(Continued from previous chapter)

- - - ===< When MacsBug won't work >===---

But, as usual, life ain't always so simple! A lot of applications won't allow you to use a-traps to crack them. I think they take away memory from MacsBug or something. To crack these you have to take a different approach!

The first thing to do is to make sure that none of the useful a-traps work. Not every program uses the a-trap ModalDialog, for example. So try some of the other a-traps I listed in the previous part of The Ultimate Mac Cracking Guide! I've seen, for example, a guy who rewrote all a-traps dealing with getting info from dialogs. Everything except the CloseDialog a-trap. So I caught him with that!

Another approach is to use the InitGraf a-trap. This a-trap is usually called by the program in the beginning while it's being initialized. If you call this, you'll have to step through the code until it reaches the subroutine dealing with registration, and then take it from there.

An a-trap that I am not very fond of myself is TEKey. This a-trap is used when getting text information from dialogs (among others). The problem with it is that it is very tedious to get out of it! It is very difficult to find your way back to the registration subroutine! But if you have a lot of time on your hand then go ahead, use it! This a-trap will almost certainly work under any conditions!

So, you've tried every a-trap you could think of, but you just won't be dropped into MacsBug! This is the point where you'll have to disassemble the program and follow through the code! Since I'm too cheap to by any of the commercial disassemblers, I will use Super ResEdit, with it's code editing feature.

The first question is "where do I start?" Well here's a hint, when a dialog is displayed on the screen, usually (unless the author doesn't use a-traps at all) the resource ID number of the registration dialog is pushed onto the stack and followed by a GetNewDialog a-trap. So now we know what to look for!

Open up the application you wanna crack with Super ResEdit, and have a look in the resources dealing with dialogs. You wanna find the dialog which is used by the application to get your registration information! When you've found it write down it's ID

number. For example, in CrackIt the dialog box responsible for getting registration info has ID number 128. Now, convert 128 into hex (=80). Now open up the CODE resource and in the case of CrackIt open up resource ID 1 . Activate the hex editor window and search for the hex string "0080". Why 0080 and not just 80? Because you gonna find a hell of a lot more 80's than 0080's!

Oh yeah! Before I forget. Some applications have a LOT of code resources. And unfortunately you'll have to go through each one of them and search for 0080. And also, make sure you're searching for a hex string not an ASCII string!

So anyway, once you've found 0080, activate the code editor window. And you should see one of the subroutines dealing with putting up the registration dialog box before you. This is great, but how do I find the registration subroutine? If you'd care to click on the "Addr" value of the first command in Annon 25, then you'll be able to find, from the Goodies menu, which instructions will branch to that specific command! This is a very useful feature that you'll be using quite a lot!

And after that it's just to follow through the code like you would in MacsBug. Unfortunately it is not as simple as stepping through something with MacsBug, since you can't have a look at what's happening to the different registers.

I have seen programs that don't use dialogs to get the registration information! In this case you should check under the "ALRT" resource. If the programmer is really paranoid he/ she might create an own resource where he stores the data for the registration dialog box. In this case...uhm look for the ID of this resource?

And if you can't decide which dialog box the program uses for registration, then have a look in the "STR#" resource or in the resources dealing with strings. Here you're looking for the ID of the resource which contains the text that is displayed in the registration dialog.

Remember one thing, however! Try to narrow down your search by using MacsBug! If you can, then try to break into the registration procedure with any a-trap. Remember, the point is to get to the main registration routine, and it doesn't matter how the hell you get there! There are no rules! There are only "easy ways" and "not so easy ways" to crack programs. It's up to you how you do it!

Oh yeah! And before I forget! Some people label their subroutines! And some people use labels such as "Registration". So when you have the code editor window active in Super ResEdit and you see under the Modules menu that one of the subroutines is called "Registration", then don't bother looking for ID numbers and shit like that! Go to the subroutine straight away!

- - - ===< Other Types Of Protection >===---

At this point Part 3 is over 4500 words long and I feel that this is enough info for another part. But HA 12 being the last issue and all, I will describe how to crack other protection system very briefly!

- Date Expiration Schemes: Programs protected with these kinds of protection are usually demos designed only to last for a month or so. They will most likely compare the current date to the date stored in the preferences file (or in the data/resource fork), and then branch accordingly. Therefore, the easiest way to defeating time protected demos is to simply turn the date back in the "Date & Time" control panel. But if you wanna crack the program the complicated way you wanna look for a-traps again. The a-traps to look for are: GetDateTime, Random and TickCount. The last two a-traps would be used if the program didn't call the time check routine at startup but at random times during the time it's running. And if that is the case, you are best off if you change the code with Super ResEdit! Again somewhere the program NEEDS to use a CMP or a TST command, to see if the time actually has expired or not. So keep your eyes open for those!

- Key disk protection: You know that this is the protection system, if the program asks for a disk (or a CD) in order to be able to start up. The way this protection would work, is that somewhere in the program, the disk will be checked for either an invisible file, or a bad bitmap (bad block). Now, in order for the program to see if the bad block exists, it has to attempt to read from the disk itself. The a-traps for the Device and File managers are: Open, Close, Read, Write, Control, Status, KillIO, GetVolInfo, Create, Delete, OpenRF, ReName, GetFileInfo, SetFileInfo, UnMountVol, MountVol, Allocate, GetEOF, SetEOF, FlushVol, GetVol, SetVol, FInitQueue, Eject, GetFPos, HSetVol, HGetVol, HOpen, HGetVInfo, HCreate, HDelete, HOpenRF, HRename, HGetFileInfo, HSetFileInfo, AllocContig, HSetFLock, HRstFLock. Have fun finding the correct one! After finding the correct a-trap, there will most likely be a conditional seeing whether the file (bad block) existed or not, and if it did then you're in the

clear! On the other hand if the program needs an invisible file on the disk, then you will have to copy that invisible file to another disk if you wanted to use two copies of the program. But again, the possibilities are endless, and it's up to you to figure them out and to defeat them!

Hardware Key Protection: The most difficult of them all, but not undefeatable! And I know this for a fact ;-) Hardware Keys are usually hooked up to the ADB port (keyboard). The program then searches for data stored within the key. Now, the only way to defeat this is to go through each and every instruction asking data from the key and then changing the code accordingly. If you want a nice project to work on, I can recommend you to design a program that will "suck" the contents out of hardware keys. Meaning that it will display all information stored within the key. Until that happens though you will have to trace through the code manually looking for places where the program asks the key for a value. In my list of a-traps these are the ones I found that had to do with ADB: ADBReInit, CountADBs, GetIndADB, GetADBInfo, SetADBInfo, ADBOp. This is not much, I know, and I don't even know if these will work. So it's up to you to find out!

- - - ===< The End >===---

This is it folks. If you've managed to read all three parts of The Ultimate Mac Cracking Guide, then you deserve a medal! I've tried to cover things in detail, but there are things that I just simply couldn't include. If things go according to plan there will be a "proper" PPC cracking guide coming out in a couple of months! But by then we won't have HackAddict to publish in. So just keep your eyes open on other locations!

And before I disappear into cyberspace I think it is time to say "Thank you!" to The Weasel! He's done something remarkable! He created a place where a group of people sharing the same dreams could meat and exchange ideas! His name shall be with us forever, and take the highest place on the "Hackintosh Hall of Fame"!

ProZaq