Untersuchungen von Betriebssystemen zur Vermeidung von Computerviren

1 Über dieses Dokument

Die komplette Beschreibung aller Virenarten und Virengattungen unter den Betriebssystemen DOS, Windows, UNIX, Novell etc.

Dieses Dokument ist ein Originalauszug aus meiner 1995 veröffentlichen Diplomarbeit "***Untersuchungen von Betriebssystemen zur Vermeidung von Computerviren***". Dieses Dokument wird wegen der Authentizität (-> Diplomarbeit) nicht gepflegt! Aus heutiger Sicht sind viele Passagen natürlich überholt, jedoch hoffentlich immer noch eine sehr gute Referenz zu Computerviren.

Durch die Konvertierung von WinWord 6.0 auf WinWord 97 gingen leider einige Formatierungen, Indexe und Gliederungen verloren (soviel zu kompatiblen Programmen☹). Heute wird diese Dokumentation in OpenOffice gepflegt.

2 Übersicht

Seit dem ersten Auftreten von Computerviren unter MS-DOS im Jahre 1986 hat sich die Anzahl der existierenden Computerviren explosionsartig vermehrt. Zurzeit[1] existieren ca. 6000 verschiedene DOS-Viren. Die ersten OS/2- und Windows-Viren sind vor kurzem gefunden worden. Bei diesen ebenfalls relativ offenen Betriebssystemen ist der gleiche Trend wie unter DOS zu erwarten, ein Ende ist generell noch nicht abzusehen! Unter UNIX gibt es bis heute fast keine Viren, was darauf hindeutet, dass ein Betriebssystem mit Sicherheitsmerkmalen vor Viren sicherer ist! Der folgende Beitrag versucht das Phänomen "Computervirus" und die damit verbundene Mystik etwas aufzuklären. Das Thema Computerviren ist nicht ganz einfach zu verstehen, deshalb ist für das Verständnis des folgenden Artikels ein "gesundes" Maß an EDV Grundwissen erforderlich!

2.1 Das Handbuch

Dieses Handbuch wurde mit WinWord erstellt und einem vom Programmautor entwickelten Hilfsprogramm nach ASCII portiert (Datei VirusDef.Doc). Sie können deshalb dieses Handbuch auf JEDEM Drucker, der mindestens 64 Zeilen pro Blatt und einen Seitenvorschub unterstützt, ausdrucken (auch auf Laserdruckern).

Befehl: COPY VIRUSDEF.DOC PRN

ACHTUNG: Das Originaldokument enthält Grafiken, Tabellen und Fußnoten, die natürlich nicht übernommen werden konnten! Seit Dezember 2001 wird dieses Dokument nicht mehr als MS-DOS ASCII Datei ausgeliefert sondern als Akrobat PDF Datei.

2.2 Urheberrecht

[1] Stand 1994, Zeitpunkt der Erstellung der Diplomarbeit

3 Inhaltsverzeichnis

Inhaltsverzeichnis

4 Einleitung

4.1 Aufbau dieser Diplomarbeit

Wie im Abstrakt bereits erwähnt, werden zunächst einmal alle relevanten Begriffe um den Ausdruck „Computervirus“ definiert. Anschließend erfolgt ein historischer Rückblick über die noch relativ junge Existenz von Computerviren. Mit einer Klassifikation der einzelnen Virengattungen werde diese genauer eingeordnet und mit der Vergabe von verschiedenen Attributen werden die Eigenschaften von existierenden Computerviren festgelegt.

Mit diesem Hintergrundwissen, wie Computerviren funktionieren, werden dann Betriebs- und Dateisysteme analysiert und deren virenspezifischen Schwachstellen kurz charakterisiert. Um solche existierende Sicherheitslücken zu „stopfen“ wird im Kapitel „Hard- und Softwarelösungen“ auf geeignete Lösungen eingegangen. Da diese Lösungen jedoch meistens nur unzureichend sind, wird im darauf folgenden Kapitel weitere Verbesserungsvorschläge für Betriebs- und Dateisysteme diskutiert.

Diese Diplomarbeit hat unter anderem auch zum Ziel, die vorhandene Sicherheitslücke bei PC-Rechnern mit INTEL 80386+ Architektur durch ein zu entwickelndes Programm zu schließen. Dieses Programm wird im Kapitel „Das Programm MBR-Kill“ vorgestellt und es werden für dieses Programm verschiedene Lösungsansätze skizziert. Das nachfolgende Kapitel widmet sich dann der Bedienung und Beschreibung des Programms MBR-Kill, das sich auch auf der beigelegten Diskette befindet.

Das Programm MBR-Kill wurde zwar so entworfen, dass es auch zukünftige Computerviren zuverlässig erkennt und blockieren kann, dennoch wird ein Ausblick auf zukünftige Trends in der Virenentwicklung, Betriebssystemprogrammierung und der zunehmenden Vernetzung gegeben. Mit einer kurzen Schlussbetrachtung und dem Anhang wird diese Diplomarbeit beendet.

4.2 Ethik

Im heutigen Zeitalter, in dem viele Arbeitsabläufe nur noch durch die Unterstützung von Computern möglich sind, stellen Computerviren eine immense Bedrohung dar. Deshalb möchte ich mit dieser Diplomarbeit niemanden dazu animieren, selbst Computerviren zu schreiben, sondern rein wissenschaftlich erläutern, in welche Gruppen die bestehenden Viren eingeordnet und kategorisiert werden können. Welche Wirkung haben sie und wie kann man gegen sie vorgehen oder sie sogar verhindern.

In dieser Diplomarbeit kommen aus diesen Gründen keine Quellcodes vor, da durch verschiedene Veröffentlichungen und Literatur genug Schaden angerichtet wurde. So soll diese Diplomarbeit eher aufzeigen, wie man sein Betriebssystem vor Computerviren schützen kann. Das Programm „MBR-Kill“ stellt einen Schritt in diese Richtung dar.

4.3 Einige Definitionen

Die meisten der hier eingeführten Begriffe sind allgemein bekannt, werden jedoch in der Literatur unterschiedlich gebraucht, so dass zur Klärung hier eine knappe formale Definition erfolgt. Eine Definition erfolgt in *kursiver* Schrift.

Eine weitere ausführliche Erklärung dieser Begriffe steht teilweise in den nachfolgenden Kapiteln oder kann z. B. folgender Fachliteratur [Bontch], [Esch93], [FAQG93], [FAQL92], [Ferb92] und [PB92] entnommen werden.

4.3.1 Softwareanomalien

Die Software verhält sich nicht so, wie sie spezifiziert wurde, also nicht nach den Regeln, nach der sie entwickelt wurde.

Tritt eine Softwareanomalie in einem Computersystem auf, so verhält es sich oder einzelne Systemkomponenten davon nicht mehr so wie erwartet. Dies hat Konsequenzen für die Umgebung, in der das Computersystem eingebettet ist, weil die Funktionalität und Systemsicherheit eingeschränkt ist. Ein anderer, gebräuchlicher Begriff hierfür ist auch der "Seiteneffekt". Der Begriff Softwareanomalie kann demnach als Oberbegriff für eine Anzahl von Phänomenen verwandt werden, unter die Computerviren und verwandte Programme fallen. Hierzu zählen u. a.:

- Computerviren
- Würmer und Kettenbriefe
- Trojanische Pferde und Dropper
- Logische Bomben und Trapdoors
- Scherzprogramme

Eine kurze Definition dieser Anomalien erfolgt auf den nachfolgenden Seiten.

4.3.2 Der Computervirus

Ein Computervirus ist ein Computerprogramm, das die Fähigkeit besitzt, sich selbst zu reproduzieren (vermehren), indem es Betriebssystemkomponenten oder Betriebssystemabläufe (Umgebung) zu seinen Gunsten modifiziert[2].

Der Computervirus hat, im Gegensatz zum biologischen Virus (das Virus, von lat. virus = giftiger Saft), den männlichen Artikel und wird oft nur kurz als "der Virus" bezeichnet.

Folgende weitergehende Folgerungen lassen sich aus dieser Definition ableiten:

- ein Virus muss von jemanden geschrieben werden

[2]Anmerkung: Diese Definition wurde allgemein gehalten, damit auch spezielle Computerviren, wie etwa „Companionviren“ erfaßt werden können.

- damit ein Virus aktiv werden kann, muss er, z. B. durch Programmstart, aktiviert werden.
- ein Virus benötigt einen Wirt (Wirtsprogramm)
- ein Virus vervielfältigt sich demnach selbständig und deshalb meistens unkontrolliert (lack of control)!
- ein Virus verändert seine "Umgebung"
- es gibt keine "gutartigen" Viren, weil Viren in den Betriebssystemablauf eingreifen, was zu Softwareanomalien führt (unauthorized data modification).

Mit einem „Research virus“ bezeichnet man Viren, die zwar geschrieben wurden, aber nie verbreitet wurden.

Es handelt sich hierbei meistens um extrem primitive Viren[3] oder Viren, die direkt vom Virenautor an Antivirenforscher geschickt wurden.

Mit „in the wild“ bezeichnet man Computerviren, die außerhalb von den Labors von Virenforschern gefunden wurden.

Der Begriff „in freier Wildbahn“ wird teilweise auch hierfür verwendet.

Eine Variante ist ein Virus, der durch Modifizieren eines bekannten Virus entstanden ist.

Beispiele hierfür sind hinzugefügte Funktionen oder Änderungen am Viruscode um eine Entdeckung zu erschweren.

4.3.3 Der Wirt

In der Biologie bezeichnet man das Medium, dass einen Krankheitserreger transportiert als Vektor einer Infektion. Im Computerbereich hat sich hierfür die Bezeichnung Wirt eingebürgert.

Ein Wirt ist eine Betriebssystemkomponente, die von einem Virus befallen (infiziert) werden kann und es dem Virus ermöglicht, sich weiterzuverbreiten. Einen infizierten Wirt bezeichnet man auch als verseucht.

4.3.4 Die Infektion

Die Infektion ist der Befall eines Wirtsprogramms durch einen Virus.

4.3.5 Würmer (Worms)

Ein Wurm ist ein eigenständiges Programm, das Kopien von sich selbst erzeugt und zum Ablauf bringt.

[3] Sogenannte „Virogen“-Viren.

Der Begriff Wurm kommt aus der UNIX-Welt. Die Bezeichnung „Wurm“ ist etwas verwirrend. Vermutlich kommt die Bezeichnung Wurm von der Eigenschaft des Wurmprogramms, sich von Benutzer zu Benutzer durchzufressen. Eine andere Erklärung für die Bezeichnung „Wurm“ könnte eventuell von dem UNIX Spiel „worm“ abgeleitet werden: In diesem Spiel wird der Wurm beim Fressen immer länger. Ein Computerwurm „frisst“ sich auch von System zu System durch und wird dabei immer „länger“, solange bis die Systemlast so groß ist, bis das System zusammenbricht.

Ein Wurm ist die Vereinigung aller einzelnen Wurmsegmente. Ein Wurmsegment ist ein lauffähiges Programm, welches sich innerhalb von Netzwerken selbst vervielfältigt, ohne hierbei Wirtsprogramme zu verändern bzw. zu infizieren. **Ein Wurm benötigt - im Gegensatz zu Computerviren- keinen Wirt.** Wurmsegmente können sich gleichzeitig in verschiedenen Systemen befinden. Dies geschieht z. B. auf vernetzten Rechnern durch Prozessgabelung (z. B. mit dem fork-Systemaufruf) oder, wie beim Tannenbaumwurm, durch Duplizieren und Versendung des eigenen Programmcodes an andere Rechner. Der Wurm kann mehrere Programme auf dem eigenen Rechner gleichzeitig ablaufen lassen (Multitasking) oder andere Rechner über Netzwerkdienste nutzen. Weil auf einem isolierten Rechner keine Verbreitung erfolgen kann, sind Würmer typisch für Netzwerke. Ein wichtiges Kriterium für einen Wurm ist, dass er sich selbst an andere Rechner versenden darf und dort sich selbst starten kann. Gerade der Fernstart ist aus Sicherheitsgründen normalerweise nicht möglich. Die Praxis hat jedoch gezeigt, dass dies möglich ist, z. B. Morris´s „Internet-Wurm“ [Morr88].

4.3.6 Kettenbriefe

Der Kettenbrief ist eine Spezialform des Wurmes, der die Aufgabe hat, in den elektronischen Postkästen des Netzes Nachrichten abzulegen (Email).

Beispiel: Clausthaler Weihnachtsbaum.

4.3.7 Trojanisches Pferd

Ein trojanisches Pferd (kurz auch Trojaner) ist ein Programm, das eine bestimmte gewünschte Funktion ausführt, jedoch auch unerwartete (und unerwünschte) Funktionen enthält.

Von diesem Gesichtspunkt aus ist ein Trojanisches Pferd einem Virus ähnlich, mit der Ausnahme, dass ein Trojanisches Pferd sich nicht vermehrt.

4.3.8 Speicherresident

Ein speicherresidenter Virus installiert sich beim Ausführen eines verseuchten Wirtsprogramms selbst als ein Teil des Betriebssystems.

Der Virus bleibt solange im Arbeitsspeicher, bis das System ausgeschaltet wird. Ist solch ein Virus erst einmal im Arbeitsspeicher installiert, kann er jeden passenden Wirt bei einem geeigneten Zugriff infizieren.

4.3.9 Tarnkappenviren

Ein Tarnkappenvirus (engl. Stealth-Virus, stealth = heimlich) ist ein speicherresidenter Virus, der versucht, eine Entdeckung von sich selbst zu verhindern, indem er sein Vorhandensein in infizierten Wirten verheimlicht.

Um dies zu erreichen, muss solch ein Virus Systemaufrufe, die einen Zugriff auf den infizierten Wirt durchführen, abfangen und entsprechend manipulieren.

4.3.10 Verschlüsselte Viren

Ein verschlüsselter Virus besteht aus zwei Teilen: aus einem kurzen Entschlüsselungsteil (decryptor) und dem verschlüsselten Rest (encrypted body).

Wird solch ein verschlüsselter Virus ausgeführt, wird zuerst der Decryptor ausgeführt, der den eigentlichen Virus entschlüsselt.

Ein variabel verschlüsselter Virus benützt verschiedene Schlüssel oder verschiedene Verschlüsselungsalgorithmen. Solche Viren werden auch als oligomorph verschlüsselt bezeichnet.

Ein polymorph verschlüsselter Virus ist ein variabel verschlüsselter Virus, der zwar funktionelle gleiche Kopien seiner selbst erzeugt, diese Kopien sind von ihrem Aussehen jedoch total unterschiedlich. Eine Routine, die solche polymorphen Decryptoren erzeugt, wird auch als "Mutation Engines" bezeichnet.

4.3.11 Selbst-upgradend

Ein Virus, der nach Vorgängerversionen seiner selbst sucht und diese durch sich selbst ersetzt, bezeichnet man als "selbst-upgradend".

4.4 Erste Attacken

Die ersten sich selbstreproduzierenden Programme waren so genannte „Mainframe rabbits“ ([Ferb92], S. 5), die 1966 am MIT, Boston von zwei Studenten geschrieben wurden. Diese Programme, die als Interpreterscript für den Batchbetrieb eines CTSS Timesharing-Systems auf einer IBM 7090 geschrieben wurden, erzeugten von sich selbst Kopien und verlangsamten so den Systemdurchsatz enorm. Aufgrund eines Fehlers des Systems wurde der Rechner lahm gelegt. Der Interpreter unterstützt oft Kommandos für Prozessgenerierung und Prozessmanipulation. Der „rabbit“ kann sich in diesem Fall zigmal kopieren und alle

Prozessqueues füllen, die erreichbar sind. Ältere Systeme, die noch keinen Prozessorscheduler besitzen, brechen dann irgendwann unter der Systemlast zusammen.

4.4.1 Die ersten Netzwerkwürmer

1972 beschrieb David Gerrold in seinem Buch „When Harlie Was One“ das Konzept eines Netzwerkwurmes, der sich über Selbstwähl-Modems auf andere Rechner kopiert, sich dort aktiviert und seine alte Kopie löscht ([Ferb92], S. 6).

Diese Beschreibung trifft grob auf die ersten Würmer („Creeper“ und „Reaper“) zu, die von zwei Forschern in den 70´er Jahren zu Demonstrationszwecken entwickelt wurden. Creeper druckte eine Datei aus dem System aus, wartete und kopierte seinen Programmcode auf einen anderen vernetzten Rechner und löschte sich auf dem Ursprungsrechner. Creeper „kroch“ somit langsam von Rechensystem zu Rechensystem. Creeper wurde dahingehend modifiziert, dass er zusätzlich noch Kopien von sich selbst erzeugte und diese nicht mehr löschte! Der erste Netzwerkwurm war geboren. Nun wurde der Wurm Reaper entwickelt, der nach Kopien des Creaper-Programmes suchte und diese löschte. Somit war auch das erste „Antivirenprogramm“ entwickelt worden.

Solche Programme, die selbst von ihrem Aufbau Viren sind, jedoch den Auftrag haben, einen anderen Virus zu suchen und zu zerstören bezeichnet man als „Anti-Anti Virus“.

Im Xerox Alto Research Center wurden 1982 die ersten Würmer für verteilte Berechnungen erstellt. Aufgrund eines Programmfehlers vermehrte sich der Wurm unkontrolliert und brachte das System zum Zusammenbruch.

Der Internet-Wurm, der 1988 von Morris freigesetzt wurde, vermehrte sich innerhalb eines Tages auf schätzungsweise hunderten bis mehreren Tausend Rechnern. Der Internet-Wurm war der erste Wurm, der weltweit publik wurde.

4.4.2 Die ersten Viren

1980 wird der erste Laborvirus auf einem APPLE II entwickelt. 1981 wird der Virus „Elk Cloner“ auf einem APPLE II entdeckt. Er führt zufällige Aktionen aus, infiziert den Bootsektor und bindet sich speicherresident in die Interruptes ein. 1982 wird von Joe Dellinger der erste selbst-upgradende Virus für einen APPLE II mit DOS 3.3 verbreitet.

4.4.3 Erste wissenschaftliche Untersuchungen

Professor L. Adelmann, Universität South California, prägt als erster den Begriff „Computervirus“.

Der erste, der sich wissenschaftlich mit der Untersuchung von Computerviren beschäftigte, war Fred Cohen. Ein Artikel von ihm „Computer Viruses - Theory and Experiments“ erschien 1984, seine Doktorarbeit über das gleiche Thema beendete er 1986 ([Treb92], S.

16f). Cohen war auch einer der ersten, die experimentelle Viren auf Großrechnern entwickelte. Die nachfolgende Tabelle gibt einen kurzen Überblick über seine Tätigkeiten.

Datum	Beschreibung
03.11.83	Entwicklung eines Virus für UNIX (VAX 11/750). Die Programmierung dauerte 8 Stunden, die Systemrechte erlangt der Virus nach durchschnittlich 30 Minuten.
10.11.83	Experiment mit einem über ein Bulletin Board (BBS) eingeschleustem Programm.
Juli 84	Tests auf Bell-LaPadula System (Univac 1108)
August 84	Test unter UNIX (VAX)
31.08.84	Veröffentlichung seiner Arbeit „Computer Viruses - Theory and Experiments“ bei Professor L. Adelmann.

Tabelle: Erste wissenschaftliche Viren

Dies waren die ersten experimentellen Viren auf wissenschaftlicher Basis.

4.5 Chronologischer Rückblick

- 1985 -

Am 11.12.85 veröffentlichte die Zeitschrift „Apples“ ein Virus im Quellcode für den APPLE II. Die Zeitschrift „Computer Persönlich“ zog mit ihrer Ausgabe Nr. 24 vom 12.11.86 nach, die ebenfalls den Quellcode für einen Virus für den APPLE II enthielt ([Treb92], S. 18).

- 1986 -

Im Januar 1986 wurde die erste Vireninfektion auf einem Großrechner für Datenverarbeitung an der FU in Berlin entdeckt. 1986[4] wurde der erste Virus für Rechner mit Inter-Architektur entdeckt, der sog. Pakistani-Brain[5] Virus. Mitte des Jahres tauchten die ersten DOS-Viren in englischsprachigen Programmen auf. Kurz darauf wurden die Dateiviren Vienna, Cascade, LeHigh sowie die Bootviren Yale, Stoned und Pingpong gefunden.

- 1987 -

Im November 1987 wird der Virus LeHigh gefunden, der eine neue Technik verwendet, ein so genannter Stackbereich Infektor. Kurz darauf (Dezember 1987) wurde der Israeli-#1 (Suriv) Virus an der Universität von Jerusalem gefunden, der der Urvirus der Jerusalem Familie[6] ist.

[4]Hier gehen die Meinungen nach dem ersten Auftauchen des Brain-Virus auseinander. Laut [Esch93] wurde er im Oktober 1987 an der Universität von Dalaware entdeckt. Laut Experten, könnte der Vorgänger des Brain-Virus, der Ashar-Virus, jedoch aus dem Jahre 1986 oder früher stammen. Sicher ist jedoch, daß er im Januar 1986 in Pakistan gefunden wurde, siehe ([Ferb92], S. 11).

[5]Ein speicherresidenter Bootvirus, mit Tarnkappeneigenschaften. Er konnte jedoch nur 360 KB Disketten infizieren.

[6]Vom Jerusalem Virus gibt es schätzungsweise 300, zum Teil sehr stark modifizierte Varianten!

Im gleichen Jahr wird in Deutschland der Cascade Virus (1701/1704) gefunden. Er ist der erste speicherresidente verschlüsselte Virus, man spricht deshalb auch von der zweiten Virengeneration. Für andere Rechner tauchen ebenfalls die ersten Viren auf,

- MAC - nVir wird in Deutschland entdeckt, im Dezember der Peace Virus.
- Amiga - Swiss Cracker Association Virus.
- Atari - Pirate Trap Virus und Aladdin (der erste Cross-Platform Virus)
- UNIX - IBM MVS 370 Virus, im Juni ein anderer UNIX Virus

In der „c´t“ wird 1987 ein Artikel mit dem Source-Code von einen Virus für den Atari ST veröffentlicht. Aus diesem Quellcode entsteht ein ganzer Virenstamm in der Nachfolgezeit. Es werden deshalb die ersten Kritiken zur Veröffentlichung von Viren Source-Code laut.

Der Clausthaler Weihnachtsbaum (Christmas Tree Exec) wird am 17.12.87 von der Hochschule Clausthal-Zellerfeld über IBM-VNet, EARN und BitNet um die ganze Welt geschickt. Es handelt sich hierbei um einen speziellen Wurm, einen so genannten Kettenbrief, der sich über Email Adressen verbreitet.

- 1988 -

1988 wird der erste Virenbaukasten (Virus Construction Set) für den Atari ST veröffentlicht. Im gleichen Jahr erlangen die Netzwerk Würmer „Internet-Worm“ und „Father Christmas“ weltweite Verbreitung.

- 1989 -

Der erste polymorphe Virus (1260, V2Px oder Washburn) wird 1989 gefunden. Es werden die ersten Dateiviren (Frodo) gefunden, welche Tarnkappeneigenschaften verwenden. Erste internationale Bemühungen, Viren zu klassifizieren. Die ersten gedruckten Dienste, die über Viren informieren (Virus Bulletin und Virus Telex) erscheinen. Weitere Würmer tauchen auf; der WANK (Worm against nuclear killers) und OILZ Wurm. Die AIDS Trojan Diskette wird weltweit versendet. Die Medien finden Interesse an der Thematik Computer Viren. Die ersten Artikel und Bücher werden veröffentlicht.

- 1990 -

Der Whale Virus wird 1990 entdeckt, er ist ein „Super“-Virus[7], kann sich selbst modifizieren, ist oligomorph verschlüsselt und enthält Tarnkappeneigenschaften. Er ist bis heute der größte speicherresidente Virus für das DOS Betriebssysteme. Auf dem Macintosh wird ein Trojanisches Pferd gefunden, dass das Laserwriter Passwort verändert. In diesem Jahr wurde der DIR-II Virus unter DOS freigesetzt. Es ist ein ganz neuer Virustyp, er infiziert nämlich nicht Programme sondern deren FAT Einträge. Das erste Virus Construction Set (VCS) vom „Verband deutscher Virenliebhaber“ für DOS wird verbreitet.

[7]Laut Gerüchten, soll er sogar von Virenforschern als „Laborvirus“ programmiert worden sein. Dieser Virus hat eher wissenschaftlichen Charakter, der wurde nie groß in der „freien Wildbahn“ (in the wild) gefunden.

- 1991 -

Im Jahre 1991 taucht eine Vielzahl neuer Viren auf, die Anzahl der neu gefunden Viren wachsen jetzt langsam aber sicher exponentiell an. Der Yankee.Login (Get Password) Virus wird entdeckt. Er wurde speziell für Novell-Netware (unter DOS) programmiert, um Passwörter auszuspähen. Die weltweite Zusammenarbeit wird besser, es werden die Organisationen CARO (Computer Anti-Virus Research Organisation) und EICAR (European Institute for Anti-Virus Research) gegründet.

- 1992 -

Die erste Mutation Engine wurde 1992 von „Dark Avenger“ als Objektdatei veröffentlicht. Seither gibt es eine wahre Flut an Mutation Engines.

- 1993 -

Ab 1993 kann man generell sagen, dass DOS Computerviren eine eigene Dynamik entwickelt haben. Fast jeden Tag tauchen zwei bis drei neue Computerviren auf und ca. jeden Monat taucht eine neue Mutation Engine oder ein neues Virenkit auf. Prinzipiell gibt es seit diesem Zeitpunkt keine neuen Viren mehr, nur altbekanntes in neuen Variationen, wie z. B. Kernel-Infektoren oder permutierenden polymorphe Viren.

- heute (2001) -

Mit den „neuen“ Betriebssystemen wie OS/2, Windows oder UNIX Derivate für PC sind auch neue Viren für diese Betriebssysteme entstanden. Im Gegenzug kam die Virenentwicklung z. B. beim Macintosh oder bei UNIX Würmern fast zum Stillstand, weil einige Virenprogrammier erfolgreich verurteilt wurden, was zu einer Abschreckung führte.

Weitere Trends in der Virenentwicklung habe ich in einem eigenen Kapitel „Trends in der Virenentwicklung“ zusammengefasst.

Aus diesen Gründen konzentriert sich diese Diplomarbeit hauptsächlich auf DOS Computerviren.

5 Klassifizierung/Typologie

Im nachfolgenden Kapitel möchte ich existierende Viren nach ihrem Infizierungsverhalten klassifizieren. Im nächsten Kapitel wird dann auf die möglichen Attribute eingegangen, die Computerviren haben können.

5.1 Einleitung

Die nachfolgende Beschreibung bezieht sich fast ausschließlich auf Computerviren, die unter dem Betriebssystem DOS[8] auftreten. Abgesehen von Tarnkappentechniken und einigen speziellen Dateiformaten unter DOS/Windows ist die Beschreibung der Virenarten auf jedes Betriebssystem übertragbar!

5.1.1 Aufenthaltsbereiche von Viren

PCs besitzen zwei Bereiche, die für eine Vermehrung von Viren generell in Betracht kommen. Es handelt sich um das Dateisystem und um den so genannten Bootrecord. Somit kann zwischen zwei Virenarten unterschieden werden, nämlich

- Dateiviren, die
  - Dateien (Programme) oder das
  - Dateisystem infizieren und
- Bootviren, die
  - den Master Boot Record (MBR) oder
  - den aktiven Bootrecord infizieren.

Kann ein Virus beide Bereiche infizieren, bezeichnet man in als Hybridvirus oder auch multipartite.

Es folgt eine Klassifizierung von Viren nach Art und Weise der Infektion, anschließend werden die verschiedenen Virengattungen genauer beschrieben.

Abb.: Grobe Klassifizierung von Viren und deren hauptsächliche Verbreitung

5.2 Bootviren

Bootviren infizieren keine Programme, sondern den so genannten Bootsektor, auch Bootrecord genannt! Man unterscheidet zwischen zwei Arten von Bootviren, die sich in der Infizierung der Festplatte unterscheiden. Bei der Infizierung von Disketten verhalten sie sich jedoch gleich. Es handelt sich um

[8] Anmerkung: Fast alle Viren laufen unter den verschiedenen DOS Versionen wie MS-DOS, PC-DOS, DR-DOS oder Novell-DOS.

- MBR-Viren (Partitionsviren)[9]
- DOS-Bootviren (DBR)

[9] MBR ist die Abkürzung für „Master Boot Record“

Abb.: Klassifizierung von Bootviren

Eine Zwitterstellung nehmen die Hybrid-Viren und Kernel-Viren ein, ihr Merkmal ist jedoch, dass sie auch den MBR, den DBR der Festplatte oder DOS Bootsektoren auf Disketten infizieren können.

Achtung: Bootviren sind normalerweise speicherresident und extrem infektiös! Bei einem aktiven Bootvirus wird jede Diskette, die nicht schreibgeschützt ist, von dieser Virenart infiziert! Bootviren verbreiten sich somit schnell und sehr effektiv. Im Zeitraum September 1994 bis Mai 1995 waren die 10 am häufigsten auftauchenden Viren Bootviren! Knapp dahinter folgen dann die Hybridviren, die ebenfalls den MBR von Festplatten infizieren.

5.2.1 MBR-Viren (Partitionsviren)

Fast alle Viren, die auf Diskette den Bootsektor befallen, infizieren auf der Festplatte nicht den DOS-Bootsektor, sondern den Master-Boot-Record (MBR), im Sprachgebrauch fälschlicherweise auch "Partition Table" (Partitionstabelle) genannt. Der MBR (Master Boot Record) befindet sich auf dem ersten physikalischen Sektor der Festplatte und hat bei allen PCs ein einheitliches Format[10].

