Wissenswertes über Dongles

Johann Walzer

Dongles sind kleine Hardware-Module, welche an die parallele oder serielle Schnittstelle von PCs angeschlossen werden. Zumeist werden sie über diese Schnittstelle auch gleich versorgt. Alle Logik-Pegel, welche auf High liegen, können für diesen Zweck eingesetzt werden. Das ist zwar kein solides Design, aber es funktioniert zumindest.

Die Hardware dieser Kopierschutz-Module besteht i.A. zumindest aus einem EEPROM für diverse Parameter und meist auch aus einem kleinen µP. Diese µPs sind ähnlich den PICs, aber kundenspezifische Ausführungen.

Aus verständlichen Gründen sind nicht allzuviele Details, weder über die Schaltung der Dongles, noch die Abfrage-Software erhältlich. Die gute oder schlechte Nachricht, je nachdem von welcher Seite Sie das Problem betrachten, ist, daß grundsätzlich ALLE Schutzmechanismen zu knacken sind, es ist nur eine Frage ob sich der Aufwand dafür lohnt. Meine Begründung für diese Aussage stützt sich auf die Tatsache, solange der Mikroprozessor in meinem PC „weiß" was zu tun ist, werde auch ich als Mensch es nachvollziehen können.

Eine Schwäche besteht meiner Meinung auch darin, daß die Dongle-Hersteller ihre Hardware-Schaltung nicht offen legen wollen und deshalb für die Ansteuerung die entsprechenden Software-Treiber mitliefern müssen. Das heißt aber, die Treiber werden erst beim Link-Vorgang mit dem fertigen Programm verknüpft. Man erkennt dann aber im fertigen EXE-Code, ein eigenes Code-Segment, das vom Linker dafür vergeben wurde. Weiters sind die Treiberroutinen auch an den vielen `PUSH` und `POP` sowie einem `RET Far` Befehl zu erkennen, da sie keine Register zerstören dürfen, außer demjenigen für den Rückgabewert. Der `RET Far` Befehl ist notwendig, weil die Routinen über Segment-Grenzen hinweg aufgerufen werden. Durch Setzen eines Breakpoints auf den Beginn der Routine läßt sich dann am Stack auch feststellen, aus welchen Programmteil(en) der Aufruf erfolgte.

Weiters ist es relativ leicht möglich ein günstiges Demo-Paket vom entsprechenden Dongle-Hersteller anzufordern. Man kommt damit in den Besitz der Treiber-Routinen, die man dann nur mehr im geschützten Programm auffinden muß.

Das „Knacken" eines kopiergeschützten Programmes läßt sich aber nur dann durchführen, wenn man auch Zugang zum Dongle hat. Man könnte z.B. einen Logik-Analysator (siehe [1]) an den PC-Bus anschließen und den Programm-Ablauf mit und ohne Dongle protokollieren. Der Logik-Analysator hat den Vorteil, daß er rein passiv arbeitet, d.h. vom analysierten Programm nicht bemerkt werden kann. Durch Vergleich der beiden Protokolle läßt sich dann leicht feststellen, wo das Programm auseinanderläuft, d.h. wo die eigentliche Auswertung der Dongle-Abfrage erfolgt. Dies ist meiner Meinung nach auch der Schwachpunkt eines jeden Dongle-geschützten Programms: Wo findet die o.k. bzw. nicht o.k. Entscheidung im Programm-Ablauf statt. Das heißt, es ist gar nicht so interessant welche Verschleierungstaktiken die Dongle-Routine selbst anwendet: Verschlüsselung, Interrupts sperren (auch der Non-maskable-Interrupt läßt sich sperren), Verbiegen der Interrupt-Vektor-Tabelle, Zeitüberwachung, Abschalten der Tastatur, selbstmodifizierender Code (siehe [2]) usw. und selbst das läßt sich wieder herausfinden, man muß z.B. nur darauf triggern, wenn in den Bereich der Vektor-Tabelle geschrieben wird.

