Full Drive Encryption Security Problem Definition - Encryption Engine

Introduction for the FDE Collaborative Protection Profiles (cPPs) Effort

The purpose of the first set of Collaborative Protection Profiles (cPPs) for *Full Drive Encryption (FDE): Authorization Acquisition (AA)* and *Encryption Engine (EE)* is to provide requirements for Data-at-Rest protection for a lost device. These cPPs allow FDE solutions based in software and/or hardware to meet the requirements. The form factor for a storage device may vary, but could include: system hard drives/solid state drives in servers, workstations, laptops, mobile devices, tablets, and external media. A hardware solution could be a Self-Encrypting Drive or other hardware-based solutions; the interface (USB, SATA, etc.) used to connect the storage devices to the host machine is outside the scope.

Full Drive Encryption encrypts all data (with certain exceptions) on the storage device and permits access to the data only after successful authentication to the FDE solution. The exceptions include the necessity to leave a portion of the storage device (the size may vary based on implementation) unencrypted for such things as the Master Boot Record (MBR) or other AA/EE pre-authentication software. These FDE cPPs interpret the term “full drive encryption” to allow FDE solutions to leave a portion of the storage device unencrypted so long as it contains no user or authorization data.

Since the FDE cPPs support a variety of solutions, two cPPs describe the requirements for the FDE components shown in Figure 1.

The *FDE cPP - Authorization Acquisition* describes the requirements for the Authorization Acquisition piece and details the necessary security requirements and assurance activities necessary to interact with a user and result in the availability of a data encryption key (DEK).

The *FDE cPP - Encryption Engine* describes the requirements for the Encryption Engine piece and details the necessary security requirements and assurance activities for the actual encryption/decryption of the data by the DEK. Each cPP will also have a set of core requirements for management functions, proper handling of cryptographic keys, updates performed in a trusted manner, audit and self-tests.

Figure 1: FDE Components

This Security Problem Definition (SPD) defines the scope and functionality of the Encryption Engine as well as the assumptions made about the operating environment and the threats to the EE that the cPP requirements address.

Implementations

Full Disk Encryption solutions vary with implementation and vendor combinations.

Therefore, vendors will evaluate products that provide both components of the Full Disk Encryption Solution (AA and EE) against both cPPs. A vendor that provides a single component of a FDE solution would only evaluate against the applicable cPP. The FDE cPP is divided into two documents to allow labs to independently evaluate solutions tailored to one cPP or the other. When a customer acquires an FDE solution, they will either obtain a single vendor product that meets the AA + EE cPPs or two products, one of which meets the AA and the other of which meets the EE cPPs.

The table below illustrates a few *examples* for certification.

Table 1: Examples of cPP Implementations

Implementation	cPP	Description
Host	AA	Host software provides the interface to a self-encrypting drive
Self Encrypting Drive (SED)	EE	A self-encrypting drive used in combination with separate host software
Software FDE	AA + EE	A software full drive encryption solution
Hybrid	AA + EE	A single vendor's combination of hardware (e.g. hardware encryption engine or cryptographic co-processor) and software

Target of Evaluation (TOE) Description

The target of evaluation for this cPP is either the Encryption Engine or a combined evaluation of the set of cPP's for FDE (Authorization Acquisition and Encryption Engine).

The following sections provide an overview of the functionality of the FDE EE cPP as well as the security capabilities.

Encryption Engine Introduction

The Encryption Engine cPP objectives focus on data encryption, policy enforcement, and key management. The EE is responsible for the generation, update, archival, recovery, protection, and destruction of the DEK and other intermediate keys under its control. The EE receives a key from the AA. The EE uses that key either for the release or the decryption of the DEK, though other intermediate keys may exist in-between those two points. Key encryption keys (KEKs) wrap other keys, notably the DEK or other intermediary keys which chain to the DEK. Key releasing keys (KRKs) authorize the EE to release either the DEK or other intermediary keys which chain to the DEK. These keys only differ in the functional use.

The EE determines whether to allow or deny a requested action based on the KEK or KRK provided by the AA. Possible requested actions include but are not limited to changing of encryption keys, decryption of data, and cryptographic erase of encryption keys (including the DEK). The EE may offer additional policy enforcement to prevent access to ciphertext or the unencrypted portion of the storage device. Additionally the EE may provide encryption support for multiple users on an individual basis.

