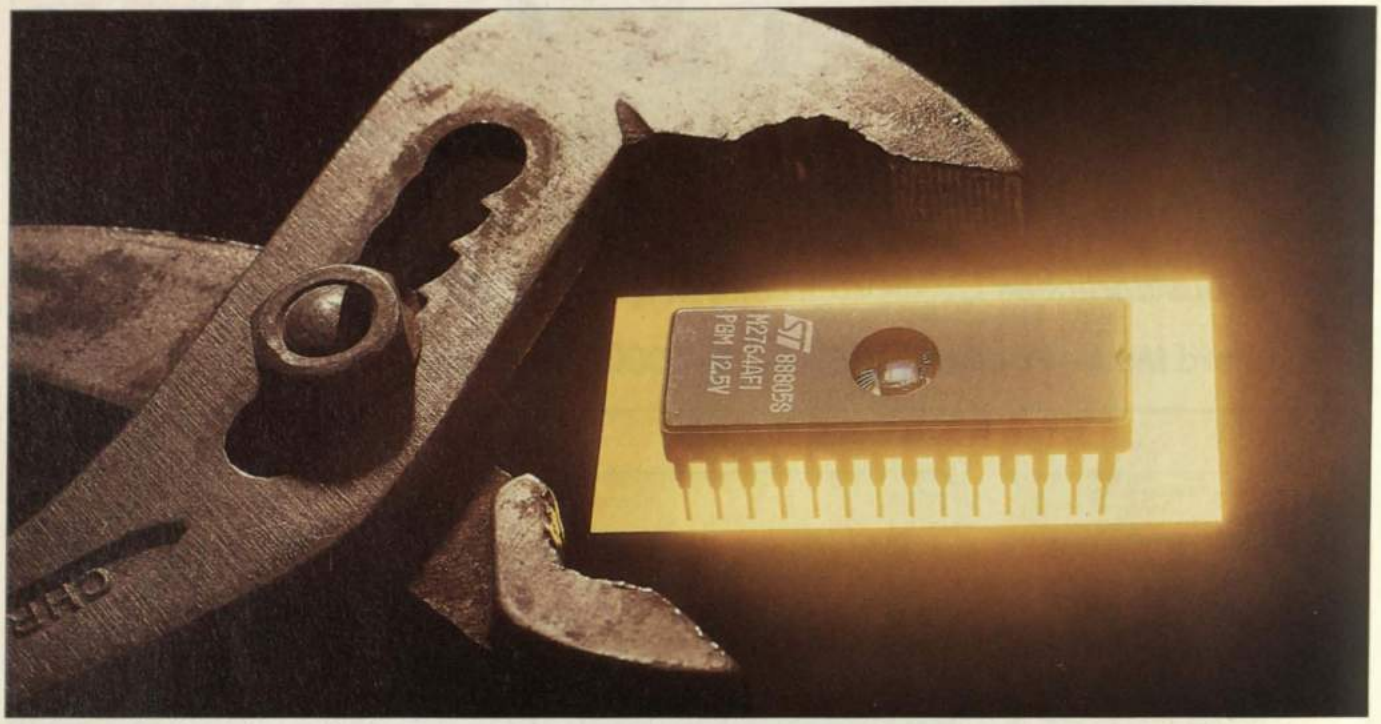

Keine Chance ohne Paßwort

Knacksicherer Paßwortschutz im V20-BIOS-ROM

Konrad Schwarz

Ist Ihr PC mehreren Personen zugänglich, die alle auch noch recht viel von Computern verstehen? Neigen diese womöglich zu 'Kollegenscherzen' via AUTOEXEC.BAT? Oder wollen Sie Ihren PC samt Daten einfach nur möglichst perfekt gegen den Zugriff Unbefugter schützen? Wenn Sie das V20-BIOS benutzen und über einen EPROM-Programmierer verfügen, dann können Sie einen Paßwortschutz direkt in das BIOS-EPROM setzen.

Den besten Schutz vor unerlaubter Zugangsberechtigung für den Rechner bietet fraglos ein Schloß, das – wie in ATs – auch gleich noch das Öffnen des Gehäuses blockiert. Abgesehen davon, daß die Schlüssel meistens verschwunden sind, wenn man sie das erste Mal wirklich braucht, sind PCs nur selten mit Schlössern ausgestattet.

Der zwar nur zweitbeste, jedoch viel bequemere Schutz besteht dann in einer ROM-residenten Paßwortabfrage, die jegliches Booten verhindert, solange das Paßwort nicht korrekt eingegeben wurde. Selbst versierte Hacker werden hier in der Regel die Waffen strecken, denn wer trägt schon ein Ersatz-BIOS-EPROM in der Jackentasche?

