25C3: Full-Disk-Encryption Crash-Course
Everything to hide

Jürgen Pabel, CISSP
Akkaya Consulting GmbH

#0 Abstract

Crash course for software based Full-Disk-Encryption.

The technical architectures of full-disk-encryption solutions for Microsoft Windows and Linux are described. Technological differences between the two open-source projects TrueCrypt and DiskCryptor are explained. A wish-list with new features for these open-source Full-Disk-Encryption offerings wraps things up.

Readers should be familiar with general operating system concepts and have some software development experience.

#1 Introduction

Software based Full-Disk-Encryption („FDE“) solutions are usually employed to maintain the confidentiality of data stored on mobile computer systems. The established terminology is „data-at-rest protection“; it implies that data is protected against unauthorized access, even if an adversary possesses the storage media. Therefore, some of the most common risks to data confidentiality are addressed: loss and theft of mobile computer systems.

The encryption and decryption of data is completely transparent to computer users, which makes Full-Disk-Encryption solutions trivial to use. The only user interaction occurs in the Pre-Boot-Authentication („PBA“) environment, in which the computer user provides the cryptographic key in order to allow for the operating system to be started and data to be accessed.

#2 Architecture

I: Pre-Boot-Authentication

The Pre-Boot-Authentication environment is loaded before the operating system when the computer is powered-on. The primary purpose of the Pre-Boot-Authentication is to load the encryption key into memory and to initiate the boot process of the installed operating system. The following methods are some common examples of how the encryption key may be provided:

- by entering a password,
- by inserting a token (Smartcard, USB token, ...) or
- by loading a cryptographic key from a TPM chip.

Perhaps the most interesting technical aspect of the Pre-Boot-Authentication environment is the question about its storage location: if the entire disk is encrypted, how is the Pre-Boot-Authentication loaded? An obvious answer is to store the Pre-Boot-Authentication on an unencrypted media, like a bootable USB stick. However, booting from a third media imposes a physical dependency and it is therefore very common to store the Pre-Boot-Authentication in:

- a dedicated (unencrypted) boot partition,
- unallocated (not assigned to any partition) or „hidden“ sectors (Host-Protected-Area),
- sectors assigned to a partition but not bound to a filesystem (ie: filesystem was shrunk),
- an encrypted filesystem, but (only) sectors containing the PBA are left unencrypted or
- a designated boot area on the filesystem (if supported by the filesystem, like NTFS [0]).

The established approach on Linux is to use a dedicated boot partition (usually mounted as /boot), which is left unencrypted and loads the Linux kernel. An accompanying RAM-disk contains the Pre-Boot-Authentication environment, which reads in the encryption key and mounts the encrypted filesystem(s) before the boot process continues as usual (init process). Technically, this approach does not strictly meet the term „Pre-Boot-Authentication“ because the Linux kernel is already booted and used to obtain the cryptographic key.

Microsoft Windows requires a „real“ Pre-Boot-Authentication environment due to its boot architecture: a (usually rather small and product-specific) Pre-Boot-Authentication environment is loaded from any of the aforementioned storage locations instead of the NT boot loader (NTLDR). The Pre-Boot-Authentication environment reads in the encryption key, hooks interrupt 13h (INT13)[1] with a cryptographic function in order to allow transparent disk access via INT13, before passing control to NTLDR for loading the Microsoft Windows kernel (NTLDR uses INT13 for disk access). NTLDR loads the Microsoft Windows kernel, including all configured drivers, before the execution control is given to the kernel.

II: Encryption driver

Microsoft Windows disk drivers implement disk access without using INT13, therefore all encryption and decryption of data must occur within the kernel. The device driver architecture for Microsoft Windows includes a component called the „lower filter driver“ [2]. Encryption drivers for Microsoft Windows are usually registered as a lower filter driver for the disk drivers. The encryption driver initializes itself by copying the encryption key used by the INT13 hook function.

