Virus vs. AntiVirus

di Bugli Fabrizio

2002-2003

Virus vs. AntiVirus

Come gia' visto, i virus sono una delle tante “piaghe” che affliggono gli utenti di computer.

Ogni volta che un nuovo virus inizia a diffondersi, sono necessarie **nuove firme** per il proprio antivirus (o nuove versioni!)

Ogni volta che un nuovo antivirus (con nuovi algoritmi per la detection o solamente con firme dettagliatamente aggiornate) entra in commercio, sono “necessari” nuovi virus per aggirare le protezioni dei suddetti.

... e' l'eterna lotta tra il bene e il male

Virus vs. AntiVirus

“La condizione necessaria per essere considerato un computer virus e' la capacita' di PRODURRE COPIE DI SE STESSO (non necessariamenteo repliche bytewise) e di INSINUARSI in reti di computer e/o files, aree di sistema e altri oggetti eseguibili. Inoltre le copie sono in grado di diffondersi ulteriormente”

http://www.kav.ch/avpve/entry/entry2.htm

Non e' **LA** definizione definitiva, cosa che mai si potra' dare.

Repliche non bytewise sono ad es. quelle dei *virus polimorfi*.
L'esempio piu' moderno di virus a grande capacita' di diffusione sono i *worms*.

Virus vs. AntiVirus

S'e' detto che non esiste un algoritmo perfetto per gli antivirus per proteggersi dai virus.

Allo stesso modo non esiste un algoritmo perfetto per i virus per nascondersi dagli antivirus.

Vediamo alcuni dei metodi che un virus adotta per evitare o solamente ritardare la notifica della propria presenza a un antivirus (sempre che l'UTENTE ne abbia installato uno..).

Stealth: Frodo.4096

All'inizio, prima che da software fatto ad hoc, era necessario nascondersi dall'utente e dai programmi.

FRODO.4096 e' il primo esempio “storico” di *stealth*.
Residente in memoria e lungo 4096 bytes, contamina sia eseguibili sia files di dati.
Un file infetto ha dimensione pari all'originale + 4096 bytes e data di ultima modifica aumentata di 100 anni. Questo fatto risulta tuttavia totalmente invisibile a DOS (e quindi all'utente e agli altri programmi, anche antivirus!).

Stealth: Frodo.4096

Caricato in memoria per la prima volta, intercetta le chiamate agli **interrupt 13h** e **21h**.

Gli interrupt software sono routines del BIOS residenti in una ROM (o fornite dal kernel del s.o. stesso) e servono al sistema operativo per compiere diverse operazioni. Affinche' il s.o. localizzi queste routines, viene caricata in RAM una tabella di puntatori al codice eseguibile di questi programmini di sistema.

In particolare 13h e' il gruppo di funzioni per l'accesso al disco, mentre 21h e' quello per le operazioni sui files.