Und durch ein wenig Verschleierungstaktik läßt sich die Sicherheit weiter steigern. So sollte der Schutzmechanismus nicht jeden Computerfreak durch eine Meldung wie 'Bitte geben Sie das Paßwort ein:' mit der Nase darauf stoßen, daß hier ein Geheimnis zu lüften sei, sondern besser ein 'dezentes Fehlverhalten' zur Schau stellen. Auch darf später kein Paßwort bei der Eingabe auf dem Bildschirm erscheinen. Und wenn Sie den 'Schlüssel' mal vergessen haben? Dann bleibt immer noch das alte EPROM, aber auch eine vorübergehende Desaktivierung des Schutzmechanismus kann man vorsehen.

Bedingungen

Verschiedene Verfahren für einen solchen Schutz per Paßwortabfrage haben wir bereits in c't vorgestellt [1, 2], allerdings bei weitem nicht so sicher vor Neugierigen (im EPROM) verborgen wie bei der hier realisierten Lösung. Um diesen Schutz aber überhaupt im EPROM des PC-BIOS unterbringen zu können,

- muß man Platz für zusätzlichen Code im EPROM haben,
- benötigt man eine genaue Dokumentation des BIOS (am besten den Quelltext), um eben diesen freien Platz auch exakt bestimmen zu können,
- und nicht zuletzt den Zugang zu einem EPROM-Programmiergerät.

Aus den beiden erstgenannten Gründen bietet sich für c't-Leser vor allem eine Änderung des V20-BIOS an [3]. Dieses ist für PCs gedacht, die bereits mit einem V20-Chip von NEC ausgestattet sind. Dieses BIOS ist optimal auf spezielle Befehle des V20-Chip ausgelegt worden (die der 8088 nicht kennt), wodurch zum einen der Rechner schneller wird. Und zum andern, obwohl bereits einige Erweiterungen gegenüber dem ROM-BIOS des IBM PC vorgenommen wurden (etwa ein integrierter Tastaturtreiber), bleibt immer noch 'Raum für Notizen', also freier Speicher.

Inwieweit man einen solchen Paßwortschutz auch in BIOS-Versionen diverser Clone-Hersteller unterbringen kann, ist uns nicht bekannt. So einfach, wie es hier vorgeführt und mit Hilfsprogrammen unterstützt wird, ist es ganz sicher nicht. An Implementierungen für andere BIOS-Versionen sollte sich nur jemand heranwagen, der wirklich fit im Umgang mit Assembler und Debugger ist. Und es sei auch gleich vor-

ausgeschickt: Einen Paßwort-Brenn-Service seitens c't wird es nicht geben.

Zwei Wege

Zum grundsätzlichen Verständnis des Schutzmechanismus wird der Sachverhalt anhand des V20-BIOS-Quelltextes dargestellt. Selbstverständlich läßt sich auf diesem Weg – also durch Änderung des Quelltextes und neues Übersetzen – auch die Implementierung vornehmen. Wem das aber zu umständlich ist, der kann sich auch mit den hier vorgestellten Hilfsprogrammen in Turbo-Pascal 4.0 behelfen, zumal Sie dann auch ohne den Quellcode des BIOS auskommen. Dazu muß man nichts weiter tun, als – etwa mit DEBUG – den Speicherbereich von F000:E000 bis F000:FFFF in eine Datei (8 KByte bzw. 2000h lang) zu schreiben, um diese anschließend mit einem der Pascal-Patch-Programme zu bearbeiten und dann in ein neues EPROM (2764) zu brennen.

Das heißt DEBUG starten und den ROM-Inhalt mit dem M(ove)-Befehl an die aktuell von DEBUG verwaltete Speicheradresse umladen:

```
mF000:E000,FFFF,DS:100
```

Jetzt muß der anzulegenden Datei mit dem N(ame)-Kommando ein beliebiger File-Name verabreicht und die Länge des abzuspeichernden Bereichs im Register CX hexadezimal angegeben werden:

```
nBIOS
rcx
CX 0000
2000
w
```

Mit dem W(rite)-Befehl wird die Datei nun auf Platte geschrieben. 'CX 0000' ist eine Meldung des Debuggers.

Gewußt wo und wie

Der Quelltext des V20-BIOS ist nicht gerade das, was man im Informatikunterricht als 'gut strukturiert' vorzeigen würde. Der Anspruch, möglichst effizienten und schnellen Code zu erzeugen und dabei dennoch die wichtigen Einsprungadressen kompatibel zu IBMs BIOS zu halten, war aber anders nicht zu erfüllen.

