Unveiling The Pirate
Part 1: Current Methods

Richard T. Evers, Editor

In this article I will be releasing information that is known by few, of which those in the know hold to be a very deep and dark secret. In my opinion this has been going on for a little too long. Software piracy has run wild for quite a few years now, and it is about time for a little information to leak out about what is happening, how it is happening and how to impede its cancerous spread just a slight bit.

Though I do not profess to know great scads of information about every form of protection and deprotection known to mankind, I have collected enough to put together a fairly thorough presentation. This article is but the first in a trilogy on this subject. The balance can be found in this issue, and are guaranteed to please even the knowledgeable reader. As I have just stated, this trilogy is for the sole purpose of enlightening programmers about the reality of piracy and how to prevent your creations from becoming just another notch on a pirates belt. This is not a diabolical scheme to hatch new flocks of hackers, even if it appears so at times.

Program protection is incorporated by people who want to protect their creations from copyright infringement. Program deprotection is an occupation taken on by those who find that normal programming is often very dull. Piracy adds a bit of spice to an otherwise terrific occupation. This spice of life costs software manufacturers millions of dollars every year in lost revenue, and it usually does not put a cent in the average pirates pocket. I stress the point that this is usually done for the thrill, not monetary gain, because this fact alone makes the prosecution of pirates in court very difficult. The courts are usually good for offenses that can be easily proven, which is rarely the case with software theft. For more information on the legal aspects, flip to part 3.

There are five methods commonly used to protect software from illegal distribution, which are as follows :

1 - Diskette protection
2 - Dongle Protection
3 - ROM Protection
4 - Program In ROM Pack
5 - No Protection

Diskette Protection

This form of protection is one which I favour the least. This technique, though cheap to incorporate and often one that will work, is pretty bad news for the average user. Once a user has purchased the protected diskette, they have one or two copies to work with. As most people can confirm, accidents do happen. If your disk drive packs up, your dog eats your diskette, or even if you happen to mess up the diskette yourself, you are in trouble. Some manufacturers give you two diskettes. Very nice. They also give you a card that entitles you to get another disk, at a nominal charge, if you return your fouled up diskette to them. The trouble with this is the software manufacturer lives in some distant city, leaving you with only one method of transporting the diskette short of an expensive courier - the mail. And in all probability, the manufacturer won't spare the expense on the former. So if the postal service is as careful with your mail as it usually is with mine, then expecting a new diskette back in workable condition is more of a fantasy than anything else. Even if the diskette does not look physically damaged, you will often find it riddled with new and improved read errors. Just terrific. It seems that troubles accumulate faster than you can get rid of them.

Diskette protection has other disadvantages. First, it really makes your disk drive work for its money. The read errors and strange formatting tricks cause your drive to virtually have a stroke every time a protected disk is read in. This protection also takes up to three times longer to load in than normal, being especially noticeable with the 1541 drive. For an encore, disk protection is often written for a specific drive, excluding all others. For those of you with a dual drive setup with your 64, you may often be out of luck trying to LOAD in a protected disk. When faced with more than one drive, or with a RAM/ROM or interface combination that the program can't figure out, the software purposely bombs out. A rotten trick to play on someone who has invested in your program.

Pirates seem to enjoy disk protection though. The challenge alone makes your program an easy candidate for the next "unprotection". Between the bit copiers available on the market, and rewriting the software to stop checking for all the errors on disk, there is a great quantity of bootleg software available that was originaly disk protected. Not only that, but these deprotected wonders are often better

than the originals. They LOAD in faster, and save you disk space by allowing more than one program per diskette. These two factors alone have the average user avoiding software that is protected this way.

Some time ago I was shown a method to break a few of the simple disk protected programs available. The trick to this is to first backup a copy of the protected disk on a 4040 drive, or any dual drive with swing down doors, then start the backup a second time but with a twist. Open and close the drive door about ten times, or until the backup procedure ends. Then remove the diskette from the drive. A read error has been created on the new copy that will closely resemble what is found on the original. If the software manufacturer has relied entirely on this single read error for protection, then the game has been lost within a five minute period. Even if their method of protection was a little more extreme, a good hacker will end up winning. For all of these reasons, I do not recommend disk protection at all.