Stealth: Frodo.4096

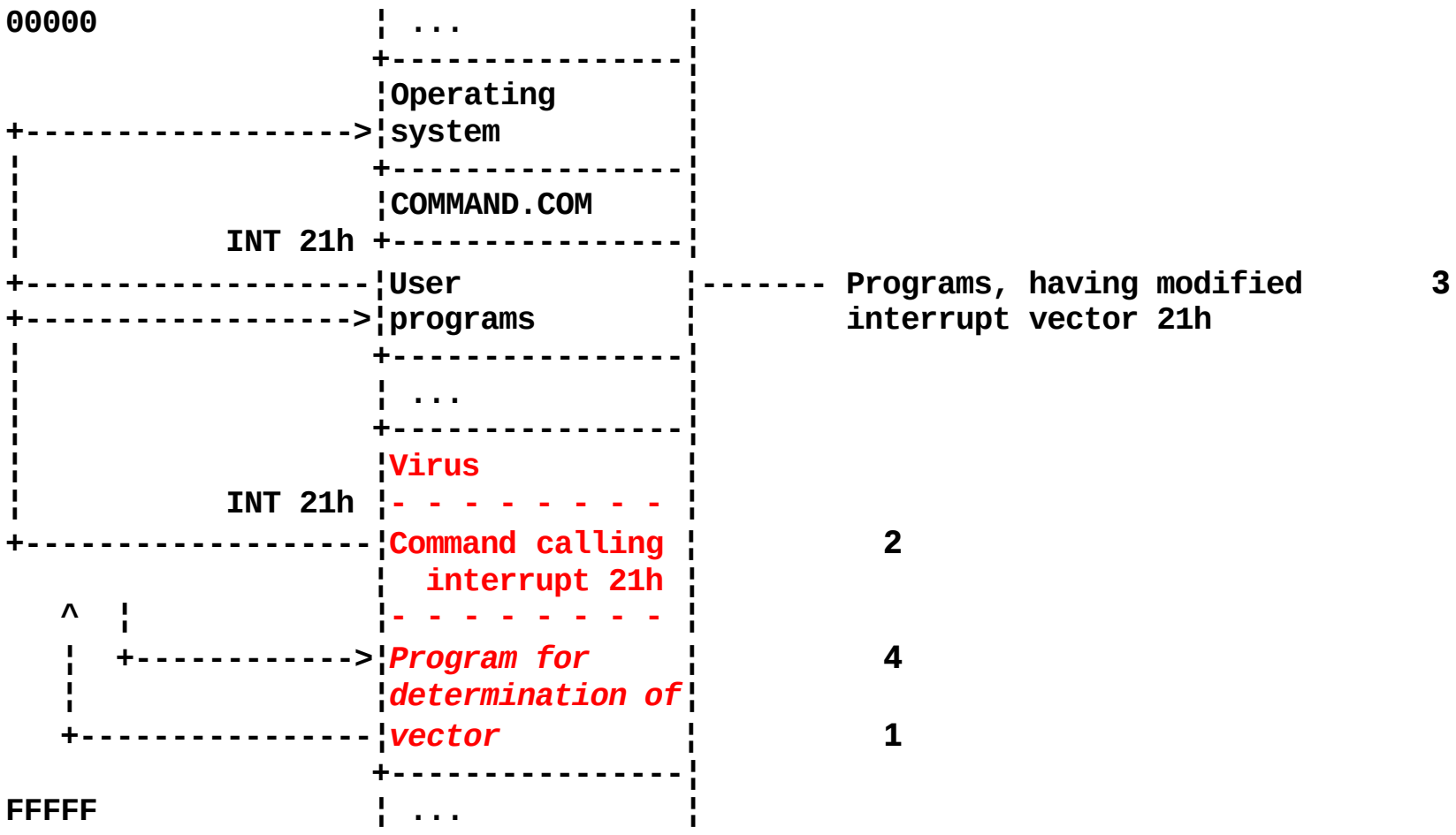
Sostituendo (alcuni dei) puntatori originali con puntatori a **PROPRIE ROUTINE**, il virus devia le chiamate del s.o. verso parti del suo codice, il piu' possibile piccole per “non dare nell'occhio”, e poi lascia che riprenda la normale esecuzione della routine corretta.

Stealth: Frodo.4096

Per localizzare l'indirizzo dei veri handler da attaccare, il virus modifica 01h (cioe' l'interrupt chiamato ad ogni istruzione processata dalla cpu) affinche' venga lanciata, dopo ogni istruzione, la sua subroutine per tracciare cio' che avviene nel sistema.

Inizia quindi l'esecuzione di una serie di programmi infetti che chiamano un INT 21h (o un altro handler di cui si vuole l'indirizzo), monitorando cosa succede nel sistema operativo. Alla fine di ogni istruzione, la subroutine maligna controlla se l'ultimo comando eseguito si trovava nell'area di memoria del sistema operativo. Se e' cosi', ferma il tracing mode e assume quell'indirizzo come la posizione dell'handler da modificare.

Stealth: Frodo.4096



Derivazione dallo Yankee Algorithm

Stealth: Frodo.4096 e DOS

Il virus esaminato e' in grado di controllare circa 20 funzioni dell'interrupt handler 21h, tra cui Read, Open, Close..

Questo gli permette di mascherarsi a utenti e programmi, poiche' essi vedono i file solo filtrati dal s.o. che ormai e' bacato.

Quando DOS cerca di accedere a un file infetto, il virus catcha la chiamata a read, restituendo la dimensione e la data ORIGINALI del file, e NON quelle modificate di 4096b e di 100 anni. La modifica avviene in modo dinamico: **in memoria c'e' un file corretto che su disco non esiste, perche' esiste solo il file corrotto**.

Inoltre c'e' anche l'autopreservazione: quando un file infetto viene aperto in scrittura, **il virus cura il file** (tramite ad esempio codici di Hamming) per evitare che parte del proprio codice venga intaccata. Al momento della chiamata a close, il TSR del virus lo ri-infetta.

Stealth: Come nascondersi da AntiVirus TSR?