So läßt sich nachträglich nicht einfach an eine freie Stelle neuer Code eintragen. Das BIOS-File darf nicht ein Byte länger werden, und auch einige Datenbereiche (etwa das Maschinenkennbyte an vorletzter Stelle im ROM) müssen weiterhin genau auf den absoluten Adressen bleiben, an denen sie im Moment liegen. Auch sei daran erinnert, daß einige Routinen im BIOS bestimmte Registerinhalte aus anderen Programmteilen erwarten, die man nicht ohne weiteres verändern darf. Damit besteht die einfachste Methode für nachträgliche Erweiterungen auch bei Quelltextänderungen in einer Art Patching.

Sehr vereinfacht wird unser Anliegen, eine Paßwortroutine nachzurüsten, durch die Tatsache, daß im V20-BIOS (im folgenden wird alles auf die aktuellste Version 3.72 bezogen) ab Adresse F730h noch 105 Bytes 'am Stück' frei sind.

Nun gilt es nur noch, eine geeignete Stelle zu finden, von der aus man in die Paßwortroutine verzweigen kann: Das BIOS führt einen Selbsttest durch, initialisiert so manches, sucht BIOS-Erweiterungen und bootet dann durch den Aufruf des Interrupt 19h. Dieser Boot-Aufruf ist ideal zum Verbiegen geeignet. Erstens ist an dieser Stelle der ganze Rechner initialisiert, zweitens benötigt die aufgerufene Boot-Routine keinerlei Registerinhalte, die man unverändert erhalten müßte, und drittens löscht das BIOS vor dem Boot-Aufruf den Bildschirm.

Wenn bei Erreichen dieses Stadiums jetzt nichts passiert, also der Rechner nicht bootet, so ist das für den Eingeweihten eine deutliche Aufforderung zur Eingabe des Paßwortes – aber eben nur für den Eingeweihten.

So wird gepatcht

Dieser Boot-Aufruf benötigt zwei Bytes, nämlich CDh 19h an der Stelle E636h im BIOS. Diese Adresse und die in den Listings sind absolute Offset-Angaben, die sich auf den Adreßbereich im BIOS beziehen, wenn es im PC eingesetzt ist (Segment F000h). Wenn Sie die lauffähige BIOS-Datei (also das EPROM-Abbild) mit DEBUG laden, so legt dieser es bei der Startadresse 100h ab. Das BIOS beginnt im PC beim Adreß-Offset E000h und ist 2000h Byte lang. Nach dem Laden mit DEBUG korrespondiert die Adresse E000h mit 100h, E636h folglich mit 736h. Ein Aufruf mit dem U(nassemble)-Befehl des DEBUG, also

```
u736
```

sollte als erste Zeile den Befehl

```
INT 19
```

zutage fördern, sonst sind Sie an der falschen Adresse (vielleicht hat das Sichern der Datei nicht geklappt, oder Sie haben eine andere Version des V20-BIOS).

Da der Weg zum frei nutzbaren EPROM-Bereich für einen 'jump short' (das wäre ein 2-Byte-Befehl, Sprungweite 128 Bytes) zu weit ist, muß man einen langen Sprungbefehl (drei Bytes Befehlslänge) verwenden. Es bietet sich an, den Befehl vor dem INT 19h mitzubenutzen, also

```
OUT NMI_CTRL,AL
```

bei E634h. Da man keinesfalls nur das zweite Byte dieses Befehls 'abzwacken' darf, trägt man bei E634h zunächst den Befehl

```
JMP KS_UMLEITUNG
```

```
BIOS-Erweiterung, DIP-Schalter-Version

1. BIOS-Ausschnitt mit dem umgeleiteten Boot-Aufruf

    E62A  0C 30                         OR   AL,30H
    E62C  E6 61                         OUT  SYS_CTRL_PORT,AL
    E62E  24 CF                         AND  AL,0CFH
    E630  E6 61                         OUT  SYS_CTRL_PORT,AL
    E632  B0 80                         MOV  AL,80H    ;ENABLE NMI
==) E634  E9 F730 R                     jmp ks_umleitung    ; hier ist die Umleitung,
==) E637  90                            nop                 ; damit die restliche
                                                            ; Adreßlage stimmt
                                        ;-----------------------------------------
    E638                                INIT_RAM:
    E638  FC                            CLD  ;THIS ROUTINE NO V20 OPS !
    E639  2B FF                         SUB  DI,DI
     ;=======================================================

2. Ab hier BIOS-Ausschnitt vor Paßwort-Routine

    F725  5F                            POP  DI   ;NEARLY ALL REGISTERS USED
    F726  5E                            POP  SI
    F727  5D                            POP  BP
    F728  1F                            POP  DS
    F729  1F                            POP  DS
    F72A  1F                            POP  DS
    F72B  1F                            POP  DS
    F72C  1F                            POP  DS
    F72D  1F                            POP  DS
    F72E  07                            POP  ES
    F72F  CF                            IRET
    F730  00 69[       FF           ]   DB   105 DUP (0FFH) ; Patch-Teil mit FF vorbesetzen
                                                            ; Start Patch-Teil
==) F730                                org  0f730h         ; bündig zu IRET oben !!

==) F730                        ks_umleitung2:
==) F730  E6 A0                         OUT  NMI_CTRL,AL    ; das stand an der alten Stelle

==) F732  E4 61                         in   al,061H        ; Inhalt von DIP-Port Steuerung
==) F734  50                            push ax             ; sichern
==) F735  24 F7                         and  al,0FFH-08H    ; unteres Nibble vom DIP-Schalter
==) F737  E6 61                         out  061H,al        ; ansteuern
```

