At-rest data encryption

Sensitive data can be stored persistently on a disk drive, flash drive, or other medium. One technique to preserve the privacy of such data is to store it in encrypted form, rather than in plaintext. Of course, encrypted data cannot be used in most computations, so if the machine where it is stored needs to perform a general computation on the data, it must first be decrypted. If the purpose is merely to preserve a safe copy of the data, rather than to use it, decryption may not be necessary, but that is not the common case.

The data can be encrypted in different ways, using different cryptographic algorithms (DES, AES, Blowfish), at different granularities (records, data blocks, individual files, entire files systems), by different system components (applications, libraries, file systems, device drivers). One common general use of at-rest data encryption is called "full disk encryption." This usually means that the entire contents (or almost the entire contents) of the storage device are encrypted. While the commonly used name suggests the approach is intended for hard disk drives, it can actually be used on many kinds of persistent storage media. Full disk encryption is usually provided either in hardware (built into the storage device) or by system software (a device driver or some element of a file system). In either case, the operating system plays a role in the protection provided. Windows BitLocker and Apple's FileVault are examples of software-based full disk encryption.

Generally, at boot time either the decryption key or information usable to obtain that key (such as a passphrase) is requested from the user. If the right information is provided, the key or keys necessary to perform the decryption become available (either to the hardware or the operating system). As data is placed on the device, it is encrypted. As data moves off the device, it is decrypted. The data remains decrypted as long as it is stored anywhere in the machine's memory, including in shared buffers or user address space. The data is never placed on the storage device in decrypted form. After the initial request to obtain the decryption key is performed, encryption and decryption is totally transparent to users and applications. They never see the data in encrypted form and are not asked for the key again, until the machine reboots.

Cryptography is a computationally expensive operation, particularly if performed in software. There will be overhead associated with performing software-based full disk encryption. Reports of the amount of overhead vary, but a few percent extra latency for disk-heavy operations is common. For operations making less use of the disk, the overhead may be imperceptible. For hardware based full disk encryption, the rated speed of the disk drive will be achieved, which may or may not be slower than a similar model not using full disk encryption.

What does this form of encryption protect against?

- It offers no extra protection against users trying to access data they should not be allowed to see. Either the standard access control mechanisms that the operating system provides work (and such users can't get to the data because they lack access permissions) or they don't (in which case such users will be given equal use of the decryption key as anyone else).

- It does not protect against flaws in applications that divulge data. Such flaws will permit attackers to pose as the user, so if the user can access the unencrypted data, so can the attacker. So, for example, it offers little protection in the face of buffer overflow or SQL injection attacks.
- It does not protect against dishonest privileged users on the system, such as a system administrator. If his privileges allow him to pose as the user or to install system components that give him access to the user's data, he will be given decrypted copies of the data on request.
- It does not protect against security flaws in the operating system itself. Once the key is provided, it is available (directly in memory, or indirectly by asking the hardware to use it) to the operating system, whether that OS is trusted and secure or compromised and insecure.

So what benefit does it provide? If a hardware device storing data is physically moved from one machine to another, the operating system on the other machine is not obligated to honor the access control information stored on the device. In fact, it need not even use the same file system to access that device. For example, it can treat the device as merely a source of raw data blocks, rather than an organized file system. However, if the data on the device is encrypted via full disk encryption, the new machine will usually be unable to obtain the encryption key. It can access the raw blocks, but they are encrypted and cannot be decrypted without the key. This benefit would be useful if the hardware in question was stolen and moved to another machine, for example.

For other forms of encryption of data at rest, the system must still address the issues of how much is encrypted, how to obtain the key, and when to encrypt and decrypt the data differently, with different types of protection resulting. There are relatively few circumstances where such protection is of value. One example is archiving data that might need to be copied and must be preserved, but need not be used. In this case, the data can be encrypted at the time of its creation, and perhaps never decrypted, or only decrypted under special circumstances under the control of the data's owner. Another unusual case involves a special form of encryption called homomorphic cryptography. Unlike traditional types of cryptography, homomorphic cryptography allows at least some operations (typically simple data transformations like adding one to a specific field in the encrypted data) without decrypting the data. In such cases, an individual file or record might be encrypted this way, with the intention of allowing some parties to alter it without decrypting it.

One important special case for encrypting selected data at rest is a password vault (also known as a key ring). Typical users interact with many remote sites that require them to provide passwords. The best security is achieved if one uses a different password for each site, but doing so places a burden on the human user, who generally has a hard time remembering many passwords. A solution is to encrypt all the different passwords and store them on the machine, indexed by the site they are used for. When one of the passwords is required, it is decrypted and provided to the site that requires it.

For password vaults and all such special cases, the system must have some way of obtaining the key whenever data needs to be encrypted or decrypted. If an attacker can obtain the key, the cryptography becomes useless, so safe storage of the key becomes critical. Typically, if the key is stored in unencrypted form anywhere on the computer in question, the encrypted data is at risk, so well designed encryption systems tend not to do so. For example, in the case of password vaults, the key used to decrypt the passwords is not stored in the machine's stable storage. It is obtained by asking the user for it when required, or asking him for a passphrase used to derive the key. The key is then used to decrypt the needed password. Maximum security would suggest destroying the key as soon as this decryption was performed, but doing so would imply that the user would have to re-enter the key each time he needed a password. A compromise between usability and security is reached, in most cases, by remembering the key after first entry for a significant period of time, but only keeping it in RAM. When the user logs out, or the system shuts down, or the application that handles the password vault (such as a web browser) exits, the key is "forgotten." This approach is reminiscent of single sign-on systems, where a user is asked for his password when he first accesses the system, but is not required to re-authenticate himself again until he logs out.