Die Partitionstabelle, welche die Daten über die einzelnen Plattenpartitionen enthält, ist ein zusätzlicher Bestandteil des MBR. In der Partitionstabelle können bis zu vier Partitionen aufgenommen werden. Eine dieser Partitionen wird normalerweise als aktiv markiert.

Der MBR enthält zusätzlich ein kleines Programm (Lader), welches bei Rechnerstart direkt vom BIOS her geladen und aufgerufen wird und den Rechner auf das Booten des eigentlichen Betriebssystems vorbereitet. Das Ladeprogramm prüft anhand der Partitionstabelle, welche Partition als aktiv markiert ist. Anhand der weiteren Daten in der Partitionstabelle wird dann der erste Sektor der aktiven Partition geladen und ausgeführt. Diesen Vorgang nennt man "bootstrapping".

Dieses Ladeprogramm wird nun von einem MBR-Infektor entweder überschrieben, verschoben und ersetzt, oder teilweise manipuliert. Viren, die so verfahren, zählen im Sprachgebrauch zu den Bootviren. Manche Viren, die Dateien infizieren, sind so programmiert, dass sie auch den MBR infizieren (siehe Hybrid-Viren).

MBR-Viren sind somit **nicht** auf das Betriebssystem DOS beschränkt, sondern können auch OS/2 und Linux-Rechner infizieren, die den PC Befehlssatz verwenden! Viren, die den MBR infizieren, können dies auf mehrere Arten tun:

- Ersetzen des MBR und Verwenden von Tarnkappeneigenschaften. Sobald der Virus aktiv ist, scheint somit der MBR unverändert zu sein.

[10] Siehe auch im Anhang: „MBR- und Partitionstabellen-Layout“ und „Boot- und MBR-Beschreibung“.

- Überschreiben oder Ersetzen des Ladeprogrammes, die Partitionstabelle wird jedoch übernommen. Dies ist sehr gefährlich, falls z. B. ein UNIX oder OS/2 Betriebssystem nachgeladen werden soll, weil im Ladeprogramm teilweise schon betriebssystembezogene Aktionen durchgeführt werden, die der Virus natürlich nicht durchführen kann. Oftmals hängt dann ein solches System.
- Es gibt auch Viren, die den MBR anders als hier beschrieben infizieren. Sie verändern z. B. nur Teile des MBR, legen für sich selbst eine neue Partition in der Partitionstabelle an, von wo aus sie dann weiterbooten lassen, oder sie codieren den gesamten MBR samt Partitionstabelle so, dass dieser nur bei geladenem Virus lesbar ist (z.B. ExeBug). Auch gibt es Viren, die den MBR aufgrund eines Programmierfehlers (oder absichtlich) einfach überschreiben (z. B. Monkey oder Azusa).

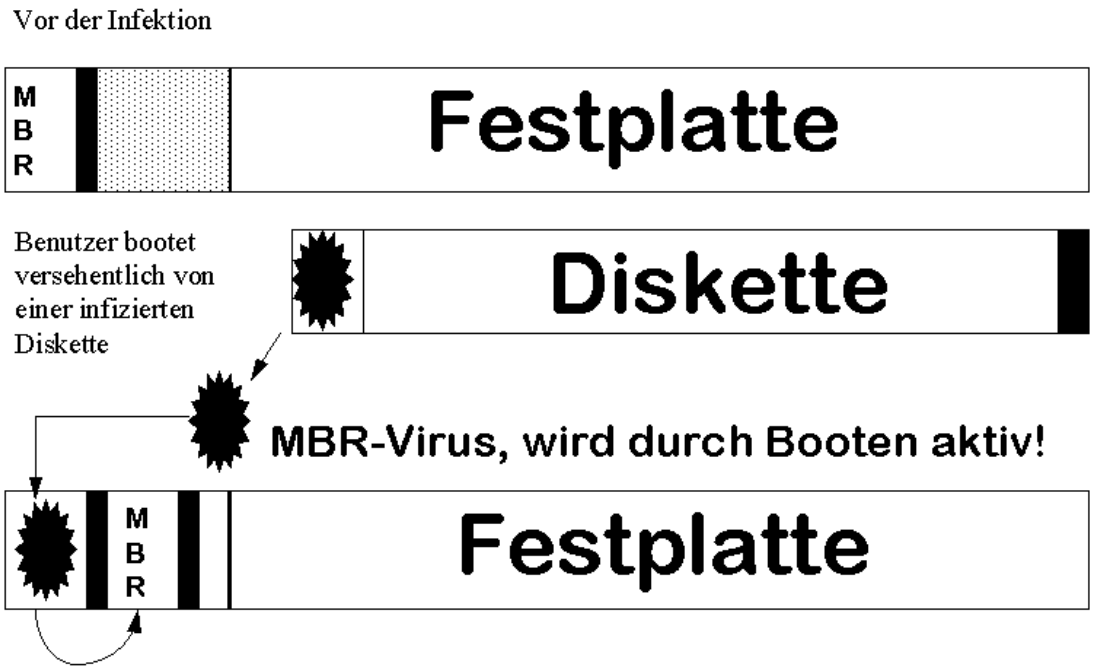


Abb.: Infektionsmechanismus von MBR-Viren

Viren, die den MBR infizieren, müssen mit speziellen Antivirenprogrammen entfernt werden, weil die Partitionstabelle nicht gelöscht werden darf und der Virus durch ein entsprechendes Ladeprogramm ersetzt werden muss. Andernfalls muss die Festplatte formatiert werden! Mit dem Programm Norton Disk Doktor kann unter Umständen solch ein Virus entfernt werden.

Bekannte Beispiele für Bootviren sind: Monkey, Jack The Ripper, Parity Boot, Stoned, Cansu (V-Sign), AntiExe, Michelangelo und Pakistani Brain.

5.2.2 DOS-Bootviren (DBS-Viren)

Diese Sorte Viren benutzen den so genannten DOS-Bootsektor als Wirt, in den sie sich einklinken. Sie werden jedes Mal beim Starten von DOS geladen. Der DOS-Bootsektor ist der erste **logische** Sektor einer Diskette oder einer Festplatte. Er enthält ein kurzes Programm,

anhand dessen DOS das Betriebssystem laden kann, wenn die Partition bzw. Diskette bootfähig ist. Dieses Programm wird nun von einem Bootsektorvirus entweder an eine andere Stelle des Datenträgers kopiert, während der Virus den Bootsektor überschreibt und eine Referenz auf das original Bootprogramm erstellt, oder es wird der Code im Original Bootprogramm manipulieren, so dass beim Booten zuerst der Viruscode ausgeführt wird und dieser sich hinterher zum Originalprogramm rückverzweigt. Es wird also zuerst der MBR ausgeführt, der als aktive Partition DOS angemeldet hat, dann der Virus und erst dann wird der original DOS-Bootsektor nachgeladen! Die meisten Bootviren lassen sich jedoch sehr einfach durch Verwendung des SYS-Befehls entfernen.

Ein Beispiel hierfür ist der Form-Virus, der schon manchen OS/2 Anwender zur Verzweiflung gebracht hat. Szenario: OS/2 ist als aktive Partition in der Partitionstabelle angemeldet. Der Form-Virus lädt jetzt den ersten Sektor der OS/2 Partition, speichert ihn am Ende der Festplatte ab und ersetzt ihn durch sich selbst. Wird jetzt OS/2 gebootet, wird zuerst Form aktiv, der dann versucht, den original OS/2 Bootsektor nachzuladen. Dies kann jedoch fehlschlagen, weil eventuell OS/2 ihn überschrieben hat. Das System hängt dann! Das gleiche Problem kann auftreten, falls eine Komprimierungssoftware wie Stacker oder DoubleSpace verwendet wird.

5.3 Hybridviren

Hybridviren sind Viren, die einerseits Programme und andererseits den Partitionssektor der Festplatte infizieren.

Die Hybridviren werden dann beim "Hochfahren" des Systems schon aktiviert und sind dementsprechend schwierig zu entfernen (Bsp.: Natas, Junkie, Flip-B und Tequila). Inzwischen gibt es Hybridviren, die zusätzlich in der Lage sind, auch Bootsektoren von Disketten zu infizieren! Bekannte Beispiele für Hybridviren sind Natas, Goldbug, Junkie oder Neuroquila.

5.3.1 Multipartite

Eine weitere Bezeichnung für Hybridviren ist "Multipartite". Ein solcher Virus kann Wirte auf verschiedene Weise infizieren, wie z. B. Companiondateien erzeugen und zusätzlich den MBR infizieren.

Der Virus Goldbug, der Partitionssektoren und Diskettenbootsektoren infiziert und zusätzlich Companiondateien erzeugt, ist z. B. ein solcher Multipartite Virus.

5.4 Systemviren (Cluster-Viren)

Diese Virenart benutzt eine besondere Eigenschaft von DOS aus, um ein DOS-Dateisystem zu infizieren. Diese Methode wird bislang ausschließlich vom DIR-II (Creeping Death) Virus und dessen Varianten benutzt. Die zusätzlich verwendeten Techniken lässt DIR-II zugleich in die Gattung der Stealthviren fallen. Geschrieben wurde der DIR-II Virus 1990 in Bulgarien, Varna, von zwei Mathematikstudenten. Der DIR-II Virus ist extrem virulent,

weshalb er zu den meistverbreiteten Viren gehört, obwohl er im Original nur mit DOS 3.0 bis DOS 5.0 Beta funktioniert.

Wirkungsweise: Wird der Virus das erste Mal gestartet, nistet er sich als Bestandteil des Betriebssystems (im CONFIG-Bereich) speicherresident ein. Der Virus bindet sich als weiterer Devicetreiber ein, deklariert sich aber als Bestandteil von COMMAND.COM. Durch die Deklarierung als Betriebssystembestandteil kann man ihn schwerer lokalisieren. Wird jedoch ein schon infiziertes System gebootet, vergrößert er den Bereich des Kommandointerpreters um ca. 1.5 KB! Als nächsten Schritt schreibt sich der Virus selbst in die letzten 1024 Bytes des Datenträgers, d. h. bei Festplatten, 360 KB & 720 KB Disketten in den letzten Cluster, bei 1.2 MB & 1.44 MB Disketten in die zwei letzten Cluster. Diese Cluster werden vom Virus als belegt markiert (als EOF). Daten, die dort standen, sind verloren! Anschließend versucht er auf dem C: Laufwerk die Datei c:" " (Alt-255) auszuführen, was dazu führt, dass DOS das aktuelle Verzeichnis und den ganzen Pfad (PATH=...) durchsucht. Weil der Virus schon aktiv ist, werden beim Durchsuchen alle Programme im Pfad infiziert!

Der Virus selbst verwendet selbst keine Interrupts, sondern fängt DOS Device Treiberaufrufe ab, die er entsprechend modifiziert. Infiziert werden vom Virus Einträge aller ausführbaren Dateien (COM & EXE) in der ersten FAT (File Allokation Table). Der Virus kopiert und verschlüsseltet deren Zeiger auf die ungenützten Einträge in der ersten FAT, während er den Originaleintrag auf sich selbst umsetzt. Ruft der Benutzer nun ein Programm auf, wird aufgrund des umgesetzten Eintrags zuerst der Virus aufgerufen. Dieser sieht nun in seiner "eigenen FAT" nach, welches Programm der Benutzer eigentlich starten wollte und lädt nun dieses von der Festplatte in den Speicher, wo es gestartet wird.

Stealthtechniken: Die Disketten- und Festplattengröße wird vom Virus als um einen Cluster kleiner deklariert, weshalb nicht auf den eigentlichen Virus zugegriffen werden kann. Wenn von DOS aus auf einen infizierten Directorysektor zugegriffen wird, verändert der Virus den Sektor im Arbeitsspeicher derart, dass er uninfiziert erscheint (100% Stealtheigenschaften). Der Benutzer merkt also vom Virus nichts, weil der eigentliche Wirt überhaupt nicht verändert wurde, sondern nur dessen Directory-Eintrag! Solange der Virus im Speicher ist, sehen nämlich alle Verzeichnisse und Einträge völlig normal aus. Bootet man jedoch von sauberen Diskette und schaut sich ein Verzeichnis auf der Festplatte an, so haben alle ausführbaren Dateien die Länge 1024 Bytes (Anmerkung: Länge des Virus) und zeigen auf die gleiche Datei, nämlich den Virus, der sich am Ende des Datenträgers befindet (s. o.). Durch die "Cluster"-Technik kann diese Virenart nicht von Checksummenprogrammen und Monitorprogrammen erfasst werden!

Dieser Virus ist so vermehrungsfreudig, dass er sich, einmal resident, bei bloßen Lesezugriffen auf nicht schreibgeschützte Disketten kopiert und dort in gleicher Weise den Datenträger infiziert. Die Aliasbezeichnung "CD = Creeping Death" (schleichender Tod) ist in diesem Fall sehr treffend! DIR-2 hat sich nach seiner "Freisetzung" durch seine Autoren wie ein Flächenbrand von Bulgarien aus durch Osteuropa über die ganze Welt verbreitet.

5.5 Kernel-Viren

Diese neue Virenart wurde erstmals im Oktober 1994 im russischen Virus "3APA3A" entdeckt. Dieser Virus benützt einen komplexen Infektionsmechanismus, jedoch einen komplett neuen. Wie herkömmliche Bootviren infiziert der Virus Bootsektoren von Disketten auf die "herkömmliche" Art. Seine Art, die Festplatte zu infizieren, ist jedoch (noch) einzigartig. Bei Festplatten wird die Datei IO.SYS bzw. IBMBIO.COM infiziert, die zum DOS Kernel gehört. Der Virus erzeugt hierbei eine Kopie der Datei IO.SYS, die dann jedoch ein anderes Attribut (Volumelabel) trägt. Wird DOS gestartet, wird der Virus durch das Ausführen der Datei IO.SYS aktiv. Weil der Virus nicht die Partitionstabelle infiziert und eine Kopie von IO.SYS als geschützte Volumendatei anlegt, kann der Virus nicht mit FDISK oder SYS entfernt werden.

5.6 Dateiviren (Fileviren)

Unter dem Betriebssystem MS-DOS gibt es verschiedene Programmarten (BAT, COM, EXE, SYS, Overlays, Linklibaries oder Windowsprogramme), die sich teilweise erheblich voneinander unterscheiden. Dateiviren unterscheiden deshalb normalerweise diese Programmarten bei einer Infektion.

Man unterscheidet generell zwischen

- überschreibenden Dateiviren und
- sich anhängenden Viren (Linkviren oder Prepend-/Appendviren)

Eine weitere genauere Klassifikation erfolgt weiter unten. Wie schon unter der Beschreibung zu Bootviren erwähnt, ist es schwierig Hybridviren und Systemviren genau einer Kategorie zuzuordnen, weshalb sie hier nochmals aufgeführt werden.

Abb.: Verschiedene Klassen von Dateiviren

5.6.1 Überschreibende Dateiviren

Der Virus überschreibt den Anfang des befallenen Wirtsprogramms. Das heißt, dass das Wirtsprogramm nicht mehr lauffähig ist! Durch das Überschreiben ändert sich die Dateilänge nicht (falls der Wirt größer als der Virus ist). So ein Virus ist selten,

1. weil er schon bald auffällt, und
2. weil bald kein Programm mehr läuft!
3. weil er meistens nicht speicherresident (= gute Vermehrung) ist

Aus diesen Gründen sind solche Viren leicht zu entdecken, ein Entfernen aus dem Wirt ist jedoch durch das Überschreiben nicht mehr möglich!

Der am meisten verbreitete überschreibende Virus unter DOS dürfte der Burger[11] (560) Virus sein. Die Mini Viren (Länge unter 80 Bytes) sind zwar auch überschreibend, jedoch so kurz (z. B. 22 Bytes), dass das Wirtsprogramm oft noch lauffähig 'bleibt'. Überschreibende Viren sind die einfachste Virengattung, weshalb z. B. der erste OS/2-Virus überschreibend ist.

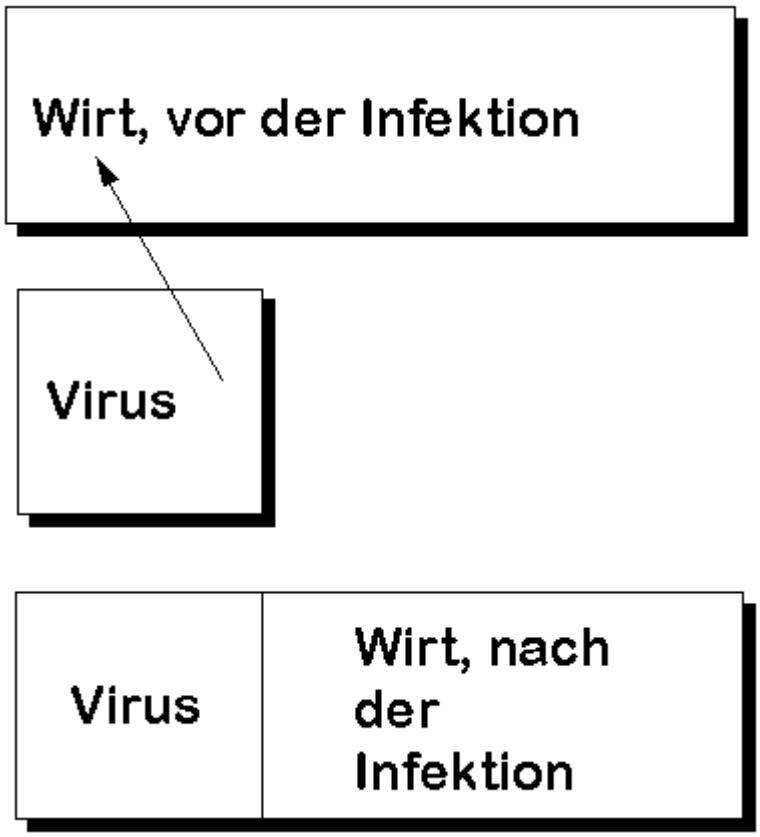


Abb.: Infektionsmechanismus von überschreibenden Viren

5.6.2 Nicht überschreibende Fileviren (Linkviren)

Werden auch Appendviren oder Linkviren genannt. Sie kopieren sich von Programm zu Programm, bis das ganze System verseucht ist. Das Weiterkopieren geschieht durch Aufruf eines verseuchten Programms. Von diesen Viren merkt man lange nichts, weil die verseuchten Programme noch lauffähig sind! Die Lauffähigkeit wird dadurch erreicht, indem sich der Virus

- an das Ende des Wirtes anhängt („to append“) und den Einsprungspunkt des Programms auf den Virencode "umbiegt" oder
- den Wirt überschreibt und die am Anfang überschriebenen Programmteile sichert („to prepend“).

[11] Der Buchautor Ralf Burger hat diesen Virus als Assemblerlisting in einem seiner Bücher veröffentlicht!

Nach erfolgreicher Infektion wird das Programm vom Virus rekonstruiert und ausgeführt. Ein Virus, der beide Arten anwendet, ist der Jerusalem Virus. COM-Dateien enthalten den Virus am Programmanfang, während bei EXE-Dateien der Virus am Ende hängt und der Einsprungspunkt geändert wurde. Dies ist eine der einfachsten Virengattungen, ca. 60 Prozent aller Viren sind Link-Viren.

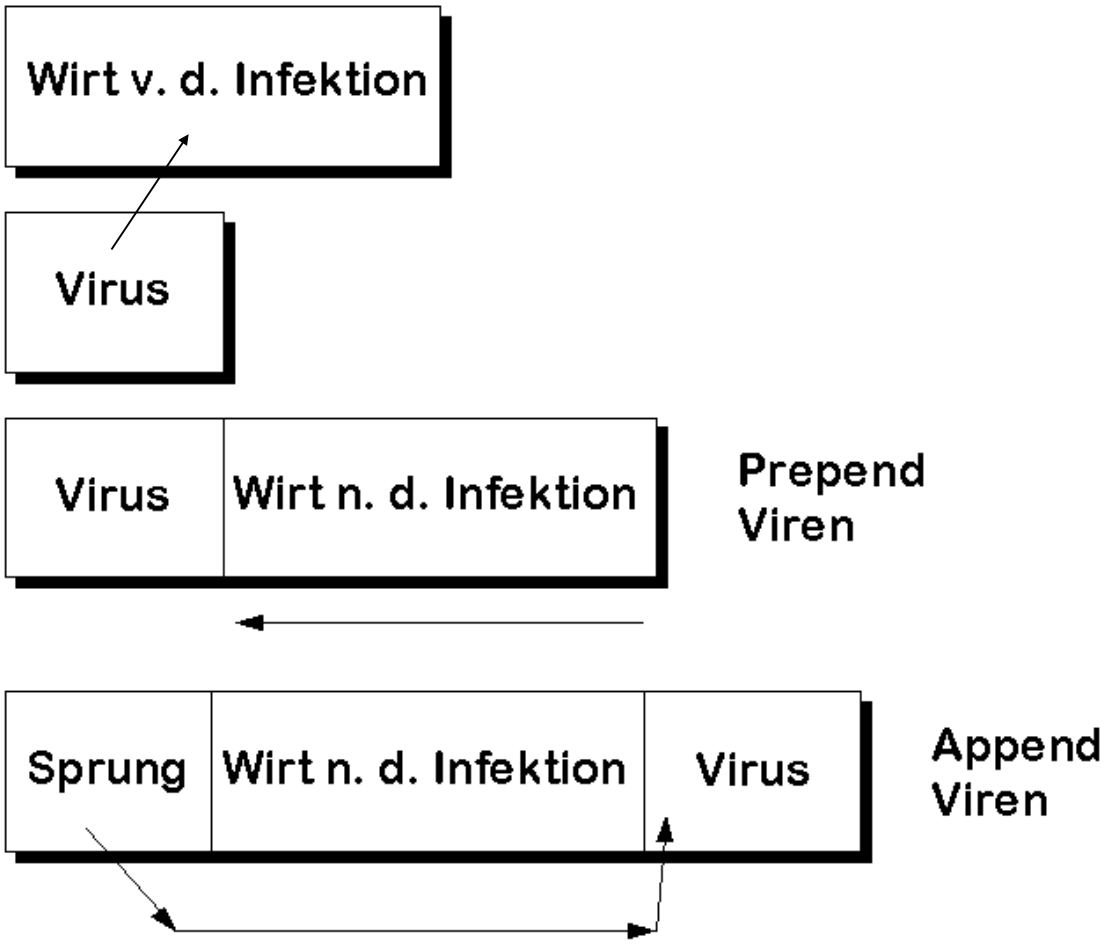


Abb.: Infektionsmechanismus von Linkviren

5.7 Companionviren

Eine eigene Gattung sind die Companionviren (Begleiter). Eine besondere Eigenschaft ist, dass sie sich nicht in den Wirt hineinschreiben, sondern sie infizieren ein Wirtsprogramm, indem sie eine neue, eigenständig ausführbare Datei erzeugen, die beim Aufruf aufgrund gewisser Umstände vor dem gewünschten Programm oder anstatt des gewünschten Programms ausgeführt wird. Diese Datei enthält den Virus, der hinterher das vom Benutzer gewünschte Programm aufruft[12]. Eine beliebte Technik solcher Viren ist es, EXE-Files zu befallen, indem sie im gleichen Verzeichnis eine versteckte COM-Datei gleichen Namens erzeugen. COM-Dateien werden von COMMAND.COM (ab DOS 3.3) immer vor gleichnamigen EXE-Files ausgeführt, so dass der Virus beim Aufruf zuerst aktiv wird. Ein relativ

[12]Dieses Verfahren wird auch von einigen UNIX Viren ausgenützt. Es wird dann z. B. der Programmname „ProgrammXYZ“ in „.ProgrammXYZ“ (bitte beachten: Punkt ProgrammXYZ) umbenannt, welches durch den ls-Befehl nicht mehr sichtbar ist!

neuer Virus (Goldbug) nennt seine Wirte einfach um (EXE-Endung wird durch Leerzeichen ersetzt) und erzeugt eine EXE-Datei, die beim Starten diesen Vorgang wieder rückgängig macht.

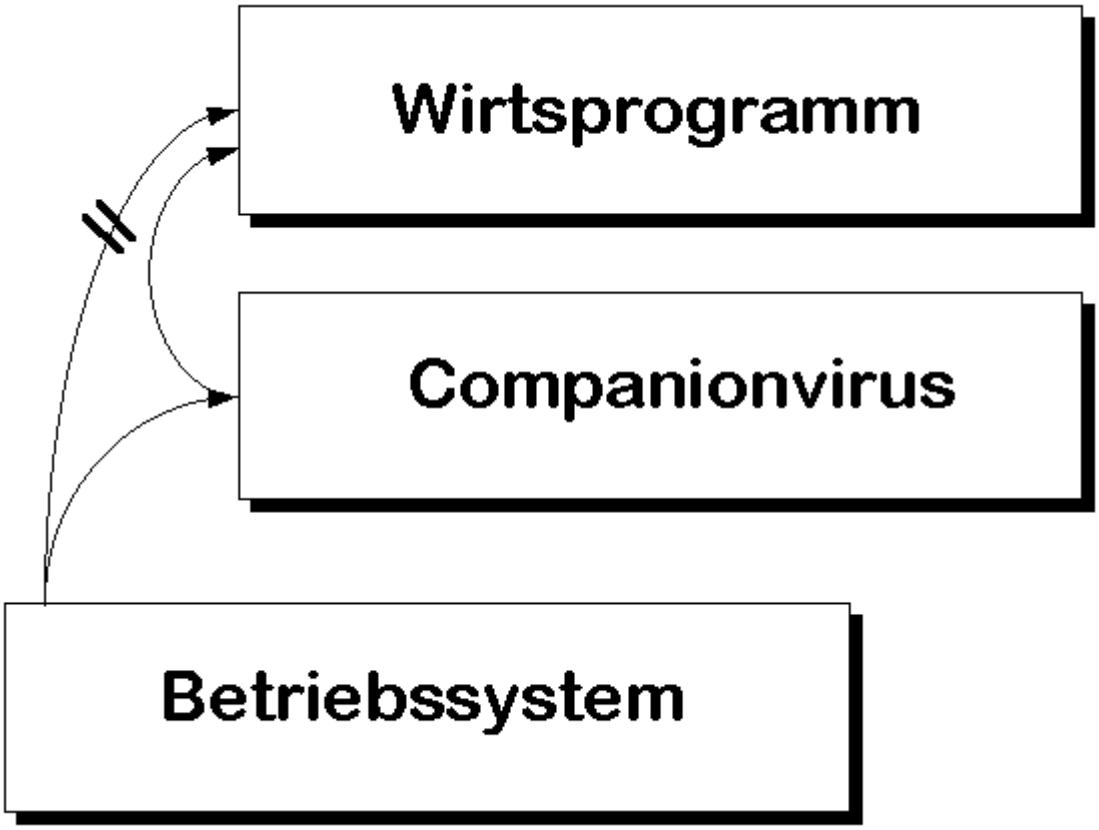


Abb.: Infektionsmechanismus von Companionviren

Companionviren werden oft in Hochsprache (PASCAL, BASIC oder C) geschrieben, weil es fast unmöglich ist, einen Linkvirus in Hochsprache zu schreiben. Da diese Viren das eigentliche Wirtsprogramm völlig unangetastet lassen, ist es für den normalen Anwender extrem schwierig, solch einen Virus zu entdecken.

Insbesondere Prüfsummen-Software, die alle bisher genannten Virengattungen finden würde, könnten an dieser Stelle keine Veränderungen der Datei bemerken. Aufgrund der Eigenart von Companionviren, ihren Wirt zu infizieren, ist es nicht so einfach, eine allgemeingültige Definition für einen Computervirus zu machen.

Einige Virenforscher sehen Companionviren immer noch nicht als Virus an! Nach meiner Definition sind Companionviren aber eindeutig Viren, weil sie Betriebssystemabläufe zu ihren Gunsten modifizieren, und zusätzlich sich reproduzieren.

Ein Beispiel für einen Companion-Virus ist "Little Brother" oder "ACME"

5.8 Trojanische Pferde, Dropper und Scherzprogramme

5.8.1 Trojaner (trojans)

Ein trojanisches Pferd (Trojaner) ist ein Programm, welches vorgibt, zum einen eine nützliche Funktion zu haben, zum anderen aber nach dem Aufruf eine unerkannte unzulässige Nebenwirkung durchführen.

Die meisten Trojaner sind Programme, die unmittelbar nach der Ausführung aktiv werden und z. B. die Platte formatieren oder sonst wie Daten durcheinander bringen (vgl. "Logische Bombe"). Trojaner haben nicht die Fähigkeit, sich zu vermehren, was sie von Viren und Würmern unterscheidet, sind jedoch weitaus zerstörerischer als die meisten Computerviren.

Trojanische Pferde haben oft einen einladenden Programmnamen (ARJ, SCAN, LESBOSEX, MANDY, RUNME oder PORNO); vgl. auch Namensvetter. Diese Programme sind sehr rar, es wurden erst wenige Fälle weltweit bekannt. Trojanische Pferde sind meistens in Hochsprache geschrieben und mit 'Online-Komprimierer' gepackt, um eine Analyse/Entdeckung zu erschweren!

5.8.1.1 Namensvettern (spoofing programs)

Namensvettern sind Trojanische Pferde, die durch ihren Dateinamen den Anwender verleiten, das Programm auszuführen.

Folgende Tabelle zeigt ein paar populäre Programme, die jedoch Trojanische Pferde sind:

Flushot	Version 4
Flushot Plus	Version 1.3
Virus SCAN	Versionen 51, 65, 68, 70 und 72

Tabelle: Einige Beispiele für Namensvetter

5.8.1.2Dropper

Eine spezielle Art eines Trojaners ist ein Dropper (to drop = fallen lassen). Ein Dropper, der meistens verschlüsselt ist, installiert einen Virus als Dateivirus oder überschreibt z. B. den Bootsektor einer Diskette mit einem Bootvirus. Ein Dropper kann sich nicht vermehren, jedoch der vom Dropper installierte Virus!