gefolgt von einem NOP (1 Byte) ein, um die Adreßlage des nachfolgenden Programmcodes nicht zu beeinflussen. Klar, daß man als erstes im neuen Programmteil den geopferten Befehl nachholt, sonst gibt's Probleme (siehe Listing).

Auch abschaltbar

Wenn man so richtig in der schönsten Programmentwicklung steckt – sprich: der Rechner stürzt alle paar Minuten ab –, kann einem das dauernde Paßworteingeben schon mal auf die Nerven gehen. Auch kann man die genaue Schreibweise des Paßwortes nach längerer Rechnerabstinenz ja mal vergessen haben.

Für diesen Fall habe ich nach einer vorübergehenden Abschaltmöglichkeit des Paßwortschutzes gesucht und zwei Methoden gefunden. Jede Methode erfordert allerdings eine andere Paßwort-Routine im BIOS, so daß hier deshalb zwei Versionen vorgestellt werden. Es mag Rechner geben, bei denen beide nicht praktikabel sind, sie sind aber die Ausnahme, und deren Besitzer müssen sich bei Bedarf nach Alternativen umsehen.

Fast alle sklavisch abgekupferten PCs (hier sind die No-name-Produkte durchaus von Vorteil) haben auch den DIP-Schalter für den POST (Power On Self Test) des IBM-BIOS. Da dieser Test nur für den IBM-Service tiefere Bedeutung hat, wurde er im V20-BIOS der Version 3.72 völlig weggelassen. Damit ist dieser Schalter (Position 1 des Blockes für die Konfiguration) bei Besitzern des V20-BIOS frei verfügbar.

Die DIP-Schalter-Version der Schutz-Routine testet zuerst den Switch 1. Dazu muß das untere Nibble der Schalter angewählt werden. Ist dieser in der von IBM geforderten Normalstellung (um beim Hacker kein Mißtrauen zu erwecken), wird das Paßwort gefordert. Andernfalls wird ohne Abfrage gebootet.

Die zweite Möglichkeit ist viel komfortabler, aber sie steht nur denen offen, die eine Hardware-Uhr mit einem IC vom Typ UM 82C8167 oder eine kompatible Variante – etwa auf einer Multi-I/O-Karte – im Rechner haben. Dieses IC hat auch eine Weckfunktion, die aber der PC nicht ausnutzt. Also fristen die Register für die Weckzeitspeicherung batteriegepuffert ihr Dasein: wertvolle Bits, die auch beim Ausschalten des Rechners ihre Information behalten.

Ich benutze das Register an der Portadresse 24Ch. Wenn Sie nicht sicher sind, ob Ihre Uhr dort ein Register aufweist, können Sie das sehr einfach und gefahrlos ausprobieren. Schauen Sie zunächst mit DEBUGs I(n)-Befehl nach, welcher Wert an dieser Stelle lesbar ist:

