Interested in learning
more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

The Encrypting File System: How Secure is It?

The Encrypting File System (EFS) is one of the many new features of the Windows 2000 operating system. It was designed to address security holes in NTFS. Namely, tools like NTFSDOS that allow attackers to bypass NTFS permissions. Microsoft states, "The encryption technology used is public key-based and runs as an integrated system service making it easy to manage, difficult to attack, and transparent to the user." (Encrypting File System, 1). EFS does provide another layer of security, but just how difficult is it to a...

Copyright SANS Institute
Author Retains Full Rights

AD

The Encrypting File System

How Secure is It?

By Howard Wright
SANS Security Essentials (GSEC)
Version 1.2f

November 2, 2001

© SANS Institute 2001, Author retains full rights

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

The Encrypting File System – How Secure is it?

Abstract

The Encrypting File System (EFS) is one of the many new features of the Windows 2000 operating system. It was designed to address security holes in NTFS. Namely, tools like NTFS-DOS that allow attackers to bypass NTFS permissions. Microsoft states, "The encryption technology used is public key-based and runs as an integrated system service making it easy to manage, difficult to attack, and transparent to the user." (Encrypting File System, 1). EFS does provide another layer of security, but just how difficult is it to attack?

How it Works

EFS works using a clever mix of symmetric/asymmetric key technology. Both of these encryption technologies have their strengths and weaknesses. Microsoft attempts to capitalize on the strengths while making the weaknesses irrelevant.

Symmetric key encryption (also known as secret key encryption) uses a single key to encrypt and decrypt data. The primary advantage of symmetric encryption algorithms is that they are very fast. This makes these algorithms ideal for encrypting large amounts of data. The problem with symmetric encryption algorithms is key distribution. If more than one person needs to access the encrypted data, they must have access to the secret key. This is a concept that does not scale well. Asymmetric key encryption algorithms were designed to solve this key distribution problem.

Asymmetric key encryption algorithms use public key/private key based encryption. With this type of encryption each user gets two keys. One is the public key and this can be freely distributed to whomever the user will be interacting with. The second key is the private key and only the user should have a copy of this one. The public key is used to encrypt a file which is sent to the user. The user can then use his private key to decrypt the file.

The first time a file is encrypted, EFS assigns the user a public key/private key pair for the account. If a certificate service is available, it will generate the keys. For standalone systems, EFS will generate a self-signed certificate. When a file is encrypted, EFS generates a random number for that file called a file encryption key (FEK). The FEK is used to encrypt the file contents using data encryption standard (DESX). DESX is an enhanced version DES. Basically, DESX processes the data three times with three different keys. The FEK is stored on the system and is encrypted using the user's public key with the RSA public key encryption algorithm.

It appears that for every file encrypted, two encryptions take place, one for the file and one for the FEK. This seems a bit redundant, but there is a reason for this behavior. Microsoft uses a symmetric algorithm called Data Encryption Standard Exclusive (DESX) to encrypt the files. Microsoft chose this method because it is fast and well suited for encrypting large amounts of data. However, Microsoft uses an asymmetric algorithm to encrypt the FEK. Because it is a small amount of data, it makes sense to use an asymmetric algorithm for encrypting the FEK.

Breaking Encrypted Files

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

Microsoft often touts EFS as a great way to secure laptops. The idea being that if the laptop is stolen, all critical data will still be encrypted and therefore, safe. This is not quite accurate. Under the right conditions, there are some relatively easy methods of cracking encrypted files. Microsoft does recommend some work-a-rounds which will also be explored.

There are three critical flaws with the encrypting file system that make it less secure than it could be. These flaws, taken separately, seem minor and even insignificant. In fact, it could be argued that these are not flaws, but legitimate features. Unfortunately, when these three "features" combine, security vanishes.

The first flaw is that the architecture of EFS does not allow encryption of system files. This means that encryption is limited to data files and various folders, but the system drive cannot be fully encrypted. So why is that a problem? Who cares if someone steals your laptop and can see the %systemroot% folder? All the data is still protected, right? The latter is technically true, but because the interloper can actually boot the system, he has a toe hold, a place to begin the task of cracking your files.

The second flaw is the default configuration. It is typical of Microsoft to configure their technologies so they work "out of the box." They do this so the average user can take advantage of the technologies without really needing to think about it or understand it. For security, this is often a problem.