Dongle Protection

This form of protection is my personal favourite. You have at least a fighting chance against the pirate with this one, with the victor often the manufacturer. In case you are unsure of what dongle protection is, let me explain. A dongle is a rude name for a hardware apparatus that is plugged into your computer. On the PET/CBM series, a dongle can be located on the user port, or on either of the two cassette ports. On the Commodore 64 and Vic 20, you can locate them on the user port, game cartridge port, joystick ports and cassette port. Quite a few options. Now for the explanation of what they do. Inside the dongle can be found anything from one piece of wire to a complete assortment of electronic components. With the proper combination, and the proper location, a program can check to make sure that the dongle is in place. For an added thrill, use the results generated by the dongle in the calculations and operation of the program itself. Anything from timers or pulse multipliers to frequency generators or filters can be included. Therefore, even if the hacker can manage to stop the program from checking for the dongle, the program may never work properly again.

There are ways around dongle protection though. The simplest method is to break into the dongle and find out what's inside. If this can be achieved the hacker has a 50/50 chance of reproducing it.

If the dongle is filled with some form of material to stop breakage, the hacker may assume that the covering has been placed there simply to disguise virtually transparent protection. Not transparent in the sense that there is none, but transparent in that it can be quickly reproduced by those in the know, if so inclined. Some dongles are merely a jumper between two pins. Protection like this lasts about as long as one cup of coffee. A little more thought an a dongle can send a hacker to a caffeine rehabilitation center. It's up to you to decide how clever to make it.

At this point the hacker has a few options. Crack off or dissolve the material encasing the components, or X-Ray the entire key to see what's inside. If the identication hasn't been removed from the components, and the wiring isn't purposely misleading, then the X Ray technique will probably work. It's amazing how a friendship with a dentist can be beneficial to a pirate.

To stop the hacker from gaining any ground by chipping away at the covering material on your dongle, place a few very important thin wires throughout the material itself. Once the chipping begins, these wires will be cut by the illustrious chipper, thus making the dongle useless. If enough wires are used, the key will become useless to the hacker by the time they reach any important components. In a proper casing, a dongle will be destroyed before it reveals itself.

Dissolving the material that covers your components is one method that can prove effective if care is not taken to disguise the operation of the circuitry or the identification of the components. There is one sure method to discourage the hacker from this technique. Use a material that is impervious to most solvents. Most software manufacturers use whatever plastic material they can find, like epoxy resin. There are many commercially available chemicals that can dissolve epoxy in relatively no time at all. And it's a shame to allow a hacker to win so easily.

There is one substance I use that is impervious to solvents, or heat for that matter. It is called methyl methacrylate, or quite simply, denture material. This can be purchased in many forms, with the easiest and least expensive being Tray Material. Tray Material is true denture acrylic, but manufactured for a vastly different purpose. Though the dental profession frowns on sales outside of its little community, try a few of the smaller dental supply companies, or smaller dental manufacturers. These companies will often deviate from normal procedures, with the correct amount of prodding. And your dongle producing department will feel much more confident. Expect to pay about $8.00 per pound up for the material.

The reason why dongle protection is a favourite with me is because it's terrific for the user. The installation of the dongle is often very easy, and there is no limit to the number of copies that can be made of the program disk. There is also that security blanket knowing that the pirate has to actually get into your code and figure it out how to stop the check for the dongle to break your beast. If care was taken in the design of the program, and if thought was given to use the results generated by the dongle in the actual operation of the program, then the hacker may be in for an indefinite amount of work.

For a final analysis on this one I recommend it whole heartedly. The cost is higher for the manufacturer in relation to disk protection, but the end result is better business. A replacement key can be shipped through the mail, without

damage in most cases. There will be no undue wear on the users drive through use of your software, and you can expect less pilferage with a dongle protected program. Pretty good all around, but still not impervious to a determined hacker. However, most, if not all, will throw in the towel after buying their third or fourth package in their attempts to unprotect your program.

