Intel Processor Based Embedded Systems Cyber Security

Author: Boris Baer, Aitech R&D Manager

January 2017

Contents

1. Introduction

In the emerging world of embedded systems, as autonomous platforms (UAVs, SUAVs, etc) are being used for defense and aerospace, there is growing interest in advanced means to secure these systems and related information from menacing intruders.

This article provides an overview of possible security threats for embedded systems utilized in defense/aerospace, and also covers available techniques for securing these systems and the information that they contain.

The defense and aerospace sector is subject to a variety of security threats through all phases of the life cycle of an embedded system. This starts with cyber-attacks to gain access to vendors' servers and steal design information, continues with efforts to take control of the embedded systems or retrieve classified information from the aircraft, UAVs, SUAVs, etc. during their infield deployment, and, in case of crashed UAV or loss of control, extract critical information from or reverse-engineer/clone different hardware and software elements of the system.

2. Protection of Firmware and Software

2.1. Authenticating Software

In order ensure security of embedded systems, care should be taken to authenticate all system firmware and software.

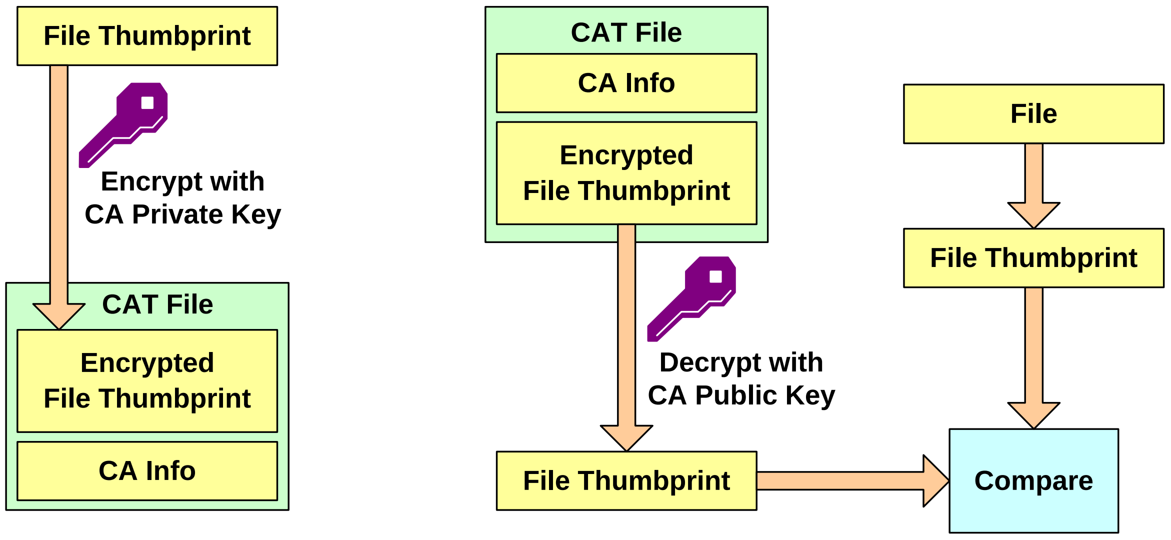
The BIOS, OS Loader and OS can be authenticated using a Trusted Platform Module (TPM) along with Intel's Trusted Execution Technology (TXT). TXT uses the TPM to generate cryptographic hashes of the code prior to loading, to verify that it has not been altered or corrupted.

In addition to authenticating the BIOS, it is also important to verify that all additional software installed in the embedded system is authentic and has not been tampered with. Authentication of software can be performed using digital signatures, which can identify the software vendor and show that the files have not been modified after signing/publication.

We can look, for example, at the process in which a Windows OS device driver is verified using a digital signature.

The driver package is signed by a Certificate Authority (CA) that has been approved by Microsoft. The CA signs the driver by generating a thumbprint (cryptographic hash) and encrypting the thumbprint using the CA's private key. The encrypted thumbprint is then embedded into the driver's CAT file.

To verify the driver, the encrypted thumbprint is extracted from the CAT file before the driver is installed, and the thumbprint is decrypted using the CA's public key. The decrypted thumbprint is then compared against a locally generated thumbprint of the driver file.



