

R-Type II

© 1989 Activision

**Deprotecting
MFM + Copylock Protection!**

Requirements for this Tutorial:

- . Amiga 500 (1 MB Chip) or WinUAE
- . Assembler (ASM-Pro or similar)
- . R-Type II Original or CAPS Image (#2065)
- . Source or Binary of Rob Northens Sectorloader (included in this archive)



Once again we begin our little journey with trying to duplicate the Original Disk:



We don't need to wait until the copy is finished... we already notice that this won't work! ☺

Let's see what is happening in the Bootblock, enter AR and read the first and only dos-readable Track to e.g. address \$50000, using "rt 0 1 \$50000".

Let's disassemble a bit ... <d \$50000> and scroll down some lines...

```
~050020 MOVE.W #7FFF,D0
~050024 MOVE.W D0,000FF096
~050028 MOVE.W D0,000FF09A
~050030 CLR.W 000FF180
~050036 LEA 5004E(PC),A0
~050038 LEA 00070000,A1
~050040 LEA 5004E(PC),A2
~050044 MOVE.L A1,-(A7)
~050046 MOVE.B (A0)+,(A1)+
~050048 CMPA.L A0,A2
~05004A BNE 00050046
~05004C RTS
=====
~05004E MOVEQ #FFFFFFF,D0
~050050 LEA 0007C000,A0
~050052 BSR 00050072
~050054 MOVEQ #0,D0
~050056 LEA 00000500,S,A0
~050058 MOVE.L A0,-(A7)
~05005A BSR 00050076
~05005C LEA 000002C,S,A0
~05005E MOVE.L #AE3B9CE3,(A0)
~050060 RTS
=====
~050072 BRA 00050096
```

Standard Stuff, killing DMA and Interrupts, copying the code from \$5004e up to address \$78000 and finally executing it with that "RTS" instruction -> (Address \$78000 has been pushed onto the stack before so the RTS will lead us there!)

The code at \$5004e already has some interesting information for us... we can see the address register A0 being set with \$500, then this value is pushed onto the stack, a routine is called (BSR \$50076), and finally the "RTS" will execute the code at \$500. So I believe that the routine at \$50076 will surely load some data to \$500. ☺

Another interesting thing is the value that is move'd into some memory region right before the RTS is executed.

A hardcoded value like #AE3B9CE3 looks suspicious to me so let's also remember this.

Reset your Amiga now, insert Gamedisk and let the loader track about 2 seconds.... Enter AR to see what is loaded up to address \$500.

```
Ready.
X
No known virus in memory!
Ready.
Breakpoint raised at address: 00000500
D0=000243CC 00000003 0000FFFF 0000FFFF 55555555 00000000 00001AF3 00001051
A0=00000E2C 000FF000 000FD000 0007CC10 00001558 00C014B6 00C00276 00000000
PC=00000500 USP = 00C014B2 SR = 2708 T=0 S=1 I=111 X=0 N=1 Z=0 V=0 C=0
$ 500
~000500 MOVEQ #0,D0
~000502 MOVEQ #0,D1
~000504 MOVEQ #0,D3
~000506 PEA 512(PC)
~00050A MOVE.L (A7)+,00000010
~000510 ILLEGAL
~000512 MOVEM.L D0-D7/A0-A7,-(A7)
~000516 PEA 532(PC)
~00051A MOVE.L (A7)+,00000010
~000520 MOVEA.L A7,A0
~000522 LINEP
~000524 ORI.B #40,D2
~000526 ORI.B #80,SR
~000528 ORI.B #7B,D0
~00052C ORI.B #7B,D0
~000530 ORI.B #40,D2
```

Ouch.. this looks like a little Copylock-Routine which is executed at first.