Gli AV TSR funzionano esattamente come i virus: scansionano **dinamicamente** i file durante l'apertura.
Quando un file viene aperto dal sistema operativo, viene intercettata la chiamata e deviata a routines che scansionano il file.

Il virus appunto si preoccupera' di fornire all'antivirus un **handler falso**, che NON lancera' la scansione ma procedera' con il normale uso.

E' tutta questione di tempismo, non tanto su chi per primo carpisce le chiamate, quanto sul fatto che intanto e' il virus a partire per primo, poiche' **risiede nel settore d'avvio** del disco corrente.
Sempre ammesso che la macchina sia stata infettata in una sessione precedentemente senza protezioni.

Stealth: Soluzioni

I virus solamente Stealth sono comunque facilmente rimovibili.

Con un **database aggiornato** e' sufficiente arrivare ad uno stato del sistema con la ram pulita e poi andare in cerca del software maligno.

Cio' e' realizzabile ad es. bootando con un floppy pulito contenente l'antivirus scanner, onde evitare che il virus diventi memory resident e protegga i files o i boot sectors infetti dal riconoscimento.

Stealth: Varianti

Stealth Boot Viruses

Come detto i virus stealth TSR per DOS possono nascondersi facilmente agli antivirus (incapaci di svuotare la ram prima di partire) intercettando le chiamate di accesso ai file e le chiamate che interrogano lo stato del disco.
Per i semplici Boot: Int13 serve appunto per restituire dei valori falsamente corretti riguardo alla FAT. Int21 serve solamente per determinare quando attivare il proprio stealth, e cioe' ogni volta che si utilizza la funzione Int21 di esecuzione su un file.

Contro gli antivirus che accedono in maniera **diretta** al disco (come faceva Norton), il meccanismo qui sopra non era sufficiente.
E quindi era necessario, oltre a gestire gli INT, anche PULIRE REALMENTE il settore infetto per poi ri-infettarlo, dato che l'av avrebbe usato procedure **proprie** per l'accesso, senza chiedere il filtro del DOS.

Stealth: Varianti

Stealth File Viruses
Si comportano in maniera simile ai Boot, ma crescono in dimensione di codice proporzionalmente al numero di chiamate di sistema disponibili su un file.
Non solo la funzione d'esecuzione di Int21 e' necessaria, ma anche MOLTE altre, come l'apertura o la chiusura.
Analogamente a cio' che fanno i Boot virus con i settori d'avvio, **i File Virus devono nascondere la propria presenza nei files**.
(La stessa cosa che faceva pure Frodo per auto-preservarsi dalla accidentale cancellazione).

Con l'avvento di Win95 hanno dovuto “imparare” a trattare anche con le chiamate sui nomi lunghi.

Stealth: Varianti

Sempre Stealth ma piu' dannosi sono infine i virus che NON intercettano le chiamate del DOS tramite la RAM, ma quelli che **modificano direttamente il kernel** del sistema operativo in modo che le chiamate siano fatte “nativamente” verso le sue routines (ad es. *Beast.512* o *Dir_II*)
In questo modo non hanno praticamente bisogno di essere memory resident.

Allo stesso modo un boot floppy contenente il kernel (se e' boot floppy, lo contiene sicuramente) di DOS e' sufficiente per un avvio pulito del sistema.

Self-encryption

Il metodo di Stealth descritto e' piuttosto superficiale ed e' facile prendere contromisure e identificare i virus che lo adottano: infatti tutti aggiungono ai file **la stessa stringa** di codice.

L'approccio per raggiungere un nuovo tipo di invisibilita' e' quello di AUTO-CRITTARSI (self-encrypting), per decrittarsi successivamente.
Questi virus sono solitamente costituiti da due parti:

codice del virus cifrato + routine di decifratura.

Durante la diffusione, il codice viene decifrato e poi ricifrato in un altro file, magari con **diversa chiave**.
In questo modo l'antivirus per cercarli e' costretto a conoscere non solo le stringhe di definizione del virus, ma anche stringhe tipiche di routine di decifratura.

Polimorfici e mutanti

Finche' qualche stringa rimane fissa (la routine di decifratura), e' facile individuarli con le virus masks (le “firme” o “definizioni” dell'antivirus).
Da qui nasce l'“esigenza” di rendere totalmente variabile il codice del virus.

I virus polimorfi cercano di ovviare a questo.
Oltre a tutte le caratteristiche dei virus auto-cifranti (codice cifrato che si ricifra con diverse chiavi a ogni infezione tramite una routine), utilizzano un algoritmo che modifica in modo **casuale** la routine di decifratura, in modo che non si possa identificarla tramite semplici tecniche di scansione.