5.8.2 Logische Bomben (bombs)

Eine logische Bombe ist eine Spezialart eines Trojanischen Pferdes. Logische Bomben sind lauffähige Programmteile, die in einen nützlichen Code eingebettet sind und aus einem "Auslöser" (engl. trigger) und einer "Nutzlast" (engl. payload) bestehen.

Ihre zerstörerischen Funktionen werden eine gewisse Zeit lang überhaupt nicht aufgerufen. Später, wenn irgend eine Triggerbedingung erfüllt ist (z. B. ist ein bestimmtes Datum erreicht oder das Programm 100 mal aufgerufen wurde), "explodiert" die Bombe und ruft ihre Zerstörungsfunktion auf.

Im erweiterten Sinn kann man auch den zerstörerischen Teil eines Virus als logische Bombe bezeichnen, wenn der Virus auf die Erfüllung seiner Triggerbedingung wartet, um dann aktiv zu werden, z. B. Löschen von Programmen am Freitag, den 13 ten oder wie beim Michelangelo Virus, das Formatieren der Festplatte am 6. März. Es gibt auch Trojanische Pferde und Viren, die harmlose Programme so abändern, dass aus diesen Programmen eine logische Bombe wird! Beispiele hierfür sind mehrere MMIR-Viren.

5.8.2.1Zeitbombe (time bomb)

Wird die Ausführung der logischen Bombe an eine Zeitbedingung gekoppelt, spricht man auch von einer Zeitbombe.

Sharewareprogramme, die z. B. die 30 tägige Testphase erlauben, haben so gesehen eine Zeitbombe einprogrammiert.

5.8.3 Scherzprogramme (jokes)

Sie sollen lediglich jemanden erschrecken und zur allgemeinen Belustigung dienen, ohne schädlich zu sein (vgl. Trojaner) oder sich selbst zu vermehren. Meist fängt der Computer nach dem Aufruf eines Scherzprogramms irgendwann an, eine Melodie zu spielen oder

sonst etwas Ungewohntes auf dem Bildschirm anzuzeigen. Scherzprogramme haben gemeinsam, dass sich das Computersystem abnormal verhält, wenn ein Scherzprogramm aktiv ist (vgl. Softwareanomalien). Gerade Computerneulinge bekommen -bei Aktivierung des Scherzprogrammes- einen gehörigen Schreck oder richten in Panik und Unwissenheit sogar tatsächlichen Schaden an. Ferner kann es durch unsaubere Programmierung des Scherzprogrammes zu ungewollten Seiteneffekten kommen.

Es gibt jedoch auch Scherzprogramme, die von Hackern in Trojanischen Pferde umgewandelt wurden (z.B.: der SLOD-Trojan). Der Cascade Virus (Herbstlaub) entstand ebenfalls aus einem Scherzprogramm, dem Programm „FALLDOWN“.

5.9 Viren, die noch nicht aufgetreten sind

Der nachfolgende Abschnitt beschreibt einige „Virenarten“, die noch nicht entdeckt wurden oder einfach nur Mythen sind, die sich hartnäckig halten.

5.9.1 Geisterviren

Geisterviren sind Viren, die von Antivirensoftware fälschlicherweise gefunden werden.

Hierzu gibt es verschiedene Gründe:

- Ein Virensuchprogramm hinterlässt seine Sucherkennungen unverschlüsselt im Speicher. Das Konkurrenzprodukt, welches die gleichen Erkennungen verwendet, findet beim Durchsuchen des Arbeitsspeichers jetzt einen "Geistervirus" (z. B. TSAFE, VSAFE oder VDEFEND).
- Ein Programm wurde mit einem selbstüberprüfenden Virenschutz versehen, der für ein Virensuchprogramm wie ein Virus aussieht (z. B. TNT-Virus, F-XLOCK oder VSS). Gerade regelbasierende Virensuchprogramme melden oft einen solchen Selbstschutz als verdächtig an, weil dieser vom Codeaufbau ähnliche Strukturen wie ein Dateivirus besitzt.
- Wenn eine verseuchte Diskette unter DOS mittels des DIR Befehles betrachtet wird: DOS speichert den Bootsektor der zuletzt eingelegten Diskette aus Performancegründen im Arbeitsspeicher zwischen. Das Virensuchprogramm findet jetzt einen Geistervirus im niedrigen Arbeitsspeicher, obwohl dieser Virus tatsächlich nie aktiv ist, sondern nur im Diskettenpuffer zwischengespeichert wurde! Beim nächsten Einlegen einer virenfreien Diskette oder nach einem Kaltstart ist der Virus durch die neuen Daten überschrieben worden.

5.9.2 "Hide and Seek" Viren

Das sind Viren, die sich nur eine gewisse Zeit innerhalb des Systems aufhalten. Als Verstecke könnten beispielsweise die Pufferbereiche intelligenter Terminals oder DFÜ-Einrichtungen dienen.

5.9.3 „Call" Viren

Diese Viren verändern unter Umständen nur ein einziges (!) Byte am Wirtsprogramm. Der eigentliche Virus wird im Massenspeicher abgelegt und wird lediglich durch den Aufruf des infizierten Programms aktiviert. Der „Nachteil" dieser Art von Viren ist aber, dass beim Fehlen des eigentlichen Virusprogramms auf dem externen Massenspeicher die infizierten Programme nicht mehr lauffähig sind! Eine Verwendung im PC-Bereich ist mir nicht bekannt. Der weiter oben beschriebene DIR-II Virus (Cluster-Virus) verwendet vom Prinzip her eine ähnliche Taktik!

5.9.4 Gepufferte Viren

Das sind Viren, die sich ins batteriegepufferte RAM einnisten und ähnliche Eigenschaften wie Hardware Viren haben. Durch Entfernen der Pufferbatterie sind diese Viren leicht zu entfernen! Es muss jedoch damit gerechnet werden, dass sich die Viren von infizierten Programmen aus beim erneuten Starten des Systems wieder ins gepufferte RAM installieren. Ein typisches Beispiel hierfür sind Viren für den Amiga.

Hinweis: Es herrscht oft die falsche Annahme, Viren könnten sich im CMOS von PCs einnisten, was aber grundsätzlich falsch ist! Das CMOS kann nur Daten speichern, jedoch keinen ausführbaren Maschinencode!

5.9.5 Hardware Viren

Diese Art von Viren ist zurzeit noch nicht existent, weil sie nur durch Veränderung der Hardware ins System eingebracht werden können. Beispiel: Austausch eines Boot-ROMs. Diese Art von Viren ist zwar recht schwierig zu installieren, aber sehr effektiv, da sie sehr schwer ausmachbar ist. Es muss jedoch beachtet werden, dass dies keine Softwaremanipulation ist, sondern eine Hardwaremanipulation! Der Virus „BIOS Menegetis" ist von seiner Eigenschaft am ehesten als Hardwarevirus einzuordnen.

5.9.6 „Virus Desinformaticus"

Die meisten Computeranwender sind schon einmal mit dem Thema „Viren" über die Presse, durch den Rundfunk, von Freunden oder auch durch entsprechende Software konfrontiert worden. Die Medien verstehen es, dieses Thema publikumswirksam aufzubauschen, jedoch nutzbare Informationen werden selten weitergegeben. Viel schlimmer noch, die meisten Anwender werden dadurch regelrecht verunsichert. In unüberlegten Aktionen und durch nicht vorhandenes Wissen werden mehr Daten, Zeit und letztlich Geld zerstört, als es je ein Virus könnte. Viele Benutzer wissen deshalb nicht, welche Folgen eine Vireninfektion des Computers haben kann und wie sie sich bei einer Infektion verhalten sollen oder sich davor schützen können. Deshalb ist es bitter nötig, ein Bewusstsein für Viren unter den Computeranwendern zu schaffen.

Für „Hygienehinweise" und Disziplinmaßnahmen zum Umgang mit Computerviren sowie zur Auswahl von geeigneter Antiviren Software möchte ich auf folgende Literatur verweisen: [Esch93], [FAQG93], [FAQL92] und [PB92].

6 Verschiedene „Attribute“ von Viren

In diesem Kapitel möchte ich die verschiedenen Attribute von Computerviren näher erläutern. Komplexere Viren machen oft von mehreren Attributen Gebrauch, um sich dadurch erfolgreicher vermehren zu können.

6.1 Unterscheidung nach Infektionsmechanismus

6.1.1 Direct Action-Viren

Diese Virenart installiert sich beim Aufruf nicht resident im Speicher, belegt also auch keinen Speicherplatz. Vielmehr durchsucht der Virus direkt beim Ausführen die Laufwerke und Verzeichnisse, für die er programmiert wurde, nach Wirtsprogrammen und versucht, diese sofort zu infizieren. Man unterscheidet grob folgende Suchverfahren:

- im aktuellen Verzeichnis (current directory)
- 'Dot Dot'-Methode (rekursiv im nächst höheren Verzeichnis -> CD ..)
- über die Variable PATH
- zufällige Verzeichniswahl

Danach läuft das Programm, welches eigentlich gestartet werden sollte, ganz regulär ab. Solche Viren brauchen keine Interruptvektoren zu verbiegen und sind von einigen residenten Wächterprogrammen, die nur Interruptvektoren kontrollieren oder überprüfen, ob sich ein Virus im Speicher einnistet, nicht zu entdecken. Sie verraten sich aber oft durch lange Zugriffszeiten bei sonst recht schnell aufgerufenen Programmen, weil das Suchen nach Wirtsprogrammen mitunter recht lange dauert. Oft haben solche Viren überhaupt keine Fehlerbehandlung und fallen dann auf, wenn z. B. ein Schreibschutzfehler bei schreibgeschützten Disketten auftritt.

"Direct-action"-Infektoren können keine herkömmlichen Stealth-Techniken ausnutzen, weil dazu der Virus resident im Speicher sein muss. Die einzigen Stealthfunktionen, die solche Viren tragen können, sind das Ausschalten von Wächterprogrammen, das Benützen von SFT (s. u.) oder Tunneln. Beispiele für "Direct-Action"-Viren sind der Vienna, Danish.Tiny und VBasic Virus sowie praktisch alle Viren mit einer Größe unter 180 Bytes. Solche Viren sind relativ einfach zu programmieren, weshalb sie die Mehrzahl der Viren darstellen!

6.1.2 Speicherresidente Viren (TSR)

Bei dieser Virengattung handelt es sich fast ausnahmslos um Linkviren, die sich als TSR (TSR = Terminate but stay resident) permanent im Speicher installieren und Funktionen von DOS übernehmen, überwachen und entsprechend manipulieren oder um Bootviren, die

Sektorzugriffe manipulieren. Hierzu werden entweder Interruptvektoren verbogen ("Interrupt-Hooking"), oder der Virus deklariert sich als Bestandteil des Betriebssystems selbst. Viren, die sich resident im Speicher befinden, können ihre potentiellen Wirte auf wirkungsvollere Weise attackieren als "Direct-action"-Infektoren, weshalb die bekanntesten Viren fast alle speicherresident sind. So können sie z. B. gewisse Stealth-Techniken ausnützen (siehe "Stealthviren"). Die meisten warten nur darauf, dass der Anwender eine potentielle Wirtsdatei startet, um sie im selben Moment zu infizieren. Oft wird dabei der "Critical-Error-Handler" (Fehlerbehandlung) von DOS ersetzt, damit bei misslungener Infektion (z. B. aufgrund eines Schreibschutzes bei Disketten) keine Fehlermeldung erscheint. TSR Viren erzeugen meistens jedoch Nebeneffekte, weil diese Virenart mit anderer Software interferiert.

6.1.2.1 Unterscheidung der TSR-Viren nach Speicherbelegungsstrategien

Es gibt verschiedene Plätze im Speicher, in denen sich unter DOS Viren einnisten. Folgende Bezeichnungen sind gebräuchlich:

- TOM
- Systemholes
- MCB
- HMA
- UMB

6.1.2.1.1 TOM (Top Of Memory)

Diese Virengattung dekrementiert den TOM-Zeiger (TOM = Top Of Memory, ist normalerweise 640 KB), der die Obergrenze des konventionellen DOS-Speichers angibt, und installiert sich in dem Bereich, der dann für DOS nicht mehr belegt wird. Diese Technik wird von vielen Dateiviren (G2 & PS_MPC) und fast ausschließlich von allen Bootviren verwendet (Ausnahmen: Horse.Boot und Hybrid-Viren). Man erkennt den fehlenden Speicher daran, dass beim CHKDSK für den "Konventionellen Arbeitsspeicher" nicht mehr 640 KB (655360 Bytes), sondern ein niedrigerer Wert angezeigt wird. Dies ist aber noch kein sicheres Zeichen für eine Infektion, da manche Rechner diesen Bereich als BIOS oder SCSI Stack-Bereich benutzen. Trotzdem sollte man in diesem Fall sehr vorsichtig sein.

6.1.2.1.2 Systemholes (Systemlöcher)

Diese Virengattung sucht nach nicht verwendeten Speicherbereichen im DOS-Kernel oder anderen residenten Programmen (COMMAND.COM) und kopieren sich dort hinein. Diese verändern die Größe des freien Speichers nicht, wodurch sie schwierig zu finden sind. Solche Viren sind jedoch oft an eine bestimmte DOS-Version gebunden. Viele, sehr kurze Viren nisten sich auch in anderen Löchern des Arbeitsspeichers ein. Als Beispiel sei hier der obere Teil der Interrupttabelle oder der ungenutzte Videospeicher genannt.

6.1.2.1.3 MCB (Memory Control Blocks)

DOS verwendet intern MCB's zur eigenen Speicherverwaltung, die auch vom Anwendungsprogramm angefordert werden können. Viren, die MCB benützen, installieren sich als TSR im normalen Speicher zwischen 0 und 640 KB. Diese kann man unter Umständen mit Speicher-Map-Utilities wie z. B. MEM von DOS, Resident oder MapMem entdecken. Fast alle Viren „entfernen“ den anforderten MCB aus der MCB-Kette und markieren diesen als DOS-Bestandteil, um eine Entdeckung zu erschweren. Modernere Viren versuchen jedoch einen MCB im hohen Speicher anzufordern (s.u.), der dann über der magischen 640 KB Grenze liegt.

6.1.2.1.4 HMA und UMB

Eine weitere Möglichkeit für TSR-Viren ist, sich resident in so genannte UMBs (Upper Memory Blocks) oder in die HMA (High Memory Area) zu installieren. Aus diesem Grunde sollte man diese Speicherbereiche immer mituntersuchen lassen. Diese Methode wird immer beliebter, weil heute fast jeder Rechner über hohen Arbeitsspeicher (HMA & UMB) verfügt.

6.2 Unterscheidung nach Infizierungsgeschwindigkeit

Man kann hier auch noch zwischen den Infizierungsarten unterscheiden, die sich dann in der Vermehrungsfreudigkeit des Virus bemerkbar machen. Hinter der Infizierungsgeschwindigkeit steht die "Philosophie" des Virenautors. Es wird grob zwischen folgenden Arten unterschieden:

- Normale Infizierung
- Schnelle Infizierung
- Langsame Infizierung
- Spärliche Infizierung

Die Infizierungsgeschwindigkeit wird unabhängig von dem Infektionsmechanismus betrachtet.

6.2.1 Normale Infizierung

Eine Infizierung erfolgt gewöhnlicherweise bei Programmausführung[13] oder beim Öffnen des Wirtes. Dies ist die gewöhnliche Verbreitungsart der meisten Viren.

[13] Hierzu wird -unter DOS- meistens die EXEC-Funktion 21h/AH=4Bh vom Virus kontrolliert.

6.2.2 Schnelle Infizierung (Fast-Infector)

Es gibt Viren, die jede aus irgendeinem Grunde geöffnete[14] ausführbare Datei infizieren oder z. B. schon beim Lesen eines Verzeichnisses darin vorhandene Wirtsprogramme infizieren. So infiziert der DIR Virus Dateien nur bei Anwendung des DOS-Befehles DIR oder der Lawine Virus zusätzlich bei einem Verzeichniswechsel. Diese Virenart wird "Fast Infector", (schneller Infektor) genannt, weil sie sich mit rasanter Geschwindigkeit in infizierten Systemen ausbreitet. Durchsucht man zum Beispiel die Platte mit einem Virenscanner, der den Virus nicht im Speicher erkennt und den Suchvorgang daher nicht abbricht, so wird durch das Scannen jede untersuchte Datei infiziert! Ein Beispiel für einen "Fast Infector" sind der "Dark Avenger" (welcher der erste war, der diese Technik verwendete), Frodo, DIR-II oder "Tiny.CPP".

6.2.3 Langsame Infizierung (Slow-Infector)

Eine weitere Variante von TSR-Viren sind die "slow infectors", die "langsamen Infektoren". Ihre große Gefahr liegt darin, dass sie selbst einen ansonsten „narrensicheren“ Schutzmechanismus eines Integritäts-Prüfers, eines Wächterprogrammes oder sogar den eines Hardwareschutzes umgehen können. Diese Viren infizieren eine Wirtsdatei nur dann, wenn der Benutzer absichtlich in diese Datei schreibt, etwas löscht oder ein Programm z. B. durch Kompilation neu erzeugt oder sonst wie modifiziert. Dadurch können Veränderungen ausführbarer Dateien von Wächterprogrammen nicht mehr als illegal erkannt werden, weil der Benutzer ja die befallene Datei selbst verändert hat. Ein neuer Virentyp dieser Gattung ist z. B. der "Objective"-Virus, der sich nur bei der Erzeugung von COM Programmen aus Objektdateien vermehrt. Anmerkung: Viele der EXE-Headerviren (s.o.) sind Slow-Infectors.

6.2.4 Spärliche Infizierung (Sparse)

"Slow Infectors" vermehren sich, wie der Name schon sagt, sehr langsam. Dem Benutzer fällt somit das Vorhandensein eines Virus im System deshalb kaum auf. Auch sind Viren so programmiert worden, dass sie nur zu bestimmten Zeiten aktiv werden (z. B. Infizierung nur an Sonntagen, nur im Herbst oder erst nach 17 Uhr usw.). Die Eigenschaft, sich nur zu bestimmten Ereignissen zu vermehren, bezeichnet man als "Sparse". Ein Beispiel ist der Toni Virus, der am Monatsersten nur Dateien infiziert, die mit „A“ anfangen, am zweiten Tag Dateien, die mit „B“ beginnen usw.

6.3 Tarnkappenviren und Tarnmethoden

6.3.1 Stealthviren

Die Stealthviren gehören zur so genannten 4. Generation der Viren und stellen eine der gefährlichsten Gattungen dar.

[14] Der Virus kontrolliert hierzu die FileOpen-Funktion von DOS, z. B. 21h/AH=6Ch oder AH=3Ch

Stealthviren werden oft auch als Tarnkappenviren bezeichnet, weil er für den Anwender nicht sichtbar ist. Ein Stealth-Virus besitzt die Eigenschaft, sämtliche Dateizugriffe zu überwachen und entsprechend zu manipulieren. Die Manipulation besteht auch darin, beim Öffnen einer infizierten Datei den vom Virus infizierten Programmteil herauszufiltern. Mittels dieser Taktik wird selbst einem Virensuchprogramm eine unverseuchte Datei suggeriert, mit dem Ergebnis, dass kein Virus gefunden wird, wenn der Stealth-Virus aktiv ist. Man spricht hier von "Pure-Stealth"-Viren (100% Stealthviren) und "disinfect on the fly" ("im Vorbeiflug desinfizieren"). Ein Längenzuwachs infizierter Dateien mittels des DIR Befehles ist **NICHT** erkennbar! Stealthviren sind typisch für das Betriebssystem DOS!

Es gibt Stealthviren, die Bootsektoren (z. B. Natas, Brain, Stoned III), Partitionssektoren (z. B. Natas, Tequila, Parity Check) und/oder Dateien (z. B. Natas, 4096, Holocaust, Tremor, Neuroquila, Lockout oder Pure) befallen!

6.3.2 Dir-Stealthviren

Dir-Stealthviren[15] sind meistens nicht -wie z. B. die Stealthviren- in der Lage, den Virencode aus dem Infizierten Programm "herauszufiltern". Eine Längenänderung ist jedoch nicht erkennbar (z. B. mit DIR). Die so genannten Slackbereichviren (LeHigh & ZeroHunt) werden ebenfalls zu den Dir-Stealthviren gezählt, weil für den Anwender ein Längenzuwachs NICHT erkennbar ist! Der Unterschied besteht darin, dass mit einem Checksummenprogramm bei aktivem Virus dieser erkannt wird, während "echte" Stealthviren selbst Checksummenprogramme umgehen! Beispiele: ZeroHunt, Diamond oder Dark Avenger III.

6.3.3 Tunneln (Tunneling)

Mit "tunnelnden" Viren bezeichnet man eine Gruppe Viren, die versuchen, residente Monitorprogramme (z.B.: SDRES oder VSAFE) zu umgehen. Diese Monitorprogramme erkennen unter anderem, wenn illegal Interrupts aufgerufen werden oder eine andere virentypische Aktion durchgeführt wird und warnen dann den Benutzer. Hierzu binden sie sich, ähnlich wie Viren, in die überwachenden Interrupts ein[16]. Anschließend warten auf sie einen Interruptaufruf, den sie dann zurückverfolgen können. Ein tunnelnder Virus verfolgt nun innerhalb des Speichers den Weg, den ein Interruptaufruf nehmen würde, und sucht auf diese Weise im BIOS bzw. im DOS-Kernel nach dem Anfang des Original-Interrupthandlers. Hat er ihn gefunden, ruft er diese Adresse direkt auf[17]. Dadurch wird das Wächterprogramm umgangen, und der Virus hat sein Ziel erreicht. Der erste Virus, der diese Technik verwendete, war der Yankee Doodle (1989), der damit das eigene Wächterprogramm (VACSINE) des gleichen Autors umging!

Man kann sich diese Technik wirklich bildlich vorstellen: Um eine Festung zu erstürmen, wird ein Tunnel gegraben, um dann von innen angreifen zu können. Man kann also tunnelnde Viren im weiteren Sinne zu den Stealthviren rechnen. Die meisten Stealthviren verwenden deshalb auch diesen Techniken, um Wächterprogramme umgehen zu können.

[15] Eine andere Bezeichnung ist auch FCB-Stealth

[16] Meistens handelt es sich um die Interruptes 01h, 09h, 13h, 15h, 21h, 2Fh und 40h, siehe auch [Brow44].

[17] Siehe auch Anmerkungen zur Tunnelroutine des Programmes MBR-Kill.

Diese Technik, das so genannte "Interrupt Tracing", wird von guten residenten Monitoren mittels spezieller Methoden verhindert. Sehr viele jedoch können so umgangen werden. Neben dem Interrupt Tracing gibt es weitere sehr effektive Tunneltricks. Eine sehr fortgeschrittene Technik benützt das Programm MBR-Kill, das eine so genannte Emulation-Engine besitzt und damit fast alle Programme und Viren umgehen kann, die das „normale“ Tunneln abfangen können! Weitere Techniken will ich hier noch kurz erwähnen: die Benutzung eines undokumentierten Multiplex-Aufrufes, der die Einsprungsadresse des original Disk-Interrupt-Handlers im BIOS frei Haus liefert[18], das Verwenden des von kaum einem Monitorprogramm überwachten Dateihandles CON und das Ersetzen des Block-Device-Treibers für die Laufwerke. Die weiteren Methoden werden weiter unten detailliert beschrieben.

Anzumerken ist, dass auch viele Viren Monitorprogramme erkennen können und teilweise deaktivieren oder entfernen. So werden z. B. die Programme VSAFE & TSAFE von jedem besseren Virus deaktiviert!

6.3.4 System File Table (SFT) Techniken

Wenn ein Virus versucht, ein Wirtsprogramm zu infizieren, so muss er hierzu die Schreiberlaubnis besitzen. Dieser Umstand wird von vielen Wächterprogrammen ausgenützt, um virentypische Aktionen erkennen zu können.

Es gibt jedoch einen Trick, solche Wächterprogramme zu umgehen, indem die zu infizierende Datei im Lesemodus geöffnet wird und intern (im Arbeitsspeicher) Systemdaten vom Virus so geändert werden, dass der Zugriffsmodus auf „Schreiben“ gesetzt wird. Dies kann sehr elegant durch die so genannten System File Tables von MS-DOS erfolgen. Alle mir bekannten Wächterprogramme können so umgangen werden! Ein weiterer "Vorteil" von der Verwendung von SFT ist, dass einige Dateizugriffe vermieden werden können, was zu einem sehr schnellen Infizieren führt!

6.3.5 Cavity Viren

Unter Cavity-Viren versteht man Viren, die in einem Wirt einen Bereich suchen, der aus lauter gleichen Zeichen (in der Regel Nullen) besteht. Wenn die Anzahl dieser Zeichen groß genug ist (größer als die Virenlänge), kann der Virus, analog zu einem Komprimierungsprogramm, diesen Bereich für seinen eigenen Code nutzen.

Hierzu merkt sich der Virus das Zeichen und die Anzahl, wie oft dieses Zeichen an dieser Stelle auftritt. Dann wird dieser Bereich mit dem eigenen Viruscode überschrieben. Der Dateianfang wird genauso manipuliert, wie bei normalen Dateiviren. Der Unterschied zu Dateiviren besteht darin, dass kein Code mehr an den Wirt angehängt, sondern direkt in den Wirt hineingeschrieben wird, ohne den Wirt aber dabei zu schädigen. Logischerweise nimmt die Dateigröße des Wirtes bei dieser Infektionsart nicht zu.

[18] Diese Technik benützt das Programm MBR-Kill ebenfalls!

Man unterscheidet zwischen folgenden Cavity-Virengattungen:

- Header-Viren
- Zerohunting-Viren
- Stackbereich-Viren

6.3.5.1 Header-Viren

1994 tauchte eine ganze Serie neuer Viren auf, die sich alle folgender Technik bedienen: EXE-Programme haben oft hinter dem eigentlichen Programmkopf (EXE-Header) einen Bereich von ca. 480 Bytes, der ungenützt ist und aus Nullen besteht. Die Header-Viren überschreiben diesen Bereich mit ihrem Code. Um aktiv zu werden. muss der EXE Programmkopf so abgeändert werden, dass der Virus zuerst aktiv wird. Dies erfolgt meistens durch das Überschreiben der „MZ“-Erkennung des EXE-Header mit einen Sprungbefehl (near jump) auf den Virus. Solch eine infizierte Datei sieht jetzt für DOS wie eine normale COM-Datei aus, weil der EXE-Header teilweise zerstört ist. Weil COM-Dateien maximal 64 KB groß sein dürfen, werden von dieser Virenart auch nur EXE-Programme infiziert, die kleiner als 64 KB sind. Eine andere Methode den Programmkopf abzuändern ist unter 4.) beschreiben.

Wird solch ein infiziertes Programm gestartet, wird dadurch der Virus aktiv. Allen Header-Viren ist jedoch gemeinsam, dass es keinen Programmzuwachs gibt, weil einfach Teile des Programms überschrieben wurden. Es gibt zurzeit vier unterschiedliche Methoden, wie der Virus das Originalprogramm wiederherstellt und startet:

1. Relocation des EXE-Programms durch den Virus mit anschließendem Start (z. B. BootEXE-Virus).
2. Installieren eines speicherresidenten Disketteninterrupthandlers, der bei Lesezugriffen den Virus herausfiltert und bei Schreibzugriffen geeignete Programme infiziert (siehe auch „Slow Infector“). Danach versucht der Virus, das Programm neu zu starten. Weil der Disketteninterrupthandler aktiv ist, wird der Virus beim Startvorgang herausgefiltert und das Programm scheint unverseucht zu sein. Solche Viren haben durch die verwendete Methode 100-prozentige Tarnkappeneigenschaften. Diese Methode wird von mehreren Viren verwendet, z.B. Pure, XYZ, Cluster oder Skid_Row.
3. Eine weitere Methode ist, beim Programmstart den Master Boot Record der Festplatte mit dem Virus zu infizieren und anschließend das System neu zu booten. Der Virus wird nun als erstes Programm aktiv und kann die gleichen Tarnmethoden wie unter 2.) beschrieben verwenden. Startet man ein verseuchtes EXE-Programm, wird der Virus beim Laden herausgefiltert, d.h. es liegt das originale Programm vor, der Virus wird gar nicht mehr ausgeführt. Das besondere an diesem Virus ist, dass er multipartite ist, d.h. er infiziert Programme, also auch den Master Boot Record der Festplatte. Zusätzlich hat solch ein Virus hervorragende Tarnkappeneigenschaften. Ein Beispiel ist hierfür der Lockout Virus, der insgesamt nur 331 Bytes lang ist!
4. Eine weitere Möglichkeit besteht darin, den EXE-Kopf intakt zu lassen und nur dessen Einsprungsadresse auf den Virus umzubiegen. Durch diese Taktik können auch EXE-Dateien größer als 65 KB infiziert werden! Bei Programmausführung installiert der Virus einen INT 13h-Handler der den Virus bei Lesezugriffen herausfiltert. Anschließend

versucht der Virus das Programm neu zu starten. Weil der Diskketteninterrupthandler aktiv ist, wird der Virus beim Startvorgang herausgefiltert und das Programm scheint unverseucht zu sein.