I 24C

Diese Version der Paßwortschutz-Routine läßt sich vorübergehend über den Schalter 1 des Schalterblockes für die Konfiguration desaktivieren.

```
==> F739  E4 62                        in   al,062H        ; und Inhalt lesen
==> F73B  D0 D8                        rcr  al,1           ; um 1 shiften in Carry
==> F73D  58                           pop  ax             ; alten Zustand am Port 061H
==> F73E  E6 61                        out  061H,al        ; wieder herstellen
==> F740  73 1D                        jnc  ks_norm        ; KEIN Paßwort-Test

==> F742  1E                           push ds             ; Paßwort testen
==> F743  B8 F000                      mov  ax,0f000h
==> F746  8E D8                        mov  ds,ax          ; neues DS
==> F748  B5 00                        mov  ch,0
==> F74A  8A 0E F763 R                 mov  cl,[ks_len]    ; Paßwortlänge !!
==> F74E  BB F764 R                    mov  bx , offset ks_pass  ; Paßwortadresse
==> F751                    ks_next_char:
==> F751  B4 00                        mov  ah,0h
==> F753  CD 16                        int  16h            ; Taste vom Keyboard
==> F755  34 7F                        xor  al,7fh         ; "verschlüsseln"
==> F757  32 07                        xor  al,[bx]        ; Mit Paßwort vergleichen
==> F759  75 06                        jnz  ks_error       ; Sprung bei falscher Taste
==> F75B  43                           inc  bx             ; nächstes Zeichen
==> F75C  E2 F3                        loop ks_next_char
==> F75E  1F                           pop  ds
==> F75F  CD 19             ks_norm:  INT 19H             ; Boot, wenn alles normal

==> F761  FA                ks_error:  cli                 ; Interrupts abschalten
==> F762  F4                           hlt                 ; System stoppen
==> F763  05                ks_len     db   05d            ; Länge des Paßwortes
==> F764  19                ks_pass    db   'f' xor 7Fh    ; einzelne Buchstaben des Paßwortes
==> F765  1E                           db   'a' xor 7fH    ; jeweils mit 07FH "verschlüsselt
==> F766  11                           db   'n' xor 7fh
==> F767  11                           db   'n' xor 7fh
==> F768  06                           db   'y' xor 7fh

==> F799                               org  0f799h         ; ab hier weiter wie im Orginal

    F799  53 79 73 74 65 6D 64  NO_SYSTEM DB'Systemdisk in Drive A einlegen,Taste drücken',1f
          69 73 6B 20 69 6E 20
          etc....

Die Zeilen mit vorangestelltem "==>" enthalten Änderungen gegenüber dem Orginal.
```

```
BIOS-Version für Paßwortsteuerung via Uhren-IC

1. BIOS-Ausschnitt mit dem umgeleiteten Boot-Aufruf

    E62C  E6 61                        OUT SYS_CTRL_PORT,AL
    E62E  24 CF                        AND AL,0CFH
    E630  E6 61                        OUT SYS_CTRL_PORT,AL
    E632  B0 80                        MOV AL,80H     ;ENABLE NMI
==> E634  E9 F730 R                    jmp ks_umleitung2
==> E637  90                           nop
                                       ;-------------------------------------------------------
    E638                               INIT_RAM:
    E638  FC                           CLD ;THIS ROUTINE NO V20 OPS !
    E639  2B FF                        SUB DI,DI
    E63B  2B C0                        SUB AX,AX
    ;=====================================================

2. BIOS-Ausschnitt mit der Paßwortroutine
    F725  5F                           POP DI   ;NEARLY ALL REGISTERS USED
    F726  5E                           POP SI
    F727  5D                           POP BP
    F728  1F                           POP DS
    F729  1F                           POP DS
    F72A  1F                           POP DS
    F72B  1F                           POP DS
    F72C  1F                           POP DS
    F72D  1F                           POP DS
    F72E  07                           POP ES
    F72F  CF                           IRET
                                       ;*****************************************************
```

Üblich ist 00. Nun schreiben Sie mit dem O(ut)-Befehl zum Beispiel 55h in das Register und lesen anschließend zur Kontrolle, ob der Wert übernommen wurde:

O 24C 55
I 24C

Wenn jetzt wieder 55 angezeigt wurde, stehen die Chancen gut. Nun den Rechner aus- und nach einigen Sekunden wieder einschalten und mit

I 24C

nachsehen, ob immer noch 55 im Register steht (es könnte ja unter Umständen ein ungepufferter Port mit anderer Funktion gewesen sein), und vor allem, ob die Anzeige der Uhr noch korrekt ist. Wenn alles stimmt, können Sie die Paßwortschutz-Abschaltung via Uhr nutzen.

