Global Information Assurance Certification Paper

Windows 2000 Encrypting File System
Eric Brock
July 27, 2000

Introduction
A widely held view among Information Technology professionals today is that the term "Microsoft security" is an oxymoron. In its latest operating system, Windows 2000, Microsoft has made some very strong improvements that could make that view a thing of the past. With the additions of Kerberos version 5 authentication, IP Security, TLS, smart cart support, and integration of Public Key Infrastructure, Windows 2000 should prove to have more credibility in terms of security than any of its Microsoft predecessors.

Another addition to the Windows 2000 security lineup is the Encrypting File System, or EFS. EFS enables users to secure data on a hard drive using public key encryption. Even if an attacker gains access to data on a hard drive, files on the drive that have been encrypted using EFS are useless without the decryption key. It is easy to see that this feature will have significant benefits for laptop users, home users on dedicated broadband connections, and organizations with the need to further secure highly sensitive data.

How W2K EFS Works
For a user to encrypt a file using EFS, there must exist a public and private key. If the user does not have one, the EFS service generates the key pair automatically. This makes the ability to encrypt files available to even the novice user who need know nothing about encryption. Files can be encrypted individually, or a folder can be designated as encrypted, so that any file written to that folder would be automatically encrypted.

Once a user issues the command to encrypt a file, or once the user attempts to add a file to a folder marked for encryption, EFS takes the following steps:

Step 1 – The file is copied to a temporary text file. This file is used for recovery should an error occur during the encryption process.

Step 2 – The file is encrypted using a randomly generated key called the File Encryption Key (FEK). The length of the FEK is 128 bits (US and Canada only). The file is encrypted using the DESX encryption algorithm.

Step 3 – The Data Decryption Field (DDF) is generated. This field contains the FEK which is encrypted using RSA encryption and the public key of the user.

Step 4 – The Data Recovery Field (DRF) is generated. The purpose of this field is to determine the users that can decrypt the file in the event the user's key is unavailable to decrypt the file (lost key, leaves company, etc). These users are called recovery agents. Recovery agents are defined in the Encrypted Data Recovery Policy (EDRP), which is a domain-wide security policy. If a domain EDRP is not set, the local EDRP is used. In either case, the EDRP must exist (and therefore at least one recovery agent defined) before any encryption can take place. The DRF contains the FEK which is encrypted using RSA encryption and the public key of the recovery agent. If there are multiple recovery agents listed in the EDRP, the FEK must be encrypted with each recovery agent's public key. Therefore, a DRF must be created for each recovery agent.

Step 5 – The encrypted file, which contains the encrypted data, the DDF, and all of the DRFs, are written to the disk.

Step 6 – The text file created in step 1 is deleted.

The following process occurs when data is decrypted:

Step 1 – The FEK is decrypted using the DDF and the user's private key.

Step 2 – The file is decrypted using the FEK.

In the case of a recovery agent recovering a file, the same process is used, except the DRF is used in Step 1 rather than the DRF.

EFS Components
EFS is made up of the EFS service, the EFS driver, the EFS File System Run-Time Library (FSRTL), and Win32 APIs. The EFS service runs as a standard system service. It is part of the W2K security subsystem. It interfaces with the CryptoAPI to generate keys, DDFs and DRFs. The EFS driver acts as part of NTFS that makes calls to the EFS service and request that keys, DDFs and DRFs be created as needed. A component of the EFS driver is the EFS FSRTL, which defines the functions that the EFS driver can perform on behalf of NTFS.

How EFS and NTFS Coexist
EFS can be thought of as a second layer of defense beyond NTFS. In order to access a decrypted file, the user must have NTFS permissions to access the file. Users who have the appropriate NTFS permissions will be able to see the file in the folder, but will not be able to open the file without the appropriate decryption keys. Likewise, a user with the appropriate keys but without the appropriate NTFS permissions will not be able to access the file. So a user needs both NTFS permissions and the decryption key to be able to open an encrypted file.

However, NTFS permissions can be overridden by a number of methods, including password-cracking programs, failure of a user to log off of a system before leaving, or Systems Internal's NTFSDOS. In NT 4.0, the game was over at that point - all of the data on the hard drive was accessible. With Windows 2000, when a file is encrypted using EFS, an unauthorized user, even having gained access to files on the disk, cannot access the data in the files without the authorized user's private key.