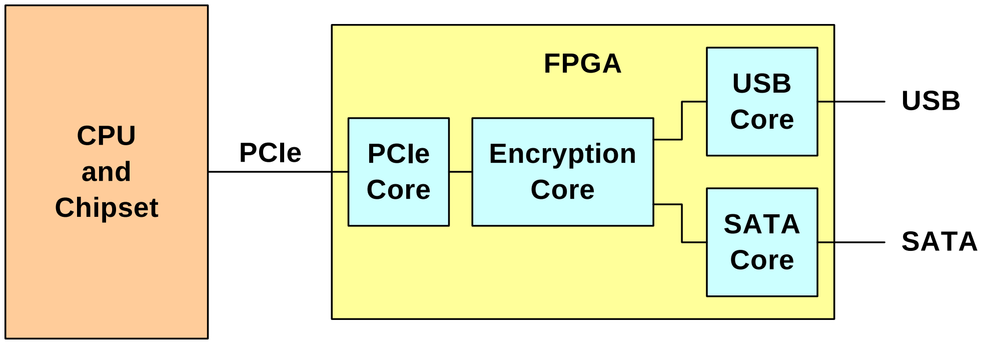
2.2. Non-Authorized External Media Recognition

Although we normally think of malware as being transmitted via the internet, transmission via removable storage media is also possible. Allowing external storage media to be used also introduces the risk that critical information can be taken off site.

For these reasons, it is often desirable to prevent the usage of external storage media on the machines that are used for the development and control of critical systems.

The easiest methods use software. The Windows OS, for example, has several ways to restrict usage of external storage devices. Group policies can be configured to prevent read and write access to various types of removable media. Group policies can also be configured to prevent usage of external media by blocking users from using removable storage devices at the driver level.

An alternate method relies on hardware rather than the OS. USB and SATA controllers are normally embedded within the CPU chipset and drivers are supplied by the OS vendor, leaving the end user with limited control. In order to prevent unauthorized media from introducing malware and from leaving the site with sensitive data, a custom USB or SATA interface could be implemented using an FPGA. The FPGA encrypts and decrypts all reads and writes to the ports without relying on software. In case a non-authorized device is installed in one of the ports, the data on the device will pass through the decryption mechanism and the device will appear to contain only random data (any data on the device will be "decrypted" and will appear to be garbage). Alternately, if critical data are copied to the device, the data will be encrypted and will be unreadable on other machines.



2.3. Protection of Storage

Write Protection

Write-protection can be valuable in multiple ways, in both workstations and embedded systems. We can use write-protection to protect software from being altered or corrupted, and can also use write-protection to prevent sensitive data from being written to non-volatile storage.

Write filtering can be implemented using software. For example, the Windows embedded OS offers the Enhanced Write Filter (EWF) tool, which places an overlay layer over the selected volume, and writes changes only to the overlay. This makes the volume appear to be writeable to applications without actually altering the contents of the volume. EWF allows the user to commit the changes in the overlay to the volume, or to simply discard the changes at reboot if a commit is not performed.

Hardware based write-protection prevents all writing/erasing of non-volatile memory. Hardware based write-protection can be enabled or disabled at the device level by toggling the write protection control bits of the device's internal registers, or by toggling the device's write protection control input lines via jumpers or via a control line from another on-board device.

Data Encryption

Encryption is another tool that can be used to protect data in both workstations and embedded systems. Regardless of other security measures that may be taken, without encryption, someone with physical access to your system can simply remove a storage device and insert it into another system to gain full access to its contents.

Encrypting your data will provide protection. When the data have been encrypted, the data will appear to be random garbage and will not be accessible without the encryption key. Encryption keys can be generated from passwords, or can be stored on a USB drive or TPM (Trusted Platform Module).

Encryption can be applied to specific files and directories on an individual basis, or it can be applied to a full disk, as well as, BIOS device so that the entire contents of the flash memory will be encrypted.

Full disk encryption is performed transparently – data are automatically encrypted/decrypted during writes/reads to the disk. Files are accessible when the key is provided and the encrypted volume is mounted (typically as if it were a physical drive). After the volume is unmounted the data will be inaccessible without remounting the volume with the correct encryption key.