Die 'Uhrenversion' ist dabei so gestaltet, daß der Rechner das Paßwort auch verlangt, wenn das Uhren-RAM den Inhalt durch eine leere Batterie verloren hat, weil dann in der Regel das unterste Bit im Register auf 0 steht. Setzt man es zum Beispiel mittels DEBUG

O 24C 01

auf 1, kann ohne Abfrage gebootet werden.

```
    F730  0069[        FF          ]  DB  105 DUP (0FFH)       ; Patch-Teil mit FF vorbesetzen
                                                                ; Start Patch-Teil
==> F730                               org 0f730h               ; bündig zu IRET oben !!

==> F730                          ks_umleitung2:
==> F730  E6 A0                        OUT NMI_CTRL,AL     ; das stand an der alten Stelle

==> F732  BA 024C                      mov dx,024cH        ; Uhr-IC Register-Adresse
==> F735  EC                           in  al,dx
==> F736  D0 D8                        rcr al,1            ; um 1 shiften in Carry
==> F738  72 1D                        jc  ks_norm         ; KEIN TEST wenn Carry

==> F73A  1E                           push ds
==> F73B  B8 F000                      mov ax,0f000h
==> F73E  8E D8                        mov ds,ax                ; neues DS
==> F740  B5 00                        mov ch,0
==> F742  8A 0E F75B R                 mov cl,[ks_len]          ; Paßwortlänge !!
==> F746  BB F75C R                    mov bx,offset ks_pass    ; Paßwort
==> F749                          ks_next_char:
==> F749  B4 00                        mov ah,0h
==> F74B  CD 16                        int 16h                  ; Taste von Keyboard
==> F74D  34 7F                        xor al,07Fh              ; Taste "verschlüsseln"
==> F74F  32 07                        xor al,[bx]              ; vergleichen
==> F751  75 06                        jnz ks_error             ; nicht gleich ==> Sprung
==> F753  43                           inc bx                   ; nächster Buchstabe
==> F754  E2 F3                        loop ks_next_char
==> F756  1F                           pop ds
==> F757  CD 19                   ks_norm:  INT 19H             ; normaler Boot-Aufruf
==> F759  FA                      ks_error: cli                 ; falsches Paßwort:
                                                                ; Interrupts abschalten
==> F75A  F4                           hlt                      ; System anhalten

==> F75B  05                      ks_len    db  05d            ; Länge des Paßwortes
==> F75C  19                      ks_pass   db  'f' xor 7fh    ; einzelne Paßwortbuchstaben
==> F75D  1E                           db  'a' xor 7fh         ; jeweils mit 07F "verschlüsselt"
==> F75E  11                           db  'n' xor 7fh
==> F75F  11                           db  'n' xor 7fh
==> F760  06                           db  'y' xor 7fh

==> F799                               org 0f799h              ; ab hier geht's normal weiter
    F799  53 79 73 74 65 6D 64         NO_SYSTEM DB  'Systemdisk in Drive A etc...',1f
          69 73 6B 20 69 6E 20
          etc...

Die Zeilen mit vorangestelltem "==>" enthalten Änderungen gegenüber dem Orginal.
```

Schutzmechanismus

Die eigentliche Schutz-Routine unterscheidet sich nur geringfügig von einer bereits vorgestellten Version in c't 3/88 [2]. Das Paßwort wird mittels Assembler 'verXORt' (exklusive ODER-Funktion), damit man es nicht gleich bei einem Dump durchs BIOS-EPROM in Klartext erkennen kann. Daher muß in dem Programmteil, der das Paßwort überprüft, diese XOR-Verknüpfung jeweils auch mit dem eingegebenen Zeichen vorgenommen werden.

Wenn das Paßwort nicht stimmt, werden die Interrupts abgeschaltet, und die CPU wird angehalten. Ein Wiedererwecken ist dann nur per Hardware-Reset möglich. Anders läßt sich schwerlich einigermaßen glaubhaft ein defekter Rechner imitieren. Wenn man keinen Rechner mit Reset-Knopf hat, ist es allerdings unerquicklich, nach jedem Tippfehler den Rechner aus- und einschalten zu müssen. Man könnte dann die Routine so ändern, daß man die CPU nur in eine Endlosschleife schickt. Dann ist der Rechner mit einem Ctrl-Alt-Del wieder zu einem Neustart zu bewegen.

In der hier vorgestellten Lösung beginnt die Paßwort-Routine bei Adresse (Offset) F730h, das Paßwort steht ab Adresse F75Ch in der Uhrenversion und ab F764h in der DIP-Schalter-Variante. Vor dem eigentlichen Paßwort wird die Länge des Paßwortes vermerkt. Das Programm kann um einiges verlängert und variiert werden, da der Bereich bis F799h noch frei ist.