6.3.5.2Stackbereich Viren und "Zerohunting"

Es gibt auch Stackbereich-Viren, die sich -ähnlich den Header Viren- in ungenutzten Programmteilen aufhalten und diese überschreiben, weshalb auch eine Längenänderung nicht auftritt! Solche Viren suchen innerhalb von Wirtsprogrammen nach Ketten gleicher Zeichen (gewöhnlich nach Nullen, z. B. Stackbereich des Programms) und überschreiben diesen Bereich, wenn er lang genug ist, mit ihrem Code. Hierzu zählen u. a. die Viren

- LeHigh (im Stackbereich des Programms COMMAND.COM)
- ZeroHunt und Darth Vader (in COM Dateien)

Da diese Bereiche vom Virus anschließend wieder mit Nullen aufgefüllt werden, sind fast keine Symptome zu erkennen. Heute sind jedoch fast alle Programme komprimiert, weshalb Stackbereich-Viren fast keine potentiellen Wirte mehr finden!

6.3.6 Slackbereich Viren

Die Slack-Area ist der Bereich des letzten Clusters eines Programms, der nicht mehr vom Programm belegt wird. Bei größeren Festplatten ist somit die Slack-Area 4 KB und größer.

Diese Virengattung ist sehr hinterhältig, weil sie sich nur sehr schwer entdecken lässt und zudem weil sie äußerst destruktiv ist. Die Wirkungsweise ist folgende:

Der Virus kopiert sich hinter den Wirt in die so genannte Slack-Area, ohne jedoch die Länge des Wirtes zu ändern. Diese Methode wurde jedoch noch nicht in existierenden Viren gefunden! Eine andere Methode ist es, den Anfang des Wirtes in die Slack-Area zu kopieren (zu sichern) und den Wirt mit dem Viruscode zu überschreiben. Beim Start des Virus lädt dieser die entsprechenden fehlenden Programmteile nach. Beispiel: Number of the Beast (666).

Eine Längenänderung des befallenen Programms ist also aus Sicht des Virus nicht nötig, weshalb z. B. Integritätsprüfer diesen Virus nur bei eingeschalteter Checksumme findet. Diese Viren sind nicht ungefährlich, weil sie in vielen Fällen zerstörte Programme und querverkettete Dateien ("cross linked files") erzeugen. Ist die Slack-Area eines Opfers nämlich zu klein für den Virus, so belegt dieser auf dem Medium auch noch den nächsten Sektor, ohne dass dafür jedoch ein Eintrag in die FAT erfolgt. Ist oder wird dieser Sektor nun legitimerweise von einer anderen Datei belegt, so entsteht ein querverketteter Sektor, und das Wirtsprogramm ist nicht mehr startfähig, da der Viruscode überschrieben wurde. Hängt der Virus sich jedoch vor den Wirt, so ist das Ende des Wirtes zerstört (z.B.: Number of the Beast)! Weil die Länge des Wirtes nicht angepasst wird, wird bei einem einfachen Kopieren auch nicht die Slack-Area mitkopiert, was wiederum zum Abscheiden von Virus- oder Programmcode führt.

Aus diesen Gründen sind Slackbereich-Viren nicht weit verbreitet. Der einzige lauffähige Slackbreich-Virus ist der Virus „Number of the Beast (666)“. Es gibt noch einen weiteren Slackbreich-Virus, nämlich den Virus „Assassin“, er ist jedoch nicht lauffähig!

6.3.7 "Live and Die" Viren

Unter "Live and Die" - Viren versteht man Viren, die sich nur für eine bestimmte Zeit in einem Programm aufhalten. Nach Ablauf eines Zeitraums entfernt sich der Virus selbständig aus dem befallenen Programm. Das Programm ist meist nach der selbständigen Entfernung noch lauffähig, unter Umständen sogar wieder im Originalzustand! Aktive Stealthviren kann man im weiteren Sinne zu den "Live and Die" Viren rechnen. Der Pure und Goldbug Virus dürfte diese Definition erfüllen, weil er sich teilweise selbst wieder entfernt, und eine weitere Infizierung des Wirtes unterlässt.

6.3.8 Armored (Gepanzert)

Eine immer beliebtere Technik ist, dass sich Viren vor Analyse oder Entschlüsselung selbst schützen. Dadurch soll den Antivirenforschern das Analysieren des Virus erschwert werden, was dazu führt, dass das Entwickeln entsprechender Antivirensoftware verzögert wird. Durch diese Techniken können auch viele Virenscanner ausgetrickst werden, die anhand von regelbasierenden Modulen (AVR-Modulen[19]) Verschlüsselungsroutinen erkennen können.

6.3.9 Hardware Stealth-Techniken

Inzwischen sind drei Methoden bekannt geworden, wie sich Viren mittels Hardwareeigenschaften Stealth-Techniken zunutze machen.

6.3.9.1 Floppy Disc Boot Simulation

Das CMOS der modernen PCs enthält Informationen, in welcher Reihenfolge gebootet werden soll (A:;C: oder C:;A:). Ein Virus kann diese Reihenfolge jedoch zu seinen Gunsten abändern, z. B. der ExeBug-Virus.

Der Virus (Bootvirus) hat schon die Festplatte infiziert und meldet durch eine bestimmte Technik die Bootsequenz so um, dass zuerst von der Festplatte und dann erst von der Diskette gebootet wird. Versucht nun der Anwender, von der Diskette hochzubooten (in der Absicht eventuelle Viren zu deaktivieren), wird zuerst der Virus von der Festplatte aktiviert. Der Virus schaut nun nach, ob im Laufwerk eine Diskette eingelegt ist und bootet gegebenenfalls von dort weiter. Weil der Virus zusätzlich Sektor-Stealthtechniken verwendet, ist der Virus nach dem "erfolgreichen" Booten von der Diskette nicht zu erkennen.

Es gibt inzwischen schon zwei verschiedene Techniken, um diesen Effekt zu erreichen:

[19] AVR = Algorithmic Virus Recognition.

- Ummelden der Bootsequenz im CMOS
- Andauerndes An- und Abmelden des Laufwerkes im CMOS (diese Technik verwendet z. B. der ExeBug-Virus).

6.3.9.2Hardware Level stealth

Das ist ein anderer neuer Trick, der in zwei existierenden Viren gefunden wurde (Strange und MegaStealth). Der Virus fängt das "Device Ready" Signal des Disk Kontroller ab, wenn ein Festplattenzugriff erfolgen soll (ATs) oder erfolgt ist (XTs). Anschließend manipuliert der Virus den Puffer, der gelesen/geschrieben werden soll zu seinen Gunsten. Selbst ein Wächterprogramm oder sogar ein Hardwareschutz, der die unterste Ebene, nämlich die Festplattenzugriffe kontrolliert, kann so umgangen werden.

6.3.9.3Flash-BIOS stealth

Neuere Computer haben unter Umständen ein so genanntes Flash-BIOS integriert[20], dass durch Softwarebefehle beschreibbar ist. Das normale BIOS ist im Gegensatz zum Flash-BIOS nicht beschreibbar. Ende 1994 tauchte ein neuer Bootvirus auf (BIOS Menegetis), der sich in das Flash-BIOS einnistet, den Bootinterrupt auf sich selbst umsetzt und Schreibzugriffe auf das Flash-BIOS abblockt. Anschließend infiziert der Virus die Festplatte. Danach verhält sich dieser Virus wie ein „normaler“ Bootvirus, d. h. er infiziert alle Disketten, die nicht schreibgeschützt sind. Wenn der Virus von der Platte entfernt wurde, wird beim nächsten Bootvorgang der Virus aus dem Flash-BIOS wieder aktiviert und infiziert sofort wieder die Festplatte! Weil der Virus Schreibzugriffe auf das Flash-BIOS blockiert, kann der Virus nicht ohne weiteres aus dem Flash-BIOS entfernt werden!

Erst Anfang Juni 1996 wurde bekannt, das die API, die der Virus „BIOS Menegetis“ verwendet gar nicht existierte, d.h. der Virus wurde selbst von seinem Autor nicht auf Lauffähigkeit getestet. Es ist jedoch bemerkenswert, dass Virenprogrammier jede Schwachstelle von Betriebssystem und BIOS ausnützen, um den Virus noch bessere „Überlebenschancen“ und Tarnkappeneigenschaften mitzugeben.

6.3.10 Abhilfe bei Stealthviren

Meistens sind die Viren im Arbeitsspeicher unverschlüsselt und können so erkannt werden. Die meisten Stealthviren besitzen folgenden 'Fehler': Der DOS-Befehl CHKDSK zeigt bei einem aktiven Virus pro infizierter Datei einen oder mehrere verlorene Cluster zu der infizierten Datei an. Diese Cluster dürfen jedoch erst mit dem Befehl CHKDSK /F beseitigt werden, wenn der Virus entfernt wurde!

6.4 Verschlüsselte und polymorphe Viren

Verschlüsselung ist die umkehrbare Umwandlung einer Informationsdarstellung in eine andere, die keine Rückschlüsse mehr auf den ursprünglichen Inhalt erlaubt. Verschlüsselung wird verwendet, damit keine virentypischen Texte im Viruscode

[20]Vor allem neuere Pentium- und 486 DX/2-Motherboards sind mit Flash BIOS ausgerüstet.

auftauchen und damit regelbasierende Virenscanner keine verdächtigen Befehle finden. Die Routine, die den verschlüsselten Virus wieder entschlüsselt, wird als „Decryptor“ bezeichnet.

Polymorphe Viren sind Viren, die verschlüsselt sind. Im Gegensatz zu den meistens verschlüsselten Viren, die eine konstante Ent- und Verschlüsselungsroutine besitzen (und somit anhand ihrer Entschlüsselungsroutine erkannt werden können), ist die Entschlüsselungsroutine bei polymorphen Viren überhaupt nicht mehr konstant. Oftmals sind nur noch drei Bytes 'Programmskelett' konstant, der Rest ist oft mit sinnlosen Maschinenbefehlen aufgefüllt. Deshalb stellt diese Virenart -wie die Stealthviren- eine ernstzunehmende Gefahr dar, weil sie teilweise nicht mehr erkannt wird. So kann z. B.:

$$1 + 1 = 2 \text{ auch durch } 1 - (-1) = 2$$

ausgedrückt werden. Gerade im Computerbereich gibt es zig Möglichkeiten, eine logische Aktion durch verschiedenste Algorithmen zu programmieren.

Manche Viren verwenden auch verschiedene Verschlüsselungsmethoden (SUB/ADD, NOT, XOR oder Negieren), damit gewährleistet ist, dass jedes infizierte Programm anders aussieht. Eine Suche nach derart komplex verschlüsselten Viren:

- ist sehr zeitaufwendig
- benötigt einen komplexen Suchalgorithmus
- ist meistens anfällig für Fehlalarme
- benötigt Erfahrung: Es ist oft sehr schwer, eine verlässliche Suchroutine für polymorphe Viren zu programmieren.

6.4.1 Mutation

Unter Mutation versteht man die selbständige Veränderung des Virus während seiner Reproduktion. Polymorphe Viren sind im höchsten Maße mutierend.

Es gibt polymorphe Viren, bei denen kein einziges Byte mehr konstant ist. Solche Routinen, die das Ver- und Entschlüsseln durchführen und teilweise als eigenständige Programmbibliotheken vorliegen (Virenbaukästen), werden auch "Mutation Engines" genannt. Beispiele: MtE (Mutation Engine), TPE (TridenT Polymorph Engine), NED (Nuke Encryption Device), DAME (Dark Angel's Mutation Engine), DSCE (Dark Slayer Confusion Engine) oder DSME (Dark Slayer Mutation Engine). Diese Liste ist noch nicht vollständig, die o. g. Mutation Engines sind jedoch die bekanntesten[21].

Solche polymorphen Viren werden von Scannern nicht mehr mit festen Bytefolgen oder Platzhaltern gesucht, sondern mit Hilfe von algorithmischer Untersuchung (AVR-Module). Das untersuchte Programm wird deshalb auf Techniken hin untersucht, die eindeutig von diesen polymorphen Viren stammen. So muss jede Verschlüsselung einen Schlüssel, den Startbereich und Endbereich und eine Entschlüsselungsvorschrift besitzen. Werden diese

[21] Weitere Mutation Engines sind: PME, SPE, GCAE, ViCE oder GPE.

Codierungstechniken in einem Programm entdeckt, so spricht der Virenscanner an. Jedoch ist es extrem schwer, Algorithmen zu schreiben, die wirklich jeden Aspekt eines polymorphen Virus abdecken. So ist es auch nicht hundertprozentig sicher, ob ein Virenscanner wirklich alle erzeugten Mutationen eines Virus überhaupt aufspüren kann. Teilweise ist es noch nicht einmal möglich, hierfür einen mathematischen Beweis zu erbringen!

6.4.2 Verschiedene Arten von Polymorphismus

Zurzeit teilt man polymorphe Viren in sechs verschiedene Klassen ein, die sich dadurch unterscheiden, wie hochgradig ein Virus mutiert [Bont94]. Die TPE ist z. B. eine der besten Mutation Engines, sie wird deshalb zur Klasse 5 gezählt. Die mit der TPE erzeugten Viren sind in ihrem Aussehen so variabel, dass fast jeder herkömmliche verschlüsselte Virus eine Untermenge der TPE darstellt.

6.4.2.1 Oligomorph

Die erste Art des Polymorphismus: Der Virus benützt eine begrenzte Anzahl (mehrere) von konstanten Decryptoren, die ebenfalls mit einer konstanten Anzahl von Sucherkennungen erkannt werden. Beispiele hierfür ist der Whale-Virus der insgesamt 34 Decryptoren verwendet. Will ein Virensuchprogramm den Whale-Virus sicher entdecken, muss es hierfür 34 Sucherkennungen verwenden! Weitere bekannte Viren sind: Cheeba, Marauder, Screaming_Fist, Slovakia und V-Sign (Cansu).

6.4.2.2 Weitere Arten von Polymorphismus

Die zweite Klasse unterscheidet sich von der ersten Klasse dadurch, dass der Decryptor zwar jedes Mal die gleichen Befehle verwendet, deren Reihenfolge jedoch willkürlich angeordnet sein kann. Beispiel hierfür HWF oder WordSwap.

Von der dritten Klasse spricht man, wenn zwischen den einzelnen Befehlen des Decryptors sinnlose Befehle eingefügt werden. Beispiele hierfür Tequila oder Flip-Virus.

In der vierten Klasse werden dann noch zusätzlich die Befehle zufällig vertauscht, die den eigentlichen Entschlüsselungsvorgang nicht beeinflussen.

Die fünfte Klasse fasst alle bisherigen Klassen zusammen. Diese Art von Polymorphismus stellt alle Entwickler von Virensuchprogrammen vor große Probleme, weil solche Viren nicht mehr zuverlässig erkannt werden können. Ein Ausweg aus diesem Dilemma ist, dass das Virensuchprogramm den Virus entschlüsselt, indem es den Decryptor des Virus selbst verwendet. Das heißt es muss in einer kontrollierten Umgebung der Decryptor des Virus solange schrittweise ausgeführt werden, bis sich der Virus selbst entschlüsselt hat! Beispiele hierfür sind fast alle Viren, die mit einer Mutation-Engine „arbeiten“.

6.4.2.3 Zerstückelt (Permutating)

Die sechste Art des Polymorphismus: Diese Technik besteht darin,

1. den eigenen Virencode in kleinere Stücke zu zerteilen und diese wahllos im Wirt unterzubringen. Es muss nur noch eine Routine hinzugefügt werden, die den Virus beim Starten wieder korrekt zusammenfügt.
2. dass der Virus sich irgendwo im Wirt unzerstückelt unterbringt und mehrere (polymorphe) Lademodule in den Wirt einfügt, die nichts anderes tun, als bestimmte Register zu laden (z. B. sinnlose Befehle) und zum nächsten Lademodul springen.
3. dass die Entschlüsselungsroutine des Virus irgendwo im Wirt über mehrere (polymorphe) Lademodule verteilt im Wirt abgelegt wird, welche nichts anderes tun, als bestimmte Register zu laden und zum nächsten Lademodul zu springen, um in solch einer Schleife den eigentlichen Virus zu entschlüsseln.

Das Problem liegt darin, dass sämtliche herkömmlichen Scannertechniken (Anfang und Ende scannen und Entrypointtracing) hier versagen; der Scanner muss die ganze Datei durchsuchen, was zu Fehlalarmen und Geschwindigkeitseinbußen führt.

Kurz nachdem ich diese Beschreibung fertig gestellt hatte, tauchte der erste polymorphe, permutierende Virus auf (One_Half, Hybrid-Virus mit Stealtheigenschaften). Dieser Virus hat die komplette Entschlüsselungsroutine (die zusätzlich polymorph ist) wahllos im Wirt untergebracht und mittels Sprüngen "verbunden". Meines Erachtens ist es fast nicht mehr möglich, solche Viren mit irgendeiner Art von herkömmlichen Scanalgorithmen zu finden, geschweige denn den Virus zu entfernen.

Die einzige verlässliche Möglichkeit ist es, den Virus im Arbeitsspeicher Schritt für Schritt abzuarbeiten (Codeablauf simulieren oder debuggen), bis er sich selbst entschlüsselt hat. Ist er erst mal entschlüsselt, kann man solche Viren mit einem einfachen Suchmuster erkennen oder mit einem geeigneten Programm entfernen. Dies ist einfacher gesagt als getan: Der Versuch, den Virus One_Half von Hand mit einem Debugger zu entschlüsseln, kann eine echte Geduldsprobe werden.

Die nachfolgenden zwei Grafiken sollen veranschaulichen, wie die oben beschriebenen unterschiedlich permutierenden Viren funktionieren.

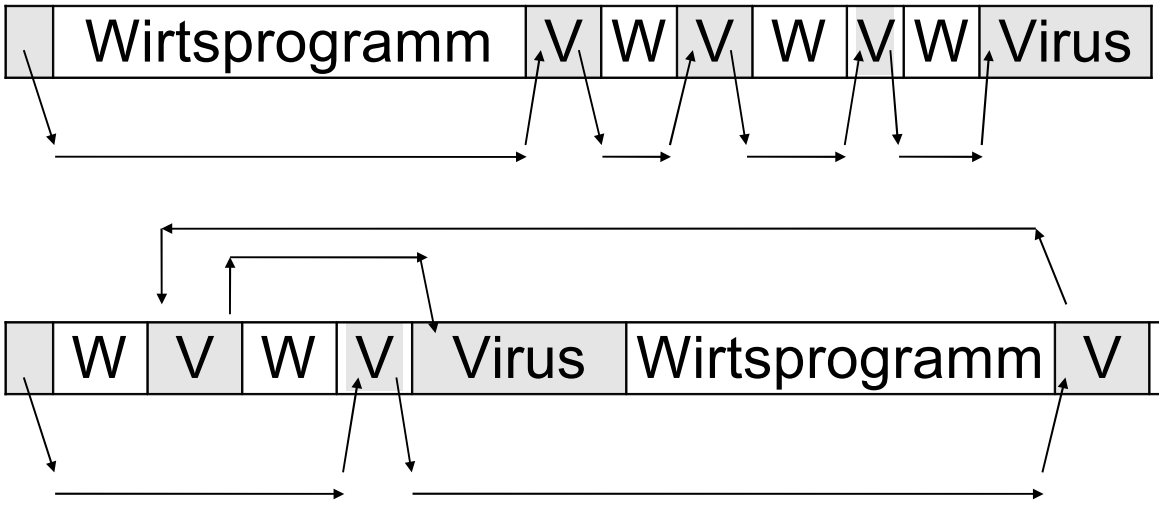


Abb.: Infektionsmechanismus von permutierenden Viren

Weitere permutierende Viren sind: Bad_Boy, Fly, Leech oder Commander Bomber.

Einen ähnlichen Infektionsmechanismus wie permutierende Viren hat die Uruguay-Virus Familie. Die Mutation Engine dieser Viren hat Polymorphismus der fünften Klasse und wird einstimmig von Virenexperten als die zurzeit leistungsfähigste Mutation Engine bezeichnet. Der Virus kann COM Dateien infizieren. EXE-Dateien, die kleiner als 65 KB sind wandelt der Virus zuerst einmal in COM-Dateien um. Anschließend wird die Mutation-Engine dazu verwendet einen Sprung zu verschlüsseln, der den eigentlichen Decryptor des Virus am Dateiende anspringt.

Der am Anfang eingefügte Sprung besteht nicht aus einem „einfachen“ Sprung sondern kann auch direkt, über Register oder auch indirekt ausgeführt werden. Alleine um den Sprung zu erzeugen, erzeugt der Virus Decryptoren oder fügt sinnlose Maschinenbefehle von bis zu 200 Bytes Länge ein. Das Problem für alle Virensuchprogramme ist, sie den Sprung auflösen müssen um den eigentlichen polymorphen Decryptor ermitteln zu können. Weil der Virus hochpolymorph ist und dadurch Decryptoren mit unterschiedlicher Länge erzeugt, kann das Suchprogramm auch nicht einfach hingehen und den Decryptor am Dateiende suchen.

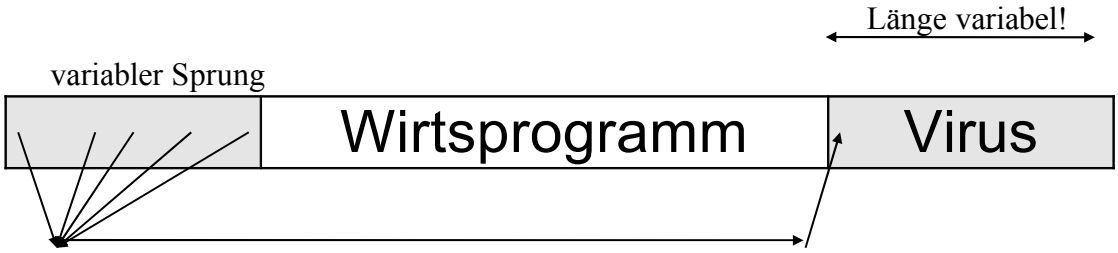


Abb.: Infektionsmechanismus der Uruguay-Virusfamilie

6.4.3 Slow Mutating (Langsames Mutieren)

Um den Antivirenforschern das Leben schwer zu machen, mutieren manche Viren nur sehr langsam. Es ist extrem schwierig, alle möglichen Varianten zu erzeugen, was dazu führt, dass der ganze Virus analysiert werden muss. Ein Beispiel ist der Tremor-Virus, der für sein Mutieren computerspezifische Parameter verwendet. D. h. solch ein Virus muss auf den verschiedenen Testmaschinen ausgetestet werden. Der Effekt beim Tremor-Virus war, dass die erste Scannergeneration nicht alle Tremorvarianten entdecken konnte, was zu einer weiteren Verbreitung führte.

6.5 Retroviren

Retroviren sind Viren, die gezielt Antivirenprogramme ausschalten, angreifen oder deren Funktion einschränken. Retroviren sind auch als Anti-Anti Viren bekannt. Folgende Techniken sind erwähnenswert (siehe auch [FBul216]):

- Modifizieren des Antivirenprogramms auf dem Datenträger oder im Arbeitsspeicher (vgl. Trojanische Pferde).
- Das Erkennen von Speicherwächtern und deren Deaktivierung oder das Ausschalten von Virenfunktionen, damit der Speicherwärter nicht mehr anspricht.
- Das Benützen von Hintertüren oder Fehlern gewisser Antivirenprogramme.
- Löschen oder Verändern von wichtigen Dateien, die zum Antivirusprogramm gehören, z. B. Checksummendateien.
- Das Verwenden von Codesequenzen im Viruscode, die für Antivirenprogramme problematisch sind.
- Das Verursachen von Fehlalarmen.
- Verwendung von Techniken, die das Erkennen, Entfernen oder Identifizieren des Virus erschweren (vgl. polymorphe Viren, Stealthviren, „Sparse“ und „Armored“).

Es sind Viren bekannt, die z. B. Checksummendateien gewisser Antivirenprogramme derart manipulieren, dass eine Infektion nicht mehr bemerkt wird. Fast jeder moderne Virus kann heute den Speicherwächter VSAFE.COM, welcher zum Lieferumfang von MS-DOS 6.xx gehört, deaktivieren. Hierzu reicht ein 8 Byte großes Programm namens „VSLAY.ASM“ das unter Virenprogrammieren weit bekannt ist. Viele Viren stoppen die Ausführung von Antivirenprogrammen, was die Entdeckungswahrscheinlichkeit eher erhöht. Oder sie unterlassen gerade aus diesem Grund das Infizieren solcher Programme. Der Virus Mirror lässt alle Programme als infiziert erscheinen: Falls versucht wird, den Virus zu entfernen, werden uninfizierte Programme zerstört! Diese Aufzählung könnte beliebig fortgesetzt werden.

7 Schwachstellen einzelner Betriebs- und Dateisysteme

Es lassen sich zwei grundsätzliche Arten von Schwachstellen in Betriebssystemen unterscheiden. Es gibt auf der einen Seite Designfehler im Systemkern oder in Betriebssystemprogrammen. Zum anderen gibt es konzeptionelle Fehler in der Spezifikation und im Systementwurf. Gerade im schnelllebigen Computerbereich werden die Betriebssystemhersteller geradezu von den technischen Innovationen „überrollt“; ältere Spezifikationen gelten dann oft nicht mehr!

7.1 MS-DOS

Anmerkung: Wenn hier über MS-DOS gesprochen wird, gilt dies gleichermaßen für die kompatiblen DOS Derivate DR-DOS, PC-DOS, Novell-DOS, PTS-DOS oder sonstige modifizierte DOS Varianten wie RT-DOS[22].

Wie alle älteren Betriebssysteme für Einzelplatzrechner (Stand Alone PC) verfügt MS-DOS über keinerlei Zugriffskontrollen. Dem Anwender stehen alle Operationen, die das System anbietet zur Verfügung. Der Anwender ist, falls keine geeigneten Maßnahmen getroffen wurden, Anwender, Operator und Systemverwalter in einer einzigen Person.

Unter MS-DOS gibt es keinen Speicherschutz. Deshalb können sich auch Viren im Arbeitsspeicher einnisten, Tunnel- und Stealth-Techniken benutzen. Die Verwendung von System File Tables (SFT) ist nur möglich weil der Virus MS-DOS interne Daten direkt im Arbeitsspeicher modifizieren kann. Fast alle fortgeschrittenen Virentricks beruhen auf der Tatsache, dass der Virus Daten und Code direkt im Arbeitsspeicher modifizieren darf und kann.

Somit kann jedes Softwareprogramm unter DOS von einem Virus umgangen werden.

7.2 Windows

Zu Windows 3.0-3.11 (für DOS) gibt es nicht viel zu sagen, wenn man sich vor Augen hält, dass Windows „nur“ ein graphischer Betriebssystemaufsatz für DOS ist.

Alle „Direct Action Infector“ Viren funktionieren unter Windows ebenso wie speicherresidente Viren, solange sie das spezielle EXE Format[23] von Windowsprogramme erkennen

[22] RT-DOS = Real Time DOS für Echtzeitanwendungen.

[23] NE-Header = New EXE-Header, ein erweiterter Programmkopf, kann auch unter DOS ausgeführt werden. Unter DOS erzeugt solch ein Programmkopf meistens die Meldung „This program requires Microsoft Windows“. Wird solch eine NE-Datei von einem Virus infiziert funktioniert meistens der NE-Programmkopf nicht mehr, das Programm ist zerstört!

können und die Infizierung dieser Programme unterlassen. Windowsviren[24], die speziell Windowsprogramme infizieren können gibt es inzwischen ebenfalls. Hier kann man in den nächsten Monaten mit wahrscheinlich einer wahren Flut von neuen Windowsviren rechnen! Das MBR-Viren unter Windows ebenfalls funktionieren, bedarf wohl keinen weiteren Erläuterung!

7.3 Windows ´95

Im Juni 1995 wurden erste Ergebnisse über das neue Windows ´95 im FIDO Forum „VIRUS.GER“ diskutiert. Demnach soll es im Juni ´95 noch kein einziges Antivirenprogramm geben, das überhaupt auf Windows ´95 läuft. Hierzu meint Microsoft auch nur lapidar, das dies Sache des Herstellers des Antivirenprogramms sein. Wird in einer DOS-Box mit einem herkömmlichen Virenscanner nach Viren gescannt, so werden diese teilweise überhaupt nicht mehr gefunden! Windows ´95 blockiert anscheinend alle Interrupt 13h und 26h Aufrufe ab. Aus diesem Grund dürften auch alle Programme (Norton Utilities, PC-Tools u.a.) nicht mehr laufen, die bisher direkt auf diesen Interrupt zugegriffen haben!

Ein Virenexperte meinte in diesem Forum, die einzige Möglichkeit nach Viren zu suchen, bestehe zurzeit darin, von einer herkömmlichen DOS-Diskette zu booten und anschließend gleich nach Viren suchen zu lassen. Man darf gespannt sein, ob Microsoft solch einen „harten“ Schritt durchzieht - immerhin funktioniert dann kein Programm mehr das direkt auf die Festplatte zugreift (also Viren und Utilities).

Bis jetzt ist noch nicht bekannt, wie jedoch Windows ´95 MBR-Viren blockieren will, weil diese ja vor dem Starten von Windows ´95 schon aktiv sind! Wenn Windows ´95 den Interrupt 13h, denn ja MBR-Viren „verbogen“ haben einfach ignoriert und durch einen eigenen Interrupt-Handler ersetzt, würde dies zwar den Virus deaktivieren, könnte wiederum ungeahnte Kompatibilitätsproblem nach sich ziehen.

Man darf also sehr gespannt sein, was sich Microsoft noch bis zum Endgültigen Release des Windows ´95 einfallen lässt. Eine Bewertung über die Sicherheit des Betriebssystems Windows ´95 möchte ich aus diesen Gründen nicht geben.

7.4 Windows NT

Windows NT ist vom Design her ein völlig eigenständiges Betriebssystem, dass mehr Gemeinsamkeit mit UNIX hat als mit Windows für DOS. Windows NT kann als C2 System eingerichtet werden, was eine hohe Sicherheit garantiert. Für Windows NT gibt es bis jetzt (April 95) noch keine Viren. Folgende „Highlights“ von Windows NT möchte ich kurz zusammenfassen, weitere Informationen finden Sie u. a. in [Micr94].

