IBM Systems MAGAZINE
Administrator

Locking Down Files With Encrypted File System

August | September 2008 | by Jaqui Lynch

In October 2007 I outlined some of the new security features in AIX* version 6.1. This article goes into detail of what's involved in implementing the Encrypted File System (EFS) and its associated prerequisites.

What is EFS?

EFS provides the capability to have a file system that includes automatically encrypted and decrypted files, provided the user has the necessary privileges. This feature depends on the CLiC libraries as it uses keys for encryption. Additionally, enhanced role-based access control (RBAC) is needed, requiring a basic understanding of RBAC, specifically for creating, assigning and using authorizations, roles and privileges.

Authorizations are used to grant access to commands or functions that one needs to perform. Roles are assigned to a user and act as a container for a set of authorizations. Privileges are used to grant the power to a process to perform certain privileged operations. When users issue a swrole they receive the authorizations and privileges assigned to that role and they then have the necessary access. (RBAC is beyond the scope of this article.)

EFS has a new attribute that indicates that the files in this file system are to be encrypted as they contain sensitive data. EFS is only available for JFS2 file systems and the EFS is created using a new file system type of EFS. The file system is encrypted on a per-file basis and the "ls –aU" command can be used to see whether a file is encrypted. Directories themselves aren't encrypted, only the files within them. Instead of showing permissions such as rwxr-xr-x, the file would show as rwxr-xr-xe, provided the –U flag is used on the ls command.

EFS has two new commands: efsmgr and efskeymgr. efsmgr is used to manage the encryption of files, directories and file systems including enabling EFS and setting up inheritance. efskeymgr is used to manage and administer the keys that are used for EFS. Users can either create their own keystores or use a group keystore.

Inheritance can be set at the filesytem or directory level. If the directory has inheritance set, files are automatically encrypted when they're created. Inheritance isn't turned on by default for the EFS. If inheritance is desired, the "efsmgr –s –E" command should be used once the file system is set up to ensure that all files are encrypted. Inheritance can be set on the file system or a directory in the file system. Lastly, EFS must be explicitly enabled using the "efsenable –a" command.

Steps to Implement an EFS

First ensure that the prerequisites are met (including CLiC, AIX version 6.1, JFS2 and RBAC-enabled). The six steps I followed were performed as root unless otherwise stated.

1. Use the “efsenable –a” command to enable EFS. This prompts you to set your keystore password, adds several lines to the /etc/security/user and group files, creates a number of files in /var/efs and puts the keystore in /var/efs/users/root.
2. Create the file system:

```
mkdir /jaqui
crfs -v jfs2 -g rootvg -m /jaqui -a size=200M -a
efs=yes
mount /jaqui
lsfs -q /jaqui
```

   This will show “EFS: yes” in the last line.

3. To enable inheritance on the file system, use “efsmgr -s -E /jaqui.” Do this up front as—if you turn inheritance on later—it only applies to any new files created, not to the ones currently in the file system.
4. To prompt for root’s EFS password, use “efskeymgr-o ksh.” This loads the keys so you can use them. If you’re unsure if the keys are loaded you can run “efskeymgr -V” to list the currently loaded keys for this user. If the user login password and their keystore password are the same, RSA keys are loaded automatically at login. Otherwise the user must use the efskeymgr –o ksh command.
5. Create a file and a directory:

```
mkdir /jaqui/dir1
touch /jaqui/freddy
```

6. Now run some checks:

```
# efsmgr -L /jaqui
EFS inheritance is set with algorithm:
AES_128_CBC
#
efsmgr -l freddy
EFS File information:
Algorithm: AES_128_CBC
List of keys that can open the file:
Key #1:
Algorithm : RSA_1024
Who : uid 0
Key fingerprint :
1b66973c:4a7c2e4d:f3d7ca03:fa13d082:1a855289

/dev/fslv00 458752 457920 1%
16 1% /jaqui

# ls -l -U freddy
-rw-r--r--e 1 root system
0 Jun 09 23:45 freddy
```

   Now freddy is encrypted and accessed by a key owned by UID 0 (root). Additionally inheritance is set on the /jaqui file system.

You can decrypt the file permanently using efsmgr –d, reencrypt it using efsmgr –e and list attributes using efsmgr –l.

Setting Up Directory Access for a General User

The following message shows that user jaqui doesn't have a keystore, so the first step is to figure out how to add one:

```
# efsmgr -a freddy -u jaqui
Unable to get public key from user "jaqui" (skipped): Keystore
does not exist
jaqui: A file or directory in the path name does not exist.
```

First list the user attributes and notice (last line) that the user has no assigned roles:

```
# lsuser jaqui
jaqui id=500 pgrp=staff groups=staff,system,sshd,freeware
home=/home/jaqui shell=/bin/ksh gecos=Jaqui Lynch login=true
su=true rlogin=true daemon=true admin=false sugroups=ALL
admgroups= tpath=nosak ttys=ALL expires=0 auth1=SYSTEM
auth2=NONE umask=22 registry=files SYSTEM=compat logintimes=
loginretries=0 pwdwarntime=0 account_locked=false minage=0
maxage=0 maxexpired=-1 minalpha=0 minother=0 mindiff=0
maxrepeats=8 minlen=0 histexpire=0 histsize=0 pwdchecks=
dictionlist= default_roles= efs_initialks_mode=admin
efs_keystore_algo=RSA_1024 efs_keystore_access=file
efs_adminks_access=file efs_allowksmodechangebyuser=yes
efs_file_algo=AES_128_CBC fsize=-1 cpu=-1 data=-1 stack=-1
core=2097151 rss=65536 nofiles=2000 roles=
```