In den Beispielen ist das Paßwort 'fanny' gewählt. Zu beachten ist, daß die Paßwort-Rou-

Wer eine Hardware-Uhr in seinem PC hat, kann in einem der Register für die Alarmzeit ein Flag setzen, um den Paßwortschutz vorübergehend abzuschalten.

```
program PASSWORD_IN_DAS_BIOS_PATCHEN;

USES CRT,DOS;       { TURBO 4.0 }


type
   HEX_STRING = string[4];

const
   sprung : array[0..3] of byte = ($E9,$f9,$10,$90);
   REC_LEN = $80;

VAR
   FILE1 : file;
   BLOCK_BUF : array [0..REC_LEN] of byte;
   SOURCE : string[80];
   OBJECT : STRING[80];
   BUFFER : array [0..$1FFF] of byte;
   ADR,i : word;
   password : string[20];

{$I bios.inc}       { Die Datei BIOS.INC muß den versionsabhängigen Teil }
                    { ( Uhr-RAM oder DIP-Switch ) enthalten }
```

tine sehr wohl zwischen Groß- und Kleinschrift unterscheidet. Sollten Sie außerdem nicht den V20-BIOS-internen Keyboard-Treiber benutzen, sondern einen zuladbaren (etwa, weil Sie eine Tastatur mit ASCII-Layout benutzen), so denken Sie daran, daß vor dem Booten die V20-BIOS-eigene Belegung gilt.

Pascal-Patcher

Da es nicht gerade zu den einfachen Dingen gehört, ROM-Code mittels MASM, LINK und DEBUG (EXE2BIN scheidet wegen Unfähigkeit leider aus, siehe [3]) zu generieren, sei hier eine bequemere Lösung angeboten, die die ganze Patcherei mit einem Pascal-Programm ermöglicht. Die ist allerdings darauf abgestimmt, daß Sie mit den hier abgedruckten Assembler-Versionen einverstanden sind, da deren Maschinencodes bereits als Daten in einer Include-Datei stecken (sie lassen sich natürlich ebenfalls ändern).

Es ist jeweils eine Include-Datei für die versionsabhängigen Teile der Uhr- und DIP-Schalter-Version vorhanden. Jede Datei ist dabei mit einer eigenen Prüfsumme zum Auffinden von Tippfehlern versehen. Vor dem Kompilieren des Hauptprogrammes mit Turbo-Pascal 4.0 muß die jeweils gewünschte Include-Datei in BIOS.INC umbenannt werden.

Das Pascal-Programm ändert das BIOS-File, fragt Sie nach dem Paßwort und trägt dieses 'geXORt' nebst seiner Länge (hier maximal 20 Zeichen) ein und korrigiert zu guter Letzt auch die EPROM-Prüfsumme. Die bearbeitete Datei muß dann 'nur' noch in ein EPROM geschossen werden.

Ausblick

Dem Vorteil des nahezu perfekten Schutzes steht natürlich auch der Nachteil gegenüber, daß eine Änderung des Paßwortes ein Neubrennen des EPROMs erfordert. Aber nur so ist der Schutz unabhängig vom jeweiligen Betriebssystem. Auch 'kennt' der Rechner vor dem Booten noch kein DOS. Ein Paßwort kann also nicht in einer Datei untergebracht werden, sondern nur in einem absoluten Sektor. Der Boot-Sektor scheidet aber zum Beispiel dann aus, wenn man von allen DOS-Versionen unabhängig sein will.

```
procedure SPEICHERN;
var J : word;
begin
    assign (FILE1, OBJECT);
    {$I-} REWRITE (FILE1); {$I+}
    if ioresult = 0 then begin
       j := 0;
       repeat
           for I := 0 to rec_len do
              block_buf[i] := buffer[i+j];
           blockwrite (FILE1, BLOCK_BUF, 1);
           J := j + rec_len ;
       until j >= $1FFF;
       close (FILE1);
    end
    else begin
      writeln('Schreiben ist nicht möglich !!');
      halt(2);
    end;
end;


PROCEDURE pruefsumme;
VAR I:word;
   SUMME,CHECK : byte;

BEGIN;
   SUMME := 0;
   for I := 0 to $1FFE do
       SUMME := SUMME + BUFFER[I];
   CHECK:=$FF-(SUMME-1);
   BUFFER[$1FFF]:=CHECK;
   writeln('Prüfsumme : ',check,' (dezimal) wurde gepatcht');
END;


procedure LADEN;
var     I, BYTE_NO : integer;
begin
    I := 0;
    BYTE_NO := 0;
    assign (FILE1, SOURCE );
    {$I-} RESET (FILE1); {$I+}
    if ioresult = 0 then begin
       while not EOF (FILE1) do begin;
          blockread (FILE1, BLOCK_BUF, 1);
          for  I := 0 to REC_LEN do begin
             BUFFER [trunc (I+BYTE_NO)] := BLOCK_BUF [I];
          end;
          BYTE_NO := BYTE_NO + REC_LEN
       end ;
    end
    ELSE begin
        writeln('Fehler beim Lesen der Datei !!');
        halt(2);
    end;
end;

begin {MAIN}
    clrscr;
    write('Welche Datei soll gelesen werden     ? ');
    readln(SOURCE);
    write('Welche Datei soll geschrieben werden ? ');
    readln(OBJECT);
    adr := 0;
    for i := 0 to 3 do adr := adr + sprung[i];
    if adr <> 642 then begin                          { sprung testen !!}
       writeln('Das array SPRUNG ist nicht richtig eingegeben!!');
       halt(2);
    end;
    adr := 0;
    for i := 0 to patch_len do adr := adr + patch[i];
    if adr <> patch_sum then begin                    { patch testen !!}
        writeln('Das array PATCH ist nicht richtig eingegeben !!');
        halt(2);
    end;
    laden;
    for I := 0 to 3 do begin
        buffer[$0634+i] := sprung[i];          { Sprung patchen }
    end;
    for I := 0 to patch_len do begin
        buffer[$1730+i] := patch[i];           { Programm patchen }
    end;
    write('Paßwort eingeben:');
    readln(password);

    buffer[laenge] := length(password);    { Paßwortlänge patchen }
```