Even if your data are encrypted, there are several attacks that may be used to attempt to bypass the encryption. These attacks include malware that is designed to capture the keys and passwords entered by the user, or to read the encryption keys from system memory.

Another attack relies on physical access to the machine rather than malware. This attack takes advantage of the fact that the contents of RAM can remain intact for a brief period after power is removed, and attempts to read the keys from system memory by performing a cold boot (reboot without clearing of RAM) and quickly rebooting from a drive that will dump the contents of RAM to non-volatile memory, which can later be searched for the encryption key.

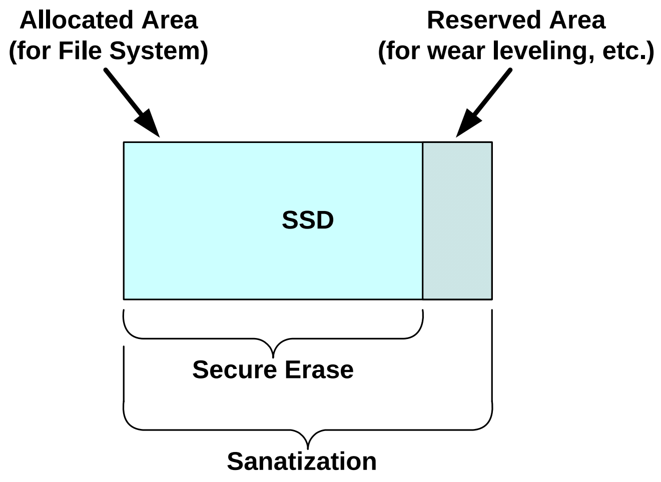
Secure Erase/Sanitization

With wide-spread usage of high-capacity flash memory devices, in both embedded systems and in workstations, it is important to understand the available methods of erasing flash storage.

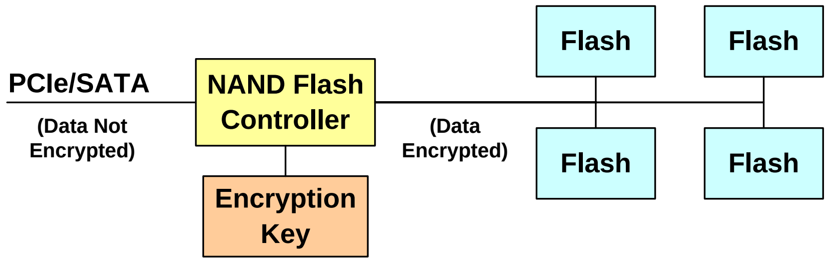
Secure erase is a feature supported by most SSDs, which clears data from blocks that are allocated as volume sectors.

Sanitization is a more powerful tool, which clears data from all flash memory devices on the SSD (not only the blocks allocated as volume sectors, but also reserved areas used for functions such as wear leveling).

Usually, performing multiple write and erase operations directly on flash devices makes data recovery effectively impossible.



For SSDs with integrated/automatic encryption, deleting the encryption key is sufficient to securely erase the disk. The encrypted data would remain on the disk, but would be inaccessible without the encryption key.



Physical destruction of flash storage can be accomplished by applying a high voltage directly to the flash devices for 3-5 seconds. This will physically destroy the devices and make them no longer usable. No data can be recovered.

Physical destruction of the flash storage may be used for an extra layer of security when discarding old hardware (after first erasing the disk as described above), and can also be used along with tamper detection circuitry to perform quick destruction of data in an embedded system that falls into the wrong hands.

For example, a tamper detection mechanism could connect a capacitor bank to the flash devices to destroy the devices within seconds of a tamper detection event.

3. Conclusion

Today's embedded systems face multiple threats throughout all phases of their lifecycles, requiring designers and operators of the systems to build robust defenses that address these threats, with multiple-layers of protection, at all stages of the lifecycle – from development to deployment to end of life. In the case of autonomous systems such as UAVs, protections should include contingencies for protection of data in systems that may fall into the wrong hands during deployment.

The security of systems that are used in critical industrial and military/aerospace applications can have real world implications that can even impact public safety. A comprehensive multi-layer protection strategy should be established from the early planning stages of a project, and carefully implemented throughout the entire project lifecycle.