The final flaw discussed is the recovery agent. This is actually a very important feature of EFS. It is important that the company be able to recover data in the event that an employee dies, gets fired or unexpectedly moves to another job. The recovery key is a second key that is managed by the administrator or designated person that can be used to unlock encrypted files. Microsoft considers the recovery key to be a critical part of the encrypting file system. So much so, that if the recovery certificate is removed, files can no longer be encrypted. There are many businesses and government entities that will not allow encryption without some recovery mechanism.

By combining these three mechanisms, we create a synergistic effect for the hacker. The following example applies to a standalone computer. Step one, a laptop is stolen. The next problem to be overcome is how to log into the computer. To some, this may seem daunting, but in reality, there are a number of ways to gain administrator access to the Windows 2000 operating system if one has physical access to the computer. Perhaps the simplest is to install a second operating system in a separate directory. Booting into the new operating system gives the attacker full access to the system with the exception of any encrypted files. Now that the attacker has file access, he can copy the SAM database from the original system. The SAM database contains an encrypted list of all users and their passwords for the local machine. Fortunately, for the would-be hacker, there are a variety of tools available on the Internet for cracking passwords. Perhaps the best known is lophtcrack. Using a tool like lophtcrack, an attacker will eventually break the administrator password. A less sophisticated user could simply delete the original SAM database and reboot into the original system. Deleting the SAM database resets the administrative logon to "administrator" with a blank password. There are other tools that make the job still faster and easier. For instance, Winternals Software (http://www.winternals.com) sells a product called ERD Commander. This product will, among other things, allow the user to change the administrator password. Simply boot from the CD, change the password and reboot. The problem with this method is that you have to actually pay for software. So, if the attacker is impoverished, there is a utility called ntpasswd (http://home.eunet.no/~pnordahl/ntpasswd/) that will boot into Linux and mount the NTFS volume and allow the administrator password to be

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

changed. The latter is a free utility. Of course, none of these cracks would work if the system files could be encrypted. Flaw number one has been exploited.

Next, the attacker needs to find a way to decrypt the desired files. Or does he? This is where flaws two and three come into play. Microsoft mandates that a recovery agent be available. For the agent to be available on a standalone machine, it must exist on that machine. By default, on a standalone system, the recovery agent is the local system administrator account. So by cracking the system administrator account, the attacker gets full control of all files… even the encrypted ones. To add insult to injury, this default configuration will make all encrypted files available to the hacker without him even knowing it. The decryption process works magically behind the scenes. Given this scenario, EFS provides no additional protection at all!

Microsoft claims that there is no problem with their technology. It is a simple case of the user incorrectly using the product that is to blame. Their position is that the configuration required for the cracks to work can only occur if the Administrator does not do his job correctly (never mind that it is the default configuration). They then reference the following help files:

> *If you are the recovery agent, you should be sure to use the Export command from Certificates in Microsoft Management Console (MMC) to back up the recovery certificate and associated private key to a secure location. After backing up, you should use Certificates in MMC to delete the recovery certificate. (Windows 2000 Server help, EFS)*
>
> *The designated recovery agent should export the data recovery certificate, secure it in a safe place, and delete the data recovery certificate from the system hard disk. In this way, the only person who can recover data for the system is the person who has physical access to the recovery certificate. (Windows 2000 Server help, Best Practices)*

Microsoft spends a lot of time and energy describing the necessity of removing the recovery key. What they neglect to tell you is that you *cannot remove (export) the user's private key*. In future releases, Microsoft hopes to make the private key exportable to some other media (i.e. smart cards), but for now, in Windows 2000, the keys are stored locally. So what does this mean? Let's return to our previous scenario. The attacker has obtained access to the administrator account. He does a quick check of the local security policy and discovers that the recovery certificate has been removed. Undaunted, he right clicks on the My Computer icon and selects manage. When the MMC opens, he selects Local Users and Groups. All user accounts are displayed. The attacker now changes the passwords for all the users to some known value. The attacker can now logon to any user account on the system. And by doing so, he obtains access to all of that user's encrypted files. So, you see, by removing the recovery key, you add maybe ten minutes to the time it takes the attacker to access your encrypted files. (Note: For this method to work, you must use one of the methods that exploit the passwords. Throwing away the SAM database would eliminate the user accounts.)