But perhaps we don't need to investigate this further, let's see where the copylock ends. If we remember right, the Bootblock just move'd that suspicious "Copylock-Like" result to address \$E2C... a place that is not too far away from the copylock start. Let's disassemble that area:

```
d e23
~000E2A CMP.L #AE3B9CE3,D0
~000E2C BNE 0000021EA
~000E34 LEA E3C(PC),A0
~000E38 MOVE.L A0,00000020.S
~000E3C MOVE.W #2000,SR
~000E40 LEA 000FF000,A0
~000E46 MOVE.W #7FFF,D0
~000E4A MOVE.W D0,96(A0)
~000E4E MOVE.W D0,9A(A0)
~000E52 MOVE.W D0,9C(A0)
~000E56 LEA 00000500,S,A7
~000E5A JSR 0000EBA6
~000E60 JSR 0000A7D8
~000E66 MOVE.L #13D9,00000014.S
~000E6E MOVE.W #0,000FF180
~000E76 MOVE.L #174500,30(A6)
~000E7E MOVEQ #FFFFFF,D0
~000E80 LEA 0002FD00,A0
~000E86 JSR 0000F924
~000E8C JSR 00001EAA
~000E92 JSR 0000AA3A
~000E98 MOVE.W #C7,D7
~000E9C JSR 0000F71F
~000EA2 JSR 0000A9D4
```

Ok, this is surely the code that is executed after the Copylockroutine is passed. It compares our #AE3B9CE3 with the value returned in D0.

Let us set a breakpoint here to see what D0 looks like, behave like shown in the next picture:

```

bs e2a
Breakpoint inserted
Ready.
Y
No known virus in memory!
Ready.
Breakpoint raised at address: 0000E2A
r
D0=AE3B9CE3 00000000 0000FFFF 00000000 55555555 00000000 00001AF9 00001051
A0=0000E2C 0007F00 0007F00 0007CC19 00001558 00C014B6 00C00276 00000000
PC = 00000E2A USP = 00C014B2 SR = 2700 T=0 S=1 I=111 X=0 N=0 Z=0 V=0 C=0
Ready.

```

So after some time we hear the well-known trackgrind and our AR will pop up again, telling us that the instruction at \$E2A is the next one to be executed. Let's have a look at the value in D0 ... it is what we expected it to be: **\$AE3B9CE3**.

For testing purposes you could restart your Amiga, reboot the game and after some tracking, set a breakpoint to the start of the Copylock at \$500. After AR pops up, insert some other disk, set a breakpoint at the end of the Copylock again and see what is happening to the D0 register. The value will be different and the game will show a green screen and stop loading!

So getting rid of this Copylock will be quite easy. We only need to overwrite the start of the Copylock, putting our special value directly to register D0, followed by a jump to the end of the Copylock. Due to the fact that we did not decode the inner loop of the Copylock, we cannot be sure if this special value has perhaps also been moved to some other memorylocations . This is happening quite often.. so the games can check these addresses later on and if not set with the correct value, lead to some strange behavior.

We can easily check this by searching for this value throughout chipmemory. Try it like this:

```

f AE 3B 9C E3. 0 00000
Search from: 000000 to: 000000
000060 000E2C 07801E 07FF0C 07FF24 07FFBA
Ready.

```

As we can see, our special value has been moved to 4 different places in chipmem. (\$7801E is part of the bootcode which was copied into highmem).

We need to remember these addresses as well and fill them with the correct value lateron.

Okayyy, this should be the copylock part... now let us find out how the Trackloader works to obtain a dump from the whole disk.

Leave Action Replay now and let the game continue loading.... while the game is tracking let's enter AR again to disassemble the trackloading code..... in my case the Program Counter was somewhere around \$FA00...

```

~00F9F4 RIS
=====
~00F9F6 LEA 000FF000,A1
~00F9FC LEA 000FF000,A2
~00FA02 MOVE.B #7D,100(A2)
~00FA08 ANDI.B #F7,100(A2)
~00FA0E ANDI.B #C0,E00(A2)
~00FA14 ORI.B #8,E00(A2)
~00FA1A MOVE.B #7F,D00(A2)
~00FA20 MOVE.W #5200,9E(A1)
~00FA26 MOVE.W #8400,9E(A1)
~00FA2C MOVE.W #8210,9E(A1)
~00FA32 MOVE.W #4489,7E(A1)
~00FA38 BIST #5,000FE001
~00FA40 BNE 0000FA38
~00FA42 RIS
=====
~00FA44 MOVE.W D1,D3
~00FA46 MULU.W #404,D3
~00FA4A MOVEA.L FB1C(PC),A3
~00FA4E ADDQ.L #4,A3
~00FA50 ADDA.L D3,A3
~00FA52 MOVEQ #7E,D3
~00FA54 MOVE.L #53555555,D4
~00FA5A MOVEQ #0,D5

```

Ok, this looks for sure like some trackloader actions here, we are at the right place.