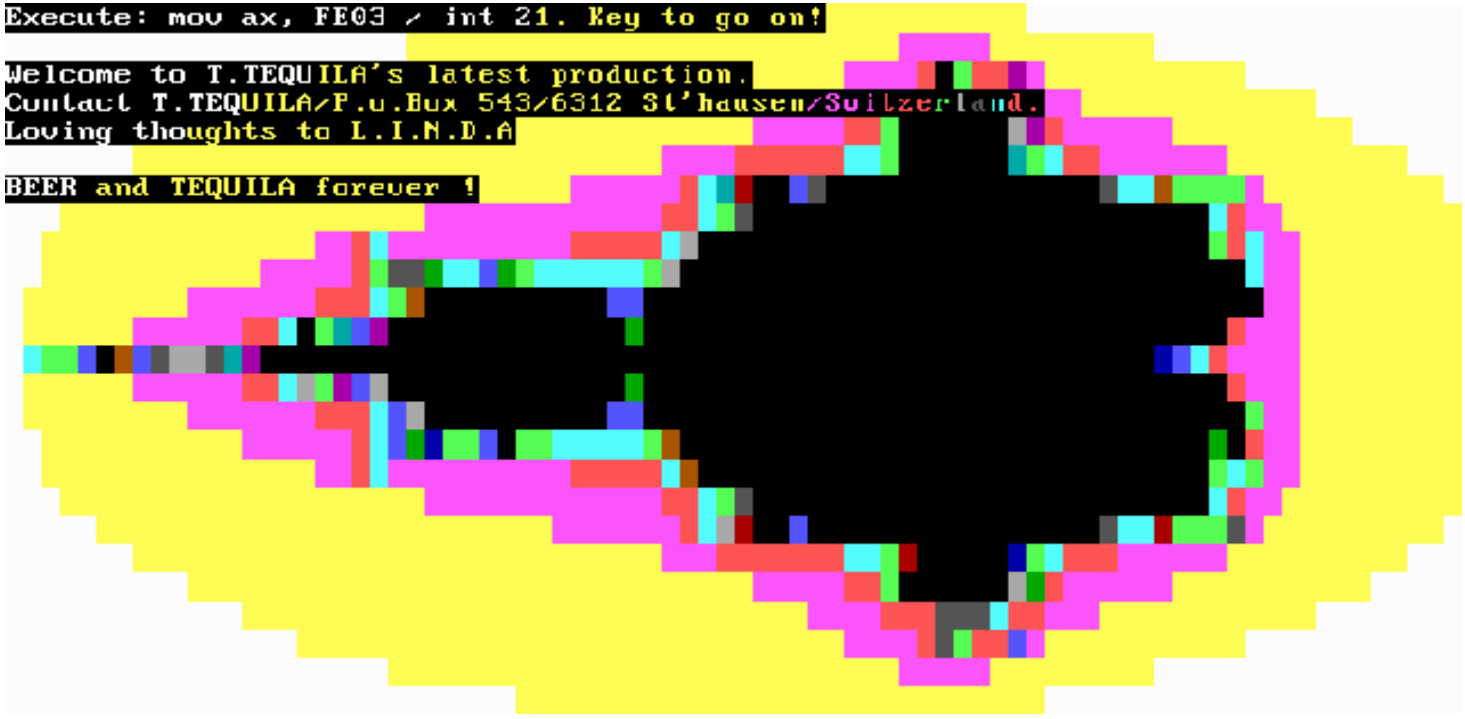
Polimorfici e mutanti

I primi esemplari (*Chamaleon* e *Tequila*) randomizzavano gli algoritmi per le operazioni di cifratura (come i self encrypting) ma anche quelli di decifratura (tipico dei polimorfi) in base a variabili piuttosto banali.

Il primo per esempio cifrava il corpo in base al timer di sistema (1000000h = 16.777.216 varianti) e il decryptor era scelto tra piu' di 3*(10^21) varianti, cioe' la grandezza dello stesso (39 bytes).

Tequila

Come per Frodo, l'autore di Tequila si era impegnato affinche' l'infezione di un file fosse il meno manifesta possibile.
Totalmente il contrario avveniva quando il codice veniva decifrato ed eseguito...



Mandelbrot Fractal Set

Polimorphic = Self-Encryption + Mutation Engine

Gli algoritmi “randomizzanti” visti sono piuttosto deboli, molto meglio automatizzare tutto!
Automatizzare anche per chi successivamente deve scrivere altri virus polimorfici.

