioni o esempi.

www.wikipedia.it
www.s0ftpj.org
ftp://ftp.ge.cnr.it/pub
www.ippari.unict.it
www.rootkit.com

0x900 – Conclusioni
Tutto ciò che ho spiegato in queste pagine è una panoramica generale del mondo dei rootkit. In questa guida ho cercato di mettere assieme tutte le informazioni frammentarie che ho imparato e ritrovato su questa tecnologia nel corso del tempo e spero vi serva anche a voi. Per un ulteriore approfondimento tecnico consiglio la presentazione di FuSyS (aka Matteo Falsetti) che trovate tra le fonti citate sopra. Ringrazio tutti. Al prossimo paper.

Saitek
Italian Net Raiders
www.saitek.altervista.org
www.netraiders.altervista.org
saitek_06@yahoo.it

"Apprendi e non smettere mai di farlo"