Lets scroll downwards in memory to obtain the beginning of the Loader.

```

~00F948 SUBI.B #0,2(A7,D0.W)
~00F94E ROXL.B #6,D0
~00F950 SUBI.B #1,0(A3,D6.W)
~00F956 ORI.B #0,-(A2)
~00F95A ORI.B #0,D6
~00F95E ORI.W #F240,(A2)
~00F962 LEA FB20(PC),A1
~00F966 ADDA.W D0,A1
~00F968 MOVE.B (A1)+,D0
~00F96A MOVE.B (A1)+,D1
~00F96C EXT.W D1
~00F96E MOVE.W (A1)+,D2
~00F970 MOVE.L (A1)+,(A7)
~00F972 ESR 0000F900
~00F974 MOVE.L (A7)+,D0

d f960
~00F960 ASL.W #3,D0
~00F962 LEA FB20(PC),A1
~00F966 ADDA.W D0,A1
~00F968 MOVE.B (A1)+,D0
~00F96A MOVE.B (A1)+,D1
~00F96C EXT.W D1
~00F96E MOVE.W (A1)+,D2

```

The code at address \$F960 could be the start of the trackloader, right in front are only senseless instructions.

Lets get started! Reset, let the game track some seconds, then set a breakpoint to the loader at \$F960.

```
X
No known virus in memory!
Ready.
Breakpoint raised at address: 0000F960
r
D0=00000001 00000001 0000FFFF 0000FFFF 55555555 00000000 AAAAF2ED 00005045
A0=0004CE58 000FF000 000FD000 00030188 00001558 00C014B6 000148F6 000004F4
PC = 0000F960 USP = 00C014B2 SR = 2000 T=0 S=1 I=000 X=0 N=0 Z=0 V=0 C=0
n 414
:0004F4 00 00 1E E4 00 00 1E B0 00 00 0E 92 70 00 72 00 ...ä.....P.P.
bs 1ed4
Breakpoint inserted
Ready.
r d0 0
D0=00000000 00000001 0000FFFF 0000FFFF 55555555 00000000 AAAAF2ED 00005045
A0=0004CE58 000FF000 000FD000 00030188 00001558 00C014B6 000148F6 000004F4
PC = 0000F960 USP = 00C014B2 SR = 2000 T=0 S=1 I=000 X=0 N=0 Z=0 V=0 C=0
r a0 40000
D0=00000000 00000001 0000FFFF 0000FFFF 55555555 00000000 AAAAF2ED 00005045
A0=0004CE58 000FF000 000FD000 00030188 00001558 00C014B6 000148F6 000004F4
PC = 0000F960 USP = 00C014B2 SR = 2000 T=0 S=1 I=000 X=0 N=0 Z=0 V=0 C=0
n fd20
:00FB20 02 00 A5 99 00 00 0B 33 EF 1A 03 00 45 00 00 0B C5 .....3....E...
:00FB30 20 01 00 39 00 00 73 8F 24 0B 00 1C 00 00 38 AF ..9..$......8.
      ^
      |
      | Tell the new values like that!
```

When breakpoint happened, edit the values and registers like in the picture above and leave AR again. The game will (hopefully) start tracking the entire disk now, in this case to address \$40000.

After a minute, loading will be finished and AR pops up again.

```
X
No known virus in memory!
Ready.
Breakpoint raised at address: 00001EE4
sm dump, 40000 40000+b33ef
Disk ok
```

The complete disk should be available in memory now, ready for being saved!! Insert an empty disk and save the dump with
<sm dump, 40000 40000+\$b33ef>.

That's it! We can care about our crackdisk now and can put the original back in it's case.