Figure 2 illustrates the components within EE and its relationship with AA.

Encryption Engine Security Capabilities

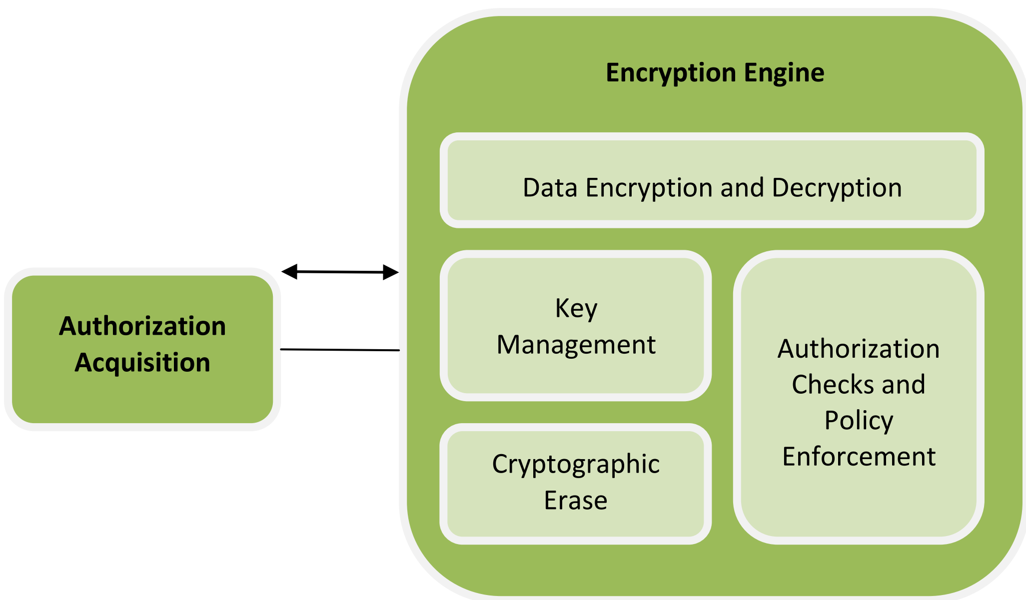

Figure 2: Encryption Engine Details

The Encryption Engine is ultimately responsible for ensuring that the data is encrypted using a prescribed set of algorithms. The EE manages the authorization for using DEKs to decrypt the data on the storage device through decryption or release of the DEK. It also manages the authorization for administrative functions, such as changing the DEK, setting up users, managing the authorizations required for decrypting or releasing the DEK, managing the intermediate wrapping keys under its control and performing a cryptographic erase.

The EE may provide key archiving and recovery functionality. The EE may manage the archiving and recovery itself, or interface the AA to perform this function. It may also offer configurable features, which restricts the movement of keying material and disables recovery functionality.

The foremost security objective of encrypting storage devices is to force an adversary to perform a cryptographic exhaust against a prohibitively large key space in order to recover the

DEK or other intermediate keys. The EE uses approved cryptography to generate, handle, and protect keys to force an adversary who obtains an unpowered lost or stolen platform without the authorization factors or intermediate keys to exhaust the encryption key space of intermediate keys or DEK to obtain the data. The EE randomly generates DEKs and intermediate keys. The EE uses DEKs in a symmetric encryption algorithm in an appropriate mode along with appropriate initialization vectors for that mode to encrypt sectors on the storage device. The EE either encrypts the DEK with a KEK or an intermediate key.

The TOE and the Operational/Pre-Boot Environments

The environment in which the EE functions may differ depending on the boot stage of the platform in which it operates, see Figure 3. Aspects of provisioning, initialization, and perhaps authorization may be performed in the Pre-Boot environment, while encryption, decryption and management functionality are likely performed in the Operating System environment.