The mechanism of choice among Linux distributions for Full-Disk-Encryption is via the „device mapper“ [3] subsystem: a virtual device is mapped over the target device. The cryptographic algorithm is implemented in the virtual device driver and delegates all I/O operations to the underlying device driver. Loopback drivers and cryptographic filesystems are other alternatives for disk encryption on Linux. Yet another approach for implementing disk encryption on Linux would be to implement a kernel driver which hooks into the block device's function pointer (like `ide_driver_t.do_request` for IDE disks).

III: Initial disk encryption

Initial disk encryption for Linux mapped devices requires for the filesystem to be created anew, encrypting an existing filesystem is not possible (yet?). Solutions for Microsoft Windows are one step ahead in this respect, encrypting an existing filesystem („in-place encryption“) is commonplace – even among open-source implementations (refer to section #4/III).

The initial disk encryption is usually managed by a dedicated service process: it continuously loads, encrypts and saves disk sectors until a disk is fully encrypted. The initial encryption phase usually takes several hours to complete, depending on factors like CPU speed and disk size. Using the computer system during the initial encryption phase is possible but performance is often severely limited due to the constant use of the CPU and continuous disk I/O. The only performance control available in (almost) all Full-Disk-Encryption solutions governs CPU limits. However, todays CPU performance levels and multi-core architectures make it much more adequate to employ I/O limiting controls in order to more precisely define the acceptable performance impact during the initial disk encryption phase.

#3 Threats

I: Passwords

Whenever password based authentication schemes are employed for the Pre-Boot-Authentication environment, it's of utmost importance to choose a strong password in order to avoid brute-force attacks on the cryptographic key. Randomly chosen or generated passwords with upper- and lower-case letters and digits add roughly 6 bit of entropy to the key space per password character. It would therefore require a password with more than 40 characters in order to properly protect a 256 bit cryptographic key. While it may not be absolutely necessary to adopt such an extremely long password, it is recommended that passwords for encryption purposes should be significantly stronger than suggested by standard password recommendations.

II: Cold boot attack

A recently publicized security risk for computer systems protected by a Full-Disk-Encryption solution is called the „cold boot attack“ [4]. The attack targets powered-on (the operating system is loaded) computer systems by extracting the cryptographic key from RAM. It is therefore of utmost importance to power-off or hibernate computer systems if they are exposed to unauthorized physical access.

III: Coercion

Legal, physical and other forms of coercion are non-technical threats to data protected by cryptographic means. Plausible deniability is a technique which may offer protection from coercion in some circumstances by hiding an encrypted virtual disk (a „hidden volume“) inside another encrypted disk. Adversaries won't be

able to determine whether a hidden volume exists, unless external evidence – like a witness who knows about the existence of the hidden volume – indicates that a hidden volume must exist. TrueCrypt is the only solution that combines Full-Disk-Encryption protection with hidden volume capabilities [5].

#4 Products

I: History

Commercial Full-Disk-Encryption solutions have been around since at least the early 90's for Microsoft DOS. TrueCrypt comes to mind as the premier open-source offering, but one of the most desired features – the encryption of the system drive – has only been implemented in version 5, which was released in February 2008. The first open-source solution with support for system drive encryption on Microsoft Windows was released one month earlier: DiskCryptor [6].

II: Commercial Software

Many commercial Full-Disk-Encryption offerings exist. The most interesting/relevant solutions are:

Name	**OS Support**	**Notes**
CE-Infosys CompuSec	Windows & Linux	Free version (as in beer) available for personal and commercial use
CheckPoint FDE	Windows, OSX & Linux	Only FDE solution supporting system drive encryption on OSX
PGP WDE	Windows	Allows mounting of encrypted disk on another system („disk slaving“)
SafeNet ProtectDrive	Windows	Includes CD/DVD&USB encryption support
Secude FinallySecure	Windows	Linux based PBA
Utimaco SGE	Windows	PBA is obfuscated/obscured in order to harden analysis

Multi-user support for the Pre-Boot-Authentication environment is a common feature among commercial Full-Disk-Encryption solutions. Most solutions allow for the user's operating system login credentials to be synchronized automatically to the Pre-Boot-Authentication environment in order to not require users to remember yet another password. Automatic credential passing from the Pre-Boot-Authentication environment to the operating system login is also a commonly implemented feature: users are not required to authenticate at the Microsoft Windows login-screen, a software component automatically handles the login by using the credentials the user entered in the Pre-Boot-Environment.

