MyKey Technology Inc.

The Fallacy of Software Write Protection in Computer Forensics

Mark Menz & Steve Bress

Version 2.4
May 2, 2004

MyKey Technology Inc.

1.0 Table of Contents

1. Table of Contents
2. Abstract
3. Introduction
4. Problems
    a. Controlled Items
    b. Unforeseen failure problems that can occur and bad luck
    c. Administrative reasons
5. Conclusion

2.0 Abstract

Abstract
The use of software for digital media protection in the field of computer forensics is not viable. The complexity of the systems and inability to control the makeup of those systems preclude the predictive nature that is required to insure a software write blocker, and other functions of media protection, can work reliably and consistently. The preceding coupled with the added element of human discretion eliminates the ability to certify that a system will work properly consistently. This ability to work consistently is imperative when dealing with original evidence, which in most cases is a hard drive from a computer. A failure, such as a write to the drive, can cause unrecoverable damage. Basically, a PC based system is designed to write to its media. Any failure of a software protected PC based system can allow it to write to its media. A properly designed hardware media protection device, however, allows no changes to the media even if it has a failure. You can certify a controllable device but not a dynamic system.

3.0 Introduction

Introduction
The first and primary concern in computer forensics is the preservation of the Court's evidence. Conflicting with that concern is the operating system and operating hardware of the computer you attach the evidence to. Take, for instance, the Microsoft Windows® operating system. There are a tremendous number of file writes that occur upon boot up of any Windows operating system. Any idea how many write commands are issued in a standard Windows boot up? Most people are not aware that data may be written to a drive during the boot process. If any changes gets through, your evidence is now questionable. With the complexity of today's operating systems, you just can't tell how many programs will load during the boot sequence. For example, run Task Manager and you'll be shocked by the number of separate Processes that are launched at boot up. If any of these launch and make a write call to the hard drive, your evidence is toast.

When you duplicate or preview the original evidence, failure is not an option. Recovery from a failure that changes the evidence from its original state during the duplication or preview phase is not possible.

The chances of a failure vary greatly for a number of reasons. For example, in most cases you do not know the true heritage of the evidence (in this case, some form of digital media) that needs to be preserved and reviewed. At no point can even a single bit be changed without serious consequences. To that end, it is imperative that the environment in which the evidence is placed and operated be predictive, controlled and protective. This is the foundation for digital media protection; of which write protection (write blocking) is one part.

For a system to be controllable it must be predictive. A system that is controllable can also be designed to be protective. The key here is predictive behavior of the system. Knowing what a system will do at any given point in time is predictability.

A personal computer today is a complex system of components, devices, data and operating and file system software. Within this complex system, an increasing number of autonomous automatic tasks are performed. Some are well known and some not as well known. Combine this with human discretion and just plain bad luck, predicting and controlling the system consistently is all but impossible. Consistently controlling the ability of a computer system to not modify the state of the evidence media via software cannot be done in this complex environment. There are too many points of failure that can have catastrophic consequences to the Court's evidence. What are some of these dynamic elements that preclude software protection? That is what we will now present.

We will first look at things that can go wrong. We will then end by looking at what we call “Administrative Reasons” for not using software write blocking. Again, we are not concerned about the system when everything is working right, but we are very concerned about the many ways that things can go wrong.

4.0 Problems

Things that can go wrong

The first thing we want to establish is a point that everyone should be able to agree with. If you are using software write blocking, the device to be protected is not protected until;

1. The computer boots correctly.
2. The Operating System (OS) is loaded and operating correctly.
3. The write blocking software is loaded.
4. The write blocking software is operational.

While there may be some overlap in these steps, all of them are necessary for software write blocking to have a chance at being effective. Until all these steps have completed and are operating correctly the evidence is exposed. With that established, we will now show why the concept of software write blocking is a fallacy.

