Simple File Encryption with Openssl
Author - Nic Maurel

Introduction
Now I'm sure you've heard in the past clear text data is a bad thing and can be easily viewed. This applies to network traffic, as well as standard files residing on your hard drives or flashdisks, or pretty much any devices that carry or transport data. Now I'm going to show a quick example of how useful openssl can be to quickly encrypt files on a linux hard drive.

A shadow file on a server is a very important file, and only root is permitted to view it, as it stores password hashes. Now say for some strange reason you want to back this file up in the event of corruption or for redundancy purposes with a backup user. If this user was compromised and if this was a mission critical server, it would be a bad thing right? So how can we raise the bar and prevent this. I could think of many, but we could just encrypt the shadow file with openssl, so if this file fell into the wrong hands, the baddies would have a hard time trying to view it.

So how does encryption with openssl? Well that's the easy part.
Well first of all openssl supports many encryption types when you use it in encryption mode eg.
- AES
- Blowfish
- CAST
- DES, Triple DES
- RC2, RC4

If you like to see in more detail just check out the openssl man file.

And a whole bunch of bitrates with each of these. I'm going to use AES-256-cbc which will suffice for this example, after all the US Government uses AES to it can't be that bad:

Okay so we're logged in as root on our linux box and we want to make a copy of our shadow file and encrypt it.

```
[root@cerebrus ~]# mkdir -p /admin/bkp
```

> Make a directory

```
[root@cerebrus ~]# cd /admin/bkp/
```

> Go to the new directory

```
[root@cerebrus bkp]# cp -pr /etc/shadow ./
```

> Copy shadow file to new directory - preserving permissions

```
[root@cerebrus bkp]# ls -l
total 4
-r-------- 1 root root 1499 2009-02-23 23:36 shadow
```

> List file

```
[root@cerebrus bkp]# openssl enc -aes-256-cbc -salt -a -in ./shadow -out ./shadow.enc
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
```

> Ok. Now the above command is now where we encrypt the file and is explained below.
> openssl : The openssl toolkit command.
> enc : encryption parameter.
> -aes-256-cbc : the type of encryption being used, as mentioned before this algorithym should suffice.
> -salt : adds additional strength to the encryption and is recommended.
> -a : creates a base64 encoding of the file, for email or web usage, if this parameter is not used, using cat to display the file will resemble catting a binary file with a lot more garbage.
> -in : the name of the file you want to encrypt.
> -out : the name of file you want to output to.
> password : this is the most important part, because the passphrase needs to be strong (8 alphanumeric characters) or else guessing it will be too easy. You can also specify a -pass passphrase parameter on the commandline, but I am paranoid and people can read the history file if it is not cleared, so I use the input method. You can also use an ssl key, by specifying -pass file:/etc/keys/shadow.key useful for batch jobs, but if this key resides on the same server you are defeating the object of having a key. All in all it is your choice, call me old fashioned but I will go with the input method.

```
[root@cerebrus bkp]# ls -l
  total 8
  -r-------- 1 root root 1499 2009-02-23 23:36 shadow
  -rw-r--r-- 1 root root 2060 2009-03-02 10:21 shadow.enc
```