Windows NT bietet unter anderen folgende Sicherheitsmechanismen an, die geeignet sind einen Virenbefall zu erschweren und zu unterbinden:

[24]Im April 95 waren ca. fünf „Direct Action infectors“ und zwei speicherresidente Windowsviren bekannt.

- Verschiedene Benutzer (Logins) mit accounting (Passwort) und beschränkte Zugriffsberechtigungen.
- Überwachungslisten für
  - Gruppen und
  - Benutzer
- Verzeichnisüberwachung und Dateiberechtigungen. Folgende Attribute können vergeben werden:
  - Lesen (R)
  - Schreiben (W)
  - Löschen (D)
  - Besitz übernehmen (O)
  - Berechtigung ändern
- Freigabeeigenschaften, d. h. Ressourcen wie Festplatten, Drucker, CD-ROM oder Bänder können freigegeben werden oder für den Benutzer gesperrt werden.
- Systemprozesse sind geschützt, sie können selbst von Benutzern mit Administrator-Rechten nicht terminiert werden. Allgemeiner Speicherschutz ist vorhanden.
- Das System verwendet ein eigenes geschütztes Dateisystem, in dem Benutzer nur zugewiese Rechte haben.

Standardmäßig ist für Windows-NT keine Gewaltenteilung vorgesehen. Dies kann jedoch durch entsprechende Konfiguration gewährleistet werden.

Durch diese Sicherheitsmerkmale (C2-Sicherheitsstufe) ist Windows-NT von Haus aus sehr gut gegen eine eventuelle Virenattacke gerüstet. Im April 1995 gab es meines Wissen nach noch keine Windows-NT spezifischen Computerviren.

7.5 Netzwerke für PCs

Stellvertretend für die vielen Netzwerke, die eine Einbindung von Rechnern mit dem Betriebssystem DOS und Windows erlauben, habe ich Novell Netware herausgegriffen. Novell Netware (oder auch kurz: Netware) ist zur Zeit der Marktführer was dezidierte Client-Server Betriebssysteme betrifft. Novell Netware bietet u. a. auch die Möglichkeit, UNIX-Systeme (z. B.: TCP/IP oder NFS) sowie andere Systeme oder Großrechner einzubinden.

Novell Netware ist ein eigenständiges Betriebssystem und läuft auf einem Rechner mit 80386+ Architektur im Protected Mode. Dieser Rechner ist der so genannte Server (Fileserver) und verwaltet u. a. Dateien, Druckaufträge, Zugriffsrechte und bedient die angeschlossenen Klient- („Kunden“ oder „Benutzer“) Rechner. Dieser Server steht normalerweise in einem getrennten Raum und kann deshalb nur von einem bestimmten privilegierten Personenkreis bedient werden.

Der Server und das angeschlossene Netz werden vom so genannten Supervisor gewartet. Ansonsten hat niemand Zugriffsrechte auf den eigentlichen Server.

Auf dem Fileserver legen der Supervisor Standardprogramme und die Benutzer ihre Daten zentral ab. Der Server speichert diese Dateien und verwaltet sie. Er übernimmt damit für alle Benutzer gemeinsam die Arbeit der Festplatte im lokalen Rechner. Zusätzlich können lokale Festplatten gleichzeitig verwendet werden (siehe auch [Smad94, S. 13], Kapitel 1: „Netzwerk-Grundlagen“). Will ein Benutzer Zugang zum Netzwerk erhalten, muss ihm der Supervisor einen „Account“ einrichten, der unter anderem ein Arbeitsverzeichnis (Homeverzeichnis), den Benutzernamen und dessen Passwort festlegt. Somit ist ein unberechtigtes Benutzen des Netzes eingeschränkt.

7.5.1 Zugriffsrechte unter Netware

Wie bei großen Netzwerken auch, so können unter Novell Netware Zugriffsrechte, wie z. B. Lesen, Schreiben oder Löschen vergeben werden. Hierzu können auch Benutzer- und Gruppenrechte vergeben werden ([Smad94], S. 21f.). Novell Netware bietet neben den Grundfunktionen der Datenspeicherung und Zugriffsrechte weitere Sicherheitsmechanismen, die Datenverlust weitgehend verhindern. Novell bezeichnet diese Mechanismen als „Security Fault Tolerance (SFT) I, II und III“ (siehe [Smad94], S. 23 ff). Auf diese Sicherheitsmechanismen möchte ich nicht näher eingehen, weil sie nicht direkt mit dieser Diplomarbeit zu tun haben.

Da die Installation der optimalen Rechtestruktur eine recht knifflige Aufgabe ist, drücken sich „Schnellinstallateure“ oft um diese Arbeit und vergeben einfach zu viele Zugriffsrechte. Schlimmstenfalls darf so jeder Benutzer alles löschen oder verändern und hat Zugriff auf alle Verzeichnisse und Dateien. Dies erhöht somit das Risiko des Datenverlustes durch Fehlbedienung oder -was für diese Diplomarbeit interessant ist- das Risiko des Virenbefalls enorm!

Novell Netware bietet eine Vielzahl von verschiedenen Zugriffsrechten an, um mögliche Virenattacken auszuschließen. Einige möchte ich kurz erwähnen, ansonsten siehe [Smad94], S. 215f.

Folgende Attribute für Dateien sind geeignet, einen Virenbefall zu vermeiden, bzw. auch zu lokalisieren:

Flag	Attribut	Beschreibung
RO	Read-Only	Nur lesender Zugriff, kein Schreiben, d. h. ein Virus kann so eine Datei nicht infizieren.
S	Shareable	Datei kann von mehreren Benutzern gleichzeitig genutzt werden. So sollten Programme im Homeverzeichnis vom Typ „Non Shareable“ sein.
SY	System	Datei gehört zum System und darf nicht gelöscht werden. Damit können z. B. Checksummen gegen Retroviren geschützt werden!
T	Transactional	Die Datei wird vom Transaction-Tracking-System überwacht. So kann z. B. die Quelle einer Infektion zurückverfolgt werden.
X	Execution only	Datei darf nur ausgeführt werden, ein Umbenennen ist verboten. Ist z. B. bei Programmen sinnvoll, die gemeinsam benützt werden.
RA	Read Audit	Hat noch keine Bedeutung. Dieses Attribut soll der Write-Audit-Überwachung von Lese- und Schreiboperationen auf Dateien dienen. Wenn dieses Attribut verfügbar ist, kann so auch eine Zurückverfolgung der Infektionsquelle realisiert werden.

Tabelle: Auszug möglicher Dateiattribute unter Novell Netware

Ferner können für Verzeichnisse ähnliche Attribute vergeben werden [Smad94], S. 216f. Bei geeigneter Auswahl der entsprechenden Attribute kann so eine Infektion durch Viren verhindert werden!

7.5.2 Viren unter Novell Netware

Viren für Novell Netware sind bis heute noch nicht aufgetaucht. Es ist nicht sinnvoll, einen Virus für Netware zu schreiben, weil dies spezielle Netwarekenntnisse erfordert und auf der anderen Seite wird ein Novell Server einmal installiert und danach nicht darauf laufend neue Software „ausprobiert“ und weitergegeben.

Für Netware gibt es auch kaum zusätzliche Programme, die einzigen Lieferanten sind Novell selbst und ein paar Systemhäuser. Ein Virenbefall des Servers ist somit alleine durch die Verfügbarkeit von Software für Novell Netware ziemlich eingeschränkt. Zusätzlich ist anzumerken, dass der normale Benutzer keinen Zutritt zum eigentlichen Server hat und somit gar nicht in der Lage ist, verseuchte Programme in das Betriebssystem Netware einzuspielen.

Anders sieht es mit dem angeschlossenen Netzwerk aus. Die angeschlossenen Client-Rechner haben als primäres Betriebssystem hauptsächlich DOS mit Windows installiert (eine

Einbindung anderer Rechner oder Betriebssysteme ist generell möglich). Ist ein solcher Client-Rechner mit einem Dateivirus infiziert, kann sich dieser über das Netz verbreiten, je nachdem, wie gut das Netz installiert wurde. Bootviren können sich **nicht** über ein Netzwerk verbreiten, weil Bootviren nur physikalische Laufwerke (also z. B. A: und C:) infizieren können, jedoch nicht die logischen Laufwerke eines Netzwerkes!

Hybrid- und Multipartite-Viren können sich wiederum über ein Netz verbreiten, weil sie lokal die physikalischen Laufwerke (wie Bootviren) und auf dem Netz entsprechend geeignete Programme (wie Dateiviren) infizieren!

Dateiviren können sich nur über das Netz verbreiten, wenn der Benutzer Schreibrechte auf ausführbare Programme hat. Dies kann dann der Fall sein, wenn er Programme im eigenen Homeverzeichnis hat, oder was viel schlimmer ist, wenn er Schreibrechte auf Programme hat, die auch von anderen Benutzern ausgeführt werden. Die Programme im eigenen Homeverzeichnis sind normalerweise für die anderen Benutzer „unsichtbar“ und stellen somit keine Gefahr dar, das Netz zu verseuchen. Anders sieht es bei gemeinsam genutzten Programmen aus. Ist solch ein Programm erst einmal infiziert, wird bei jedem Programmaufruf der Rechner des Aufrufers infiziert. Ist dies der Fall, kann ein Netzwerk innerhalb eines Tages vollständig verseucht sein.

Abhilfe: Wie schon weiter oben erwähnt, liegt die Ursache an der mangelnden Konfiguration des Netzwerkes durch den Administrator. Sind allgemein genutzte Programme durch Novell schreibgeschützt, kann kein Virus ein solches Programm infizieren, weil das Betriebssystem vollständig von DOS/Windows gekoppelt ist. Wenn ein Netzwerk sauber konfiguriert ist, kann sich ein Virus (fast) nicht über ein Netz ausbreiten. Es gibt DOS-Viren, die speziell für Novell Netware programmiert worden sind, jedoch kann keiner dieser Viren das getrennt laufende Netware Betriebssystem umgehen. Diese Viren sind fast alle zum Ausspähen von Passwörtern entwickelt worden.

Fazit: Somit kann gesagt werden, dass Novell Netware unter der Voraussetzung, dass das Netz richtig konfiguriert wurde, sehr virensicher ist!

7.5.3 Und es ist doch möglich!

Folgende Punkte zeigen, wie es trotzdem zu einer Verbreitung von Viren über ein Netz kommen kann. Das ist nicht ein Mangel von Novell Netware (oder eines entsprechenden äquivalenten Netzwerk-Betriebssystems), sondern liegt an der Schwachstelle „Mensch“! Entsprechende Gegenmaßnahmen habe ich soweit wie möglich realisierbar gleich mit aufgeführt.

Szenario: Der Rechner des Supervisors ist infiziert. Der Supervisor führt bei der Netzpflege verschiedene Programme aus, die auch von den anderen Benutzern ausgeführt werden. Innerhalb kürzester Zeit ist das Netzwerk komplett verseucht! **Abhilfe**: Der Rechner zur Netzwerkpflege wird **nur** für diesen Zweck eingesetzt und neu einzuspielende Software wird sorgfältig überprüft. Der Supervisor verwendet für sich einen normalen „Account“ ohne zusätzliche Rechte auf einen **anderen** Rechner! Zusätzlich wird der Benutzer „Systemadministrator“ angelegt, der ähnliche Rechte hat wie der Supervisor, mit der Ausnahme

von Schreibrechten auf ausführbare Programme. Nur wenn es zwingend notwendig ist, wird dann das Netz als Supervisor gepflegt.

Szenario: Ein Benutzer „U1“ hat einen infizierten Rechner „R1“ und infiziert Programme in seinem Homeverzeichnis (auf dem Server). Jetzt geht er zu einem anderen Rechner „R2“ und meldet sich dort an. Er führt die infizierten Programme im eigenen Homeverzeichnis aus und infiziert indirekt somit den zweiten Rechner „R2“. Am nächsten Tag meldet sich Benutzer „U2“ auf diesem Rechner „R2“ an. Weil dieser Rechner jetzt infiziert ist, werden die Programme im Homeverzeichnis des Benutzers „U2“ infiziert. Ist dieser Benutzer der Supervisor, so kann dies fatale Folgen für das Netz haben! **Abhilfe**: Regelmäßiges Benützen eines Virensuchprogramms. Zusätzlich sollte das Netz mit einem Checksummenprogramm überwacht werden.

Szenario: Es gibt ein Verzeichnis im Netzwerk, auf das alle Benutzer Schreib- und Leserechte haben. Dort werden firmeninterne Dateien abgelegt, z. B. die neuesten Tabellen oder ein wichtiges Dokument. Eines Abends spielt ein Benutzer ein Spiel ein, das mit einem Virus infiziert ist. Ein paar Arbeitskollegen kopieren sich das Spiel und infizieren somit ihre Rechner und anschließend Programme im Homeverzeichnis. **Abhilfe**: Keine Schreibrechte in diesem Verzeichnis für alle Benutzer, z. B. nur für die Abteilungsleiter. Ein Virensuchprogramm durchsucht pausenlos dieses Verzeichnis nach Viren. Ablage gemeinsam benutzter Daten in einem eigenen schreibgeschützten Verzeichnis, das vom Supervisor aktualisiert wird.

Fazit: Novell Netware bietet von Haus aus hervorragende Sicherheitsmerkmale, um eine Infektion von Programmen auf dem gemeinsam genutzten Server zu verhindern. Die einzige Schwachstelle ist der Mensch, der oftmals zu leichtfertig mit seinem Rechner umgeht.

7.6 UNIX, VMS und MVS

UNIX, VMS und MVS[25] sind verschiedene Betriebssysteme für Großrechner mit Multitasking- und Multiuser-Eigenschaften. Für die Diplomarbeit möchte ich UNIX herausgreifen, weil dieses Betriebssystem am weitesten verbreitet ist.

UNIX ist ein universelles Betriebssystem für Großrechner mit Sicherheitsmerkmalen wie Zugriffskontrolle, einer virtuellen Speicherverwaltung, Multitasking und Multi-Using Eigenschaften. UNIX gibt es seit ca. 20 Jahren und wird vorrangig auf Großrechnern eingesetzt, heute gibt es jedoch auch mehrere UNIX Derivate für PCs.

UNIX entstammt der Forschung und wurde maßgeblich an Universitäten in den USA entworfen und entwickelt. UNIX wurde von der Firma AT&T (American Telephon and Telegraph) in den Bell Laboratorien weiterentwickelt. Die Entwicklung begann 1969. Der Name UNIX ist deshalb ein geschützter Begriff der Firma ATT.

Im Jahre 1976 spaltete sich die Entwicklung auf. Verschiedene Firmen bauten auf dem vorhandenen UNIX-System auf und entwickelten eigene UNIX-Versionen, die oft speziell auf die von den verschiedenen Firmen eingesetzten Rechner optimiert wurden. So entstanden

[25] UNIX wurde von AT&T, VMS von DEC und MVS von IBM entwickelt.

Versionen, wie z. B. HP-UX von der Firma Hewlett-Packard, sowie andere Formen wie XENIX, ULTRIX, Coherrent, SINIX, SCO-UNIX, MUNIX oder LINUX. Seit 1983 gibt es eine verbesserte und einheitliche UNIX Version: System V. Auf dieser Version bauen die heute eingesetzten UNIX-Versionen auf.

Bis heute sind -neben ein paar Würmern- keine nennenswerten Viren unter UNIX aufgetreten. Die nachfolgenden Abschnitte sollen deutlich machen, warum es schwieriger ist, unter UNIX Viren zu schreiben und zu verbreiten.

7.6.1 Zugriffskontrolle

UNIX verfügt als Betriebssystem für Großrechner über Zugriffskontrollsysteme auf Datei- und Verzeichnisebene. Zusätzlich muss jeder Benutzer einen Account haben und sich mit Passwort dem System gegenüber identifizieren.

Jeder Benutzer kann für seine Dateien individuelle Zugriffsrechte verteilen. Dadurch kann er seine Daten vor unberechtigtem Zugriff, Löschen oder Manipulation durch andere Systembenutzer schützen. Es gibt mehrere Arten von Zugriffsrechten (Attributen), u. a.:

- Datei lesen (r)
- Datei schreiben (w)
- Datei ausführen (x)

Diese Attribute können wiederum an drei "Parteien" vergeben werden:

- Den Besitzer der Datei (owner)
- Der Gruppe des Besitzers (Group/Team)
- Alle anderen Benutzern (Rest)

Beispiel: Der Befehl "ll" (HP-UX und SCO-UNIX, ansonsten ls -l oder l) zeigt die Dateien des jeweiligen Verzeichnisses an. Man erhält beispielsweise folgende Ausgabe:

```
-rwx--x---  1  Ralph  users   148032  Oct 2 1989  10:05   Hallo
```

Ein Strich "-" bedeutet, dass die betreffe "Partei" keinen Zugriff auf diese Datei hat. Die drei darauf folgenden Zeichen definieren die Rechte des Eigentümers (Ralph). Hier hat er Lese- (r), Schreib- (w) und Ausführungsrechte (x). Die nächsten drei Zeichen definieren die Rechte der Gruppe (users). Alle Benutzer, die zur Gruppe users gehören, dürfen diese Datei ausführen. Die letzten drei Zeichen definieren die Zugriffsrechte aller anderen Benutzer. In diesem Fall kann kein "fremder" Benutzer auf die Datei „Hallo“ zugreifen. Die weiteren Informationen geben an, wie groß die Datei ist (148032 Bytes) und wann sie das letzte mal geändert wurde (2. Oktober des Jahres 1989).

7.6.2 Orange Book und Red Book

Das DoD (Departement of Defense) veröffentlichte 1983 seine „Trusted Computer System Evaluation Criteria“, Kriterien zur Bewertung vertrauenswürdiger Computersysteme, im so genannten „Orange Book“.

Nach den Bewertungsmaßregeln wird ein Computersystem in bestimmte Kategorien (Levels) eingeordnet. Diese Einordnung gilt nur für eine bestimmte Konfiguration der Software mit einer bestimmten Hardware und ist Voraussetzung für den Betrieb des Systems im militärischen Bereich. Weil dieses Bewertungssystem noch unzureichend ist, wird seit 1985 an einem „Red Book“ gearbeitet. Auch bemühen sich zivile Behörden wie das NIST (National Institute of Standards and Technologie) seit 1987 um flexiblere und auf zivile Anforderungen zurechtgeschnittene Kriterien. Das „Red Book“ ist eine Gruppe von nationalen Sicherheitskriterien und enthält neben dem „Orange Book“ das „Green Book“ und Sicherheitskriterien aus „Information Technology Security Evaluation Criteria“ (ITSEC).

Level	Bedeutung
D	Unsicheres System, minimaler Schutz. Keine Gewaltenteilung[26] vorhanden.
Levels mit willkürlicher Kontrolle (DAC = Discretionary Access Control). Der Anwender kann die Zugriffsrechte für seine Dateien festlegen.	
C1	Anwender muss sich einloggen; Benutzergruppen sind erlaubt.
C2	Anwender muss sich einloggen; Passwort und Logdatei sind erforderlich.
Levels mit obligatorischer Kontrolle (MAC = Mandatory Access Control). Der Zugriff erfolgt nach den DoD Richtlinien.	
B1	Freigabeebenen nach DoD.
B2	System ist testbar, kein Fluss von Informationen zu niedrigeren Freigabeebenen möglich.
B3	System wird durch ein mathematisches Modell beschrieben.
A1	System wird durch beweisbares mathematisches Modell beschrieben (höchste Sicherheit). Verifiziertes Design, keine neuen Spezifikationen.

Tabelle: Sicherheitsstufen des Orange Book [Treb92], S. 23

Kommerzielle UNIX Systeme sind in den Sicherheitsstufen C2 bis B2 erhältlich. Bestehende kommerzielle UNIX Systeme können mit minimalen Modifikationen (Shadow Passwortdatei und Auditing) zu C2-Systemen „umgerüstet werden“.

7.6.3 Fehler und Hintertüren

Komplexe Betriebssysteme wie UNIX enthalten eine Vielzahl von Fehlern und Hintertüren im Benutzerinterface. Solche Fehler ermöglichen es Viren, Trojanischen Pferden und Würmern, illegal Root-Rechte zu erlangen, siehe auch „Internet Worm“.

Fehler können auf zwei Arten entstehen:

[26] Unter Gewaltenteilung versteht man die Trennung von Daten und ausführbaren Programmen.

- Konfigurationsfehler - falsche oder unvollständige Konfiguration der Zugriffskontrolle und der Identifizierungsmechanismen. Siehe auch Bemerkungen zu Novell Netware.
- Softwarefehler - Fehler im Kernel und in Dienstprogrammen, die während der Ausführung besondere Rechte haben.

Unter UNIX gibt es auch die Möglichkeit, Passwörter zu knacken, besonders dann, wenn das System noch keine C2-Sicherheitsstufe besitzt. So existiert bei dem PC UNIX Betriebssystem „LINUX“ das alte Format für die /etc/passwd Datei, welche die Benutzerpasswörter noch verschlüsselt enthält. Mit einem geeigneten Programm können innerhalb kürzester Zeit dann etliche Passwörter geknackt werden. Schlimmer ist noch, dass es immer standardmäßig den Benutzer „LINUX“ gibt, der das Passwort „linux“ besitzt. Bei dem Betriebssystem SINIX von Siemens ist es ähnlich, hier lautet das Passwort „Siemens“...

7.6.4 Viren unter UNIX

Unter UNIX gibt es generell vier verschiedene Gruppen von Computerviren:

- Bootviren, z. B. PC-Bootviren, die unter der Intelarchitektur laufen. So funktionieren die meisten DOS-Masterbootrekordviren unter den PC UNIX Betriebssystemen LINUX oder SCO-UNIX.
- Batchfileviren, das sind Viren, die aus einem Shellscript bestehen. Diese Virenart kann sich theoretisch auf fast allen UNIX Rechner verbreiten.
- Linkviren, die sich an ausführbare Programme anhängen oder dynamische Linklibraries (DLL) infizieren. Diese Virenart kann sich nur auf Rechnern verbreiten, die das gleiche Betriebssystem und einen kompatiblen Maschinenbefehlssatz verwenden.
- Überschreibende Viren, entweder als Scriptfiles oder als binäres Objektfile.

7.7 Andere Systeme

Auf folgenden Betriebssystemen bzw. Computern wurden in den letzten Jahren ebenfalls Computerviren gefunden:

- Amiga
- Atari
- Archimedes
- Macintosh

Jedoch muss erwähnt werden, dass heute diese Systeme eine eher untergeordnete Rolle spielen. Es liegt hauptsächlich daran, dass auf diesen Systemen:

- kaum neue Viren auftauchen
- nur sehr wenige Viren überhaupt existieren

- die Systeme teilweise technisch überholt sind
- Viren keine so gute Angriffsflächen zur Vermehrung haben, wie z. B. unter DOS
- es keinen Assembler gibt, in dem normalerweise Viren programmiert werden.
- kommerzielle Software eingesetzt wird - andere Programme werden deshalb nicht „ausprobiert“ oder installiert. Deshalb haben Viren kaum Möglichkeit sich zu vermehren.

8 Hard- und Softwarelösungen

Im folgenden Kapitel möchte ich aufzeigen, was zurzeit an Hard- und Softwarelösungen zum Erkennen, Schützen, Entfernen und Vorbeugen gegen Computerviren auf dem Markt erhältlich ist. Wie immer, gibt es für das Betriebssystem MS-DOS die größte Anzahl von Softwarelösungen, weil hier auch die Anzahl der verschiedenen Viren am größten ist. Hardwarelösungen sind generell effektiver, werden aber bis heute kaum ausgenützt.

8.1 Allgemeine Lösungen

8.1.1 Closed Shop Betrieb

Unter Closed Shop Betrieb versteht man die räumliche Trennung von Rechnern und Peripheriegeräten in einem besonderen, für den normalen Benutzer nicht zugänglichen Raum. Der Benutzer fordert dann von diesem Rechner Dienste an. Durch die Zentralisierung und die räumliche Trennung können somit einfache aber wirkungsvolle Schutzmechanismen installiert werden! Dieses Konzept findet z. B. bei UNIX Maschinen sowie bei Novell Netware Servern Anwendung.

8.1.2 Diskless Workstations

Wenn man mechanisch sicherstellen kann, dass auf das Diskettenlaufwerk des Client-Rechners nicht mehr zugegriffen werden kann spricht man von „Diskless-Workstations“. Dies wird meistens durch Ausbau des Diskettenlaufwerkes realisiert. Der Vorteil von Diskless Workstations ist, das der Benutzer nicht mehr unkontrolliert Daten in sein System oder in das Netzwerk einspielen kann. Der Nachteil liegt klar auf der Hand: Möchte der Benutzer Daten auf Diskette abspeichern, so muss er dies umständlich über ein an das Netzwerk angeschlossenes Laufwerk tun.

8.2 Softwarelösungen

Softwarelösungen -speziell für MS-DOS, Windows und OS/2- für die Erkennung, Identifizierung, Vorbeugung und Entfernung von Viren gibt es wie Sand am Meer. Auch gibt es für UNIX Softwarelösungen, die sich jedoch meistens auf Virenscannung beschränken.

Das Entwickeln von Antivirensoftware muss lukrativ sein, nur so kann man sich die Vielzahl diverser Antiviren-Softwareprogramme erklären. Man muss sich nur einmal das Verzeichnis „virus“ auf dem SimTel Server anschauen, es ist richtig voll gestopft mit Antivirensoftware. Generell kann keine Empfehlung gegeben werden, welches Programm nun besser, schneller, schöner oder teurer ist. Deshalb möchte ich nur kurz die Philosophie der verschiedenen Softwarekonzepte erläutern (siehe auch [Rada93]).

8.2.1 Integry Checking

Zunächst wäre einmal die Gruppe der Integritätsprüfer (Integry Checking). Diese Programme versuchen, einen Virus daran zu erkennen, indem sie Manipulationen an möglichen Wirten durch den Virus entdecken. Dies wird dadurch erreicht, dass auf einem **virenfreien** System von jedem potentiellen Opfer (MBR, DOS-Bootsektor, FAT und Programme) ein „Fingerabdruck“ ermittelt und dieser in einer Datenbank gesichert wird. Infiziert nun ein Virus das System, verändert er den Fingerabdruck des Wirtes und kann somit erkannt werden. Integrität Checker können als Programme oder als speicherresidenter Schutz eingesetzt werden. Oft werden beide Möglichkeiten angeboten.

Vorteile:

- Ist eine zuverlässige Methode, alte und neue Viren zu entdecken und braucht deshalb nicht so oft upgedatet zu werden.
- Erkennt auch polymorphe Viren.
- Ist relativ schnell in der Überprüfung des Systems.

Nachteile:

- Kann nicht ermitteln, um welchen Virus es sich handelt (Klassifizierung).
- Ich nicht für unerfahrene Benutzer geeignet.
- Ist nicht geeignet für Systeme, in denen oft Programme geändert werden (z. B. Softwareentwicklung oder Mailboxen).
- Ist nutzlos, wenn ein Stealth-Virus aktiv ist oder wenn vorher das System schon infiziert war.
- Datenbank wird oft von Viren manipuliert (siehe Retroviren).
- Kann nur Viren erkennen, **nachdem** er sich vermehrt hat!

8.2.2 Scannen

Unter Scannen versteht man das Auffinden von Viren in ausführbaren Programmcode. Man unterscheidet zwischen zwei grundsätzlichen Scantechniken,

- dem heuristischen Scannen und
- dem Signaturenscannen.

Heuristisches Scannen sucht im Gegensatz zum Signaturenscannen nicht nach bekannten Viren, sondern nach virentypischen Sequenzen, wie z. B. Datei öffnen, Code anfügen und Datei wieder schließen. Heuristisches Scannen ist sehr anfällig für Fehlalarme und wird deshalb oft nur zusätzlich zum Signaturenscannen verwendet. Zum Entfernen von Viren muss das Virenentfernungsprogramm den Virus genau kennen, weshalb im Allgemeinen das heuristische Scannen hierfür nicht verwendet werden kann.

Beim Signaturenscannen sucht der Virenscanner nach kurzen virenverdächtigen Sequenzen, die eindeutig von genau diesem Virus stammen. Virenscanner sind sehr weit verbreitet, weil der normale Benutzer wenig Grundwissen benötigt, um einen Scanner einzusetzen und weil Scanner sehr zuverlässig arbeiten.

8.2.3 Monitorprogramme

Monitorprogramme werden speicherresident geladen und überwachen von dort an suspekte Dateizugriffe, suchen nach bekannten Viren, eventuell auch nach unbekannten Viren (heuristisches Scannen) oder führen Integrity Checking durch. Gute Monitorprogramme überwachen ferner das System auf weitere virentypische Aktionen wie Interrupt Tracing, das sich Einnisten im Arbeitsspeicher, das Einpflanzen eines MBR-Virus oder das Erzeugen von Companion Dateien.

Das Problem bei Monitorprogrammen ist, dass sie sehr speicherhungrig sind, den Systemdurchsatz verlangsamen und Inkompatibilität erzeugen können. Ferner sind Monitorprogramme wiederum nur von erfahrenen Benutzern einsetzbar, weil sie ein gutes Grundwissen über Viren voraussetzen. Aus diesem Grund sind Monitorprogramme nicht so weit verbreitet. Ferner sind die meisten Monitorprogramme ungeschützt, d. h. Viren können solche Monitorprogramme direkt im Speicher „patchen“ und damit wirkungslos machen (z. B. Neuroquila)!