Mutation Engine

Un virus polimorfico e' composto dal corpo eseguibile crittato e da un **generatore casuale di algoritmi di decifratura**.
Quando viene eseguito un programma infetto, viene scelto un metodo, con il quale si decifra il corpo. Quest'ultimo infetta un altro file e, portandosi dietro il mutation engine (lo scopo del virus e' che anche le sue copie siano capaci di replicarsi), genera random un nuovo algoritmo per cifrarsi all'interno della nuova vittima.

Mutation Engine

Esempi di MtE: *Dedicated*
(http://www.kav.ch/avpve/poly-gen/mte.stm)
Nella parte di decifratura, le operazioni SUB ADD XOR ROR ROL possono comparire un numero arbitrario di volte, senza un ordine.
La parte di cifratura con nuova chiave allo stesso modo presenta un numero non prevedibile di istruzioni assembler con diversi modi di indirizzamento.

Altri esempi di virus totalmente polimorfici (cioe' dove tra una istanza e l'altra non c'e' nemmeno un byte uguale): *Bootache, CivilWar, Crusher, Dudley, Fly, Freddy, Ginger, Grog, Haifa…*

Polymorphic

In sintesi: NON c'e' alcuna uguaglianza tra due file infetti dallo stesso virus, poiche' ogni volta si nasconde tramite un algoritmo che e' deciso RUNTIME.

A parte i polimorfici “imperfetti”, per cui alcuni bytes rimanevano costanti e tramite essi potevano essere individuati (le sequenze costanti in realta' sono talmente corte che necessitano di **ulteriori decifrature** per recuperare gli oggetti infetti), per gli altri la detection implica metodi speciali diversi dalla scansione tra cui l'**emulazione** dell'esecuzione del file infetto e complicati algoritmi matematici per il recupero del codice e dei dati del virus.

Ma se e' l'antivirus a girare gia' in emulazione di DOS? (cfr. Virtual Mode 86)

Polymorphic: Livelli di polimorfismo

Vengono catalogati in base alla complessita' del decryptor e alle maschere necessarie per individuarli..

- Livello 1: hanno un set finito di decifratori con codice costante, sono percio' oligomorfici, o imperfetti. Bastano maschere multiple.

- Livello 2: i decifratori istanziabili sono potenzialmente infiniti, ma usano un set finito di istruzioni. Si puo' pensare a maschere con wildcards.

Polymorphic: Livelli di polimorfismo

? Livello 3: nel set finito di istruzioni combinabili compaiono anche junk instructions. Possono causare falsi allarmi se si applicano maschere: infatti le junk instructions possono essere potenzialmente TUTTE le istruzioni del set i8086.

? Livello 4: il decryptor usa un sottoinsieme del set di istruzioni, variandone l'ordine, ma con un algoritmo finale fisso. Puo' essere individuato ad esempio da tecniche di **emulazione**, unite a leggi matematiche (o statistiche tipo analisi euristica).

Complicando a piacere quanto detto si arriva a livelli in cui e' possibile decifrare solo parti (o nessuna!) del corpo del virus.

Polymorphic Generators

Ovviamente nacquero anche tools in grado di trasformare un comune virus in un virus con caratteristiche polimorfiche. Si tratta di moduli OBJ, ai quali **basta linkare l'object module del virus di partenza**.
Per un periodo ('92 – '93) la gara tra i virus writers non era tanto a creare un nuovo virus piu' cattivo dei concorrenti, ma a creare il generatore piu' furbo con il quale creare **tanti** virus capaci di nascondersi meglio degli altri.

Esempi sono: *MTE*, *TPE*, *NED* e *DAME*
Curiosita': NED (Nuke Encryption Device) era persino fornito open source e conteneva la seguente stringa “[NuKE] Encryption Device v1.00 (C) 1992 Nowhere Man and [NuKE]”

Oltre DOS

Con l'avvento di nuove architetture e nuovi sistemi operativi, i vecchi virus per DOS sono diventati praticamente inutilizzabili. C'e' chi li definisce presenti solo a livello teoretico.
Cioe' fanno parte del bagaglio culturale di un ricercatore nel campo degli antivirus, ma anche degli ingredienti cui un virus writer attinge.

Per esempio: un virus stealth per Windows non e' ancora apparso. Questo perche' su Windows e' piu' facile di prima creare programmi con comportamenti maligni senza bisogno di particolari camouflage.
(Anteprima di Outlook!)

