Security Reminder - using the Linux kernel in industrial projects

Nicholas Mc Guire

< der.herr@hofr.at >

Revision History	
Revision 0.1	2010-12-21
First release	

Table of Contents

1. Introduction

The Stuxnet virus incident should remind us that we need to know what is going on inside the software that we trust not only our economic livelihood on but also our safety. Open Source is not exempted in any way from use with malicious intent. It is your responsibility as developer to take precocious steps to minimize the likelihood of a malicious use of your industrial GNU/Linux system. Following are a few notes pertaining to securing the Linux kernel sources for your projects.

2. Downloading the kernel sources

First thing is to download your kernel - might sound a bit trivial to dedicate a section to it - but by the time you are done with this section, it should be clear why.

2.1. Downloading the kernel

Downloading the kernel can be achieved by grabbing one of the vanilla kernels from kernel.org as tar.bz2 file or via git.

The download is trivially achieved with **wget**, **ftp** or the like. Note that there are kernel mirrors in many countries all over the world so you should not be downloading from kernel.org unless that is really your fastest mirror. The kernel mirrors are ftp.COUNTRYCODE.kernel.org and all have the same structure; as this manual is being written in .cn, that is what we use.

```
rtl22:~# cd /usr/src
rtl22:~/usr/src\# wget ftp://ftp.cn.kernel.org/pub/linux/kernel/v2.6/linux-2.6.33.tar.bz2
rtl22:~/usr/src\# wget ftp://ftp.cn.kernel.org/pub/linux/kernel/v2.6/linux-2.6.33.tar.bz2.sign
```

Without further comments, here is the **git** command line to download the linux-2.6 kernel git - note that this is though a few 100 megabytes at the time of this writing. The git repository contains all kernels as of 2.6.11 including all rc (release candidates) releases. Updating it for a new kernel version is of course then a fast business once you have the clone:

```
git clone https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6
```

You can use the http protocol as well - but then you are not verifying where your download is actually coming from.

2.2. Verifying your kernel source

It is only a matter of time until Industrial Linux will become the target of attacks - if you grab sources on the Internet and don't verify there trust-worthiness, you are asking for problems. In proprietary software you have to trust the vendor - and the consequences can be grave as the current Stuxnet virus incident unfortunately demonstrates. Open Source can be modified by anybody, with any intention, so it is up to you to ensure trustworthy sources!

Once it is downloaded you should verify that this really is an authentic Linux kernel:

```
rtl22:~/usr/src\# gpg --verify linux-2.6.33.tar.bz2.sign linux-2.6.33.tar.bz2
gpg: Signature made Wed 09 Sep 2009 06:40:47 PM EDT using DSA key ID 517D0F0E
gpg: Can't check signature: public key not found
```

The error message shows that the key ID that is needed is 517D0F0E - the key ID is always a hexadecimal number - so below we prefix it with a 0x. The key needs to be queried from a public key server - now strictly speaking if someone bothered to spoof both then you would still be in trouble - but that is very unlikely that anybody would be willing to go through such troubles. Also you only need to download this kernel key once - it will stay the same in the future unless compromised (which you will here of if you are - as you should be - on the appropriate mailing lists). Note also that it is not uncommon to get timed out on public key servers - seems like there are too few to cover the needs. If you see something like

```
rtl26:/usr/src\# gpg --keyserver wwwkeys.pgp.net --recv-keys 0x517D0F0E
gpg: requesting key 517D0F0E from hkp server wwwkeys.pgp.net
gpg: keyserver timed out
gpg: keyserver receive failed: keyserver error
```

simply retry a few times - eventually you should see something like

```
rtl22:~/usr/src\# gpg --keyserver wwwkeys.pgp.net --recv-keys 0x517D0F0E
gpg: requesting key 517D0F0E from hkp server wwwkeys.pgp.net
gpg: key 517D0F0E: public key "Linux Kernel Archives Verification Key <ftpadmin@kernel.org>" imported
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0 valid: 2 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: Total number processed: 1
gpg: imported: 1
```