ROM Protection

ROM protection is a form of protection that is disliked by many. A technique that applies more to the PET/CBM user than any of the others, this type of setup uses a single ROM placed in either the $9000 or $A000 socket in the computer. The ROM will have anywhere from 2 to 4K of code burnt into it to help stop illegal usage. The installation of this ROM by a user is the pitfall here. Broken pins and improper installation are too often the end result for the inexperienced. It is also a simple challenge for most pirates, and can be financially prohibitive. The initial cost to produce a ROM far outweighs the effectiveness in most cases.

The most common method programmers use to bypass ROM protection is to have a soft ROM built into their computer. A soft ROM is a device made from RAM that can appear to be ROM in the eyes of the software. ROM contents can be saved to disk and then loaded into the soft ROM, thus fooling the program into believing the ROM is actually there. Soft ROMs are available from many sources, and can be installed in little time. Their average cost is about $150.00, and can be used for the $9000 socket, $A000 socket, or both. Your choice. This method of piracy is not one that software manufacturers worry about though. They worry about the hacker that actually rewrites their program to work without the ROM.

This method of deprotection was more common a few years ago than it is now. All the truly fine programs that were ROM protected have already been broken. The method used to break the program is usually to relocate the contents of the ROM somewhere else in RAM, then rewriting the program to access it in the new spot. The user may lose a bit of memory, but the programs will still work.

Another method used was to rewrite the program to not check for the ROM at all. Some programmers, when designing ROM protection, never actually put much thought into how they were going to protect it until it was too late. The routines in ROM were not used for anything, therefore the programs were usually very simple to break. Many other ROM protected programs available have been been well designed in the protection department, with vital code actually placed within, but still to no avail. The pirate knows where the protection is, and can often see the access points with a simple disassembly. The balance of breakage occurs with a little time and effort. But it can usually be accomplished. Pretty rotten, but possible.

The final method that the hacker would take to break a ROM protected program is to physically copy the ROM itself. ROM burners are available from numerous sources, with average cost riding around $100.00. What these burners allow you to do is mass produce most ROMs with EPROMs, as long as you have an original copy of the ROM on disk. For the price of a ROM burner, and the price of the EPROMs, about $15.00 each, the pirates can do whatever they like. Some pirates have been known to photo copy manuals, burn new ROMs and sell the pirated packages for reduced prices, which is a very sleazy way to steal a dollar.

When all has been taken into consideration, ROM protection is not the best way to protect your program. Not only is it a pain for the average user, especially if a few ROM protected packages are used, but it's an easy mark for most pirates. If at all possible, stay away from this one.

Program Located In RON Pack

This is the protection most encountered today in the games that Commodore releases. This technique is one that is pretty good, and very nice for the user. A simple plug in of the ROM pack into the game socket, and your program comes alive. Very nice, but hardly impervious to the inventive hacker.

There is one device designed and used by pirates that is constantly in use destroying ROM Pack protected games. After the computer and cartridge combine to allow the program to begin, a press of two buttons concurrently will bring this poor game to its knees. This high tech black box has effectively stopped the program in its tracks without destroying the colour table, zero page or any of the other equally important areas in RAM. With this step taken, a simple SAVE to disk and a bit of work later will produce another broken game to add to an already overflowing collection of pirated programs. What a rotten trick, to design a electronic pirate. Write a bit of code, toss in a bit of hardware, and presto, instant hacker. The human element has finally been taken out of the piracy game.

Another method used to break ROM pack protection is so simple that most do not think to use it. Reproduce the ROM and make your own pack. The trouble that pirates find with this is the cost of the ROM pack. When mass produced, the cost for raw materials is very low. When bought by the average hacker, the price is very high. And the cost of the ROM pack has to be shelled out each and every time a copy is to be made. Too high for most pirates, which is exactly what the industry hopes for. A hacker is usually too cheap to spend the money on the programs themselves, when they know that most of the their software came for the price of a diskette, if they bought the diskette at all.

Making cartridges can get pretty involved too. Some use several PC boards laminated together with enough interconnections to make X-Rays of the unit so confusing that they're useless. And like a good dongle, any attempt to dismantle the boards usually ends up destroying them.