Oltre DOS

PMBS e *PM.Wanderer* furono tra i pochi virus a sfruttare la modalita' protetta (pm = protected mode) dei 386 e successivi.
All'avvio tramite Boot Sector infetto, il proprio codice viene eseguito in modalita' supervisore, che successivamente lancia DOS in modalita' finestra virtuale V86.

Questo tipo di virus e' **IMPOSSIBILE** da individuare se si e' solamente sotto DOS perche' il sistema operativo gira in modalita' virtuale su UN PROCESSORE 8086 **emulato** dall'hardware.
Il virus invece gira in modalita' reale.

Non si diffusero molto, grazie all'avvento di sistemi operativi che non lavoravano piu' in DOS puro.

Windows 95

Con l'abbandono del DOS, i virus iniziarono a diffondersi tramite nuove strategie.

Il primo modo di diffondersi in assoluto per Windows 95 era allegarsi ai dischi demo forniti direttamente da Microsoft ai collaboratori. Si trattava del “vecchio” *Form* per DOS che venne scoperto da un beta tester che scansiono' i dischetti ricevuti da MS. (Febbraio 1995)
(17 giugno 2002: Microsoft ha ripetuto questo errore, distribuendo il package Visual Studio .NET in Koreano infettato da *Nimda*
http://www.sophos.com/virusinfo/articles/nimda_korea.html)

Con i vecchi BIOS (486 e Pentium I) inoltre, se la protezione antivirus behaviour blocker era attivata (cioe' veniva protetto in scrittura l'MBR), l'installazione di Win95 si bloccava al 96% senza dare ALCUN messaggio.

Macro

Una delle prime “nuove tecnologie” (che ancora non sfruttava il formato PE NewEXE di Windows 95) fu quella tramite MACRO di Microsoft Office.

Gli antivirus, per forza di cose, devono monitorare anche l'esecuzione di miniprogrammini all'interno di file.
E non si tratta di andare in cerca di stringhe di istruzioni assembler, ma di statements in un linguaggio object oriented.

Essendo un linguaggio di programmazione ad alto livello, e' piu' facile nascondersi a scansioni tramite stringhe, sfruttando la flessibilita' del linguaggio.

Vediamo un paio di tecniche anche recenti.
http://vx.netlux.org/lib/vjt01.html

Macro: Nascondersi agli scan

Si tratta di riutilizzare 10 anni di storia dei virus a proprio vantaggio.

- Stringhe tipo `Options.Virusprotection=false` all'interno di una macro sono fin troppo evidenti. E' un tentativo di disattivare la “protezione” da macro virus che Word offre.
E' sufficiente sostituirle con
`Options.Virusprotection = (Rnd * 0)`

L'antivirus esegue scansioni “non intelligenti”, semplicemente parsando il codice alla ricerca di stringhe stabilite e NON calcolando nulla (numero_random moltiplicato per zero equivale a falso). Un po' come facevano alcuni polimorfici generando istruzioni assembler junk o con modi di indirizzamento “strani”.

Macro: Nascondersi agli scan

Questo metodo spesso e' inutile, perche' molti utenti di MS Office che fanno uso di Macro preferiscono disabilitare la funzione per evitare il nag screen ad ogni apertura di documenti.

(Stessa cosa piu' o meno con la protezione dall'esecuzione di script VB o ActiveX o JavaScript, di cookie, download di file eseguibili e quant'altro con Office, Outlook e Internet Explorer)

Macro: Nascondersi agli scan

- In seguito l'AV cerca per variabili sospette di oggetti.

Stringhe come

```
Set Norm = NormalTemplate.VBProject.VBComponents(1).CodeModule
```

settano come Modello di default (Normal.dot), il contenuto della variabile a destra dell'uguale. L'AV scanna per associazioni di questo tipo, ma basta rimbalzare attraverso diverse variabili per ottenere lo stesso risultato.

```
Set a = Normaltemplate: Set b = a.VBProject
Set c = b.VBComponents(1): Set d = c.CodeModule
```

E' esattamente equivalente alla stringa precedente ma confonde l'antivirus.

Sempre che stia scansionando tramite maschere.

Il comportamento del virus infatti RIMANE LO STESSO.

(Perche' l'antivirus si confonde? Qualita' di un antivirus)

L'antivirus ha delle esigenze, essendo un prodotto commerciale destinato a degli utilizzatori che lo hanno pagato.