Let's look at the way a system boots up and specifically the system's BIOS. A software write block may be heavily dependent on a particular BIOS. Did the manufacturer of the write block test different BIOSes with their software? Any change to a system's BIOS, such as a firmware upgrade using onboard FLASH, may also affect the operation of a software write block. Every system has a different BIOS, a company X BIOS in a Gateway may be different from the company X BIOS in a Dell. In fact it could vary by model number and even within the same model. Also different versions of BIOSes could vary greatly on their startup and operational characteristics. In addition to the system's BIOS there are BIOSes that load from peripheral cards, such as SCSI, IDE controller cards or SATA controller cards. What is the effect of this on a software write block? Has your supplier tested it? What version did they test? Do you know what your system does?

Remember, until the OS and then write blocking software loads and is operational your media is unprotected, and even after that it may be unprotected.

Also, in this pre OS loading time frame, any number of boot errors can occur, including a failure to load off the forensic drive or CD. A change or error in the system BIOS could cause the system to load off an attached drive. Some of the newer BIOSes can boot not only from standard drives such as IDE, but also from external USB devices. Since the BIOS has code to both read and write to any attached device, any kind of a glitch in the boot sequence can have catastrophic results to an evidence drive. Even more troubling is that modern motherboards have the ability to boot from a network source. Couple this with the proliferation of wireless devices, such as Bluetooth, WiFi, and ZigBee, and it is getting increasingly more difficult to have full and true knowledge of the state of a machine.

Any software write block will be heavily dependent on the OS in which it will be used. Care must be taken to insure that the OS you have is compatible and that the manufacture of the write blocking software has extensively and correctly tested it. Software written for one version of the OS may not work with another version or even a different build and/or Service Pac. Something as seemingly innocuous as a Windows Update may render the software write blocker ineffective. In addition, care must also be taken to insure that the OS was installed with all required components and that no errors occurred during installation. Even an OS that purports to have the ability to “mount read only” can write to a drive (Reiser and EXT3 in Linux is a good example, it will write to the drive to change the Journal Count). Do you “know” what your OS and FS (file system) are really doing?

Application software can have catastrophic effects on software write blockers, both in the installation and the operation. Applications can load drivers that can supersede your OS files, thereby making changes to the OS, or work around the normal operations of the OS. Since device drivers affect the stability of the OS and may have direct access to the PCs hardware, it is important to know of any interactions. How is this to be tested? Do you know what all your applications are doing? Are they all working correctly? Given that once a program can access an I/O port, it can write to a drive directly, can there ever be any true write protection in software alone? All major operating systems have a mechanism to allow direct I/O programming. It is this method that allows OS device drivers talk with the underlying hardware. If the OS has the ability to write but is relying on a “bit” to tell it where or where not to write, what if the bit is wrong?

There is also the integrity of write blocking drivers. Is the driver using the correct code? Was the installation proper and correct? Did the code take a "hit" on your drive? Is every bit correct? Was the code designed correctly? Is the source code available, and if it were, how would you know what to look for?

Then there is the hardware that needs to be controlled. Is it working correctly? Seemingly minor problems can cause software to install incorrectly or operate erratically.

Unforeseen failures and problems that can occur (aka… just plain bad luck)
Let's look at a list of possible failures that can occur that could alter the system and affect the media. Remember, until the operating system and write blocking drivers are loaded and working correctly, your attached media is completely exposed to harm. In fact even after the OS is loaded and drivers operating, it can be exposed to harm.

How many times have you experienced your Windows system crashing or rebooting due to power glitches or other unexplainable reasons. How often have you started to image a drive and walked away due to the extended amount of time it takes? If at any time during this process the system glitches and reboots, the media is unprotected. If the media contains evidence collected as part of an investigation, all the hard work in obtaining that evidence may be wasted in a millisecond.

For another example, suppose that we have a software write blocker that loads into windows and protects the USB ports from writes. The trouble is, you are going to use a USB to IDE tailgate device to connect that IDE drive to the USB port. What is the tailgate device doing and what can it do. Does it have the ability to pass a write to the drive? If so, what prevents it from doing so unintentionally? Beyond the tailgate is the software itself. What method do they use to determine when a write is occurring verses a read? Does it really block all ways to write to the device? Given that data has to be passed from the computer to the tailgate to instruct it what to do, what happens if the command is misinterpreted? If the tailgate device doesn't have a form of write protection, it can cause a write.