After some testing with the crackimage I noticed that the Filetable is missing in the dumped data... the game loads it seperately before the trackloader is called for the first time. But that's not a big deal.

If you still remember, the first file in the filetable begins on track #2 sector 1. The filetable itself is stored in sector #0 right in front. So we already grabbed it because we dumped from that position.

We only need to remember this when coding the crackimage... we have to copy it into place before tracking the image back onto disk !

So boot up your favourite assembler now, reserve about 800 KB Chipmem and begin to code the diskimage. The sourcecode is included in this archive -> so no need to retype it.

This time I used the well known sectorloader from Rob Northen. The binary is included in this tutorial-archive.

It works with the following parameters:

- D0 = Drivenumber (0=DF0:)**
- D1 = Startsector**
- D2 = Sectors to read**
- D3 = Some Drivemotor on / off parameter**
- A0 = Loadaddress**
- A1 = MFM-Buffer**

Enough said, lets type some code!

```
Asm-Pro OS V1.17 By Genetic Source 0 >>DISKIMAGE_RNC.s
#####
## R-TYPE 1 (C) ACTIVATION
## DISKIMAGE (133 TRACKS) FOR FLASHTRO TUTORIAL.
## !! USE CHIPMEM FOR THIS FOR WRITEBACK !!
#####
##
## USAGE:
## 1. <A> ASSEMBLE SOURCECODE
## 2. <J> EXECUTE SOURCE TO MAKE NECESSARY PATCHES IN DISKIMAGE
## 3. <W> WRITE BACK 133 TRACKS FROM LABEL #DISK_START
## 4. <C> CALCULATE BOOTBLOCK-CHECKSUM
#####
DISKOFFSET_COPYLOCK_CODE = 2*$1800+$200
DISKOFFSET_ORIGINAL_LOADER = 2*$1800+$200+$F960-$500
DISKOFFSET_FILETABLE = 2*$1800+$200+$FB20-$500

INCDIR "SOURCES:TUTORIAL/"
; OVERWRITE OLD_COPYLOCK CODE WITH OUR NEW LOADER

LEA NEW_COPYLOCK_CODE(PC),A0
LEA DISK_START+DISKOFFSET_COPYLOCK_CODE,A1
MOVE.L (COPYLOCK_CODE_END-NEW_COPYLOCK_CODE)-1,D0
COPY.CL.MOVE.B (A0)+,(A1)+
Line 1 Col 1 Bytes 4984 Free 6584 1681 ---- 06:00:50
```

The fully commented sourcecode comes up on the last 2 pages.

When finished typing or whatever, follow the steps that are described at the top of the source to create the cracked diskimage.

```
Asm-Pro OS V1.17 By Genetic Source 0 >>DISKIMAGE_RNC.s
INCDIR "SOURCES:TUTORIAL/"
; OVERWRITE OLD COPYLOCK CODE WITH OUR NEW LOADER
;
LEA NEW_COPYLOCK_CODE(PC, A0
LEA DISK_START+DISKOFFSET, COPYLOCK_CODE, A1
MOVE.L # (COPYLOCK_CODE_END-NEW_COPYLOCK_CODE)-1, D0
COPY.CL: MOVE.B (A0)+, (A1)+
Line 1 Col 1 Bytes 5021 Free 6171 561 -*- 19:57:52
>a
Pass 1..
Pass 2..
Incbin : "SOURCES:TUTORIAL/RNLOADER.BIN" = 1582 (= $00005DE )
Incbin : "SOURCES:TUTORIAL/DUMP" = 734191 (= $000033EF )
No errors
>u
Updating .. sources:Tutorial/DISKIMAGE_RNC.s
File length = 5106 (= $000013F2 )
>j
D0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A0: 00075E1 00004F81 00000000 00000000 00000000 00000000 00000000 004253D0
SSP=004263CC USP=004253D0 SR=0004 -- -- PL=0 --Z-- PC=EOP VBR=00000000
>wt
RAM PTR:DISK_START
DISK PTR:>0
LENGTH:>133
>■
```