In the Operating System environment, the Encryption Engine has the full range of services available from the operating system (OS), including hardware drivers, cryptographic libraries,

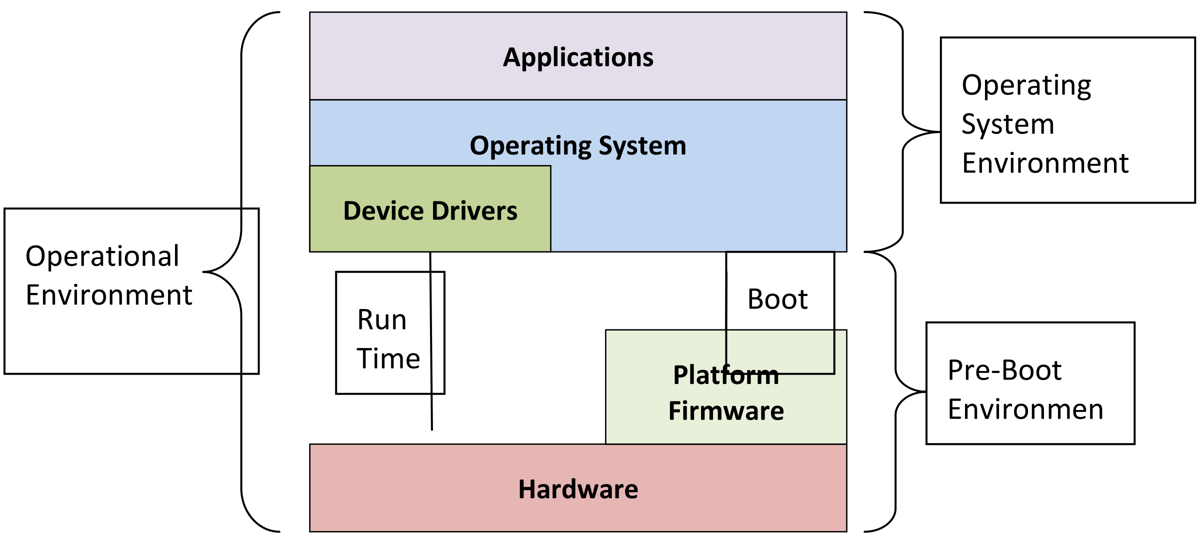

Figure 3: Operational Environment

and perhaps other services external to the TOE.

The Pre-Boot environment is much more constrained with limited capabilities. This environment turns on the minimum number of peripherals and loads only those drivers necessary to bring the platform from a cold start to executing a fully functional operating system with running applications.

The EE TOE may include or leverage features and functions within the operational environment.

Functionality Deferred until the Next cPP

Due to time constraints, this SPD defers requirements for some important functionality until the next version of the cPP. These include requirements for partition/volume management, remote management, and power management (requirements for power state protection).

Threats

This section provides a narrative that describes how the requirements mitigate the mapped threats. A requirement may mitigate aspects of multiple threats. A requirement may only mitigate a threat in a limited way.

A threat consists of a threat agent, an asset and an adverse action of that threat agent on that asset. The threat agents are the entities that put the assets at risk if an adversary obtains a lost or stolen storage device. Threats drive the functional requirements for the target of evaluation (TOE). For instance, one threat below is T.UNAUTHORIZED_DATA_ACCESS. The threat agent is the possessor (unauthorized user) of a lost or stolen storage device. The asset is the data on the storage device, while the adverse action is to attempt to obtain those data from the storage device. This threat drives the functional requirements for the storage device encryptor (TOE) to authorize who can use the TOE to access the hard disk and encrypt/decrypt the data. Since possession of the KEK, DEK, intermediate keys, authorization factors, submasks, and random numbers or any other values that contribute to the creation of keys or authorization factors could allow an unauthorized user to defeat the encryption, this SPD considers keying material equivalent to the data in importance and they appear among the other assets addressed below.

It is important to reemphasize at this point that this Collaborative Protection Profile does not expect the product (TOE) to defend against the possessor of the lost or stolen hard disk who can introduce malicious code or exploitable hardware components into the Target of Evaluation (TOE) or the Operational Environment. It assumes that the user physically protects the TOE and that the Operational Environment provides sufficient protection against logical attacks. One specific area where a conformant TOE offers some protection is in providing updates to the TOE; other than this area, though, this PP mandates no other countermeasures. Similarly, these requirements do not address the “lost and found” hard disk problem, where an adversary may have taken the hard disk, compromised the unencrypted portions of the boot device (e.g., MBR, boot partition), and then made it available to be recovered by the original user so that they would execute the compromised code.

(T.UNAUTHORIZED_DATA_ACCESS)

The cPP addresses the primary threat of unauthorized disclosure of user data stored on a storage device. If an adversary obtains a lost or stolen storage device (e.g., a storage device contained in a laptop or a portable external storage device), they may attempt to connect a targeted storage device to a host of which they have complete control and have raw access to the storage device (e.g., to specified disk sectors, to specified blocks).