Other unforeseen problems that can occur include;

Corrupted BIOS caused by

- BIOS Flash failure
- A re-flash of the BIOS code is done with the wrong code
- BIOS flash gets corrupted (virus, accident etc.)
- Hardware failure in reading the BIOS
- The BIOS flash for an update is incompatible with old code

Motherboard Failures

- IDE channel failure
- IDE connecting cable failure (A common cause of drive failures)
- Drive power cable failure (and yes I have one that causes a drive to act errantly)
- Special drivers from motherboard manufacturer used that supersede the software blocking drivers.
- USB or Firewire port failure
- Cable failure
- Incorrect boot device used (CD-ROM/NET/Wrong drive)

Miscellaneous problems

- Drive dock/sled failure
- Additional device controller failure (i.e. Promise cards, etc…)

Both drives set to master
Drive jumper setting failure or drive reads jumpers incorrectly
Incorrect drive label causes incorrect jumper settings
Human error
Just plain bad luck coupled with Murphy issues.

OS problems

Unknown design flaws (for forensics). Examples are the Linux Reiser and EXT3 file systems that will write to the drive even if mounted read only. It will update a journal log on the mounted read only drive.

Corrupted drivers

Drivers experience corruption while installing
Reading failures of drivers during startup cause unpredictable results
Change of drivers via Microsoft Service Pac installation
Change of drivers via Microsoft Windows Update installation
Slave/Master bit reading failure
Driver incompatible with a specific build of the kernel

RAM failures

A read error occurs in RAM disabling write blocking or corrupting OS

Operator errors

Write Block software not run or installed correctly
A program has run that bypasses write block software
Wrong boot sequence is set in the BIOS
Wrong boot device is used or the boot device fails causing a boot to the suspect media.
External device accidentally left connected during boot

File system problems

Corruption (virus)
Corruption of software

In a recent class in computer forensics put on by the California Department of Justice Advance Training Center, there was a slide rack that caused SUSE Linux to not install correctly. This was not seen until after the OS was loaded and operating for a few minutes, at which time the keyboard would constantly fail. It was traced back to corrupted keyboard drivers caused by the slide rack. (It was repeatable using the slide rack on different systems.)

A national bank recently experienced some of their ATM machines randomly rebooting and then failing to load the ATM application. The machines were then showing the windows desktop with a live keypad.

Administrative problems
The simplest reason against software write blocking is an administrative reason that is financially based. Given the multiple points of failure that can occur, why risk it? A single failure could more than eliminate the saving you incurred by not having hardware media protection.

In addition, the first thing a defense council or opposing party should and will ask for are your maintenance logs and operation logs for the computer used with the original media. They should also ask to see the forensic computer you used to verify the state of the hardware and the software against the logs.

This means there will be an extra expense for management and proactive auditing to insure your logs are correct and personnel are following the operational guidelines. There will also be the added expense for the time it takes in maintaining those logs and the testing of equipment after any change.

But let's say you use software and damn the torpedoes. The cost in employee time to defend your practice in court will far exceed the cost of buying a hardware media protection device. Also if you plan on using software write protection, what is your test plan? Who will put it together and implement it.

Why risk it? Your reputation is on the line. The Court's evidence is on the line. Any possible short-term savings are more than offset by the long-term risks and costs associated with software write protection.

5.0 Conclusion

Conclusion
Systems that are designed to write but rely on some type of control system to prevent a write can experience a failure of the controlling system. A computer system using write blocking software can fail and write to the drive. Media protection devices are systems that are designed not to write and thus have no controlling system to fail.

Preserving the Court's evidence, an original hard drive in most cases, is your primary concern. The greatest exposure to unrecoverable failure occurs during the duplication process. Given the uncontrollable nature of the systems we use, the safest and least expensive route is to use hardware media protection. You can certify a controllable device but not a dynamic system.