III: Open-Source Software

TrueCrypt and DiskCryptor are open-source software projects and feature Full-Disk-Encryption support for Microsoft Windows. Neither have currently implemented any key or user management features and are therefore only of limited use in enterprise environments. TrueCrypt is a container based encryption solution at heart: it was primarily designed to use encrypted files as virtual disks. Support for encrypting real disks and partitions was added in version 5, but only for Microsoft Windows. Computer users with only modest computer experience will find TrueCrypt rather confusing due to its rather involved procedure for activating its Full-Disk-Encryption features. Technically versed computer users on the other hand value TrueCrypt's extensive cryptographic capabilities. DiskCryptor's user interface (which for the most part consists of the installer application) offers more of a traditional Full-Disk-Encryption experience from a computer user's perspective: the user is presented a dialog during installation to select the disk drives for encryption. The final installation step requires the user to set an encryption password, after which the selected disk drives are being initially encrypted.

TrueCrypt and DiskCryptor are entirely independent, but compatible, implementations: DiskCryptor adheres to the the TrueCrypt volume specifications. The TrueCrypt volume specifications requires a 512 byte volume header to be located before the encrypted volume. The header contains data required for mounting the encrypted volume (this applies to disks, partitions and file-back containers alike). File-backed containers

contain this header at offset 0, while the header is stored in the sector located just before the lower partition boundary for partition-back containers. However, this only works reliably for the first partition as the first partition always starts on (at least) sector 63, leaving sector 62 (and others) unused and available for the volume header. This workaround has a negative impact on TrueCrypt's Full-Disk-Encryption capabilities: only the first partition may be encrypted in-place as other partitions are unlikely to have a spare sector at their lower partition boundary. DiskCryptor on the other hand stores the volume header inside the first sector of the partition and therefore allows for any partition to be encrypted in place. However, this approach introduces two issues:

- it reduces the available partition size for the filesystem by one sector[1] and
- it requires relocating every sector in the partition.

DiskCryptor implements a custom filesystem shrinkage function for NTFS/FAT12/FAT16/FAT32 filesystems in order to shrink the filesystem on Microsoft Windows XP and 2003 as necessary. DiskCryptor running on newer versions of Microsoft Windows employs native filesystem shrinkage support, if available. The sector relocations are conducted during the initial encryption process, which reads data from an unencrypted sector and writes the encrypted data to the target sector. The sector shifting logic is implemented in DiskCryptor's lower-level filter driver and therefore hides all aspects about the sector shifting to upper layers. For example: if the filesystem driver initiates a read operation for sector `X` then DiskCryptor intercepts the request in the lower-level filter driver, loads sector `X+1` from disk and returns the decrypted data as sector `X`.

#5 Oddities

I: TPM Support

Current computer systems – especially laptops – are often equipped with TPM chips for secure cryptographic key storage and most commercial Full-Disk-Encryption products claim TPM support. However, none of the commercially available products support storing cryptographic keys on TPM chips. Instead, the TPM support is limited to TPM "binding": the ID of the TPM chip (its public key) is saved to the settings during installation and compared with the TPM chip's current ID. The comparison is executed either at the start of Microsoft Windows or for every user login; mismatches cause either an immediate shutdown or a rejected user login. This unintuitive use of TPM chips is easily explained: TPM chips are vendor-specific devices and require proprietary drivers. Full-Disk-Encryption vendors dread the cost of adapting proprietary drivers to their Pre-Boot-Authentication environments. TPM chips are therefore only used from within Microsoft Windows by calling the standardized TSS interface, which in turn calls the vendor drivers. BitLocker, the Full-Disk-Encryption solution provided with some versions of Microsoft Windows Vista, is the only Full-Disk-Encryption solution with support for storing cryptographic keys in TPM chips.