- Deve essere ovviamente privo di errori di programmazione.
Quanto e' credibile un AV che va in crash durante la scansione?

- Deve garantire una scansione efficace (no miss)

(Perche' l'antivirus si confonde? Qualita' di un antivirus)

? Deve avere una velocita' sopportabile, rendendo il suo lavoro il piu' possibile trasparente all'utente

? Deve fornire il **minor numero di crying wolves**, cioe' di falsi allarmi.

Su questi due ultimi elementi si basano soprattutto le tecniche all'interno dei macro virus.
Sarebbe non commerciabile un antivirus che perde molto tempo nell'analisi estensiva di un linguaggio ad alto livello come Visual Basic. Un'analisi “troppo zelante” al contrario porterebbe a troppi falsi positivi.

Macro: Nascondersi agli scan

- E' tramite stringhe simili a queste (note agli antivirus), il virus macro si replica:

```
ActiveDocument.VBProject.VBComponents("DUAL").Export "c:\dual.sys"
                                    ....
Normaltemplate.VBProject.VBComponents.Import "c:\dual.sys"
```

L'antivirus cerca **solo** gli usi impropri della coppia `import/export`.

Usando un metodo grezzo di copia, si aggira questo controllo:

```
viruscode = ActiveDocument.VBProject.VBComponents("DUAL") _
            Lines(1,20)
Open "c:\dual.sys" for output as #1
    Print #1, "Attribute VB_Name = ""dual"""
    Print #1, viruscode
Close #1
....
Normaltemplate.VBProject.VBComponents.Import "c:\dual.sys"
```

Ovviamente la parola *viruscode* non viene usata.

- Altro: AMI Technic, AddModuleInfection, *W97M.Uebel*

Macro: Cifratura

Si sfrutta l'alto livello del linguaggio, riproducendo comportamenti tipici dei selfencrypting viruses (cioe' corpo crittato + subroutine di decifratura in chiaro).

- ASCII Exchange Encryption: *W97M.Optiz.D*, spostare i caratteri ascii di n posti, l'esempio piu' facile di cifratura ma efficace se dall'altra parte NON c'e' alcun tentativo di decifratura.

- XOR Encryption: “a XOR b = c” => “c XOR b = a”

- Junk Code Encryption (ricorda piu' la steganografia): aggiunge al codice del virus dei simboli ascii con valore maggiore di N e la procedura di decifratura poi filtra le stringhe eliminando quei valori (ad es. `if Asc(z) > 177 Then z = “”`)

Macro: Cifratura

Character Encryption: nato con la funzione di Word97 `Char()`

```
MsgBox "This is a virus"
```

equivale a

```
MsgBox Chr(84) + Chr(104) + Chr(105) + Chr(115) + Chr(32) +
Chr(105) _ + Chr(115) + Chr(32) + Chr(97) + Chr(32) + Chr(118) +
Chr(105) _ + Chr(114) + Chr(117) + Chr(115)
```

In un certo senso equivale a cio' che certi siti (di spam, contenenti virus VBS eccetera) viene fatto con Internet Explorer, ma in questo caso ci si nasconde dalla lettura a video. (vedi dopo)

Windows 9x

E' ovvio che queste tecniche sono combinabili in maniera arbitraria, tenendo conto tra l'altro che con Win9x sono stati introdotti MOLTI nuovi formati di file eseguibili, oltre alle macro.

- VXD
- DLL …

Infettare questo tipo di files e' come salire di un livello rispetto ai virus memory resident del DOS.

Gli antivirus per Win95 (salvo i primi realizzati ancora per DOS) diventavano funzionanti con l'avvio dell'interfaccia grafica (si avviavano tramite delle entries nel registro di sistema, che e' l'ultima cosa letta prima di lanciare l'interfaccia o addirittura con il menu “Esecuzione automatica), ma i virus venivano caricati in memoria PRIMA, insieme ai drivers del sistema operativo.

Windows 9x: Due esempi

Navrhar (http://www.viruslist.com/eng/viruslist.html?id=3029) seguiva questo cammino la prima volta:

Documento con macro virale -> PE NewEXE Dropper -> VXD ---> Documenti

Win95.Punch e' stato il primo Memory Resident in nuovo formato EXE (cioe' Win32). Invece di usare gli handler degli interrupt, usava intercettare le chiamate di sistema Kernel 32 per diffondersi e per nascondersi. Per il resto era come un TSR per DOS.

Windows 9x

I due esempi precedenti e MOLTI altri virus sono identificabili PRIMA DELL'INFEZIONE tramite un antivirus con i database che li contenga, poiche' basta la scansione per stringhe.