In my opinion, this form of protection rates just below the dongle. Your program will be quite safe, but a little more costly to produce. You be the judge, for you alone know what market your program is geared towards. If the market is huge, there is a pretty good chance that your profits will still be huge, even with breakage.

No Protection At All

The last form of protection for this article is zero protection in the major sense of the word. I know that most people don't agree too much with that one, but it could be viable. All it would take would be some careful prior planning.

By prior planning I mean that you should write your program with a really fine manual in mind. Games players usually don't need the manual, but business and application software can be worthless without it. Your manual can be printed so it's tough to photo copy, and photocopied books stand up better as evidence in court. Besides, photocopying is too expensive for chintz hackers. The software might be fabulous, but supplying a manual with every copy they wish to hand out will have pirates moving on to less documented packages.

Other protection might be considered like a dealer/user contract, to be signed by the user, and a serial number placed on each diskette, in a spot few would ever think to look. Make it clear with a message in your program that without the contract the user is in violation of the law. Forged contracts are even more frightening to a reputable business than a copied manual. Produce a truly fine overall package and most would rather own one legally than accept a duplicate.

Serial numbers will help too. Write the serial number in the manual in a few locations, so that it will reproduce if photo copied. Remember that most people don't look through the entire manual before photocopying it. Copyright law does apply to manuals, so guard yourself well. The combination of dealer–user agreement and manual will stand up in court if necessary. The serial number written on to the diskette is also a great help. Don't inform anyone that the serial number is on the diskette, just keep a log of it somewhere, like in your records, and on the contract that the user signs. If you ever find that copies of your program are circulating, a quick check onto the disk surface will determine where the diskette originated. With that determined, legal action is your next step.

A word on the placement of the serial number. Write the number in a spot on the surface of the diskette that is sure not to be disturbed by any disk activity. For all of the disk types numerous hiding places do exist, you just have to locate them. For an encore, encrypt the serial number in such a way that if it was found, it could not be easy deciphered. The low/high ASCII value is fine, but the low/high number EOR'd with a number known only to your company, would be just right. Have your program check for the number, just to make sure it is there. If the number isn't, fry the disk itself. Then put the computer into a death loop, just to even the score a little for them trying to steal your creation. When your rights are at stake, protect yourself to the hilt.

Another point to ponder with this technique is to store the serial number on disk along with a constant used for calculations within your program. Numerous programs on the market use their protection as part of the calculation process, for setting up the screen dimensions, some mathematical calculations, and for a variety of other reasons. Why not do the same thing. Instead of frying their diskette, let them continue using the program. If the serial number isn't there, the constant will be absent from their calculations.

Just imagine how much money it would cost a company to use a pirated accounting package protected in this manner. They might not realise that anything was wrong until year end. Then suddenly they would have an entire fleet of auditors ripping through their records to find out how they made/lost all that money during the year. For the few hundred dollars the company saved on the program, they lost it a hundred times over in man hours to correct the mistakes. A pretty good way to get even with a cheap firm. If the firm has the nerve to complain, ask them for the diskettes, manual and user contract from the point of purchase. Try not to react too quickly, and the unsuspecting firm may lead you to the source. Then bring the curtains down on both of them. A lawyer is the next step, and you have a pretty good shot at winning too.

And Finally, The End

Other methods exist to protect and deprotect programs, but these are really just further extensions to those already covered. As mentioned at the beginning, this article is one of three that has been prepared for the occasion. As you have discovered, this one deals with the methods used to copy protect your programs. Part two I enjoyed writing most; Programming Sleight of Hands, an article that will take you through some practical methods to protect your programs, and how these methods are often superceded. The final article is about the legal aspects of piracy, a product of some quite extensive research.

Whatever your choice of reading, I hope that you have enjoyed this issue. Your views on this subject, and on everything that we have printed in this issue, are welcome. We will be able to refine our magazine into the jewel we know her to be capable of becoming, with just a little help from you. Express your views on paper, disk or cassette tape, and send them in to us. By piecing together what our readers like and dislike about our publication, we can produce the highest quality magazine found anywhere.

May you find that all of your bugs have four wheels and an engine in the rear. RTE