Finally reboot your machine and have a play.
This game rocks!! ☺



--	--

```
; #####
; ## R-TYPE II (C) ACTIVISON ##
; ## DISKIMAGE (133 TRACKS) FOR FLASHTRO TUTORIAL. ##
; ## !! USE CHIPMEM FOR THIS FOR WRITEBACK !! ##
; ## ----- ##
; ## USAGE: ##
; ## 1. <A> ASSEMBLE SOURCECODE ##
; ## 2. <J> EXECUTE SOURCE TO MAKE NECCESSARY PATCHES IN DISKIMAGE ##
; ## 3. <WT> WRITE BACK 133 TRACKS FROM LABEL #DISK_START ##
; ## 4. <CC> CALCULATE BOOTBLOCK-CHECKSUM ##
; ## ----- ##
; ##### (ALPHA ONE) #####
```

```
DISKOFFSET_COPYLOCK_CODE      = 2*$1800+$200
DISKOFFSET_ORIGINAL_LOADER    = 2*$1800+$200+$F960-$500
DISKOFFSET_FILETABLE          = 2*$1800+$200+$FB20-$500
```

```
INCDIR "SOURCES:TUTORIAL/"
```

```
; OVERWRITE OLD COPYLOCK CODE WITH OUR NEW LOADER
; -----
```

```
LEA    NEW_COPYLOCK_CODE(PC),A0
LEA    DISK_START+DISKOFFSET_COPYLOCK_CODE,A1
MOVE.L # (COPYLOCK_CODE_END-NEW_COPYLOCK_CODE)-1,D0
COPY_C: MOVE.B  (A0)+, (A1)+
        DBF     D0,COPY_C
```

```
; OVERWRITE FIRST INSTRUCTION OF ORIGINAL LOADER WITH
; A "JMP $500" -> SO OUR LOADER IS EXECUTED!
; -----
```

```
LEA    DISK_START+DISKOFFSET_ORIGINAL_LOADER,A0
MOVE.W #$4EF9, (A0)+
MOVE.L #$500, (A0)+
```

```
; THE FILETABLE IS LOCATED ON TRACK #2 SECTOR #0 ON DISK
; AND IS LOADED SEPERATELY BY THE ORIGINAL CODE. WE SKIP THIS PART
; AND COPY IT INTO THE RIGHT PLACE HERE ($FB20 IN MEMORY).
; -----
```

```
LEA    DISK_START+2*$1800,A0                ; TR #2 SEC #0
LEA    DISK_START+DISKOFFSET_FILETABLE,A1
MOVE.L #$90,D0                               ; SIZE OF FILETABLE
COPY_F: MOVE.B  (A0)+, (A1)+
        DBF     D0,COPY_F
        MOVEQ   #0,D0                        ; END OF PATCHING!!
        RTS
```

```
; -----
; =====
; THE NEW TRACKLOADER
; -----
; HERE COMES OUR NEW LOADER THAT IS WRITTEN TO THE PLACE OF THE
; ORGNAL COPYLOCK, IN MEMORY AT $500, ON DISK LOCATED ON TRACK #2, SECTOR #1
; (FILE #1 IN FILETABLE)
; =====
```

```
NEW_COPYLOCK_CODE:                ; <- $500 IN MEMORY !
        MOVEQ   #0,D1                ; HANDLE THE FILES LIKE THE ORIGINAL
        MOVEQ   #0,D2
        ASL.W    #3,D0                ; FILENUMBER * 8
        LEA     $FB20,A1              ; FILETABLE PTR
        MOVE.B  (A1,D0.W),D1          ; D1 = TRACKNUM
        MULS    #1800,D1              ; * TRACKSIZE $1800
        MOVE.B  1(A1,D0.W),D2         ; D2 = SECTORNUM
        MULS.W  #512,D2               ; * SECTORSIZE $200
        ADD.L   D2,D1                 ; GET CORRECT BYTEOFFSET
        DIVS.W  #512,D1               ; AND TURN IT INTO AN SECTOROFFSET
        MOVE.W  2(A1,D0.W),D2         ; GET FILESIZE IN SECTORS
        ADDQ.W  #1,D2                 ; +1
        MOVEQ   #0,D3