EFS Benefits
The ability to encrypt data on the disk is a tremendous benefit in and of itself. There are additional benefits that Microsoft's implementation of EFS brings over other third-party encryption methods:

- Accessing an encrypted file requires no action from the user. Previous third-party file encryption tools required users to enter passwords each time the file was accessed, and they did not integrate with the file system or the operating system in a seamless manner, which contributed to user frustration and a reluctance to utilize file encryption.

- The EFS cipher suite combines the efficiency of symmetric encryption (DESX) with the security of asymmetric encryption (RSA). Rather than using asymmetric encryption for the data, only the FEK is encrypted in this manner. The data is encrypted using symmetric encryption.

- Backup processes back up files in their encrypted form, eliminating the need for backup operators to access the data.

- EFS integrates into the file system, so that the file system cannot be bypassed when a malicious user has access to the hard drive. Also, all of the EFS drivers run in kernel mode and are inaccessible directly by users.

- Windows 2000's CryptoAPI architecture allows users to store their private keys on smart cards, which is much more secure than having the key reside on a hard drive or floppy disk. This also enables access from multiple locations.

Limitations
Seldom does an increase in security come without a cost. Any encryption process will add processing overhead and decrease performance to a certain extent. Here are some more issues with the practice of encrypting files on a hard drive:

- Encryption only takes place when the file is stored on the disk, not over the network. Other network encryption methods, such as IPSec, must be implemented in order to secure network transmissions.

- Anti-virus scanners will not be able to scan encrypted files unless they have access to the user's keys.

- It is conceivable that if a file or hard drive is stolen, and the malicious user is skilled in deciphering encrypted data and has unlimited amounts of time, that the data could be decrypted.

Here are some more limitations that pertain specifically to Microsoft EFS:

- EFS only works on an NTFS volume. There is not currently support for EFS on FAT volumes.

- EFS currently uses DESX as its encryption algorithm. Stronger algorithms exist, and Microsoft has promised that they will be supported in future versions of EFS.

- If system files are encrypted, the system will become unusable. EFS is intended for user data only. Files needed by the OS to boot the system cannot be encrypted or they would be inaccessible at startup. To guard against this, encryption is prohibited on files where the system attribute is set. However, on 7/25/00, SecuriTeam reported a DoS relating to EFS, which was that if the autoexec.bat file (which does not have the system attribute) was encrypted, the system would no longer be bootable.

- Defining too many recovery agents can significantly impact performance. For every recovery agent, the FEK must be encrypted and a DRF created. This causes two problems. First, more disk space is needed for the multiple DRFs stored with the file. Second, creating multiple DRFs takes more time and processing resources.

- File sharing is not supported in this release. Only the user whose key was used to create the DDF can access the file. Microsoft has announced that it will support file sharing in future versions of EFS.

- EFS requires increased administration for system administrators, most importantly in the area of managing encryption keys.

- The text backup file created in step 1 of the encryption process is stored in unencrypted form for the duration of the process. It is conceivable that a malicious user could gain access to this file while it exists.

Conclusion

EFS is a welcome addition to a Microsoft operating system. It operates transparent to the user and helps reduce some of the dangers of poor physical security. As with any security technology, it has its limitations and issues, and it will not provide all the answers. But as part of an overall security strategy, EFS can be a very valuable tool for preventing unauthorized eyes from seeing sensitive data.

References

Ahmad, Zubair. "Windows 2000 EFS." 1 March 2000. Windows 2000 Magazine. URL: http://www.winntmag.com/Articles/Index.cfm?ArticleID=7977

Buhrmaster, Gary. "Windows 2000 EFS (Encrypting File System)." 12 February 2000. URL: http://www-project.slac.stanford.edu/windows2000/updates/efs1.htm

Ludens, Douglas. "Encrypting File System - The Basics." URL: http://windows2000.about.com/compute/windows2000/library/weekly/aa000514a.htm

Microsoft Corporation. "Step-by-Step Guide to Encrypting File System (EFS)." URL: http://www.microsoft.com/TechNet/win2000/efsguide.asp

SecuriTeam. "Logon DoS when Windows 2000 is combined with EFS." 25 July 2000. URL: http://www.securiteam.com/windowsntfocus/Logon_DoS_when_Windows_2000_is_combined_with_EFS.html