Brian A. Troha

Softkey for...

EDD 4

Utilico Software

■ **Requirements:**

☐ Apple][series
☐ *EDD version 4*
☐ 48K slave disk

Essential Data Duplicator has been upgraded to version 4. New features like the hi-res disk scan and examine disk drive are now included. Like the three version before a bitcopy seems hopeless. This is due to the very precise syncronization routines used when the program was recorded. Only the tracks from $ØØ to $ØD are used and are written using the 4 by 4 nibble encoding scheme. However there is one thing you should never forget: There are NO unbreakable protection schemes. This is true for at least one reason, the program must somehow load into the machine. So if you can follow the boot of the program you could (eventually) unprotect any program. Being able to "Boot code Trace" a program is a valuable skill. I say skill because, the loading of each program will change and the loader will try every trick in the book to hide what it is doing.

The theory behind boot code tracing relies on two things. First, there is a program in ROM on the disk controller card that will start the loading of all programs. This program will load in sector $ØØ of track $ØØ. If the number located at $8ØØ is one then control will be passed to $8Ø1. If $8ØØ has a number larger than one the rom code will load in more sectors. Electronic Arts is famous for loading in five to sixteen sectors to start the boot. Second, if you where to move this program down to RAM you could change it to do anything you would like. The following will give a basic example of this technique.

To begin, move the boot ROM down so we can modify it to jump to the monitor.

```
86ØØ<C6ØØ.C6FFM
86F8:4C 59 FF
```

Insert the original *EDD 4*, and start the boot.

```
86ØØG
```

Listing through the $8ØØ page to the end I looked for jumps to the next stage of the loader.

At $8AD you will find a JMP $4ØØ, this is the start of what is known as BOOT1 or the first boot stage. This is a jump into the text page, so we will have to move it up to examine it.

To do this three things must be done. One, we must change the code to maintain control and move the text page memory up. Two, you will also have to change the loading page of the zero sector, so the modified boot will not be overwritten. Third and last, you must redirect the boot zero code at $86ØØ from the monitor to the modified first stage code.

```
Jump to the move routine
  8AD:4C ØØ ØF   JMP $ØF00
Load sector zero into $8000
  8659:8Ø
Continue the boot
  86F8:4C Ø1 Ø8  JMP $Ø801
```

The memory move:

```
Four pages to move
  FØØ:A2 Ø4      LDX #$Ø4
Start at zero
  FØ2:AØ ØØ      LDY #$ØØ
Load a byte from the text page
  FØ4:B9 ØØ Ø4   LDA $Ø4ØØ,Y
Store it in a safe place
  FØ7:99 ØØ 14   STA $14ØØ,Y
Increment the pointer
  FØA:C8         INY
Move a full page
  FØB:DØ F7      BNE $ØF04
Increment the text page
  FØD:EE Ø6 ØF   INC $ØF06
Increment the store location
  F1Ø:EE Ø9 ØF   INC $ØF09
Out of pages to move
  F13:CA         DEX
No, then move more
  F14:DØ EE      BNE $ØF04
Yes, jump to the monitor
  F16:4C 59 FF   JMP $FF59
```

Now list through the moved code and you will find a JuMP to $CØØ (at $1477, which should be $477). This is the real start of the program, and the point at which it must be stopped.

Well, you have a computer, so let it do the work for you. This means writing a "Tape-worm", or a program to load in each stage, changing the jumps out to maintain control. When the program has loaded in, return control to the user. This is not as hard as it sounds for this boot process.

The ROM loads in code at $8ØØ, the jump to $8Ø1 (the first jump). The new code at $8Ø1 loads more code into the text page then jumps to it from $8AD (the second jump).

The last piece of code loads in the rest of the program then jumps to the start from $477 (the third and last jump). There are now three jumps we need to manipulate. To start the worm, move the ROM boot code to $86ØØ and add the code that follows:

```
  8600<C600.C6FFM
Overwrites the JMP $0801
  86F8:A9 Ø5
  86FA:8D AE Ø8
  86FD:A9 87
Change the JMP $04ØØ to JMP $87Ø5
  86FF:8D AF Ø8
Now jump to the first stage loader
  8702:4C Ø1 Ø8
  8705:A9 59
  8707:8D 78 Ø4
  870A:A9 FF
Change the JMP $0CØØ to the monitor
  870C:8D 79 Ø4
Jump to the second stage loader
  870F:4C ØØ Ø4
```

An 86ØØG will start the worm, which loads all of *EDD 4* into memory and leaves us in the monitor. *EDD 4* uses the memory from $CØØ through $5FFF (this includes the hi-res title page). Part of the memory range from $BØØØ through $BFFF is used by the program for disk access. These parts must be moved down and later replaced. Lastly the whole thing can be saved out of memory. The $1CØØ page was empty so I placed the start up routine there.