8.2.4 Virenentfernung

Virenentfernung wird dann angewandt, wenn ein Virus erfolgreich wichtige Programmteile infiziert hat. Hier gibt es mehrere Ansätze, den Virus wieder zu entfernen. Das Programm, das den Virus entfernt, wird als Killer oder Cleaner bezeichnet. Folgende Methoden werden zurzeit in existierenden Programmen verwendet:

- Der Virus ist dem Cleaner bekannt, er kann deshalb das Programm wieder herstellen. Weil von den bekanntesten Viren teilweise bis zu 150 modifizierte Varianten existieren, ist dieser Ansatz inzwischen nicht ungefährlich, weil bei einer ungenauen Erkennung das Wirtsprogramm zu „Tode“ repariert wird. Cleaner für bekannte Viren arbeiten jedoch in der Regel relativ zuverlässig. Die nachfolgend erwähnten Verfahren sind in der Erfolgsrate schlechter.
- Heuristisches Cleanen: Der Cleaner lädt das Programm in den Arbeitsspeicher und führt den Virus Schritt für Schritt aus. Der Cleaner überwacht jeden Befehl und unterbindet eventuelle virenspezifische Angriffe auf das Betriebssystem. Der Vorteil des heuristischen cleanen ist, dass der Cleaner keine Informationen über den Virus braucht, der Cleaner kann bei hochkomplex verschlüsselten oder ganz neuen Viren eingesetzt werden. Der Nachteil eines solchen Cleaners ist, dass er ziemlich langsam ist und die Erfolgsrate nach meinen Erfahrungen bei ca. 50 bis 60 Prozent liegt. Ein solcher heuristischer Cleaner ist z. B. RVK[27], der mittels Tracing und Emulation den Virus Stück für Stück abarbeitet. Ein ähnliches Verfahren verwendet das Programm MBR-Kill.
- Restoring. Dieses Verfahren wird besonders von Integry Checker angewandt, die den Fingerabdruck des zu reinigenden Programms vor der Infektion abgespeichert haben. Bei den meisten Viren reicht es aus, den Programmkopf zu rekonstruieren und den Virus abzuschneiden und anschließend das gereinigte Programm mit dem alten Fingerabdruck zu vergleichen. Dieses Verfahren funk-

[27] RVK = ROSE Virus Killer, regelbasierender Virenkiller.

tioniert nicht bei permutierten Viren, EXE-Header Viren und Zerohunting, sondern nur bei Link-Viren, deshalb liegt die Erfolgsrate auch nicht so hoch.

8.3 Hardwarelösungen

Hardwarelösungen[28] werden speziell für INTEL PCs angeboten, sie haben sich jedoch nie richtig durchgesetzt. Vielleicht liegt es daran, dass man für Hardwarelösungen zahlen muss, während Antivirensoftware oft kostenlos als Shareware erhältlich ist. Ferner erfordert eine Hardwarelösung wiederum einige Voraussetzungen, z. B. beim Einbau oder nachher in der Lernphase des Hardwareschutzes. Es gab über die letzten Jahre mehrere Firmen, die versucht haben, ihren Hardwareschutz erfolgreich zu vermarkten, aber die einzigen, die sich durchgesetzt haben, möchte ich kurz ansprechen.

Es gibt u.a. eine Karte „ExVira“ - sie soll seit 1991 zu einem Preis von DM 399,-- vermarktet werden, eine weite Verbreitung fand diese Karte jedoch nicht. Laut Aussagen eines Virenforschers soll diese Karte jedoch teilweise schlechter sein, als herkömmliche Softwarelösungen! Nähere Informationen liegen mir leider nicht vor.

8.3.1 AMI-BIOS

Das AMI-BIOS (neben anderen BIOS-Derivaten) bietet die Möglichkeit, die Bootsequenz (A:-C: oder C:-A:) einzustellen und Schreibzugriffe auf den MBR abzufangen und gegebenenfalls abzublocken. Werden diese Optionen im Setup eingestellt, kann eine Infizierung durch die meisten MBR- und Hybridviren erfolgreich abgewehrt werden. Weil die Bootsequenz und der Status für das Blockieren von MBR-Schreibzugriffen im CMOS gehalten werden, gibt es schon Viren, die gezielt diesen Schutz umgehen können. Trotzdem muss gesagt werden, dass diese Funktionen „kostenlos“ bei einem AMI-BIOS mitgeliefert werden und sie bei richtiger Anwendung immerhin 99 Prozent aller MBR-Viren blockieren können.

8.3.2 Thunderbyte Hardware Card

Leider stand mir nicht eine solche Karte zur Verfügung, weshalb ich mich auf die Produktbeschreibung und Erfahrungsberichte von Benutzern beschränken muss.

Die Thunderbyte Hardware Karte soll grob die gleichen Funktionen erfüllen, wie die Antivirenprogramme von der gleichen Firma, d. h.

- Schutz vor MBR- und Bootviren
- Überwachen kritischer Dateioperationen
- Überprüfung auf Veränderungen (Integry Checking)

Das Problem an dieser Hardwarelösung ist, dass sie wiederum nur von sehr erfahrenen Benutzern eingesetzt werden kann und dass die Karte anscheinend nicht im Graphikmodus

[28] Oft spricht man hier auch von Virenschutzkarten.

funktioniert. Auch soll die Karte ziemlich „scharf“ eingestellt sein, was zu einer Vielzahl von Fehlalarmen führen soll.

Das alles spricht gegen eine solche Karte, weil der Benutzer das Vertrauen in solche eine Karte verliert, und sie deshalb nicht entsprechend einsetzt oder schlimmstenfalls ganz deaktiviert. Heute werden vermehrt graphische Betriebssysteme eingesetzt, unter den eine solche Hardwarekarte nicht mehr vollständig funktioniert. Auch ist der Preis einer solchen Karte (ca. DM 200) nicht gerade ein Argument, solch eine Karte sich anzuschaffen.

Virenerkennungssoftware muss ständig upgedatet werden, was bei einer Hardware Lösung nicht ganz so einfach ist. Ist der Hardwareschutz durch externe Software programmierbar, so wird in kürzester Zeit ein Virus auftauchen, der solch einen Hardwareschutz durch Softwarebefehle deaktiviert![29]

8.3.3 Netcard-PROM

Seit neuestem können Netzwerkkarten mit einem Hardwareschutz ausgerüstet werden, der laut Herstellerangaben, das Einschleusen von Viren durch den angeschlossenen Client-Rechner verhindern soll. Nur stellt sich hier die Frage, wie solch ein PROM monatlich um die neusten Viren upgedatet werden soll. Es reicht ja nicht nur neue Signaturen hinzuzufügen - polymorphe Viren benötigen meistens einen eigenen Entdeckungsalgorithmus. Auch hier stellt sich die Frage, ob sich solch eine Lösung sich auf dem Markt etablieren wird.

8.3.4 Boot-PROM

Unter Novell-Netware ist es z. B. möglich vom einen Boot-PROM der Netzwerkkarte aus DOS zu booten. Hierzu legt der Supervisor auf dem Netz für jeden Benutzer ein so genanntes Boot-Image an, dass das eigentliche benutzerdefinierte Booten auf dem Novell Server erst erlaubt. Rechner mit Boot-PROM Lösungen haben in der Regel überhaupt keine Festplatte mehr integriert. Der Benutzer kann nur auf dem Server Daten bearbeiten und speichern. Fällt somit der Zugriff auf die lokale Festplatte weg, haben MBR- und Hybridviren keine Chance solch einen PC zu infizieren!

8.4 Virenschutz in der Praxis

Die nachfolgende Tabelle wurde der Zeitschrift LANline entnommen ([LANl95]). Gefragt wurden Unternehmen der Elektrotechnikbranche, welchen Virenschutz sie für ihre Netzwerke einsetzen. Die nachfolgende Tabelle zeigt keinen repräsentativen Querschnitt, jedoch ist es erstaunlich, wie wenig Firmen überhaupt einen Virenschutz einsetzen!

[29] Siehe hierzu auch „Hardware-Stealth Techniken“, insbesondere der Virus „BIOS-Menegetis“

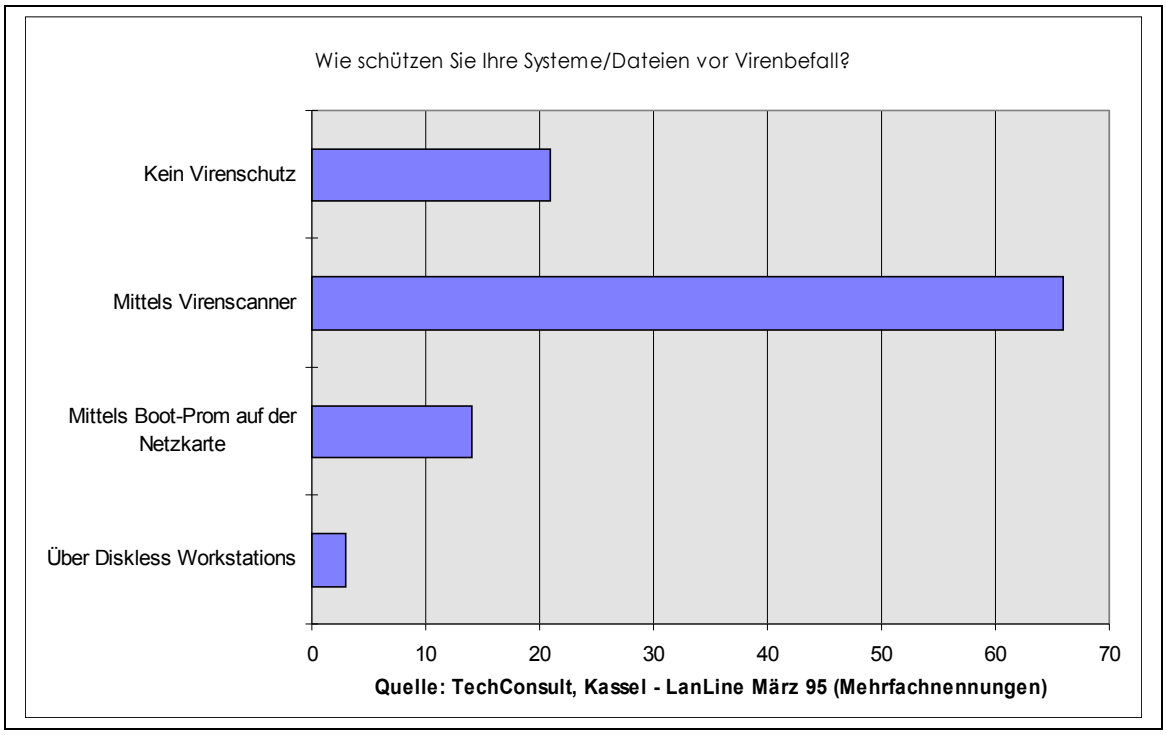


Abb.: Welcher Virenschutz wird bei Netzwerken verwendet

8.5 Fazit

An Softwarelösungen, speziell für das Betriebssystem DOS, mangelt es nicht, um eventuelle Viren zu erkennen oder zu entfernen. Das Problem hierbei ist, dass die meisten Softwarelösungen dafür konzipiert sind, Viren **nach** einem Befall zu erkennen und zu entfernen.

Softwarelösungen, die **präventiv** einen Befall verhindern sollen, werden aus folgenden Gründen kaum eingesetzt:

- Sind für den normalen Benutzer oft zu kompliziert oder benötigen für den effektiven Einsatz spezielles Grundwissen über Computerviren.
- Sind nicht zuverlässig, d. h. neue Viren können durchschlüpfen, während z. B. ein Kompilieren von einem Quellcode als virentypische Aktion gemeldet wird (Slowinfektor). Die Gefahr von Falschalarmen liegt in der Natur dieser Softwarelösungen!
- Sind normalerweise nur für Dateiviren geeignet. Hybridviren und MBR-Viren können deshalb solch ein Schutz leicht umgehen.
- Sind oftmals inkompatibel mit vorhandener Software und verlangsamen den Systemdurchsatz oft erheblich.
- Jedes Softwareprogramm kann von einem Virus direkt im Arbeitsspeicher gepatched werden.
- Monitorprogramme sind unter graphischen Oberflächen, wie Windows oder OS/2 meistens nutzlos.

Die Gründe, die gegen einen effektiven Schutz durch Softwarelösungen sprechen, lassen sich fast ausnahmslos auch für Hardwarelösungen, wie etwa die Thunderbyte Card aufführen.

Eine positive Ausnahme, ist in meinen Augen der Hardwareschutz des AMI-BIOS, dass gezielt nur einen Bereich überwacht, nämlich den MBR der Festplatte. Leider haben dies die Virenprogrammier auch schon erkannt, Viren, die speziell diesen Schutz umgehen können, sind deshalb auch schon im Umlauf.

9 Verbesserungsvorschläge für Betriebs- und Dateisysteme

Wie aus den vorhergegangen Kapiteln ersichtlich wurde, hat das Betriebssystem DOS die größten Sicherheitslücken; hier ist meiner Meinung nach, auch noch der größte Handlungsbedarf. Die nachfolgenden Vorschläge sind fast nur rein theoretischer Natur, weil kein Betriebssystemhersteller hergeht und von heute auf morgen sein Betriebssystem komplett redesigned. Auf der anderen Seite sind manche Vorschläge einfach zu realisieren, ohne dass Teile am Betriebssystem geändert werden müssten.

9.1 Schutz durch selbstüberprüfende Programme

Vom technischen Standpunkt aus betrachtet, wäre es ein Leichtes, das sich die Betriebssystemkomponeten selbst überprüfen. Solche Impfprogramme gibt es schon lange für das Betriebssystem DOS. Würde nun die nächste DOS-Version mit selbstüberprüfenden Programmcode auf den Markt kommen, würden ca. 70 Prozent aller Viren unter DOS sofort auffallen, weil früher oder später ein Betriebssystemprogramm eine Infizierung melden würde.

Diese Impfprogramme sind teilweise als Shareware erhältlich. Für den normalen Einsatz würde es also ausreichen, wenn ein paar Programme immunisiert würden. Dies wäre eine erste Hürde, die von ca. 70 Prozent aller Viren nicht überwunden werden könnte.

9.2 Restricted Shell

Dieser Begriff kommt von UNIX. Eine Restricted Shell ist eine eingeschränkte Benutzeroberfläche (Shell).

Der Benutzer kann z. B. nicht das Verzeichnis wechseln oder hat keine Schreibrechte. Wenn es möglich ist, dem Benutzer eine solche Shell einzurichten, ohne dass er in seiner Arbeit behindert wird, ist schon ein sehr sicherer Virenschutz eingebaut.

Unter DOS könnte z. B. sofort ein Menü gestartet werden, in dem nur Anwendungsprogramme gestartet werden können, d. h. der Benutzer hätte **keine** Möglichkeit neue (eventuelle) infizierte Programme zu starten!

9.3 Speicherschutz

Das Problem bei Betriebssystemen, die nicht Multitasking oder Multiuser fähig sind, ist oft ein nicht vorhandener Speicherschutz. Gerade unter DOS hat das fatale Folgen: Viren können den DOS-Kernel, die System File Tables (SFT) oder Antivirenprogramme direkt im Speicher patchen, deaktivieren oder modifizieren!

Gerade das sich Einnisten von Viren in den Arbeitsspeicher wäre nicht möglich, wenn Viren nicht direkt die Speicherverwaltung von DOS manipulieren könnten. Windows bringt zwar einen minimalen Speicherschutz, dieser ist jedoch relativ unwirksam[30], ein manipulieren des DOS-Speichers wird auf jeden Fall nicht unterbunden!

Bei Betriebssystemen mit Speicherschutz können sich speicherresistende Viren zwar auch einnisten, der Benutzer hat jedoch die Kontrolle über solche Prozesse. Ferner können Viren unter solchen Betriebssystemen kaum Tarnkappeneigenschaften verwenden, weil ihnen hierzu die Erlaubnis fehlt, direkt im Speicher Daten zu manipulieren.

9.4 Gewaltenteilung

Im Computerbereich versteht man unter Gewaltenteilung das getrennte Behandeln von Programmen (Code) und Daten.

Unter UNIX werden z. B. die allgemein benötigten Programme (Dienstprogramme) in ein spezielles Verzeichnis (z. B. /bin) kopiert. In diesem Verzeichnis hat niemand Schreibrechte außer dem Systembetreuer! Diese Programme können deshalb nicht von dem normalen Benutzer infiziert werden. Ein Virus kann sich deshalb nicht über diese Dienstprogramme verbreiten.

Moderne Betriebssysteme sollten eine Art von Gewaltenteilung integriert haben. Denkbar wäre auch, dass es ein weiteres Attribut gibt (z. B. Ext-Read-Only), das nur mit einem Passwort zurückgesetzt werden kann. So könnten z. B. alle standardmäßig ausgelieferten Dienst- und Systemprogramme dieses Attribut besitzen. Soll nun solche eine Datei verändert werden, so muss der Benutzer diese Änderung mit seinem individuellen Passwort bestätigen!

[30] Meistens werden von Windows nur „Allgemeine Schutzverletzung“ gemeldet.

10 Das Programm MBR-Killer

Das nachfolgende Kapitel stellt in groben Zügen ein Softwareprogramm vor, das geeignet ist, Virenbefall auf PC Rechnern mit der Intelarchitektur 80386+ zu erkennen, zu verhindern und eventuelle aktive Viren zu entfernen. In die Entwicklung des Programms flossen alle die Überlegungen und Anforderungen aus den letzten Kapiteln ein, um bestehende Sicherheitslücken bei PCs mit INTEL 80x86 Architektur zu schließen.

10.1 Übersicht

Die Entfernung von Bootviren auf PCs mit 80x86 Prozessoren, welche den Master Boot Record (MBR, auch Partitionstabelle genannt) infizieren, bereitet vielen Anwendern enorme Probleme. Es gibt sogar einige Bootviren, welche die Rekonstruktion des originalen MBR nicht mehr zulassen, weil dieser vom Virus überschrieben wurde (siehe Virengattungen). **Bootviren gehören zu den am häufigsten auftretenden Viren**[31]. Daher wäre es sinnvoll, ein Programm zu entwickeln, welches generell in der Lage ist, solche Viren entfernen zu können. Ferner sollte das zu entwickelnde Programm einen neuen MBR erzeugen, der sich selbst auf Virenbefall untersucht. Somit wäre gewährleistet, dass das System zur Bootzeit frei von Bootviren ist, ohne dass der Benutzer hierzu eine Aktion durchführen muss.

Es wäre also wünschenswert, ein Programm zu haben, welches einen infizierten MBR mit einem virenfreien funktionellen gleichartigen MBR ersetzt, welcher die zusätzliche Eigenschaft der Selbstüberprüfung besitzt.

10.2 Mögliche Angriffspunkte

Mögliche Angriffsstellen durch MBR-Viren sind der eigentliche Masterbootrekord und der nachfolgende Bootsektor (entspricht dem ersten logischen Sektor der aktiven Partition). Diese Systembereiche, welche eine Veränderung des MBR und dessen Partitionstabelle sowie des als aktiv markierten Bootsektors nach sich ziehen, werden von folgenden Virengattungen infiziert:

- MBR - Bootviren (z. B. Jumper, Stoned, Ripper, AntiExe, Stoned.Angelina oder Parity_Boot)
- Bootsektorviren (z. B. Form oder Boot_437), sehr seltene Virengattung!
- Multipartite-Viren (z. B: Tequila, Natas, Goldbug, Neuroquila, Delwin.1759, Junkie oder Flip)

Diese Virengattungen stellten im Zeitraum Oktober 1994 und Mai 1995 jedoch die „Top Ten“ der am weitest verbreiteten Viren dar! Eine ausführliche Beschreibung zu den einzelnen Virengattungen finden Sie im Kapitel „Verschiedene Virenarten/Klassifizierung“.

[31] Siehe auch „Top-Ten“ im Anhang!

10.3 Was ist zu überwachen?

Das grundsätzliche Problem ist, dass unser Programm sicher sein muss, dass es das erste Softwareprogramm im System ist, welches ausgeführt wird. Falls ein anderes Programm vor unserem Schutzprogramm ausgeführt wurde, deutet dies auf eine Infektion hin.

Alle gängigen Bootviren versuchen, sich speicherresident im Arbeitsspeicher einzunisten. Mögliche Aufenthaltsbereiche sind hierzu einzig und allein die Interrupttabelle (IVT), Videospeicher, HMA oder der TOM-Bereich, weil andere Bereiche später vom Betriebssystem genutzt werden. Um potentielle Opfer zu infizieren, bzw. sich später wieder in den MS-DOS Interrupt einklinken zu können, müssen vom Virus einige Interrupts belegt werden, vor allem der Disketten-/Festplatteninterrupt INT 13h. Folgende Lösungen sehen meines Erachtens Erfolg versprechend aus:

1. **Lösung**: Überprüfung auf Änderungen an Bereichen, die von Viren verursacht werden können. Dazu zählen unter anderem:
    - TOM-Bereich, Interrupttabelle und Videospeicher
    - Interruptvektoren (13h, 40h und eventuell Keyboard- und Timerinterrupt)

Fast alle Bootviren verändern den Interrupt 13h (Disketten- und Festplatteninterrupt) und den Wert des TOM-Zeigers. Für eine minimale Selbstüberprüfung würde es ausreichen, diese zwei Bereiche zu überwachen.

2. **Lösung**: Tunneln des Disketten-/Festplatteninterruptes, bis dieser in das schreibgeschützte oder gespiegelte BIOS zeigt. Mit diesem „sauberen“ Interrupt den MBR nachladen und mit dem eigenen Code vergleichen. Treten beim diesem Vergleich Unterschiede auf, ist ein Virus aktiv.

10.4 Softwareanforderungen

10.4.1 Minimalanforderungen

- Ersetzen von MS-DOS/Novell DOS MBR (Partitionen).
- Unabhängigkeit von der DOS-Version.
- Englischsprachige Benutzerführung

1. Überprüfung des Systems durch den neuen MBR auf eventuelle Bootviren. Überprüfung folgender kritischer Bereiche:
    - Testen, ob der TOM-Zeiger erniedrigt wurde.
    - Wohin zeigt der Interrupt 13h?

oder

2. Nachladen des MBR an einen anderen Bereich und Vergleich mit dem eigenen Code auf Veränderung. Mit dieser Technik würde jeder Virus, der keine Tarnkappeneigenschaften besitzt, erkannt werden! Falls möglich könnte der Diskettenterrupt direkt angesprungen oder getunnelt werden um eventuelle Tarnkappenviren auszuschalten.

10.4.2 Weitere wünschenswerte Anforderungen

Die Frage hierbei ist, ob es überhaupt möglich ist, solche Zusatzfunktionen aufgrund des begrenzten Speicherbereiches im MBR zu implementieren.

- Bevor der original (eventuell verseuchte) MBR ersetzt wird, sollte sichergestellt werden, dass das System virenfrei ist (re-infektion). Eventuell das Bootlaufwerk überprüfen.
- Weitere Überprüfung der Interrupts 40h, 67h, Keyboard- und Timerinterrupt
- Checksumme über nachfolgenden Bootsektor bilden und bei Start prüfen.
- Ersetzung von LINUX, SCO-UNIX, Novell Netware und OS/2 Partitionen.
- CMOS und Controllerinterrupt überprüfen, um Hardware Bootstealthviren erkennen zu können.
- Tunneln, um eventuelle (schon aktive) Stealthviren umgehen zu können. Verwendung eines Codeemulators.
- Option /**save**, um den MBR auf Diskette sichern zu können
- Option /**load**, um einen gesicherten MBR wieder zurückschreiben zu können.
- Selbstheilung, falls MBR verändert wurde. Wird jedoch aus Platzproblemen im neuen MBR nicht möglich sein.
- Verschlüsselung des neuen MBR, um eine Analyse oder ein Patchen durch Viren zu erschweren[32].
- Immunisieren aller Festplatten per Parameter.

10.5 Probleme

1. Der TOM-Wert, der normalerweise 640 KB beträgt, wird jedoch von etlichen Programmen und Konfigurationen erniedrigt! Hierzu zählen vor allem SCSI-Platten, der BIOS-Datenbereich und andere Hilfsprogramme. Glücklicherweise sind Hilfsprogramme beim Booten noch nicht aktiv, was die Anzahl der möglichen Verursacher einschränkt!
2. Die Partitionstabelle, die Informationen über die Größe, Art, Lage usw. der verschiedenen Partitionen enthält, darf nicht einfach ersetzt werden, sondern muss in den neuen MBR gesichert werden.
3. Der eigentliche MBR ist sehr klein (ca. 450 Bytes), weshalb die Ladefunktion, Selbstüberprüfung, Meldetexte usw. sehr kurz gehalten werden müssen.
4. Regelbasierende Virensuchprogramme könnten einen neuen Virus finden.
5. Es existiert mindestens eine logische MBR-Bombe (vgl. Trojanisches Pferd), die nicht resident ist und nur am 1.1. des Jahres aktiv wird. Diese Bombe wird nicht speicherresident, belegt keine Interrupts und dekrementiert auch nicht den TOM-Zeiger. Von dieser Bombe gibt es jedoch auch eine speicheresidente Variante mit Stealtheigenschaften!
6. Exotische Betriebssysteme könnten schon Betriebssysteminitialisierungen im MBR durchführen, welche das Schutzprogramm nicht durchführt[33].

[32]Dies ist leider bei der Verwendung des zweiten Lösungsvorschlages nicht möglich, weil sich sonst der eigene Code vom nachgeladenen Code unterscheiden würde!

[33]Der alte LILO-Lader vom PC UNIX Betriebsystem LINUX führt z.B. den eigentlichen Ladevorgang schon im MBR aus, weshalb der LILO-Lader nicht mit MBR-Kill (oder MBR-Viren) verträglich ist!

7. Der MBR könnte verändert werden, ohne dass eine Virusinfektion vorliegt. Zum Beispiel dann, wenn die aktive Partition durch einen Bootmanager verändert wurde.

10.6 Lösungsvorschläge

Lösungen, analog zur Problemaufzählung.

1. Falls der TOM-Bereich um ein bis acht KB erniedrigt wurde und der Interrupt 13h in diesen Bereich hineinzeigt, ist wahrscheinlich ein Bootvirus aktiv! Fast alle AT-Rechner, die ein CMOS besitzen, haben den Wert des TOM-Bereiches zusätzlich im CMOS abgespeichert. Ein Vergleich des TOM-Wertes BIOS mit dem des CMOS könnte deshalb ein Erfolg versprechender Ansatz sein.
2. Die Partitionstabelle, wird einfach vom infizierten bzw. originalen MBR übernommen. Hier könnte noch eine Überprüfung durch das Impfprogramm auf eventuelle exotische Viren hinzugefügt werden.
3. Die Implementierung des neuen MBR kann meines Erachtens in 80386 Assembler (optimiert) erfolgen, eine Überprüfung der Zielhardware durch das Impfprogramm wäre dann wünschenswert. Eventuelle Meldungstexte müssen notfalls sehr kurz gehalten werden (z.B.: ERROR! oder VIRUS!).
4. Saubere Programmierung, keine Tricks verwenden, welche auch Viren verwenden.
5. Falls die Bombe den MBR voll ersetzt (überschreibt), kommt keine Meldung des neuen selbstüberprüfenden MBR mehr -> sehr auffällig. Falls die Bombe den original MBR nachlädt, kann ein Nachladen des MBR durch das Schutzprogramm und anschließend ein 1:1 Vergleich die Bombe entdecken. Sollte die Bombe Stealtheigenschaften ausnützen, müsste sie speicherresident werden, was sich in der Veränderung des Interruptes 13h und des TOM-Zeigers bemerkbar machen würde.
6. Hier hilft nur ein Test des Betriebssystems auf Kompatibilität mit dem neuen MBR.
7. Überprüfung der Partitionstabelle auslassen. Hier könnte sich jedoch für solche Viren eine Sicherheitslücke auftun, die eine neue Partition anlegen und von dort weiterbooten lassen. Solche Viren sind mir jedoch noch nicht bekannt!

10.7 Ausgewählter Lösungsansatz

Meines Erachtens ist der Lösungsansatz 2.) der bessere, aus folgenden Gründen:

Vorteile des 1.) Lösungsansatzes:

- Mir sind zurzeit keine besonderen Vorteile gegenüber dem Lösungsansatz 2.) bekannt.

Nachteile des 1.) Lösungsansatzes:

- Es existiert eine Vielzahl von BIOS-Varianten, die den Wert des TOM-Zeigers beim Bootvorgang schon erniedrigen.
- Der Interrupt 13h könnte auf einen Festplattentreiber im konventionellen Speicher zeigen.
- Es gibt ein paar Bootviren, die nicht den TOM-Zeiger verändern (z. B. Horse und mehrere Hybridviren).

- Tarnkappenviren können eventuell nicht erkannt werden.
- Nichtspeicherresidente (Lader) Viren können nicht erkannt werden.

Vorteile des 2.) Lösungsansatzes:

- Es wird jeder Virus erkannt, der den MBR verschiebt, ersetzt oder verändert hat.
- Durch das Tunneln können auch aktive Tarnkappenviren erkannt werden.
- Es werden auch nicht speicherresidente Viren erkannt.
- Es werden auch die Viren erkannt, die sich im HMA-, UMB-Speicher oder in der Interrupttabelle (IVT) einnisten.

Nachteile des 2.) Lösungsansatzes:

- Das Tunneln könnte mit dem BIOS inkompatibel sein[34].
- Die Tunnelroutine ist aufwendig zu programmieren, deshalb passt sie eventuell nicht mehr in den neuen MBR hinein.
- Eine Tunnelroutine ist sehr virentypisch und wird wahrscheinlich von regelbasierenden Virensuchprogrammen als unbekannter Virus erkannt.
- Viren können, -so wie Neuroquila- den neuen MBR zu ihren Gunsten modifizieren (patchen).
- Ein Verschlüsseln des MBR ist nicht mehr möglich (s. o.).