Then check what the authorization is for efs:

```
# lsauth ALL | grep efs
aix.security.efs id=6130 dfltmsg=Encrypted Filesystem
Keystores Administration msgcat=sysauths.cat

msgset=8 msgnum=19
```

Then create a role for EFS security called jlefssec:

```
# mkrole dfltmsg='jl efs security'

authorizations=aix.security.efs jlefssec
```

Check that the role was created correctly:

```
# lsrole ALL | grep efs
jlefssec authorizations=aix.security.efs rolelist= groups=
visibility=1 screens=* dfltmsg=jl efs security msgcat=
auth_mode=INVOKER id=12
```

Add the role to user jaqui and check it got added:

```
# chuser roles=jlefssec jaqui
# lsuser jaqui
jaqui id=500 pgrp=staff groups=staff,system,sshd,freeware
home=/home/jaqui shell=/bin/ksh gecos=Jaqui Lynch login=true
su=true rlogin=true daemon=true admin=false sugroups=ALL
admgroups= tpath=nosak ttys=ALL expires=0 auth1=SYSTEM
auth2=NONE umask=22 registry=files SYSTEM=compat logintimes=
loginretries=0 pwdwarntime=0 account_locked=false minage=0
maxage=0 maxexpired=-1 minalpha=0 minother=0 mindiff=0
maxrepeats=8 minlen=0 histexpire=0 histsize=0 pwdchecks=
dictionlist= default_roles= efs_initialks_mode=admin
efs_keystore_algo=RSA_1024 efs_keystore_access=file
efs_adminks_access=file efs_allowksmodechangebyuser=yes
efs_file_algo=AES_128_CBC fsize=-1 cpu=-1 data=-1 stack=-1
core=2097151 rss=65536 nofiles=2000 roles=jlefssec
```

Now activate it:

```
# setkst
Successfully updated the Kernel Authorization Table.
Successfully updated the Kernel Role Table.
Successfully updated the Kernel Command Table.
Successfully updated the Kernel Device Table.

Check if the role is assigned:

# lsuser -a roles jaqui
jaqui roles=jlefssec

Now log in as jaqui and switch roles:

$ swrole jlefssec
jaqui’s Password:
$ ls /jaqui
dir1 file2 freddy lost+found
$ cat /jaqui/freddy
cat: 0652-050 Cannot open /jaqui/freddy.

The previous message is because root didn’t authorize jaqui to
access the file, so do that now using: “# efsmgr -a freddy -u
jaqui.”

Then as jaqui, load the keys:

$ efskeymgr -o ksh
jaqui’s EFS password:
$ cat freddy

Now you should see the data in the file.
```

Ongoing Issues

```
The main issue when working with encryption is key management.
```

Once a file is encrypted, the key is necessary to decrypt it. This means that key management is critical and copies of keys and authorizations need to be part of any disaster-recovery plan. Data on tape or disk is useless without the keys to access it. Backup of the keystores is critical.

Decisions must be made up front to determine roles, authorizations and privileges. As an example, it's possible to use group keystores rather than individual ones. This allows you to control a number of files with one key. Other considerations include:

- The file system must have enough space free to encrypt or decrypt files so it's important to monitor file system usage.
- /var must have enough space to hold the keystores and to grow them as necessary.
- An encrypted file uses an additional 4 KB to store encryption metadata. In file systems with many files it's important to include this in sizing estimates.
- Once a JFS2 file system is changed to an EFS it's a permanent change.
- Performance tests should be run if this is a performance-critical file system or if it's CIO- or DIO-enabled.
- EFS file systems can't be NFS exported and certain OS file systems (/, /usr, /var and /opt) can't be EFS enabled. Backup software must support EFS.
- You must plan for how keys will be loaded automatically if applications need access to encrypted data.

The Final Report

EFS provides an attractive solution for protecting sensitive data and automatically encrypting and decrypting the data. Setup requires some understanding of RBAC as well as careful planning. It's important to involve those who handle areas such as disaster recovery and backup in the plans to ensure that data isn't protected from legitimate access in a recovery situation. EFS is a great way to ensure data on the disk is always encrypted while it's on the disk, but it's important to ensure that performance during encryption and decryption is acceptable. For this article no performance tests were run, as it's a basic primer on bringing up an EFS.

References

1. SG24-7430 AIX v6 Advanced Security Features – Introduction and Configuration (Sep 2007)
www.redbooks.ibm.com/redbooks/pdfs/sg247430.pdf

2. SG24-7559 AIX v6.1 Differences Guide
www.redbooks.ibm.com/redbooks/pdfs/sg247559.pdf