La confusione nell'identificare e rimuovere virus con Win9x **dopo** l'infezione e' dovuta anche al fatto che da dischetto non si riesce a avviare il sistema al 100% (interfaccia grafica) e quindi non si puo' usare l'antivirus “solito”.
Si puo' avviare da floppy e arrivare a un prompt dal quale poi lanciare antivirus funzionanti anche in DOS (ma On-Demand!).

Problemi: se e' una versione nata per Win ma adattata a ricevere comandi via console e' possibile che NON riconosca alcuni vecchi virus stealth o polimorfici.
Se e' viceversa, un vecchio antivirus per DOS non e' sicuramente aggiornabile con l'ultima firma disponibile, sempre che sia in grado di leggere gli EXE Win32.

Windows 9x: ancora eseguibili?

Per come e' impostata l'interfaccia di Win9x, e' facile per i virus tentare ancora la strada dell'inganno dell'utente.

Windows distingue i tipi di file in base all'estensione e di **default** visualizza i file di documenti (bmp, gif, ppt, doc, txt, xls.....) conosciuti privi dell'estensione, ma SOLO col nome e con un'icona che *dovrebbe* ricordare il programma ad essi associato.

In pratica e' come se il file di documento fosse un eseguibile, se per “esecuzione” significa fare doppio click sopra un'icona, il cui risultato da' l'apertura di un programma.

La complessita' delle tecniche adottate dai virus per nascondersi e' inversamente proporzionale al numero di “incorrettezze” che possiede il sistema operativo da attaccare.

Windows 9x, ME, 2000: gli utenti

Uno dei primi metodi per far si' che gli utenti Windows eseguano dei programmi maligni e' proprio quello di sfruttare l'associazione di files.
(Prima ancora si gioca sulla loro pigrizia nell'installare o tenere aggiornato un av).

Ci sono almeno due modi:

- Doppia estensione al file *(Iloveyou.txt**.vbs***): soprattutto se l'utente ha le impostazioni di default. Quando venne utilizzato la prima volta, gli antivirus ancora non scansionavano gli allegati e-mail.
Possibilmente nel caso di file .exe si associa al programma del virus un'icona che ricorda un file testo o un'immagine.

Windows 9x, ME, 2000: gli utenti

? Utilizzare l'associazione “nativa” ai file (http://www.guninski.com/):

Win non usa solo le estensioni .* per identificare i files, ma anche la CLSID, cioe' un numero di 128 bit esadecimale che identifica una certa estensione.

Anche se un utente ha la visualizzazione delle estensioni per i files conosciuti abilitata, la CLSID rimane invisibile ugualmente.

Esempio:

fotomia.jpg.{00020C01-0000-0000-C000-000000000046}

verra' trattato come un file audio!

Ovvie le conseguenze se si aggiunge la CLSID di un file .exe...

Internet: Esempio 1, Url Spoofing

Dato che il browser integrato in Windows molto spesso esegue senza alcun messaggio di conferma gli script contenuti nelle pagine internet, l'ennesimo modo di nascondersi all'utente (e persino attirarlo!) e' sfruttare questo “errore” unito ad altre arguzie:

1. Molti browser permettono di scrivere degli url tipo
www.guardiadifinanza.it@h4x0r.da.ru
ignorando la parte prima di @, ma andando subito al link dopo.
2. E' possibile anche inserire url contenenti il numero di IP
www.guardiadifinanza.it@213.77.78.8 (che e' l'IP di h4x0r.da.ru)
3. Al posto dell'IP in dot notation, si puo' usare la versione dotless
www.guardiadifinanza.it@3578613256
4. Oppure mascherare alcuni caratteri con la loro versione codificata in ascii esadecimale (e con un indirizzo leggibile piu' interessante)
www.makemoney.com/getdollars.php?%40%684x%30r.%64%61%2Eru

Internet: Esempio 2, quando l'utente non serve

Gli stragemmi usati fino ad ora sono assolutamente validi anche se non e' l'utente che si va a cercare i guai, ma gli vengono comodamente recapitati nella casella di posta elettronica. Sta comunque sempre a lui l'onore di ESEGUIRE il programma infetto (tipo *happy99*).

E quando e' **il programma di posta che esegue** allegati maligni?
Klez usava un buco di Internet Explorer riguardo alla funzione IFRAME.ExecCommand(), per partire automaticamente durante la visualizzazione del messaggio di posta elettronica.