[34]Dies konnte ich jedoch in zahlreichen Tests auf verschiedenen Rechner nicht nachweisen.

10.8 Implementierungsansatz

Der neue MBR soll nach Lösungsansatz 2.) folgende Funktionen durchführen:

Stack und Datenregister initialisieren	
Eigenen Code an sichere Stelle 0000:1000h verschieben, dabei eventuell den nachfolgenden Programmcode entschlüsseln	
>>> Tunneln des Diskketteninteruptes INT 13h <<<	
Laden des MBR an 0000:7c00h mit neuem INT 13h	
Vergleichen des eigenen Codes, mit dem Code an 0000:7c00h	
Unterschiedlich?	
Ja?	Nein
Meldung: Virus aktiv! Stopp!	%
Ist eine aktive Partition vorhanden?	
Ja	Nein
%	**Fehlermeldung „NO Part.!“, Stopp!**
Aktiv markierten Bootsektor nachladen	
Ist der Bootsektor (System) überhaupt ladbar?	
Ja	Nein
%	**Fehlermeldung „NO SYS!“, Stopp!**
Bootsektor durch Sprung nach 0000:7c00h aktivieren	

Abb.: Schematische Funktion des MBR-Ladeprogrammes

10.8.1 Die einzelnen Teilphasen

Die Implementierung kann und soll in Teilphasen erfolgen. Grob kann zwischen zwei verschiedenen Phasen unterschieden werden:

Erste Phase: Nachbilden der Ladefunktion des eigentlichen MBR, wie z. B. der MBR von FDISK von MS-DOS**.**

Zweite Phase: Vergleich des eigenen Codes mit dem nachgeladenen MBR auf Veränderungen.

Dritte Phase: Zusätzliches Tunneln des Interruptes 13h.

Zusätzliche Phasen: Verschlüsselung des MBR Laders oder sonstige Funktionen, die sinnvoll sind.

Phase Eins (minimal lauffähiger MBR-Lader) hat grob folgenden Ablauf:

Stack und Datenregister initialisieren	
Eigenen Code an sichere Stelle 0000:1000h verschieben	
Ist eine aktive Partition vorhanden?	
Ja	Nein
%	**Fehlermeldung, Stopp**!
Bootsektor der aktiven Partition nach 0000:7c00h laden	
Bootsektor durch Sprung nach 0000:7c00h aktivieren	

Abb.: Entwicklungsphase „Eins“ des MBR-Lader

Zusätzlich zum neuen MBR muss noch das eigentliche „Impfprogramm“ entwickelt werden, das den alten MBR einliest, die Partitionstabelle sichert und einen neuen MBR mit den alten Partitionsdaten zurückschreibt. Weil der neue MBR sowieso in Assembler entwickelt werden muss, ist es meines Erachtens sinnvoll, das komplette Impfprogramm in Assembler zu entwickeln. Unter C ist es aufgrund der verschiedenen Compilerdialekte nicht möglich ein portables Programm zu schreiben, welches BIOS Funktionen benützt. Ferner ist in Assembler der Zugriff auf den Partitionssektor um einiges einfacher zu implementieren.

Die Tunnelroutine kann nur in (Inline-)Assembler implementiert werden, weshalb sich die Frage nach der Programmiersprache fast erübrigt. Gewählt wurde der Makro Assembler von Microsoft (MASM 6.0), weil mit diesem Compiler elegant mit Makros und Hochsprachenkontrollkonstrukten programmiert werden kann. Ein weiteres Programm (TRACER.PAS) befindet sich auf der Diskette, das den Interrupt 21h tunnelt. Unter Turbo-Pascal ist es möglich, Inline-Assembler zu verwenden, wie der Quellcode TRACER.PAS zeigt. Wie gesagt, es wäre möglich gewesen, das Impfprogramm in C/C++ oder Turbo-Pascal zu programmieren, dann könnte das Programm jedoch nicht so elegant Methoden verwenden, um eventuelle Stealthviren zu umgehen (siehe unten).

10.9 Inkrementelle Softwareentwicklung

Bevor das eigentliche Programm (MBR-Kill) entwickelt wird, sollte sichergestellt werden, dass die Tunnelroutine funktioniert und kurz genug ist, um in den MBR-Lader integriert zu werden. Aus diesem Grund entstand das Programm TUNNEL.ASM. Es zeigte sich schnell, dass der erste Ansatz für die Tunnelroutine von einem Virus umgangen werden konnte, nämlich dem Virus „AntiExe“. Dieser Virus ruft nicht den original Festplatteninterrupt (INT 13h) per Interruptchaining[35] auf, sondern legt den INT 13h auf den INT D3h um, wodurch die Tunnelroutine vorzeitig abbricht.

Ferner waren einige Fragen bezüglich TOM-Zeiger, CMOS und BIOS zu klären. Aus diesem Grund entstand das Programm BOTSTLTH.ASM (Bootstealth), das diese kritischen Bereiche untersucht und auch auf dem Bildschirm anzeigt. Einige Routinen aus dem Programm Bootstealth fanden dann auch Verwendung im Programm MBR-Kill!

[35] Darunter versteht man das Einklicken in einen Interrupt, der zuerst auf den eigenen Programmcode zeigt. Nach Abarbeitung des eigenen Programmcodes wird per FAR Call oder per FAR Jump die original Interruptadresse angesprungen.

Ein weiteres Programm (MBRINFO.C) wurde in ANSI-C (Quick-C 2.5) entwickelt, das interessante Informationen über die Partitionstabelle wie Größe oder Art ausgibt. Ferner kann mit einer Option der Inhalt des Masterbootrekords als „ASCII-Dump“ auf den Bildschirm ausgegeben werden.

Alle hier erwähnten Programme befinden sich als Quellcode und als ausführbare Programme auf der Diskette im Anhang. Für die Programmierung wurde hierzu die Interruptliste von Ralf Brown [Brow44] verwendet.

10.10 Tunneltechniken und Emulation-Engine

Wie schon weiter oben erwähnt, habe ich den ersten Ansatz für ein Tunnel-Engine verworfen, weil diese Lösung vom AntiEXE-Virus umgangen werden konnte. Zunächst ist nochmals kurz zu erklären, wie Tunneln funktioniert; eine kurze Erklärung wurde schon im Kapitel „tunnelnde Viren“ gegeben.

Unter Tunneln versteht man die schrittweise Abarbeitung von Maschinenbefehlen im Einzelschrittmodus durch die CPU, wobei die Tunnelroutine vor jedem neuen Maschinenbefehl aufgerufen wird. Dieser Vorgang wird auch als Tracing bezeichnet und das Programm, dass das Tunneln überwacht (die aufzurufende Tunnelroutine) als Tracer.

Für das Tunneln muss die CPU mittels des sog. Trap-Flag (TF) in den Einzelschrittmodus umgeschaltet werden. Danach wird nach jedem Befehl von der CPU aus die Register CS und IP sowie die Flags auf dem Stack gesichert und die aktuelle Programmausführung unterbrochen. Anschließend wird das Trap Flag gelöscht und ein Interrupt 01 ausgelöst. Normalerweise zeigt der Interrupt 01 auf IRET (Interrupt Return), der die gesicherten Flags und Register vom Stack holt und dann zur aufrufenden Routine zurückkehrt.

Um einen Tracer zu schreiben muss also eine Tracerroutine erstellt werden, der Interrupt 01h auf diese Routine „umgebogen“ werden und das Trap-Flag gesetzt werden. Nach erfolgreichem Abarbeiten des zu tracenden Codes muss natürlich das Trap-Flag wieder ausgeschalten werden, und anschließend der Interrupt 01h wiederhergestellt werden.

Ich möchte hier auch noch kurz darauf hinweisen, dass Tracing oft für das Ermitteln der originalen Einsprungspunkte von Interrupts verwendet wird, dem „Interrupttracing“. Es gibt jedoch noch andere Verwendungsmöglichkeiten von Tracing, die jedoch meistens in Hackertools oder regelbasierenden Virenkillern (z. B.: RVK) Verwendung finden. Ich will nochmals auf das Programm TUNNEL.ASM verweisen, welches demonstriert, wie man den Interrupt 13h (Festplatteninterrupt) tunneln kann.

Nun ist es jedoch relativ einfach möglich, wiederum ein Stück Programmcode so zu schreiben, dass wenn es getraced wird, einfach das Trap-Flag löscht und somit die Tunnelroutine abhängt! Um dies zu verhindern hat das Programm MBR-Kill zusätzlich eine hochkomplexe Emulation Engine integriert, die aufgrund ihres erheblichen Codeumfanges (über 200 Bytes) nicht im MBR Platz hat und deshalb nur vom Programm MBR-Kill direkt benützt werden kann. Diese Emulation Engine, die man getrost als Kleinod der Assemblerprogram-

mierung bezeichnen darf, verwendet ebenfalls einen Tracer, der jedoch einen kompletten Codeanalyser integriert hat und entsprechenden kritischen Code erkennt und gegebenenfalls einfach emuliert! Das heißt konkret dass z. B. **sämtliche** Antivirenprogramme, die Interrupttracing erkennen und blockieren können von dieser Emulation Engine umgangen werden[36]! Hierdurch ist es auch möglich, sich durch extrem komplexe und technisch hoch entwickelte Viren „durchzutracen“!

Durch die Verwendung dieser Emulation Engine ist die Verwendung von MBR-Kill extrem sicher und zuverlässig, weil es zurzeit noch keinen Virus gibt, der die Emulation Engine umgehen kann. Das heißt wiederum, dass MBR-Kill deshalb zurzeit **alle aktiven Tarnkappenviren** tunneln kann und sie somit **deaktivieren** kann - der Anwender von MBR-Kill braucht deshalb nicht einmal einen Kaltstart durchzuführen! Nur durch die Verwendung dieser Emulation Engine ist es **überhaupt** möglich, aktive überschreibende Hybridviren wie Neuroquila zu entfernen! Wird nämlich von einer virenfreien Diskette gebootet, ist die Festplatte physikalisch nicht mehr ansprechbar. Wird von der Festplatte gebootet, kann der MBR nicht gecleant werden, weil der Virus Tarnkappeneigenschaften hat!

Ich möchte zu dieser Emulation Engine nicht näheres mehr sagen, weil sie einerseits spezielle Assemblerkenntnisse voraussetzt und weil ich andererseits nicht potentielle Virenprogrammier animieren möchte.

10.11 Wie effektiv ist das Programm?

MBR-Kill wurde mit dem Ziel entwickelt, einen nahezu 100 prozentigen Schutz vor MBR-Viren zu gewährleisten. Das Programm erreicht dies durch mehrere Ansätze:

- Komplett in 80386-Assembler entwickelt.
- Verwendet virenspezifische Techniken und undokumentierte Interruptfunktionen, um Viren zu umgehen.
- Umgeht eventuelle Viren durch spezielle Tunnel- und Emulationstechniken, siehe „Emulation Engine“.
- Das Programm verwendet eine bis jetzt **einzigartige** Technik, den MBR auf eventuelle Viren zu untersuchen.
- Zurzeit wird **jeder** MBR-Virus entdeckt und kann auch wieder **entfernt** werden!
- Es werden auch MBR-Viren wie Neuroquila erkannt, die sich im UMB-, HMA-Bereich oder in der Interrupttabelle (IVT) einnisten.

Durch mehrere wahlfreie Optionen kann das Programm sehr effektiv und flexibel eingesetzt werden, z. B.:

- Laden und Speichern des MBR und der Partitionstabelle auf Diskette.
- Unterstützt werden bis zu zehn Festplatten (BIOS Erkennung 80h-89h).
- Impfen eines virenfreien Systems

[36] Dies wurde von mir mit mehreren Antivirenprogrammen wie etwa VSAFE (MS-DOS 6.xx), SD-Res (Novell-DOS 7.0), VDEFEND oder TSAFE getestet - keines dieser Antivirenprogramme konnten eine MBR Immunisierung durch MBR-Kill blockieren!

10.12 Fazit

Die Verwendung einer Tunnelroutine für die Selbstüberprüfung des MBR ist bis heute vom Konzept her eine Weltneuheit! Ferner verwendet MBR-Kill eine hochkomplexe und extrem zuverlässige Emulation Engine, die eine **allgemeine** Anwendung von MBR-Kill zulässt, wo bisher spezielle Antivirenkiller entwickelt werden mussten!

MBR-Kill ist ein extrem einfach zu bedienendes Programm mit außerordentlichen Sicherheitsmerkmalen, die eine allgemeine Anwendung von MBR-Kill zulassen. MBR-Kill ist somit ein absolut universelles Programm zur Entfernung, Immunisierung und Erkennung von MBR-Viren -egal welche speziellen Tricks ein solcher Virus auch ausnützen kann. Die Programmspezifikationen wurden um einiges übertroffen!

11 Anmerkungen zu den einzelnen Programmen

Im nachfolgenden Kapitel werden die von mir für diese Diplomarbeit entwickelten Programme näher beschrieben. Diese Programme, sowie der Quellcode und teilweise auch Kurzanleitungen („DOC“) befinden sich auf der Diskette im Anhang.

Die Benutzerführung der Programme und die Kommentare sind ausschließlich in Englisch, mit Ausnahme des Programms „TRACER.PAS“. Die Programmbedienung erfolgt über Kommandozeilen Parameter. Die Programme, Quellcode und Anleitungen sind nach folgendem Schema auf der Diskette angeordnet:

- DOC - Anleitungen zu den einzelnen Programmen
- EXE - Ausführbare Programme
- SRC - Quellcodes
  - MBRINFO - Programm MBRINFO in ANSI-C
  - TRACER - Programm TRACER in Turbo Pascal 6.0
  - MBRKILL - Programm MBR-Kill in MASM 6.0
  - SONST - Restliche Assemblerprogramme in MASM 6.0

Als Programmierstandard wurde die Ungarische Namenskonvention „Hungarian Naming“[37] für C++ Programme benutzt und entsprechend an Pascal und Assembler angepasst. Eine Kurzreferenz für meine eigene erweiterte Ungarische Namenskonvention befindet sich im Anhang. Bei den Assemblerprogrammen wurden die Daten und Texte in einem eigenen Abschnitt untergebracht, um später eine Anpassung oder eine Übersetzung in eine andere Sprache zu vereinfachen.

11.1.1 Copyright und Weitergabe

Sämtliche o.g. Programme sind (C)opyright by:

Ralph Roth, Finkenweg 24, D 78658 Zimmern

und dürfen auf gar keinen Fall -auch nicht teilweise- weitergegeben werden! Aus diesem Grund fehlen teilweise wichtige Includedateien, ohne die das Programm nicht kompilierbar ist! Aus diesem Grund sind Ausführbare Programme im Verzeichnis \EXE enthalten!

Ich habe mich dennoch entschlossen, sämtliche Quellcodes zu den unten beschriebenen Programmen auf Diskette dieser Diplomarbeit beizulegen. Somit ist es möglich, anhand der ausführlich kommentieren Quellcodes den Programmablauf nachzuvollziehen.

[37] Hungarian Naming nach Charles Simonyi, siehe [Nort92].

11.2 MBR-Killer

Das Programm MBR-Kill dient dazu, einen neuen (sich selbstüberprüfenden) Master Boot Record (MBR) auf ein System zu übertragen. Der neue MBR-Lader ist im Programm MBR-Kill integriert und wird von diesem auch auf die Festplatte geschrieben. Es gibt mehrere Möglichkeiten, MBR-Kill zu benützen; benötigt wird jedoch ein 80386-SX Prozessor (oder besser) sowie DOS 3.3 (oder besser).

Hinweis: Das Programm MBR-Kill verwendet mehrere undokumentierte Interruptfunktionen [Brow44], um eventuelle Stealthviren umgehen zu können. Diese undokumentierten Interruptfunktionen stehen bei MS-DOS ab Version 3.30 zur Verfügung. DR-DOS 5 unterstützt diese Interruptfunktionen ebenfalls.

Es gibt mehrere Möglichkeiten, MBR-Kill anzuwenden:

1. Impfen (immunisieren) eines virenfreien Systems: Einfach MBR-Kill starten und Sicherheitsabfragen bestätigen.
2. Entfernen eines MBR-Virus aus dem MBR:
   - Falls es sich um einen Virus **ohne** Stealtheigenschaften handelt (z. B. Stoned), kann einfach MBR-Kill ausgeführt werden und der Virus wird entfernt.
   - Sollte der MBR durch einen Stealth-Virus (z. B. Parity_Boot, AntiExe oder Ripper) infiziert sein, sollte zur **Sicherheit** von einer Systemdiskette gebootet werden und darauf direkt MBR-Kill ausgeführt werden. Nach dem anschließenden Kaltstart müsste der Virus entfernt worden sein.
   - Sollte es sich um einen „überschreibenden“ Stealth-Virus handeln (z. B. Neuroquila, Monkey.B oder Goldbug), sollte MBR-Kill direkt auf dem verseuchten System gestartet werden. Das Programm liest dann den MBR **mit Hilfe** des **aktiven** Virus ein und erhält somit den **ursprünglichen** MBR! Anschließend tunnelt und emuliert MBR-Kill sich durch den aktiven Virus hindurch und schreibt dann den neuen MBR-Lader mit den richtigen Werten zurück, ohne dass der Virus hiervon etwas „bemerkt“.
3. Sichern und Rekonstruktion von Partitionsdaten und des zugehörigen MBR Laders auf Disketten.

11.2.1 Unterstützte Funktionen

Das Programm MBR-Kill kann mit zusätzlichen Parametern gestartet werden. Parameter können wahlweise mit einem Minuszeichen „-“ oder einem Schrägstrich „/“ eingeleitet werden. Der nachfolgende Buchstabe darf klein oder groß geschrieben werden. Zwischen den einzelnen Optionen muss ein Leerzeichen stehen! Die Reihenfolge der Parameter ist ohne Bedeutung.

MBR-Kill liest und schreibt grundsätzlich seine gesicherten Daten nur auf Disketten im Laufwerk A:, damit der Benutzer die Daten nicht „versehentlich“ auf der Festplatte speichert. Wenn nämlich ein MBR-Virus die Festplatte infiziert hat, kann beim Entfernen des Virus oftmals nicht mehr auf die Festplatte zugegriffen werden. Dann ist es nützlich, wenn man noch eine Kopie des alten MBR auf einer Diskette hat! Der gesicherte MBR wird in

der Datei MBRPART?DAT gespeichert. Das Fragezeichen repräsentiert hierbei das physikalische Laufwerk, also „0“ für die erste Festplatte, eine „1“ für die zweite Festplatte usw.

Wird MBR-Kill ohne Parameter gestartet, so versucht das Programm den ursprünglichen MBR zu lesen, sichert dessen Partitionsdaten und schreibt einen neuen sich selbstüberprüfenden MBR Lader zurück.

Es stehen zurzeit folgende Parameter (Optionen) zur Verfügung:

Option	Beschreibung
/?	Eine kurze Hilfestellung zum Programm ausgeben.
/l /r	Laden des speicherten MBR von Diskette mit anschließendem Zurückschreiben.
/s /w	Speichern des MBR auf Diskette, bevor dieser durch den neuen, sicheren MBR Lader ersetzt wird.
/q	Nur Laden oder Abspeichern auf Diskette und anschließend das Programm beenden. Ist z. B. sinnvoll, wenn nur der MBR gespeichert oder wiederhergestellt werden soll.
p*x*	Verwendet anstatt der standardmäßigen ersten Festplatte im System die physikalische Festplatte mit der Nummer x. Die Festplatten werden mit Null beginnend aufsteigend durchnumeriert. Unterstützt werden die Festplatten 0-9, was der BIOS Laufwerksnummer 80h-89h entspricht.

Tabelle: Mögliche Aufrufoptionen für MBR-Kill

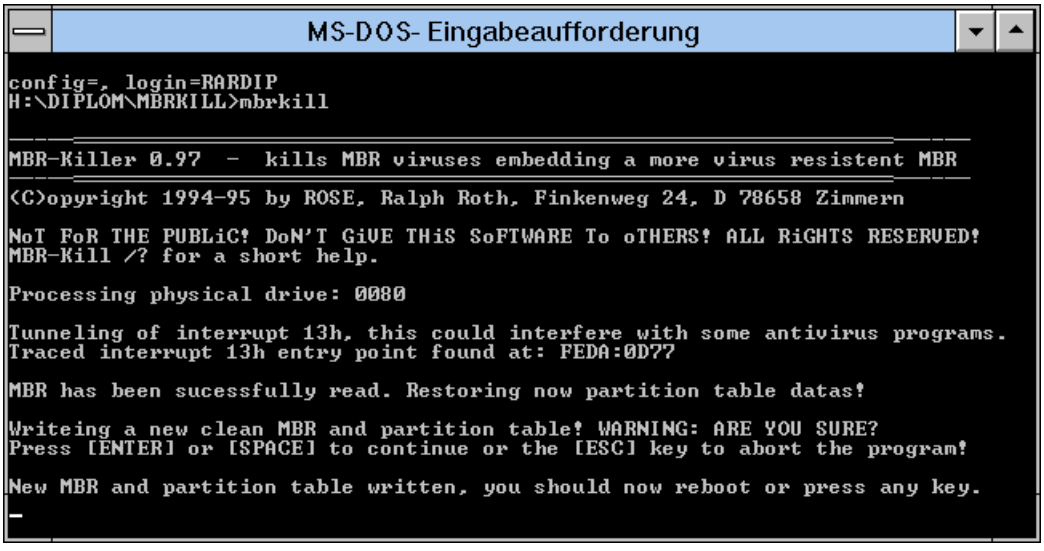


Abb.: Bildschirmmeldung des Programms MBR-Kill beim „Impfvorgang“

Beispiel:

MBR-Kill p1 -s -q

Sichert den MBR und die Partitionsdaten der zweiten Festplatte auf Diskette und bricht danach ab.

11.2.2 Wie kann ein MBR-Virus entfernt werden?

Nachfolgend wird beschrieben, wie z. B. der Parity_Boot Virus entfernt werden kann. In unserem Beispiel hat der Virus zusätzlich die zweite Festplatte infiziert, kann aber von dieser nicht aktiviert werden.

Generell empfehle ich vor dem Einsatz von MBR-Kill einen Backup des Systems zu machen, weil das Überschreiben des MBR und der Partition durch MBR-Kill von einem Virus abgefangen werden könnte, der speziell an MBR-Kill angepasst wurde, was zu Datenverlust führen könnte! Ferner wird eine Diskette ohne Schreibschutz benötigt, die unter DOS formatiert wurde, um den alten MBR abspeichern zu können.

Zuerst muss der Virus von der ersten Festplatte entfernt werden. Dies wird durch folgenden Befehl erreicht (Speichern des alten, [infizierten][38] MBR auf Diskette mit anschließendem Ersetzen des MBR):

MBR-Kill -s

Wurde nicht von einer virenfreien Diskette gebootet muss jetzt der Virus im Arbeitsspeicher durch einen Kaltstart zerstört werden (in diesem Fall ist die Diskette mit dem gespeicherten alten MBR auch vom Virus infiziert!).

Anschließend wird die zweite Festplatte gereinigt und vorher der alte MBR wieder auf Diskette gespeichert. Dies erfolgt durch folgenden Befehl:

MBR-Kill p1 -s

Anschließend ist das System virenfrei. Sollten Sie nicht von der Festplatte booten können so können Sie den alten MBR wiederherstellen. Dies erfolgt analog der obigen Prozedur: Statt des Parameters **-s** wird dann der Parameter **-w** verwendet!

11.2.3 Entfernen von „überschreibenden" Stealthviren

Wie schon weiter oben erwähnt, ist es für das Programm MBR-Kill kein Problem, überschreibende (aktive) Tarnkappenviren wie Neuroquila, Goldbug oder Monkey zu entfernen. Hierzu muss nur folgendes beachtet werden:

Das System wird ganz normal gestartet, der Virus ist nun aktiv. MBR-Kill wird gestartet und liest den MBR ganz normal ein. Weil der Virus aktiv ist, wird der original MBR an

[38] Der Parity-Boot Virus ist ein Tarnkappenvirus, deshalb erhalten wir den original MBR vom Virus zurück!

MBR-Kill zurückgeliefert, der dann z. B. auch auf Diskette gespeichert werden kann. Anschließend tunnelt und emuliert sich das Programm durch den aktiven Virus hindurch und schreibt einen sauberen MBR auf die Festplatte, ohne dass der Virus etwas „bemerkt“. Anschließend muss sofort ein Kaltstart durchgeführt werden, weil ansonsten der aktive Virus sofort wieder den MBR infizieren wird!

11.2.4 Der Quellcode von MBR-Kill

Der nötige Quellcode zum Programm MBR-Kill befindet sich auf der Diskette im Verzeichnis \SRC\MBRKILL und kann mit dem Programm PR.BAT kompiliert werden. Das Hauptprogramm ist in der Datei MBR.ASM enthalten.

11.3 Tunnel-Demo

Das Programm TUNNEL.ASM enthält den Quellcode für einen Interrupt-Tracer für den Interrupt 13h (Festplatteninterrupt). Der Interrupt 13h muss ja vom Programm MBR-Kill getraced werden, um den original Interruptvektor ermitteln zu können. Nur so ist es überhaupt möglich, eventuelle Tarnkappenviren zu umgehen!

Die Codierung erfolgte aus Platz- und Effizienzgründen in 80386 Assembler. Zusätzlich enthält dieses Programm zu Test- und Demonstrationszwecken die komplette Emulation-Engine.

Ich habe dieses Programm vor der Implementierung des Programms MBR-Kill geschrieben, um zu testen, ob der Interrupt-Tracer funktioniert. Im ersten Anlauf war es jedoch dem Virus „AntiExe“ möglich, diesen Tracer zu umgehen. Die zweite Programmversion konnte jedoch auch diesen Virus ausschalten, siehe Kommentare im Quellcode. Zusätzlich hat nun die dritte Programmversion die oben beschriebene Emulation Engine integriert. Mit dieser Emulation-Engine ist es nun möglich, jeden zurzeit existierenden Virus durchzutracen und somit die originalen Festplatten Interrupt zu ermitteln!

11.4 Tracer

Das Programm TRACER.PAS enthält ebenfalls den Quellcode für einen Interrupt-Tracer, diesmal jedoch für den Interrupt 21h (MS-DOS). Dieses Programm soll demonstrieren, wie es in einer Hochsprache generell möglich ist, mit Inline-Assembler eine brauchbare Tracerroutine zu implementieren. Zusätzlich ermittelt dieses Programm verschiedene Systemparameter und gibt diese auch auf dem Bildschirm aus.

Als praktische „Anwendung“ zählt das Programm mit, wie viele Codesegment-Wechsel (CS) durchgetraced wurden, bis der original MS-DOS Einsprungspunkt gefunden wurde. Dies lässt u.a. Rückschlüsse zu, wie viel Programme den Interrupt 21h überlagern. Zusätzlich hat der Tracer einen kleinen Codeanalyser integriert, der mitzählt, wie viel Programme die Funktion 21h/4Bh-EXEC (siehe [Brow44]) überlagern. Dieses sollte normalerweise nur ein Programm tun: Der Kommandointerpreter (COMMAND.COM). Alles andere könnte auf einen speicherresidenten Virus hinweisen, der Programme infizieren kann.

11.5 MBR-Info

Das Programm „MBRINFO.C“ wurde in ANSI-C geschrieben und kann mit dem QC 2.5 oder dem MSC Compiler kompiliert werden. Das Programm dient dazu

- detaillierte Informationen über alle vier Partitionen auszugeben
- einen ASCII Dump des Master Boot Record auf dem Bildschirm darzustellen.
- kann auch per Kommandozeile die zweite Festplatte untersuchen.

Eine besondere Eigenschaft des Programms ist es, zurzeit über 40 verschiedene Betriebssystemtypen unterscheiden zu können und diese in „Textform“ ausgeben zu können. Die Konstanten und Bezeichnungen hierfür befinden sich in der Datei „MBRINFO.H“

12 Zukünftige Betrachtungen

Computerviren sind die erste und bis jetzt einzige Form von künstlichen Leben, die einen nachweisbaren Eindruck bei der Bevölkerung hinterlassen haben. Zurzeit sind Computerviren noch eine „lästige“ Begleiterscheinung, die noch einigermaßen beherrscht werden kann. Dass dies jedoch in Zukunft nicht mehr so sein wird, möchte ich im Folgenden erläutern.

12.1 Trends in der Virenentwicklung

Die Qualität der neu auftauchenden Viren (Hybrid-, Stealth- und polymorphe Viren) hat zwischenzeitlich sehr stark zugenommen, eine Entfernung stellt den Anwender deshalb oft vor nicht mehr überwindbare Hürden. Durch die internationale Vernetzung verbreiten sich die Viren jetzt auch viel schneller als in den vergangenen Jahren.

Von vielen „alten“ Viren tauchen modifizierte Varianten auf, die oft so verändert wurden, dass sie nicht mehr von einem Virensuchprogramm erkannt werden. Die Rate neuer Viren hat in den letzten Jahren stetig zugenommen, es herrscht zurzeit fast eine exponentielle Verdopplung der neu gefundenen Viren (siehe auch [Keph94]). Laut [Keph94] dürfte dann die Anzahl der gefunden Computerviren für MS-DOS im Jahre 2000, etwa 10 Millionen erreichen, was ich aber eher für unwahrscheinlich halte.

Zurzeit werden täglich zwei bis drei neue Viren für das Betriebssystem MS-DOS gefunden! Die Hersteller von Antivirenprogrammen kommen zurzeit kaum nach, alle neuen Viren zu erfassen, ein paar kleinere Hersteller sind deshalb auch schon „auf der Strecke geblieben".

12.1.1 Virenkits und Mutation Engines

Die „Profis“ unter den Virenschreibern programmieren nicht mehr Viren, sondern Virenkits oder Mutation Engines.