        MOVE.L  4(A1,D0.W),-(A7)      ; SAVE REAL BYTESIZE OF FILE ON STACK
        LEA     $2FD80,A1            ; MFM POINTER
AGAIN:  MOVEQ   #0,D0                 ; DRIVENUMBER (DF0:)
```

```

        BSR.W  TLOADER          ; CALL RNC TRACKLOADER
        TST.B  D0               ; LOADING OK?
        BEQ.B  LD_OK            ; YEAH -> END TLOADER

BLINKER:
        BTST   #6,$BFE001      ;
        BNE.B  FLASH           ; OTHERWISE MAKE SCREEN FLASH
        BRA.B  AGAIN           ; AND WAIT FOR LEFT MOUSE TO
FLASH:  MOVE.W  D0,$DFF180      ; REPEAT PROCEDURE :P
        SUBQ.W  #1,D0          ;
        BRA.B  BLINKER         ;
LD_OK:  MOVE.L  (A7)+,D0        ; RESTORE BYTESIZE OF FILE
        RTS                ; FOR FURTHER GAME-USAGE

TLOADER:
        INCBIN "RNCLOADER.BIN" ; ROB NORTHEN SECTORLOADER

COPYLOCK_CODE_END:

; ==<DISKIMAGE>=====

DISK_START:

        DC.B   "DOS",0         ; TRACK 0 (BOOT)
        DC.L   0
        DC.L   0

        MOVE.W  #2,$1C(A1)     ; READTRACK
        MOVE.L  #2*$1800+$200,$2C(A1) ; DISKPOSITION OF FILE #1 IN FILETABLE
        MOVE.L  #24400,$24(A1) ; SIZE
        MOVE.L  #50000,$28(A1) ; LOADADDRESS
        JSR     -$1C8(A6)      ; DOIO()

        LEA     COPY_C(PC),A0   ; COPY NEXT PART OF THE
        LEA     $78000,A1       ; BOOTCODE TO $78000
        MOVE.L  # (BOOTEND-COPY_C)-1,D0 ; SO THAT WE DONT OVERWRITE
COPY:  MOVE.B   (A0)+,(A1)+     ; OUR OWN BOOTCODE WHEN
        DBF     D0,COPY        ; COPYING GAMEDATA DOWN TO $500 !!

        MOVE.W  #7FFF,D0       ; KILL DMA/INTERRUPTS
        MOVE.W  D0,$DFF096
        MOVE.W  D0,$DFF09A
        MOVE.L  #78000,$80.W    ; CONTINUE AT $78000 IN
        TRAP    #0             ; SUPERVISOR MODE

COPY_C: LEA     $50000,A0       ; COPY OUR GAMEDATA DOWN
        LEA     $500.W,A1      ; FROM $50000 TO $500 !!
        MOVE.L  #24400,D0

COPY_D: MOVE.B  (A0)+,(A1)+
        MOVE.W  D0,$DFF180
        SUBQ.L  #1,D0
        BNE.B  COPY_D

        MOVE.L  #$4E714E71,$E86.W ; NOP OUT LOADING OF FILETABLE
        MOVE.W  #$4E71,$E8A.W    ;

        MOVE.L  #$AE3B9CE3,D0    ; MOVE COPYLOCK KEY TO D0
        MOVE.L  D0,$60.W         ; AND TO ALL MEMORYLOCATIONS
        MOVE.L  D0,$E2C.W
        MOVE.L  D0,$7FF0C
        MOVE.L  D0,$7FF24
        MOVE.L  D0,$7FFBA
        JMP     $E2A.W           ; NO NEED FOR A JMP TO $500.
                                ; WE GO DIRECTLY TO $E2A!

BOOTEND:
        BLK.B   $1800-(BOOTEND-DISK_START),0 ; ZEROES REST OF LONGTRACK 0

        BLK.B   $1800,0         ; TRACK 1 (EMPTY)

DUMP:  INCBIN   "DUMP"          ; TRACK 2 - END

        BLK.B   2577,0         ; ZEROES TO GET AN EVEN DISKSIZE
                                ; 133 DOS TRACKS IN ALL

DISK_END:

```