> Lists the newly created enc file

```
[root@cerebrus bkp]# cat shadow.enc

  U2FsdGVkX1+cCnT38MhselREo1svICu1MZiWCXq+FfYbEhehB9eiDk3vOYJLZDOv
  JDFbfX4ODOb/73eyenNfLBvdmNZN1wYE6WVsXetoXXVgFrRUcaeSMX9y0HRenury
  8QZCdFLHOIzZnuRiyTDdSRONJN7N+mRY2bzSX4PVRWFM55DkIz1gevJpxDfF2zex
  GXDBh74+529esv9JI1859y/HFvBnIxtNbwdE+PNTql6kSLzWmzZI8qNJxpwbZLJ9
  rYwi2nx+F8jUhdSlQlwYS22/k5pwh4nAwHYF32R/sUuwRipkldBje5DUOSC5ruof
  ZtUbfARQ1G5z41voME+VkO50oAaFXpzlCiVzwRCbAorlMEAM3mlgVuVQjiqxx2fc
  uwG9vkyfXsQIZI+qUl20e6t3dgjaOYW3qheygnnyKE4k+iw0TQ1+Dh1Em8rzImuE
  DkAvdW1ykXg047dtaGgyy2pmttb6y2fYOyrlSExuSc3UIsqqbk8PySKUIvp9jxoU
  s+YXXCAcO5NGzQmGyTacQ9Arpfwid+bjN4BfNfuxDRPmulETpb2NUGGQjxaRKfD/
  rGbzJbX1QPiE5WE/7MhUut6g4Obnon5+JS2SuaPoLaZAeWqvF1+tk97oprNohrkf
  D4AOei3jYTl0HeS/oyKeOex700Pq1RGBMcaNFAZ48Mx7lUe9qQTiRgS9KYGoimbq
  5W7HCBkMo8fI2bsbVPSXphvQ5C+ceQmMDfdeZjNIyS0tqxNMVh5lULaQlP7Tl3sD
  DcgD9pPgOVUi7T5Wdbb75TJHvuLSEDaa9NoJ+d5UDujBoIfKMyHTcOVYn5YOs3fU
  /Mtfn5KE8SFVutUX2EBuUzd37no5oBPeHiazgNIMueDnGezn39IKrcfz5fBqJlNU
  6cfLpid6kCqsKyyC7i+Li5auLLI/PygUZILYW4wdUBKPQxMgS65jFPn+byzKg1nz
  P0zXJPEZyF8vG1o3LpcERiUumIt33EcK5n93fPZaGT63br1SIbUYkoyMztQconnS
  KxVdM+VQ9ZttNA99voOpFPpVSxGVIadgr6SRP48B3vDMh4AW1lf+913C+3cErIzP
  LXiZDm0CSJsmxdtqZwrL0vbxD3d+Eow+K1C4sOu2vGRfSrlcEHNlDkrgo1vh1WqR
  ivm4kVPNF2CP0uC6uDAIY+s8+i2fTY0F7CdUhpVvydu1y8kzATKB+InS6fW53uJu
  tJZxGyomcFEWHJ1AYdlEKr+T1sQjzqqj8v5ny4R0cH1kH7UHBudy7MALY6dcR44h
  fKhaTvo2RNDFY1TwQx8UmltBaxLT0qICTYp9O+UuyncHHJMjuPZd5BhoV77M7PHU
  HdnoGIJbjh651W7gytyYeTl4JKOl7ha56tFE85l4C6oD4IXg31aJk7X98SEyqile
  CD/fnABeuyVfbwBhQFVYIt4c4XmIQ0P9dsCXC7H0IRZ8jXi6WdBdL5jUAfxxwfk/
  ulc90r7DHv+5ndPIzSIH01LEpX0vgR+XeObD9Br+42+1+QgdYsNj6IRPMInDRaCh
  CzWM1v0YqrmBlLqXjy1/ERQ30Notw3OmBwSaP/1Li34lXD4Z+jJIKpR61O3i7/+O
  6eEXE0FENgcIb8M3wdXFQrHsBoaVkc8zueNE7AOx05zeYR04WSNXStlxGd9KSKex
```

```
2mKs7TcRljgZqsWtF4R++XayB2xbziFIDPwCV6ZqR2XarxY3M1rPM1bdojSYw9Ja
sukQ19pzoO9N7GVGdSQa2l4We3x8+5DvRGotpjiRPMqVdwDSp3j07e7f491FoNL9
kgei6aAOCWLmdEKHIKpRQRsEwgeB+k8iI5yXuvo1sfX6RWsKl1aNG7WKk4IPkKBc
ZY2Ke0qfjKUfk2j5uEkoiYLPnFMnVqDXHN+Q/TRXhE2OGt3pgGYHQDhlvvT/tXNo
WZfcBaWJHtGXofHAy549BRjZ+ugXnuFbdrj3l/rdq9cVGLWVzykJsITIQeGWqR/q
gwLNWw+xWPo8BlkHc/y45HBmdEtxlPKEfDaqcCqZ7K8=
```

> Shows the file is encoded in Base64

> Now if we want to decrypt the file then the parameters are pretty similar, it is explained below.

```
[root@cerebrus bkp]# openssl enc -d -aes-256-cbc -salt -a -in ./shadow.enc -out ./shadow.plain
enter aes-256-cbc decryption password:
```

> openssl : The openssl toolkit command.
> enc : encryption parameter.
> -aes-256-cbc : the type of encryption being used.
> -salt : if salt is being used
> -a : if the file is encrypted with base64
> -in : the name of the encrypted file
> -out : the name of file you want to output to.
> password : the passphrase you used to encrypt the file

```
 [root@cerebrus bkp]# ls -l
total 12
-r-------- 1 root root 1499 2009-02-23 23:36 shadow
-rw-r--r-- 1 root root 2060 2009-03-02 10:21 shadow.enc
-rw-r--r-- 1 root root 1499 2009-03-02 10:31 shadow.plain
```

> And Voila! The decrypted file is listed

Please note: the permissions of the output file have now changed, this is the result of the umask that is set on the server. you should change the permissions to prevent the hashes from being read.

End Notes
Openssl is very useful, and when I found out that it could do this, I immediately had to write a paper. Use this info or don't use it, but if you come across an .enc file one day, you might just know what to do with it.