Virenkits ermöglichen es auch dem blutigen Anfänger verschiedene lauffähige Viren bzw. kompilierbaren Assemblerquellcodes zu erzeugen. Das Fatale daran ist, dass jeder Virus anders aussieht und sich auch anders verhält. Der Benutzer kann auch den erzeugten Quellcode ändern, modifizieren oder erweitern, um eine eventuelle Entdeckung durch Virenscanner zu vermeiden. Zurzeit existieren über 30 verschiedene Virenkits! Es existieren z. B. von den bekanntesten Virenkits schon mindestens 500 verschiedene Viren, welche teilweise hervorragend programmiert sind.

Ein weiterer Trend sind polymorphe (polymorph = vielgestaltig) Viren, die in den letzten zwei Jahren sprunghaft zugenommen haben. Polymorphe Viren sind so komplex verschlüsselt, dass sie teilweise nicht mehr mit einem Virensuchprogramm erkannt werden können!

Jeder Virenschreiber, der etwas auf sich hält, „muss“ eine Mutation Engine programmiert haben. Seit „Dark Avenger“ als erster seine MtE veröffentlicht hat, gibt es eine Flut von Mutation Engines, die oft als Objektfile verteilt werden. Wie schon unter „polymorphe Vi-

ren“ erwähnt, wird es immer schwieriger, Erkennungsroutinen für polymorphe Viren zu entwickeln, die zuverlässig sind.

Es ist auch schon vorgekommen, dass einfach eine neue Mutation Engine zu einem alten Virus hinzugefügt wurde. So entstand z. B. aus dem CoffeeShop-Virus der MtE:CoffeeShop und aus diesem der TPE:CoffeeShop.1_0a. Aus dem TPE:CoffeeShop.1_0b-Virus entstand dann die TridenT Polymorph Engine (TPE.1_1), eine weitere Mutation Engine.

12.1.2 Multipartite Viren

Ein weiterer Trend sind Multipartite- oder Hybrid-Viren mit Stealtheigenschaften, die sowohl Dateien als auch Bootsektoren und/oder Masterbootrekords infizieren. Da solche Viren schon zum Zeitpunkt des Bootens aktiv werden, können sie sehr schwer entdeckt und entfernt werden. Zusätzlich ist diese Virenart extrem virulent, weshalb sich ein Hybrid-Virus wie z. B. Natas oder Junkie innerhalb von ein paar Wochen weltweit verbreitet. Zusätzlich sind die meisten Hybrid-Viren verschlüsselt, teilweise sogar polymorph. Der Tequila- und Flip-Virus (ca. 1992) waren die ersten „richtigen“ polymorphen Hybrid-Viren mit minimalen Stealtheigenschaften, die sich rasant vermehrten und deshalb wahrscheinlich viele Nachahmer fanden. Für die Antivirenforscher ist es zudem mühsam, solch einen Virus zu analysieren, den sie haben ohne weiteres eine Größe von 3 bis 6 KB (teilweise sogar 8 bis 10 KB), was ungefähr 40-120 Seiten disassemblierten Assembler-Code entspricht!

12.1.3 Devicetreiberviren unter Windows

Unter Windows gibt es die Möglichkeit, so genannte Gerätetreiber (Devicedrivers oder Devicetreiber) für Windows zu programmieren. Dies geht relativ einfach mit dem SDK[39] für Windows. Diese Devicetreiber können, nach Einbindung in Windows, sämtliche Dateizugriffe überwachen - und entsprechend manipulieren. Wenn man sich den Clustervirus DIR-II[40] anschaut, der sich innerhalb kürzester Zeit weltweit verbreitet hat, kann man sich vorstellen, wie effektiv sich ein solcher Windows Devicetreiber-Virus verbreiten könnte. Anscheinend ist noch niemand auf diese Idee gekommen oder es fehlt immer noch an genügend Hintergrundwissen, solch einen Virus zu entwickeln. Vielleicht liegt es auch an der Tatsache, dass das SDK relativ teuer ist, und deshalb nicht von Virenprogrammiern verwendet wird.

Meiner Meinung nach, könnte 1995 das Jahr sein, in dem eine Flut von Windowsviren auftauchen könnte. Der Hintergrund: Viele Virengruppen sind nicht mehr daran interessiert die hundertste gehackte Variante vom xyz-Virus zu veröffentlichen. Die Programmierung von einem Windowsvirus ist hier noch eine echte Herausforderung[41]. Den gleichen Trend gibt es zurzeit bei Mutation Engines und Hybridviren, der hoffentlich in nächster Zeit wieder abflacht.

[39] SDK = Software Developper Kit von Microsoft. Zum Programmieren von Windowsanwendungen.

[40] DIR-II fängt Devicetreiberaufrufe ab und infiziert dabei geeignete Wirte.

[41] Sozusagen als Bestätigung kam eine Woche, nachdem ich diesen Abschnitt geschrieben habe, der erste residente Virus (Winsurfer) für Windows von der Gruppe „VLAD“ heraus. Er soll auch unter Windows-95 funktionieren!

12.2 Internationale Vernetzung

Seit der Zugang zu Netzen wie FIDO oder Internet relativ einfach ist, wird dieses Medium von den Virenprogrammiern benutzt. So wird auf IRC[42] im Kanal #virus offen Quellcodes von Viren und lauffähige Viren verteilt. Es existieren eine Vielzahl von BBS Mailboxen und ftp-sites, in denen man anonym Viren, Quellcode, Mutation Engines und Virenkits herunter- oder hochladen kann. Hier stellt kein Sysop oder Sysadmin lästige Fragen, weshalb hier auch viele illegale Transaktionen stattfinden!

Ein weiterer Trend ist das Fluten von Nachrichtenbereichen (news groups) mit Viren und Quellcodes, wie etwa die Nachrichtenbereiche „FIDO: Virus.Germany“ oder „alt.comp.virus“. Diese Nachrichtenbereiche beschäftigen sich hauptsächlich mit der Bekämpfung von Viren! Die Moderatoren sind meistens absolut hilflos, dieses Fluten zu unterbinden, weil die Mails meistens von sog. „Fake-Accounts“[43], anonymen Accounts oder gehackten Accounts versendet werden!

So wurden alleine im April 1995 mehr als 3200 **verschiedene** Computerviren in dem Forum „alt.comp.virus“ veröffentlicht, eine wahre Fundgrube für jeden Hacker und angehenden Virenprogrammierer! Das sich so neue Viren innerhalb von nur einer Woche weltweit verbreiten können, dürfte mit diesem Hintergrundwissen jetzt einleuchtender sein.

[42] Internet Relay Chat - Foren, in denen man sich Online unterhalten kann.

[43] z. B. vom Benutzer „dunno“ oder „slash“ vom Rechner „i.dont.know“

13 Schlussbemerkungen

Rückblickend hatte ich primär mit dem folgenden Problem am meisten zu kämpfen:

- Es war überhaupt kein qualifiziertes Informationsmaterial über Computerviren und deren Aufbau erhältlich.
- Es ist extrem schwierig für „Nichtvirenforscher“, an existierende Computerviren zu Testzwecken heranzukommen, um z. B. das Programm MBR-Kill effektiv austesten zu können.
- Zur Programmierung in 80386 Assembler sowie zur Programmierung eines Interrupt-Tracers gibt es so gut wie keine Literatur.
- Zur Programmierung einer Emulation-Engine gibt es keine Literatur - sie ist aus meinen eigenen Überlegungen und Erfahrungen entstanden.

Dass ich dennoch das Programm MBR-Kill fertig gestellt bekommen habe, liegt darin begründet, dass ich aus diesem Grund über viertausend verschiedene MS-DOS Viren analysiert habe und spezielle Vireneigenschaften wie z. B. die des AntiExe, Parity_Boot oder Neuroquila Virus in das Design des Programms MBR-Kill berücksichtigt habe. Ferner kommt hinzu, dass ich seit mehr als fünf Jahren in Assembler programmiere, was natürlich die Programmentwicklung von MBR-Kill um einiges beschleunigt hat.

Zusammenfassend kann ich sagen, dass das Programm MBR-Kill voll und ganz meinen Erwartungen entspricht. Es erfüllt alle gestellten Anforderungen und lässt sich auch ohne Kenntnisse über Computerviren vom Normalanwender bedienen. Die eingebauten Sicherheitsmerkmale können bis jetzt von keinem existierenden Virus umgangen werden. Das macht MBR-Kill zukunftssicher, d. h. es wird in nächster Zeit kaum Erweiterungen benötigen.

Alleine während der Erstellung der Diplomarbeit wurde MBR-Kill von mehreren hundert Benutzern erfolgreich eingesetzt. Aus diesem Grund habe ich vor, das Programm MBR-Kill als Shareware zu veröffentlichen.

14 Anhang

14.1 Literaturverzeichnis

Folgende Literatur wurde in der Diplomarbeit verwendet:

Abkürzung	Autor, Titel, Beschreibung und Verlag
[Bont94]	Bontchev, Vesselin „Known Polymorphic Viruses“, 1994 ftp.informatik.uni-hamburg.de:/pub/virus/texts/viruses/plymrphs.zip
[Bont94b]	Bontchev, Vesselin „Are ´Good´ Computer Viruses Still A Bad Idea?“, 02.11.94 ftp.informatik.uni-hamburg.de:/pub/virus/texts/viruses/goodvir.zip
[Bontch]	Bontchev, Vesselin; "Future Trends in Virus Writing". ftp.informatik.uni-hamburg.de:/pub/virus/texts/viruses/trends.zip
[Brow44]	Brown, Ralf; Interrupt List, Revision 44 Site: FTP.CS.CMU.EDU [128.2.206.173] Dateien: /afs/cs.cmu.edu/user/ralf/pub/inter??a.zip bis inter??d.zip
[Dier90]	Dierstein, Rüdiger. Kommentar: Amerika und die europäischen IT-Sicherheitskriterien. Nr. 6 1990, S. 405-407
[Esch93]	Virus Ratgeber Version 2.0; Reiner Eschen; 11. Juli 1993 BBS: Reign in Blood, 30455 Hannover, FIDO: 2:241/23, Magic: VIRRAT, Telefon: 0511/493521
[FAQG93]	VIRUSGER.FAQ - FAQ der FidoNet-Area VIRUS.GER, Malte Eppert, Version 1.2, 03.03.1993
[FAQL92]	VIRUS-L.FAQ - Frequently Asked Questions on VIRUS-L/comp.virus, Last Updated: 18 November 1992, 7:45 AM EST, Anonymus FTP on cert.org (192.88.209.5) in the file: pub/virus-l/FAQ.virus-l
[FBul216]	F-PROT Professional Update Bulletin 2.16 - Information on the global computer virus situation. ASCII version. 1995 ftp.datafellows.fi:/pub/f-prot/bull-216.zip
[Ferb92]	Ferbach, D. „A Pathology of Computer Viruses“, 312 pp. pb. ca. DM 74,-- Springer Verlag London Limited 1992
[Jour86]	Jourdain, Robert L. „Programmer´s problem solver for the IBM PC, XT and AT“; Prentice Hall Press, 1986
[Keph94]	Kephart, Jeffrey O. „A Biologically Inspired Immune System for Computers“; 1994 ftp.informatik.uni-hamburg.de:/pub/virus/texts/viruses/immune.zip
[LANl95]	Zeitschrift „LANline - Das Magazin für Netze und Kommunikation“, AWI LANline Verlagsgesellschaft mbH, Ausgabe 3/95, S. 20 ff.
[Micr94]	Microsoft, „Systemhandbuch Microsoft Win NT Workstation - Version 3.5“, 1994

[Morr88] „Die großen Systeme reizten Robert“. Der Spiegel, Nr. 47 1988, S. 252-265. Anmerkung: Der „Internet-Worm“

[Nort92] Norton, Peter: Windows 3 - Programmiertechniken für Profis - „Hungarian Naming nach Charles Simonyi“, S. 56, Markt & Technik, 1992.

[PB92] Polk, W. T. & Bassham, L. E.; "A Guide to the Selection of Anti-Virus Tools and Techniques", National Institute of Standards and Technology - Computer Security Division, 2. Dec. 1992

[Rada93] Radai, Yisrael „The Anti-Viral Software of MS-DOS 6“, 9. Sept. 1993

ftp.informatik.uni-hamburg.de:/pub/virus/texts/viruses/msaveval.zip

[Slad95] Slade, Robert M.; „VirEthics“, 20.03.1995

ftp.informatik.uni-hamburg.de:/pub/virus/texts/viruses/virethics.zip

[Smad94] Smadja, Birgit „Novell Netware: Einführung, Arbeitsbuch, Nachschlagewerk“, Markt und Technik, Haar bei München, ISBN 3-87791-487-X; 1994

[Treb92] Treber, C. „Systemprogramme gegen Computerviren“, 1992, Hanser-Verlag

Folgende Literatur möchte ich dem interessierten Leser empfehlen. Sie wurde nicht direkt in dieser Diplomarbeit verwendet, ich habe jedoch einige Anregungen darin gefunden.

Bontchev, Vesselin „Possible Virus Attacks Against Integrity Programs And How To Prevent Them“, Proc. 2nd Int. Virus Bulletin Conf., September 1992

Cohen, F. A. „Short Course on Computer Viruses“, 1990, ca. $48.00, ASP

Davidson, Iolo „The Death of the Clean Boot?“, Virus News International, April 1993

Gold, Steve „Novell Netware Security Breach“, Virus News International, November 1992

Muftic, S. et. al. „Security Architecture for Open Distributed Systems“, 1993, 293 pp. ca. DM 92,--, Wiley-Verlag

Solomon, Alan „PC Viruses: Detection, Analysis and Cure“, 1991. 305 pp. ca. DM 74,--, Springer Verlag

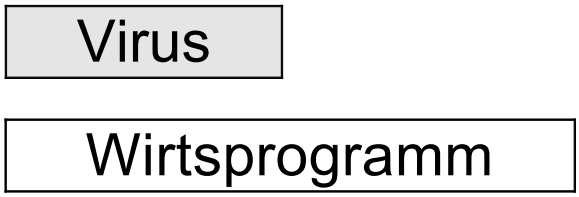
Solomon, Alan „Mechanisms of Stealth“, Proc. 5th Int. Comp. Virus and Security Conf., New York, March 1992

14.2 Abbildungs- und Tabellenverzeichnis

14.3 Infektionsmethoden verschiedener Virengattungen

Dies ist nochmals die Zusammenfassung der verschiedenen Infektionsarten, die bei Dateiviren üblich sind.

14.3.1 Vor der Infektion



14.3.2 Überschreibende Viren

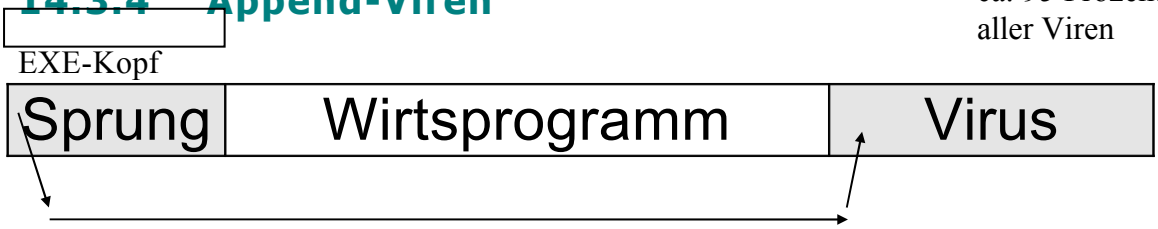
Mini
Trivial
Burger
Leprosy

14.3.3 Prepend-Viren

Jerusalem (COM)
Suriv

14.3.4 Append-Viren

ca. 95 Prozent aller Viren



14.3.5 Permutierende Viren

Commander-Bomber

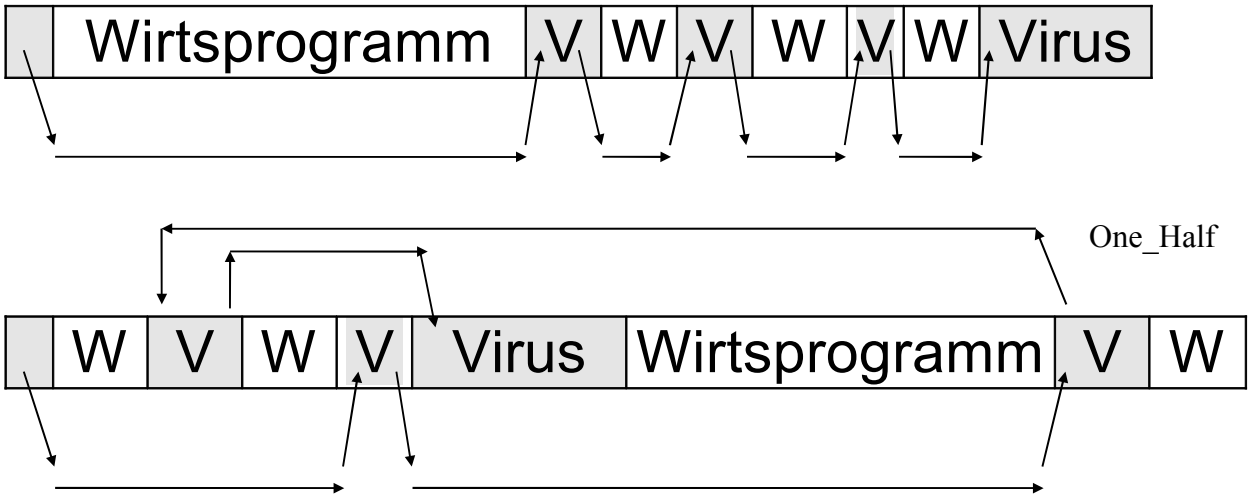


Abb.: Infektionsmechanismus von Dateiviren

14.4 MBR- und Partitionstabelle-Layout

MBR-Lader	Partitionstabelle				Marker
Benutzereigenes Programm	Eintrag 0	Eintrag 1	Eintrag 2	Eintrag 3	55AAh
Offset: 000-1bdh	1beh	1ceh	1deh	1feh	1ffh

Tabelle: MBR- und Partitionstabelle-Layout

14.5 Boot- und MBR-Beschreibung (Englisch)

INT 19 - SYSTEM - BOOTSTRAP LOADER/MBR LAYOUT

Quelle: Interrupt List v44 [Brow44]

Description:
This interrupt reboots the system without clearing memory or restoring interrupt vectors. Because interrupt vectors are preserved, this interrupt usually causes a system hang if any TSRs have hooked vectors from 00h through 1Ch, particularly INT 08.

Notes:
Usually, the BIOS will try to read sector 1, head 0, track 0 from drive A: to 0000h:7C00h. If this fails, and a hard disk is installed, the BIOS will read sector 1, head 0, track 0 of the

first hard disk. This sector should contain a master BOOTSTRAP loader and a partition table (see #0481). After loading the master boot sector at 0000h:7C00h, the master bootstrap loader is given control. It will scan the partition table for an active partition, and will then load the operating system's bootstrap loader (contained in the first sector of the active partition) and give it control.
True IBM PCs and most clones issue an INT 18 if neither floppy nor hard disk have a valid boot sector. To accomplish a warm boot equivalent to Ctrl-Alt-Del, store 1234h in 0040h:0072h and jump to FFFFh:0000h. For a cold boot equivalent to a reset, store 0000h at 0040h:0072h before jumping.
VDISK.SYS hooks this interrupt to allow applications to find out how much extended memory has been used by VDISKs (see #0480). DOS 3.3+ PRINT hooks INT 19 but does not set up a correct VDISK header block at the beginning of its INT 19 handler segment, thus causing some programs to overwrite extended memory which is already in use.
The default handler is at F000h:E6F2h for 100% compatible BIOSes MS-DOS 3.2+ hangs on booting (even from floppy) if the hard disk contains extended partitions which point at each other in a loop, since it will never find the end of the linked list of extended partitions.
Under Windows Real and Enhanced modes, calling INT 19 will hang the system in the same was as under bare DOS; under Windows Standard mode, INT 19 will successfully perform a cold reboot as it appears to have been redirected to a MOV AL,0FEh/OUT 64h,AL sequence.

SeeAlso: INT 14/AH=17h,INT 18,INT 5B"PC Cluster"

14.6 Tabellen MBR und Partitionslayout

Format of hard disk master boot sector:

```
Offset  Size    Description     (Table 0481)
00h 446 BYTEs   Master bootstrap loader code
1BEh 16 BYTEs   partition record for partition 1 (see #0482)
1CEh 16 BYTEs   partition record for partition 2
1DEh 16 BYTEs   partition record for partition 3
1EEh 16 BYTEs   partition record for partition 4
1FEh    WORD    signature, AA55h indicates valid boot block
```

Format of partition record:

```
Offset  Size    Description     (Table 0482)
00h     BYTE    boot indicator (80h = active partition)
01h     BYTE    partition start head
02h     BYTE    partition start sector (bits 0-5)
03h     BYTE    partition start track (bits 8,9 in bits 6,7 of sector)
04h     BYTE    operating system indicator (see #0483)
05h     BYTE    partition end head
06h     BYTE    partition end sector (bits 0-5)
07h     BYTE    partition end track (bits 8,9 in bits 6,7 of sector)
08h     DWORD   sectors preceding partition
0Ch     DWORD   length of partition in sectors
```

SeeAlso: #0481

(Table 0483) **Values for operating system indicator:**

```
00h    empty
01h    DOS 12-bit FAT
02h    XENIX root file system
03h    XENIX /usr file system (obsolete)
04h    DOS 16-bit FAT
05h    DOS 3.3+ extended partition
06h    DOS 3.31+ Large File System
07h    QNX
07h    OS/2 HPFS
07h    Advanced Unix
08h    AIX bootable partition, SplitDrive
09h    AIX data partition
09h    Coherent filesystem
0Ah    OS/2 Boot Manager
0Ah    OPUS
0Ah    Coherent swap partition
10h    OPUS
11h    OS/2 Boot Manager hidden 12-bit FAT partition
12h    Compaq Diagnostics partition
14h    (resulted from using Novell DOS 7.0 FDISK to delete Linux Native)
16h    OS/2 Boot Manager hidden 16-bit FAT partition
17h    OS/2 Boot Manager hidden HPFS partition
18h    AST special Windows swap file
24h    NEC MS-DOS 3.x
40h    VENIX 80286
42h    SFS (Secure File System) by Peter Gutmann
50h    Disk Manager, read-only partition
51h    Disk Manager, read/write partition
51h    Novell???
52h    CP/M
52h    Microport System V/386
56h    GoldenBow VFeature
61h    SpeedStor
63h    Unix SysV/386, 386/ix
63h    Mach, MtXinu BSD 4.3 on Mach
63h    GNU HURD
64h    Novell NetWare
65h    Novell NetWare (3.11)
70h    DiskSecure Multi-Boot
75h    PC/IX
80h    Minix v1.1 - 1.4a
81h    Minix v1.4b+
81h    Linux
81h    Mitac Advanced Disk Manager
82h    Linux Swap partition
83h    Linux native file system (ext2fs/xiafs)
84h    OS/2-renumbered type 04h partition
93h    Amoeba file system
94h    Amoeba bad block table
A5h    FreeBSD
B7h    BSDI file system (secondarily swap)
B8h    BSDI swap partition (secondarily file system)
C1h    DR-DOS 6.0 LOGIN.EXE-secured 12-bit FAT partition
C4h    DR-DOS 6.0 LOGIN.EXE-secured 16-bit FAT partition
C6h    DR-DOS 6.0 LOGIN.EXE-secured Huge partition
C7h    Cyrnix Boot
DBh    CP/M, Concurrent CP/M, Concurrent DOS
```

```
DBh     CTOS (Convergent Technologies OS)
E1h     SpeedStor 12-bit FAT extended partition
E4h     SpeedStor 16-bit FAT extended partition
F2h     DOS 3.3+ secondary
F4h     SpeedStor
FEh     LANstep
FFh     Xenix bad block table
```

SeeAlso: #0482

Tabelle: Betriebssystem Indikator der Partitionstabelle

14.7 Ungarische Namenskonvention

Hungarian Naming nach Charles Simonyi. Referenz: Peter Norton, Windows 3 - Programmiertechniken für Profis, S. 56, Markt & Technik, 1992.

Präfix	Englische Bezeichnung	Datentyp
a	array	zusammengesetztes Feld
b	boolean	Flagge (TRUE/FALSE)°
c	class	Objekt (Klasse) in C++°
cb	count of bytes	Byteanzahl
ch	character	Zeichen
d	double	Gleitpunktzahl, doppelte Genauigkeit°
dw	unsigned long	Vorzeichenlose 'long' Zahl
e	enum	Aufzählungstyp°
f	float	Gleitpunktzahl, einfache Genauigkeit°
h	handle	(Datei)handle
hdc	handle device context	Handle auf einen Gerätekontext
hwnd	handle window	Handle auf ein Fenster
i	index	zusammengesetzter Index
l	long int	
lp	long pointer	long Zeiger (zusammengesetzter Typ)
n	integer	
np	near pointer	near-Zeiger (zusammengesetzter Typ)
p	pointer	Zeiger (vgl. mit np und fp)°
pt	point	Punkt (aus windows.h: POINT)
r	rectangle	Rechteck (aus windows.h: RECT)
s	struct	Struktur (Record)°
sz	ascii zero	C-Zeichenkette (ASCIIZ string)
ul	unsigned long	vergleiche mit dw°
v	void	untypisierter Datentyp°
w	unsigned int	Wort

Tab.: Ungarische Namenskonvention

14.8 Hardware Interrupts

Der 8259 Baustein ist ein programmierbarer Interruptkontroller. Weil mehr als eine Interruptanforderung zum gleichem Zeitpunkt auftreten kann, besitzt der Chip Prioritätsmanagment. Der 8259 Baustein kann acht Interrupts bearbeiten, die Interrupts 8-15 werden von einem zweiten, kaskadierten 8259 bearbeitet.

Folgende Hardwareinterrupts stehen beim PC zur Verfügung, siehe auch [Jour86], S. 19.

IRQ	owner
0	timer
1	keyboard
2	I/O channel
8	real-time clock
9	software redirected to IRQ2
10	
11	
12	
13	maths coprocessor
14	fixed disk controller
15	
3	COM2
4	COM1
5	fixed disk, LPT2
6	diskette controller
7	LPT1

Tabelle: Hardwareinterruptes des AT sortiert nach Rangfolge.

Der 8259 besitzt drei Register (IRR, IMR & ISR[44]). Interessant für die Tunneltechniken ist das IMR-Register, auf das mit dem Port 21h zugegriffen werden kann. Dieser Port repräsentiert die IRQ´s 0-7 und können mit einer „1“ zugelassen und mit einer „0“ gesperrt werden, siehe [Jour86], S. 19 ff und Kommentare in TUNNEL.ASM und MBRKILL.ASM.

14.9 Liste der am meisten verbreiteten PC-Viren

Anmerkung: Die folgenden Tabellen wurden den Signaturenupdates des Virensuchprogrammes Norton Antivirus (NAV) entnommen. Diese Signaturenupdates sind auf den meisten ftp-Server, die Antivirenprogramme anbieten, erhältlich.

OCT94.TXT - What's New in NAV 3.0's Definition Update
AntiVirus Lab, SYMANTEC/Peter Norton Product Group
October 1, 1994
****

[44] ISR = Interrupt Status Register, IMR = Interrupt Mask Register, IRR = Interrupt Request Register

Ten most commonly reported viruses (worldwide):

Form	Boot infector
Stoned.Std	Boot infector
Monkey.B	Boot infector
V-Sign	Boot infector
AntiEXE	Boot infector
Michelangelo	Boot infector
Stealth.B	Boot infector
NoINT	Boot infector
Parity Boot.B	Boot infector
AntiCMOS	Boot infector

DEC94.TXT - What's New in NAV 3.0's Definition Update
December 1, 1994
****

Virus Alerts:

Ten most commonly reported viruses (worldwide):

Form	Boot infector
Stoned.Std	Boot infector
Monkey.B	Boot infector
V-Sign	Boot infector
AntiEXE	Boot infector
Michelangelo	Boot infector
Stealth.B	Boot infector
NoINT	Boot infector
Parity Boot.B	Boot infector
AntiCMOS	Boot infector

JAN95.TXT - What's New in NAV 3.0's Definition Update
January 1, 1995
****

Virus Alerts:

Monkey.B	Boot infector
Form	Boot infector
Stealth.B	Boot infector
AntiEXE	Boot infector
NYB	Boot infector
Michelangelo	Boot infector
Stoned.Std	Boot infector

```
Natas                File infector
V-Sign               Boot infector
NoINT                Boot infector
```

FEB95.TXT - What's New in NAV 3.0's Definition Update
AntiVirus Lab, SYMANTEC/Peter Norton Product Group
February 1, 1995
****

Ten most commonly reported viruses (worldwide):

```
Monkey.B             Boot infector
Form                 Boot infector
Stealth.B            Boot infector
AntiEXE              Boot infector
NYB                  Boot infector
Michelangelo         Boot infector
Stoned.Std           Boot infector
Natas                File infector
V-Sign               Boot infector
NoINT                Boot infector
```

MAR95.TXT - What's New in NAV 3.0's Definition Update
AntiVirus Lab, SYMANTEC/Peter Norton Product Group
March 1, 1995
****

Ten most commonly reported viruses (worldwide):

```
AntiEXE              Boot infector
Form                 Boot infector
Michelangelo         Boot infector
Monkey.B             Boot infector
Natas                File infector
NoINT                Boot infector
NYB                  Boot infector
Stealth.B            Boot infector
Stoned.Std           Boot infector
V-Sign               Boot infector
```

14.10 Der Adressraum von INTEL PCs

Abb.: Adressraum von INTEL PCs