Microsoft also suggests that the original exploit that allows access to the administrator account could be circumvented if the syskey utility were properly enabled (Analysis of Reported, 2). Syskey is a utility that first appeared as a post service pack 2 hotfix for NT4. In the NT world, it has always been an optional security feature. In Windows 2000, it is applied by default. Syskey enables strong encryption for the passwords in the SAM database. It encrypts the passwords by

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

using a 128 bit random key. In Windows 2000, the default implementation stores the key locally. This allows the computer to boot without user intervention. The syskey default configuration adds little security in the context of this paper. Freely available utilities can still be used to modify the Administrator password. One such example is ntpasswd. However, the syskey utility has optional configurations. There are two additional behaviors that can be applied using syskey. The first allows the user to store the startup key on a floppy disk. In this mode, the system will not boot without the floppy disk inserted in the drive. The second option generates the key based on a user-determined passphrase. In this mode, the user is required to enter the passphrase or the boot process will not complete. Microsoft recommends either of the latter two configurations. This partially ameliorates the first critical flaw, which did not allow the encryption of system files. This accomplishes almost the same thing. At least, the password attacks will not work because the system cannot be booted. The ntpasswd tool can be used to disable syskey, allowing the system to boot without a passphrase or floppy. However, when syskey is disabled, encrypted files are no longer accessible. It should also be noted that syskey cannot be re-enabled once it is turned off.

The latter two methods do significantly enhance security, but they also add a significant degree of complexity. Using syskey is appropriate for a single user who manages his own system. However, in a medium sized company that has some type of computer support staff, implementing syskey would be cumbersome. Syskey can only be run by an Administrator. Therefore, all systems would need to be pre-configured by the support team. If the option to require a floppy disk to boot were selected, the support staff would have to make multiple floppies for each configured system. One floppy would need to be given to the user and the others would need to be filed and stored in a secure location in case the Administrator needed to recover files or the user lost his floppy disk. If the second option is chosen, requiring a user to enter a passphrase before for the system will boot, then the Administrator will need to file and store the passphrase in a secure location. Again, the latter is required so that files can be recovered if a user forgets his passphrase or leaves the company. In either case, the user would require training to deal with the added security. These limitations mean that the use of syskey as Microsoft recommends is not practical beyond an individual user or small company. It is interesting that Microsoft, a company that puts such a huge emphasis on the value of the recovery certificate (remember, if the recovery certificate is removed, encryption no longer works) would recommend the use of a utility that has no recovery mechanism.

Up to this point, all the failures of EFS have been related to standalone systems. Workstations that are members of a Windows 2000 domain are much more resistant to the types of attacks previously discussed. Domain members are more resistant for a couple of reasons. The first is related to the recovery agent. By default, the EFS Recovery Agent is the domain Administrator, not the local Administrator. This physically separates the recovery key from the system that contains the data. The user passwords are protected in much the same way. In a domain environment, a central server authenticates all systems. The actual password hashes live on the central server. So once again, the actual passwords are physically removed from the local machine. So while an attacker can still gain control of the local administrator account, he still cannot access encrypted files. There is no recovery agent or SAM database for him to manipulate.

While using EFS as a domain member is far superior to a standalone system, there are still avenues open for the dedicated hacker to explore. By default, Windows 2000 allows users to cache their logon credentials. The purpose of the cached credentials is to allow the user to logon

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

even if the domain controller is unavailable. These cached credentials reside on the local machine. It may be possible for an attacker to capture these credentials and use them to break into encrypted files. The other area that may be exploited is the user's private key. You may recall that in the current implementation of Windows 2000, the user's private key is stored locally. There is no way to export it. If an attacker can obtain access to the private key, he will be able to decrypt the files. However, there are no free tools or any off the shelf software that will exploit these weaknesses as yet.

Conclusion

The Encrypting File System can provide another layer of security if properly configured. Standalone computers can be made somewhat secure by exporting the recovery key and using the syskey utility to force users to enter a passphrase or insert a floppy to boot the system. Unfortunately, the use of syskey does not scale well. Only people with Administrator privilege can run the utility and there is no means of installing an "admin passphrase." In a moderate sized company, this can place a significant burden on the support staff. A secure EFS configuration is not really practical for a standalone system. EFS was really designed to work in a Windows 2000 domain environment. In the domain environment, neither the SAM database nor the recovery key can be attacked because neither is physically on the local machine. Because nothing physically resides on the local disk, there is no reason to use the syskey utility to establish a passphrase. This means that all management functions can be handled remotely by the domain administrator and that files can be safely encrypted with minimal impact on support staff.