II: Multi-disk support

Support for encrypting multiple disks is a commonly named feature by Full-Disk-Encryption vendors. However, adding or removing disks once a Full-Disk-Encryption solution is installed is not so commonly supported. Several products allow the use of newly installed disks but do not support encrypting the new disks. The only way to encrypt additionally installed disks is through a work-around: decryption of all encrypted disks, software uninstallation, software re-installation and re-encryption of all drives from anew.

Various recovery and migration scenarios might make it necessary for an encrypted disk drive to be installed on another system as a data disk. This seemingly trivial procedure is also not well supported by the majority of Full-Disk-Encryption solutions. The most adequate concept for such tasks is called „disk slaving" (refer to section #3/II): it allows for an encrypted disk to be added to another computer system and for the encrypted disk to be used by loading the new disk's cryptographic key in the Pre-Boot-Authentication environment through an additional authentication step. A permanent migration of an encrypted disk to other system is therefore at least cumbersome, if not even impossible to manage for larger deployments.

1 A different header/footer is stored at the end of the partition, therefore the filesystem must be shrunk by about 30KB in total

#6 Open-Source wish-list

I: TrueCrypt compatible user and key management

The current TrueCrypt volume specifications are key agnostic. It is therefore impossible to implement any user and/or key management features. However, user and key management functions are a necessity for enterprise deployments. Breaking into the enterprise Full-Disk-Encryption segment with TrueCrypt compatible solutions therefore seems to require modifying the current TrueCrypt volume specifications. A specification compatible alternative would be to store all user and key management related data not within the volume header, but as part of the Pre-Boot-Authentication environment's configuration. The Pre-Boot-Authentication environment would than safeguard the encryption password for the encrypted volume, instead of protecting the volume's encryption key.

II: Alternative Pre-Boot-Authentication storage implementations

The Pre-Boot-Authentication storage locations listed in #2/I should be implemented for open-source projects in order to allow for flexible deployment solutions. External Pre-Boot-Authentication loading methods like USB and PXE boot implementations already exist and should be complimented by auxiliary key provisioning systems like loading the cryptographic key from a High-Security-Module (“HSM”).

#7 Conclusion

Full-Disk-Encryption is a fairly straight-forward technology, albeit different architectures have evolved for Microsoft Windows and Linux operating systems. TrueCrypt and DiskCryptor implemented Full-Disk-Encryption support for Microsoft Windows in 2008 and both are viable open-source solutions for personal computer systems. However, neither open-source solution currently offers critical management features, thus enterprise environments still turn to proprietary products for their Full-Disk-Encryption needs.

#A References

[0] http://technet.microsoft.com/en-us/library/cc781134.aspx

[1] http://en.wikipedia.org/wiki/INT_13

[2] http://msdn.microsoft.com/en-us/library/aa490241.aspx

[3] http://sources.redhat.com/dm/

[4] http://citp.princeton.edu/memory/

[5] http://www.truecrypt.org/hiddenvolume.php

[6] http://www.diskcryptor.de/

#B Author

Jürgen Pabel studied Computer Science and Information Assurance at Georgia Tech and Norwich University. He's one of the first German CISSPs and works as an IT-Security consultant at Akkaya Consulting GmbH in Cologne (Köln), Germany. His blog is located at http://blog.akkaya.de/jpabel/ and covers a broad range of IT topics with a slight focus on security.

#C Acknowledgments

Martin Modahl, Jens Neuhalfen, Pit Linnartz and another person (who wishes to remain anonymous) reviewed this paper: thank you guys!