(T.KEYING_MATERIAL_ COMPROMISE)

Possession of any of the keys, authorization factors, submasks, and random numbers or any other values that contribute to the creation of keys or authorization factors could allow an unauthorized user to defeat the encryption. The cPP considers possession of keying material of equal importance to the data itself. Threat agents may look for keying material in unencrypted sectors of the storage device and on other peripherals in the operating environment (OE), e.g. BIOS configuration, SPI flash, or TPMs.

(T.AUTHORIZATION_GUESSING)

Threat agents may exercise host software to repeatedly guess authorization factors, such as passwords and pins. Successful guessing of the authorization factors may cause the TOE to release DEKs or otherwise put it in a state in which it discloses protected data to unauthorized users.

(T.KEYSPACE_EXHAUST)

Threat agents may perform a cryptographic exhaust against the key space. Poorly chosen encryption algorithms and/or parameters allow attackers to brute force exhaust the key space and give them unauthorized access to the data.

(T.KNOWN_PLAINTEXT)

Threat agents know plaintext in regions of storage devices, especially in uninitialized regions (all zeroes) as well as regions that contain well known software such as operating systems. A poor choice of encryption algorithms, encryption modes, and initialization vectors along with known plaintext could allow an attacker to recover the effective DEK, thus providing unauthorized access to the previously unknown plaintext on the storage device.

(T.CHOSEN_PLAINTEXT)

Threat agents may trick authorized users into storing chosen plaintext on the encrypted storage device in the form of an image, document, or some other file. A poor choice of encryption algorithms, encryption modes, and initialization vectors along with the chosen plaintext could allow attackers to recover the effective DEK, thus providing unauthorized access to the previously unknown plaintext on the storage device.

(T.PERSISTENT_INFORMATION)

As a courtesy to the user, the TOE and/or the Operational Environment goes into a power saving mode in which it leaves the data or key material unencrypted in persistent memory to facilitate a speedy recovery upon powering up. Threat agents look for unencrypted keying material and data giving them unauthorized access to data.

(T.UNAUTHORIZED_UPDATE)

Threat agents may attempt to perform an update of the product which compromises the security features of the TOE. Poorly chosen update protocols, signature generation and verification algorithms, and parameters may allow attackers to install software and/or firmware that bypasses the intended security features and provides them unauthorized to access to data.

Assumptions

Assumptions that must remain true in order to mitigate the threats appear below:

(A.TRUSTED_CHANNEL)

Communication among and between product components (e.g., AA and EE) is sufficiently protected to prevent information disclosure. In cases in which a single product fulfills both cPPs, then it assumes that the communication between the components does not breach the boundary of the TOE. In cases in which independent products satisfy the requirements of the AA and EE, the physically close proximity of the two products during their operation means that the threat agent has very little opportunity to interpose itself in the channel between the two without the user noticing and taking appropriate actions.

(A. INITIAL_DRIVE_STATE)

Users enable Full Drive Encryption on a newly provisioned or initialized storage device free of user data. The cPP does not intend to include requirements to find all the areas on storage devices that potentially contain user data. In some cases, it may not be possible - for example, data contained in “bad” sectors.

While inadvertent exposure to data contained in bad sectors or un-partitioned space is unlikely, one may use forensics tools to recover data from such areas of the storage device. Consequently, the cPP assumes bad sectors or un-partitioned space contains no user data.

(A_AUTHORIZED_USER)

Authorized users follow all provided user guidance, including keeping password/passphrases and external tokens securely stored separately from the storage device and/or platform.

(A.PLATFORM_STATE)

The platform in which the storage device resides (or an external storage device is connected) is free of malware that could interfere with the correct operation of the product.

(A.MEMORY_REMNANCE)

The user does not leave the platform and/or storage device unattended until FDE solution clears all volatile memory after a power-off, so memory remnant attacks are infeasible.

Authorized users do not leave the platform and/or storage device in a mode where sensitive information persists in non-volatile storage (e.g., Lockscreen). Users power the platform and/or storage device down or place it into a power managed state, such as a “hibernation mode”.

(A.STRONG_CRYPTO)

All cryptography implemented in the Operational Environment and used by the product meets the requirements listed in the cPP. This includes generation of external token authorization factors by a RBG.

(A.TRAINED_ADMINS)

Authorized administrators received appropriate training and follow all administrator guidance.