All.Net Analyst Report and Newsletter

Welcome to our Analyst Report and Newsletter

Countering hardware storage device Trojans

Recent revelations from a wide range of sources indicate that you can no longer reasonably trust your hardware storage devices. In essence, SD, USB, SSD, HDD, and many other controllers use general purpose processors to mediate between their external interface and the physical storage devices. This is necessary in order to compensate for hardware faults, which are the result of the tradeoff between low-level reliability and volume for price. By building large amounts of low quality components for far less per bit in cost, we get enormous amounts of unreliable memory which act as reliable memory of smaller size through the use of software on the controller that uses redundancy in storage and wear leveling to provide more reliable, faster, and larger storage than would be available for the same price otherwise.

Why programmable?

With the programmable micro-controller comes the ability and necessity to update the code that runs it. This is necessary because it is less expensive to send out imperfect firmware and fix it later, and because it is less expensive to allow programming than to burn the code into the hardware because the same controller can be used across many product lines and just be reprogrammed for different storage devices, which change technologically over time.

And the security implications?

With general purpose programmability comes general purpose function, allowing the device that appears to be just memory to act as an arbitrary programmable mechanism with its own storage and analytical capability. That means that when you send something to storage, a maliciously coded micro-controller can alter it, refuse to return it, encrypt it, store it, etc. If you use the device to store information that you interpret later, the micro-controller could add code to software, alter content of spreadsheets, combine information from different sources to, as an example leak confidential information by encoding it in less confidential information, and so forth. If the device is used for cache memory, temporary files in compilation, to store the operating system, libraries, other similar code, or other similar uses, this could make arbitrary changes to the operating environment producing unlimited effects and full control to a malicious actor. In other words, if I try, you lose.

What can we do about it?

It may seem like a lost cause, but it's not. Notionally, one approach that makes this far harder for the potential attacker, is to encrypt and authenticate everything sent to storage. In practice you need to start with some trustworthy storage that allows you to boot the system or provide detection for untrustworthy startup. This harkens back to the old days where doing a secure boot to defeat low-level viruses was analyzed and the need for trusted channels to storage was identified. Nothing is all that new here.[1] Cryptographic keys require secure storage and execution, so encryption and decryption should really happen in a separate hardware device.

1 At http://all.net/books/integ/vmodels.html you can find “Models of Practical Defenses Against Computer Viruses” (1988) and at http://all.net/books/integ/bootstrap.html you can find “A Note On High Integrity PC Bootstrapping“ (1991).

Another approach is microzoning detailed recently in related articles,[2] and now part of our standard of practice.[3] In essence, this uses encrypted storage and transport with different keys for each of many microzones and storage, computation, and communication separated from each other. Virtual machines are invoked for each segregated content set, with the net effect being that once a VM is brought (back) up in its permanent initial state, altering stored content will cause failure of filesystems to mount, content to be used, encrypted tunnels to form, etc.

Another approach, at least for storage, is to take apart each of the memory devices you get and verify that they work as you think, perhaps including adding hardware to verify that changes don't occur between storage and presentation during operation. Of course you then have the problem of trusting the mechanisms you use for this purpose, and the fact that this likely drives cost up so high that you cannot afford the certainty. You may as well build your own... but then you have to trust your insiders...

A few key points to make

All of this may sound great, but in fact it is problematic in various ways. There are two fundamental issues that ultimately need to be addressed:

- You need to somehow get a trustworthy environment operating somewhere in order to leverage it for creating the rest of a trustworthy infrastructure.
- Redundancy can support detection of alteration except in the case of common mode failures, but this is also expensive, slower, and hard at higher levels of operation.

And the problem still isn't solved

In the end, work-arounds for this challenge can be made, and we use these sorts of things all the time. But that doesn't change the underlying issue. The real problem is that now you and everyone else is starting to find out that they cannot trust government, the manufacturers, the software companies, the cloud providers, or anyone else to deliver products or services that you can count on. At their sole discretion they may, and in many cases have, provided back doors into everything they provide to support their own greed or the will of those more powerful than they. Whether under threat or payment, regardless of where they are or who created them, we simply cannot trust them, and they apparently will not trust us.

Whether it is Microsoft surreptitiously deleting software from millions of computers through their automated update process, Apple removing installed software as part of their major updates, Cisco routers with back doors, Google infrastructure broken into by the NSA, Amazon removing books from Kindle platforms, hardware manufacturers altering code or adding disablement functions to their devices, the NSA intercepting shipments and planting Trojan horses, France requiring keys to encrypted content, or any of the long long list of other parties at the table does not matter. The truth is, you simply cannot trust the mechanisms.

Summary

What we have here is a global failure of trust brought about by a global effort to weaken trustworthiness. They try to keep it secret or tell us it's for our own good, but it often backfires. The only real hope today is to find a way to regain trust by composing untrusted components.

2 http://all.net/Analyst/2013-03-13-MicroZoning.pdf
3 http://all.net/SoP/SecDec/index.html and elsewhere on all.net.