Eine mögliche Abhilfe für diese Schwäche besteht darin, die Dongle-Abfragen an verschiedenen Stellen im Programm durchzuführen, die eigentliche Auswertung aber erst in späterer Folge durchzuführen. Weiters ist es möglich, daß die Dongle-Routine nicht bloß eine ja/nein Entscheidung liefert, sondern eine algorithmische Antwort, die man z.B. für den weiteren Programm-Ablauf benötigt. So könnte z.B. ein CAD-Programm ein Argument an den Dongle liefern, und dieser antwortet dann mit dem Sinus-Wert des Arguments, der in weiterer Folge für das Zeichnen benötigt wird. In diesem Fall genügt es also nicht bloß die Dongle-Abfrage selbst zu finden, sondern man muß auch noch die Programmlogik verstehen und nachbilden. Eine andere Variante wäre z.B. Teile des Programmcodes aus dem Dongle dynamisch nachzuladen.

Weiters kann man eine Prüfsumme über den Programm-Code bilden, um so Veränderungen zu finden und daraufhin die Funktion des Programms verändern bzw. einstellen (am besten natürlich nicht unmittelbar auf die Abfrage der Prüfsumme sondern erst später im Code).

Andere Schutz-Varianten bestehen z.B. darin eine Demo-Version zu erzeugen. Bei dieser fehlen dann z.B. die Ausgabe-Routinen bzw. sind die Daten-Felder kleiner dimensioniert. Dies läßt sich sehr leicht durch Ändern einiger Programm-Konstanten und neu kompilieren durchführen. Es besteht so keine sinnvolle Möglichkeit das Programm zur Vollversion auszubauen. Man müsste das Programm komplett disassemblieren und auf den Source-Code zurückführen und man muß vor allem auch die Programmlogik verstanden haben. Ein Disassemblieren von Programmen ist aber meines Erachtens nur bis zu einer Größe von ca. 2 kByte sinnvoll durchzuführen, darüber ist es weniger Aufwand das Programm komplett neu zu schreiben bzw. zu kaufen. Weiters sind bei einer Demo-Version keine Kosten für die Kopierschutz-Hardware erforderlich.

In die gleiche Kategorie fallen auch Programme, die bis zu einem bestimmten Datum verwendet werden können. Das Auffinden der Uhrzeit-Abfrage sollte aber kein unüberwindliches Hindernis darstellen, da die IO-Ports bei allen „Kompatiblen" auf den gleichen Adressen liegen. Dies ist ja auch der erste Ansatzpunkt bei allen Hardware-Schutzsteckern, die auf der parallelen oder seriellen Schnittstelle stecken: die Port-Adressen sind eindeutig festgelegt!

Weiters besteht die Möglichkeit den Programm-Code zu verschlüsseln. Dazu ist nicht unbedingt ein Hardware-Schutz erforderlich. Nur wer den richtigen Code kennt bzw. die kopiergeschützte Original-Diskette hat, kann das Programm wieder entschlüsseln. Die Schwäche liegt aber darin, daß zumindest ein Startteil unverschlüsselt vorliegen muß. Um das zu Verschleiern kann man mehrfach rekursiv verschlüsseln, damit wird es schwieriger im Debugger Break-Points zu setzen, aber wie gesagt, nicht unmöglich.

Zum Abschluß sei noch wertfrei an eine andere Vorgangsweise erinnert: Ein Programm-Hersteller wußte, daß von seinem kopiergeschützten Programm geknackte Kopien im Umlauf sind. Wie aber konnte er an die Besitzer der Raubkopien herankommen? Er warb mit einer neueren Demo-Version des Programms. Hatte man die Demo-Version installiert, so untersuchte sie die Festplatte des PCs, ob vielleicht eine geknackte Vorgänger-Version zu finden ist. Falls das der Fall war, erschien eine unverfängliche Mitteilung, daß man zu dieser Demo-Version noch weitere Literatur anfordern konnte. Tat man das, so erhielt man statt der erhofften Literatur den Brief eines Rechtsanwaltes ...

Literatur:

[1] PC-Hardware-Debugger „Hardbreaker"
(c't 1/93 S.188-195, c't 2/93 S.12,206-214 und c't 4/93 S.228-234)

[2] „Vor die Füße" / Die Queue als Disassemblier-Sperre
(c't 6/89 S.252)

[3] „Harte Nüsse geknackt" / Kopierschutzsysteme näher betrachtet
(c't 5/90 S.78 - 85)