Analysis of the NTFS file system for Windows 11 in a security context

Released in 1993 for Windows NT 3.1, NTFS (New Technology File System) replaced FAT as the default file system on Windows (Custer, 1994). As a modern file system, NTFS places a large importance on security.

Some examples of NTFS security features are:

* Access control lists (ACLs)
* Encrypting File System (EFS)
* BitLocker

ACLs (introduced in 1995 for Windows NT 3.51 (Russinovich, 2006)) are a way of restricting access to system objects on a network or computer. Both users and groups (referred to as security principals) can be defined and are represented by a unique SID (Security identifier). An administrator can then add security principals and assign read, write, modify, or full control permissions for specific objects such as files, folders, printers, and registry keys. Sub-folders of a folder with an access control list will inherit permissions (Microsoft, 2022)

EFS (introduced in 2000 in NTFS 3.0 alongside Windows 2000) is a NTFS feature that allows a user to encrypt individual files and directories of their choosing (Microsoft, 2023). To encrypt files, EFS uses a combination of a public and a private key – known as “asymmetric encryption”. When a file is encrypted with EFS, a unique File Encryption Key (FEK) is generated. The FEK is used to encrypt the contents of the file in blocks. The FEK is stored in a header of the encrypted file and is then encrypted with a public key taken from the user’s certificate from the user profile. (Microsoft, 2012)

Diagram

Description automatically generated

Figure 1 diagram showing the structure of a file encrypted with EFS (Microsoft, 2012)

As shown in the diagram above, “recovery agents” can be created. This is in the event that the private key is damaged or lost. The local or domain administrator will automatically be assigned as recovery agents. Private keys will be created for each agent and can be decrypted with the public keys (Microsoft, 2012). Recovery agents can also be manually created (Microsoft, 2022)

EFS uses transparent encryption, this means that when an authorized user opens the folder the user certificate public key is automatically used to decrypt the private key and decrypt the contents as plaintext. From an authorized user’s perspective, the decryption is seamless (Microsoft, 2012)

There are four different services that EFS uses to encrypt files: CryptoAPI, EFS Service, EFS Driver, and EFS FSRTL (EFS File System Run-Time Library). EFS service uses CryptoAPI to generate the FEK and public key and then creates the header for the encrypted file. EFS driver communicates with EFS service to request key management service from CryptoAPI for encryption and decryption which is then passed to EFS FSRTL to perform file system operations (open, read, write, and append) (Microsoft, 2012)

Diagram

Description automatically generated

Figure 2 diagram to show full operation of EFS (Microsoft, 2012)

BitLocker (introduced in 2006 alongside Windows Vista) is a full drive encryption solution for the NTFS file system. It is recommended that BitLocker is enabled in conjunction with a TPM (Trusted Platform Module) chip. However, BitLocker can be used to encrypt a NTFS drive without a TPM, but this requires that a USB startup key to be inserted into the computer on startup (Microsoft, 2022).

When BitLocker is installed, an unencrypted system partition is automatically created and is used to boot, decrypt, and load the operating system (Microsoft, 2022).For encryption, BitLocker uses AES (Advanced Encryption Standard) with a default key length of 128 bits – this can be changed to 256 bits through group policy (Microsoft, 2022). To reduce encryption time, BitLocker can be configured to use “used disk space only encryption” and only encrypt areas of the disk that contain data (Microsoft, 2022). BitLocker can also be configured to only startup after a PIN has been provided or a USB startup key inserted (Microsoft, 2022)

**[comparison][mention MacOS FileVault vs BitLocker][Disk Utility vs EFS]**