Linux Data Security Procedures

Contents

1 Introduction

Prevention is the best solution for accidents and incidents, which includes data security. Using the ''Need to Know'' principle should allow development ''Dual Use'' without the overhead associated with data security. The first priority is encouragement of security clean requirements, source code, and documentation. However, when forced with data security, a process with specific practices must be implemented.

This paper creates a specific Linux procedures for data security to be used on a project with a customer restricted to RSA signatures and AES256 encryption. The goal is to meet federal policies and to use public and widely available GNU Privacy Guard (GnuPG) (**gpg**) and the Advanced Encryption Standard (AES). Actually it shall meet federal policies and guidelines for data security found at `http://csrc.nist.gov`. Therefore, the specific GNU's PGP (**gpg**) configuration and parameters must be used. Improperly generated key will not meet these requirements.

A secondary goal is to support the long term strategic plans to includes remote support of sensitive

equipment. A secure data process, includes supporting the Federally defined insecure media's and networks..

This draft targets the Lenny Debian Linux 5.0.

2 Identification


```
$Id: data_security.tex 20 2011-12-02 23:54:43Z ty $
$HeadURL: svn+ssh://InfiniteDelta.com/svn/papers/data_security/data_security.tex $
```

3 Federal Policy Sources

The following link federal covers handling of International Traffic in Arms Regulations 2009 (ITAR) data: `http://www.pmddtc.state.gov/regulations_laws/itar_official.html` However, it did not specify the transferring of data over less secure media.

The National Institute of Standards and Technology does set forth several policies and guidelines:

`http://csrc.nist.gov/groups/ST/toolkit/digital_signatures.html` `http://csrc.nist.gov/groups/ST/toolkit/block_ciphers.html`

Specifically, the federal government approves using AES256 can be used to encrypt data. See policy (6) of CNSS Policy No. 15, Fact Sheet No. 1 National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information (June 2003).

`http://csrc.nist.gov/groups/ST/toolkit/documents/aes/CNSS15FS.pdf`

Note: the RSA/SHA1 signature is still approved, but SSHA2 is encouraged. We should solve this before we lock upon our public keys.

4 Recommended Data Security Policy

First priority is to avoid contamination and the ''Need to Know''. However, we may still need the data security.

4.1 ITAR BSP ''Dual Use'' cleaning

ITAR **Dual Use** projects such as kernel's and their BSPs are forever plagued by information that needs to be protected. However, most derived requirements can be clean of this information and its restrictions.

Example a secure system requires a battery with X specifications. Neither the battery or its actual capabilities needs to be restricted. Only the tie to the secure system is critical with its specifications.

Specifically: Use an Agile approach by offering ‘‘Dual Use’’ capabilities on specific IO for hardware. No need to know the customer or project or specific use, etc.

Note: Non ‘‘Dual Use’’ still needs special treatment.

4.2 Data Encryption

Consider still treating the computer as unsecured media may only require an approved encrypted partition and a memory stick for keys. This also solves the stolen computer and/or removed hard drive issues, and system crashes. A stolen unsecured media with approved encryption, with a separate and still secure key, is a less sever ITAR incident (No jail time!).

Even the systems physically locked behind closed doors, such as the SVN server, shall have encrypted partitions for critical data. The svn+ssh protocol shall be used so that the company network can be treated as an insecure media. All work to the servers shall use the same Secure SHell (ssh) protocol for any remote support. A guard against an intruder between system boots is required on the device used for secured keys.

5 Data Security Procedures

All proprietary or sensitive data shall be maintained on an AES-256 encrypted partition. When an OpenPG encrypted file is received, is is placed onto the encrypted partition, decrypted, and used there.

5.1 Public Key Handing

Many organizations do not actually make public keys public, not that they keep strict controls on them either. Public Keys shall not be made generally public for those who only interface with specific people. All keys and specific customer keys can be kept on a local key server and only distributed on a need basis.

5.2 Data Tracking

All ITAR shall be tracked, even while it is fully encrypted on insecure media. Tracking changes is not necessary. Only the locations and potential locations shall be documented. Encrypted data is not accessible outside the project assigned ITAR personal.