Step by step

1 Enter the monitor to make the tape-worm.

```
CALL -151
```

2 Move the boot ROM down and enter the code to finish the worm.

```
8600<C600.C6FFM
86F8:A9 Ø5 8D AE Ø8 A9 87 8D
8700:AF Ø8 4C Ø1 Ø8 A9 59 8D
8708:78 Ø4 A9 FF 8D 79 Ø4 4C
8710:ØØ Ø4
```

3 Insert original and run the tape-worm to load *EDD 4*.

```
8600G
```

4 Move the used portions of high memory down.

```
6000<B000.B3FFM
6400<B700.BFFFM
```

5 Boot a slave disk with a short hello program.

```
C600G
```

6 Enter the monitor again.

```
CALL -151
```

7 Add the start up code and memory moves:

```
1CØØ:A2 Ø4 AØ ØØ B9 ØØ 6Ø 99
1C08:ØØ BØ C8 DØ F7 EE Ø6 1C
1C1Ø:EE Ø9 1C CA DØ EE A2 Ø9
1C18:AØ ØØ B9 ØØ 64 99 ØØ B7
1C2Ø:C8 DØ F7 EE 1C 1C EE 1F
1C28:1C CA DØ EE A9 ØØ 8D F2
1C3Ø:Ø3 A9 C6 8D F3 Ø3 49 A5
1C38:8D F4 Ø3 AD 57 CØ AD 55
1C4Ø:CØ AD 52 CØ AD 5Ø CØ 2C
1C48:1Ø CØ AD ØØ CØ 1Ø FB 2C
1C5Ø:1Ø CØ 6Ø
```

8 Fix the start of the program so it will run:

```
BFD:2Ø ØØ 1C
```

9 Save the whole program (at last!)

```
BSAVE ESSENTIAL DATA DUPLICATOR 4,
    A$BFD, L$6ØF8
```

For versions higher than *EDD v4.3*

When *EDD 4* was upgraded to support the new Apple IIgs (version 4.4 or later) the loader and program was changed. Some of the absolute addresses given have changed and you need to save four more pages of memory.

If you follow along with the article to verify the right locations you can crack any new version that comes out. The changes are as follows:

1. The JMP $4ØØ is now at $8AF instead of $8AD
2. The JMP $CØØ is now at $47E instead of $477
3. There is a short routine from $8AØØ - $8DFF that is needed for the character table used by the hi-res disk scan. It moves itself down to $8ØØ-$BFF, and is called near the start of the program.

Here are the needed modifications to the step-by-step method:

2 Move the boot ROM down and enter the code to finish the worm.

```
86ØØ<C6ØØ.C6FFM
86F8:A9 Ø5 8D BØ Ø8 A9 87 8D
87ØØ:B1 Ø8 4C Ø1 Ø8 A9 59 8D
8708:7F Ø4 A9 FF 8D 8Ø Ø4 4C
871Ø:ØØ Ø4
```

4 Move the used portions high memory down.

```
6ØØØ<BØØØ.B3FFM
64ØØ<B7ØØ.BFFFM
6DØØ<8AØØ.8DFFM
```

7 Add the start up code and memory moves:

```
1C00:A2 04 A0 00 B9 00 60 99
1C08:00 B0 C8 D0 F7 EE 06 1C
1C10:EE 09 1C CA D0 EE A2 09
1C18:A0 00 B9 00 64 99 00 B7
1C20:C8 D0 F7 EE 1C 1C EE 1F
1C28:1C CA D0 EE A2 04 A0 00
1C30:B9 00 6D 99 00 8A C8 D0
1C38:F7 EE 32 1C EE 35 1C CA
1C40:D0 EE A9 00 8D F2 03 A9
1C48:C6 8D F3 03 49 A5 8D F4
1C50:03 AD 57 C0 AD 55 C0 AD
1C58:52 C0 AD 50 C0 2C 10 C0
1C60:AD 00 C0 10 FB 2C 10 C0
1C68:60
```

9 Save the whole program.

```
BSAVE ESSENTIAL DATA DUPLICATOR 4,
    A$BFD, L$64F8
```