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

Bibliography

Analysis of Reported Vulnerability in the Windows 2000 Encrypting File System (EFS)
URL: http://www.microsoft.com/TechNet/security/topics/analefs.asp (10 Oct. 2001).

Determine whether Syskey has been applied to a system
URL: http://is-it-true.org/nt/registry/rtips226.shtml (10 Oct. 2001).

Encrypting File System for Windows 2000
URL: http://microsoft.com/windows2000/techinfo/howitworks/security/encrypt.asp
(10 Oct. 2001).

Inside Encrypting File System, Part 1
URL: http://www.win2000mag.com/Articles/Index.cfm?ArticleID=5837 (10 Oct. 2001).

Inside Encrypting File System, Part 2
URL: http://www.win2000mag.com/Articles/Index.cfm?ArticleID=5592 (10 Oct. 2001).

SAVANT Windows 2000 Advisory No: 00/26 Dated: 10 May 2000
http://archives.indenial.com/hypermail/ntbugtraq/2000/May2000/0039.html
(15 Oct 2001).

Schmidt, Jeff. Microsoft Windows 2000 Security Handbook.
Indiana: Que, 2000.

Solomon, David A. & Russinovich, Mark E. Inside Windows 2000
Washington: Microsoft Press, 2000.

Windows 2000 Encrypting File System (EFS) Vulnerability
http://www.deepquest.pf/win32/win2k_efs.txt (No Longer available)

Windows 2000 Server help, Best Practices

Windows 2000 Server help, EFS

Windows NT System Key Permits Strong Encryption of the SAM
http://support.microsoft.com/support/kb/articles/q143/4/75.asp

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

Last Updated: May 19th, 2013

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Malaysia @ MCMC 2013	**Cyberjaya, MY**	**Jun 03, 2013 - Jun 08, 2013**	**Live Event**
SANS Pen Test Berlin 2013	**Berlin, DE**	**Jun 03, 2013 - Jun 08, 2013**	**Live Event**
Industial Control Systems Security Training - Houston	**Houston, TXUS**	**Jun 10, 2013 - Jun 15, 2013**	**Live Event**
Security Impact of IPv6 Summit 2013	**Washington, DCUS**	**Jun 14, 2013 - Jun 16, 2013**	**Live Event**
SANSFIRE 2013	**Washington, DCUS**	**Jun 14, 2013 - Jun 22, 2013**	**Live Event**
SANS Canberra 2013	**Canberra, AU**	**Jul 01, 2013 - Jul 13, 2013**	**Live Event**
SANS London Summer 2013	**London, GB**	**Jul 09, 2013 - Jul 16, 2013**	**Live Event**
Digital Forensics & Incident Response Summit 2013	**Austin, TXUS**	**Jul 09, 2013 - Jul 16, 2013**	**Live Event**
SANS Rocky Mountain 2013	**Denver, COUS**	**Jul 14, 2013 - Jul 20, 2013**	**Live Event**
SANS Mumbai 2013	**Mumbai, IN**	**Jul 22, 2013 - Jul 27, 2013**	**Live Event**
SEC528 SANS Training Program for the CompTIA® New Advanced Security Practitioner (CASP) Certification	**Washington, DCUS**	**Jul 22, 2013 - Jul 26, 2013**	**Live Event**
SANS San Francisco 2013	**San Francisco, CAUS**	**Jul 29, 2013 - Aug 03, 2013**	**Live Event**
SANS SEC 560: Network Penetration Testing @ Bangalore 2013	**Bangalore, IN**	**Aug 05, 2013 - Aug 10, 2013**	**Live Event**
SANS Boston 2013	**Boston, MAUS**	**Aug 05, 2013 - Aug 10, 2013**	**Live Event**
Critical Security Controls Summit	**Washington, DCUS**	**Aug 12, 2013 - Aug 18, 2013**	**Live Event**
Industrial Control Systems Security Training - DC	**Washington, DCUS**	**Aug 12, 2013 - Aug 16, 2013**	**Live Event**
Mobile Device Security Summit 2013	**OnlineCAUS**	**May 30, 2013 - Jun 06, 2013**	**Live Event**
SANS OnDemand	**Books & MP3s OnlyUS**	**Anytime**	**Self Paced**