5.3 User gpg RSA key generation.

Start clean by moving the HOME/.gnupg directory aside or remove it if you do not have any outstanding dependencies on the current keys.

Create a default .gnupg by running **gpp --list-keys**. It should create the directory and default configuration files within.

Edit the HOME/.gnupg/gpg.conf file to replace the default HKP server to use keyserver hkp://debian-42.dw.local.

Generate an 2048 RSA key with your name, email, and position by **gpp --gen-key**.

Generate an RSA subkey for encryption and make it indefinite by **gpg --edit-key** *keyid* using the **addkey** command. Select the **RSA (encrypt only)** option. Select **2048** or larger bits. Select **0 = key does not expire**.

Send the key to the local hkp server debian-42.dw.local server **gpg --send-key** *keyid*.

5.4 PGP/gpg compatibility issues

The PGP utility used by DornerWorks and their customer's Windows computer uses the **.pgp** extension for an encrypted Zip file:

```
gpg -er customer -o proj_data.pgp proj_data.zip
```

Or to decrypt:

```
gpg -o proj_data.zip proj_data.pgp
unzip -l proj_data.zip
```

5.5 Getting vendor/customer keys from the key server.

To get keys submitted by the project use **gpg --search-keys** *search name*. Keys located on this server does not make them trusted.

5.5.1 Configuring a local group.

To create a group for encryption add a group command to your .gnupg/gpg.conf file:

group customer_group = 2E13B5F5 BB3D9AE5 D933B644 32021130

To encrypt to the group:

```
gpg -er customer_group data_file.pdf
```

5.6 Adding vendor/customer keys.

An ASCII public key is needed from the third party **gpg --import** < *name*.**asc** Then sign it for you and others to use **gpg --sign-key** *keyid*.

```
gpg --sign-key Tom
```

Send it to the local keyserver **gpg --send-key** *keyid*.

```
gpg --send-key 3438E13C
```

5.7 Deleting vendor/customer keys.

To delete a key out of your personal rings perform a **gpg --delete-key**, example:

```
gpg --delete-key Ty
```

5.8 Deleting vendor/customer from debian-42, the onak server.

To remove from the debian-42 computer running the onak server perform a **onak delete**. Note the **onak index***user* is handy to get the keyId.

```
onak index Ty
onak delete 438EB5F5
```

Output:

```
Type   bits/keyID    Date       User ID
pub   2048R/384C2245 2010/08/18 Ty Zoerner (Engineer) <Ty@InfiniteDelta.com>
```

5.9 Setting up an AES 256 partition

The following configures and AES-256 partition

```
apt-get install loop-aes-modules-2.6.26-2-686
apt-get install loop-aes-utils
```

Mount a USB flash drive to keep the secret key ‘‘/media/flash’’. To setup /dev/sda3 as an encrypted file system the fstab must include:

```
/dev/sda3       /opt            ext3    noauto,encryption=aes-256,gpgkey=/media/flash/.J32dse9.gpg   0 0
```

The following will prepare the partition:

```
head -c 3705 /dev/urandom | uuencode -m - | head -n 66 | tail -n 65 | gpg --symmetric -a > /media/flash/.
dd if=/dev/urandom of=/dev/sda3 bs=1M
losetup -e aes-256 -K /etc/opt.gpg /dev/loop0 /dev/sda3
mkfs.ext3 -v /dev/loop0
losetup -d /dev/loop0
mount /opt
```

5.10 Mounting an AES 256 partition

The system integrity must be checked before mounting an encrypted drive. Typically, the secured USB drive contains the script to mount the drive, and includes an md5sum check of the core system, and a check of the running processes. This is guard against an intruder between system boots, which a simple boot disk can alter the system.

Consider using the USB device as the only viable boot device to mount the encrypted drive.

5.11 Lost, incidents, and violations.

If any data falls outside the project document locations, an incident must be reported. It is important to include the condition of the data and any keys so the appropriate response can be determined.

5.12 Avoid requiring shred

If possible, avoid plans depending on the **shred** utility. Most modern file systems backup the files and recovering them can be a simple procedure. Secondly, the use of simple backup procedures also defeats the effectiveness of shred.