So now we got the key and we can verify the download:

```
rtl22:~/usr/src\# gpg --verify linux-2.6.33.tar.bz2.sign linux-2.6.33.tar.bz2
gpg: Signature made Wed 24 Feb 2010 02:16:18 PM EST using DSA key ID 517D0F0E
gpg: Good signature from "Linux Kernel Archives Verification Key <ftpadmin@kernel.org>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: C75D C40A 11D7 AF88 9981 ED5B C86B A06A 517D 0F0E
```

Note the WARNING is due to us not having verified that the key actually comes from the person claimed - as we have no way of verifying the key's authenticity any further; in consequence, we simply must trust the source. If this is critical, you would need to contact the key owner (i.e. by e-mail/fax) to get the fingerprint and then compare it. If you have an appropriate chain of trust through suitable certificates and PKI servers in your company, then they can make sure that they get a trusted key.

2.3. Locally signing

If your local IT department does not have a policy for signature management in place, then the following is a weaker but still recommendable alternative to simply accepting the above warning.

As the Linux kernel key is in wide use and it certainly would make the headlines if compromised, you can be quite sure that it is valid by checking it on multiple mirror sites. Assuming that you download the key at the start of some project - not immediately when you need it - and then store it locally to verify kernels you download, it is reasonable to create a locally copied and non-exportable signature.

So let's check the signature of the ftpadmin of kernel.org. First we list it:

```
rtl26:/usr/src\# gpg -ka ftpadmin@kernel.org
pub 1024D/517D0F0E 2000-10-10
uid Linux Kernel Archives Verification Key <ftpadmin@kernel.org>
sub 4096g/E50A8F2A 2000-10-10
```

Then we download that key file:

```
rtl16:/usr/src\# gpg --keyserver wwwkeys.pgp.net --recv-key 0xE50A8F2A
```

and now you can locally sign the key or you can verify the key via e-mail/fax/web - verifying via web is, of course, the weakest - but if you can do it from a different network, it also helps to establish trust.

```
rtl26:/usr/src\# gpg --edit-key ftpadmin@kernel.org lsign
gpg (GnuPG) 1.4.9; Copyright (C) 2008 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
pub 1024D/517D0F0E created: 2000-10-10 expires: never usage: SCA
trust: unknown validity: unknown
sub 4096g/E50A8F2A created: 2000-10-10 expires: never usage: E
[ unknown] (1). Linux Kernel Archives Verification Key <ftpadmin@kernel.org>
pub 1024D/517D0F0E created: 2000-10-10 expires: never usage: SCA
trust: unknown validity: unknown
Primary key fingerprint: C75D C40A 11D7 AF88 9981 ED5B C86B A06A 517D 0F0E
Linux Kernel Archives Verification Key <ftpadmin@kernel.org>
Are you sure that you want to sign this key with your
```

```
key "Nicholas Mc Guire <der.herr@hofr.at>" (DEE15405)
The signature will be marked as non-exportable.
Really sign? (y/N) y
You need a passphrase to unlock the secret key for
user: "Nicholas Mc Guire <der.herr@hofr.at>"
1024-bit DSA key, ID DEE15405, created 2010-02-02
Command> q
Save changes? (y/N) y
```

If you now verify the kernel source, then you should no longer get any WARNING from gpg. Any malicious source or signature file can now be detected with reasonable reliability:

```
/rtl26:/usr/src# gpg --verify linux-2.6.33.tar.bz2.sign
gpg: Signature made Wed 24 Feb 2010 02:16:18 PM EST using DSA key ID 517D0F0E
gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0 valid: 4 signed: 1 trust: 0-, 0q, 0n, 0m, 0f, 4u
gpg: depth: 1 valid: 1 signed: 0 trust: 1-, 0q, 0n, 0m, 0f, 0u
gpg: Good signature from "Linux Kernel Archives Verification Key <ftpadmin@kernel.org>"
```

This all might sound a bit paranoid - but security of industrial Open Source software starts at establishing trustworthy sources from the very outset - adding this small effort will not provide 100% security but, nevertheless, a clear increase.