Allerdings wurde in [2] auch eine Lösung aufgezeigt, bei der nur ein einziges Zeichen als 'Paßwort' verwendet wird. Dabei wählt man typischerweise ein Zeichen, das sich nur über eine Kombination aus Zahlen über den Ziffernblock bei niedergehaltener Alt-Taste eingeben läßt. Ein solches Zeichen ließe sich zum Beispiel gut in einem der bereits erwähnten Uhrenregister unterbringen. Es bleibt also genügend Spielraum – auch im ursprünglichen Sinne des Wortes. (gr)

```
    for i := 1 to length(password) do   { Paßwort "verschlüsselt" patchen }
        buffer[laenge+i] := ord(password[i]) xor $7F;

    pruefsumme;
    speichern;
end.
```

```
{ Diese Datei muß als BIOS.INC abgespeichert sein, wenn man das Paßwort }
{ mit dem Uhren-IC UM 82 C 8167 oder einem Ersatztypen steuern will }
const
   patch  : array[0..42] of byte = (
            $E6,$a0,$ba,$4c,$02,$ec,$d0,$d8,$72,$1d,$1e,$b8,$00,$f0,$8e,
            $d8,$b5,$00,$8a,$0e,$5b,$f7,$bb,$5c,$f7,$b4,$00,$cd,$16,$34,
            $7f,$32,$07,$75,$06,$43,$e2,$f3,$1f,$cd,$19,$fa,$f4);
   patch_sum = 5523 ;
   laenge    = $175B;    { Adresse für Paßwortlänge }
   patch_len = 42;
```

```
{ Diese Datei muß als BIOS.INC gespeichert sein, wenn die }
{ Paßwortabfrage mit dem DIP-Schalter gesteuert werden soll }
const
   patch  : array[0..50] of byte = (
            $E6,$a0,$e4,$61,$50,$24,$f7,$e6,$61,$e4,$62,$d0,$d8,
            $58,$e6,$61,$73,$1d,$1e,$b8,$00,$f0,$8e,$d8,$b5,$00,
            $8a,$0e,$63,$f7,$bb,$64,$f7,$b4,$00,$cd,$16,$34,$7f,$32,$07,
            $75,$06,$43,$e2,$f3,$1f,$cd,$19,$fa,$f4);
   patch_sum = 6796 ;
   laenge    = $1763;    { Adresse für Paßwortlänge }
   patch_len = 50;
```

Dieses Programm in Turbo-Pascal 4.0 nimmt Ihnen die ganze Arbeit der BIOS-Änderung ab. Sie brauchen nur die gewünschte Include-Datei bei der Kompilierung auszuwählen, und der Patcher baut Ihnen aus der alten EPROM-Datei eine neue mit Paßwortabfrage.

Literatur

[1] W. Schrader, D. Grell, Paßwort-Knobeleien, 'Kindersicherung' für Papas PC, c't 8/87, S. 98 ff.

[2] Ch. Brocks, T. Stein, L. Stohlmann, Paßwort-Knobeleien, Teil 2: Neue Varianten eines Paßwortschutzes beim Booten von PCs, c't 3/88, S. 204 ff.

[3] P. Köhlmann, V-Chip-Power, V20-ROM-BIOS für PCs und Kompatible, c't 10/87, S. 208 ff.