Using SELinux security enforcement in Linux-based embedded devices

Björn Vogel
nomiko e.V.
Bochum, Germany
Bjoern.Vogel@uni-dortmund.de

Bernd Steinke
Nokia Research Center
Bochum, Germany
Bernd.Steinke@Nokia.com

ABSTRACT

This contribution describes how Security Enhanced Linux (SELinux) is enabled on a Nokia 770 Internet Tablet. It will refer to a procedure done under a Debian [1] Testing (Etch) Environment. The procedure will also be possible under other Linux distributions but since Maemo is built upon Debian, this approach is the most preferrable way to extend Maemo Linux. An SELinux enabled device will provide the possibility of a convenient configuration of the device. Different stakeholders can define detailled access control to the components they maintain. This ensures the interests of the stakeholders by providing the benefits of a Linux based embedded device.

Keywords

selinux, security, embedded system

1. INTRODUCTION

Nowadays, the acceptance and the need of mobile systems is growing constantly. Nearly everyone is equipped with some kind of system storing some of his personal data like phone numbers, important contacts, appointments and other information. These systems can be mobile phones, PDAs, and other embedded systems which are growingly interconnected. Along with these interconnections the kind and amount of available services are growing as well. These growing amount makes it necessary to allow several service providers to maintain the settings for their services, since in the majority of cases the user is not interested in getting involved in this configuration process. The other way round, providers are interested in a protection against specific modifications of configurations of e.g. GSM/UMTS network settings by the user. So, with the increasing amount of used services and the need for access to configure these services, this means that many different actors need to have access to the device and each actor needs his specific area of access which is protected against manipulation by other actors.

MOBILWARE 2008, February 13-15, Innsbruck, Austria

DOI 10.4108/ICST.MOBILWARE2008.2927

Another trend is the usage of Linux-based operating systems. The benefits such as the flexibility, the stockpile of solutions and the open source paradigm open up advantages for both manufacturers and customers.

On the other hand, the classical Linux systems are not capable of providing the security mechanisms required by the new demands of these multi-user environments. In this context SELinux offers the possibility of a more fine grained multi domain access control.

In special, Security Enhanced Linux (SELinux) [12] is an enhancement for the standard Linux Kernel, which implements fine-grained Access Control based on the FLASK concept [13], [14]. The FLASK concept architecture was developed in the early '90s by the NSA and others. It which provides Mandatory Access Control.

In contrast to the classical Linux access control with its *owner, group, others*-mechanism, SELinux attaches a special collection of security attributes to each subject and object within the system. It uses the mechanisms of Role Based Access Control (RBAC) and Type Enforcement (TE) to decide access requests.

Until now, there was no embedded device running SELinux and so it will be described how a first proof-of-concept was created on a Nokia 770 Internet Tablet (N770). The N770 is one of the first completely Linux-based mobile computers. It is running Maemo [10], a Debian based Linux distribution.

2. PROBLEM DESCRIPTION

Although SELinux is currently available for and within most of the bigger Linux distributions (like Debian, Fedora Core, Ubuntu, etc.) SELinux is currently not included within the Linux distribution of the Nokia 770 Internet Tablet. The distribution used by this device is Maemo. It is slightly changed to the specific design properties of the N770.

One main difference between PC Systems where SELinux is mainly used and the Nokia 770 Internet Tablet is the changed boot procedure. There is no boot loader like lilo [6] or grub [4], which provides the capability of passing boot options to the kernel. This is a problem since SELinux can be switched on and off via boot parameters, and when it is turned on (enabled) it is possible to set the mode it should run in. The first one is the enforcing mode. This is the mode where SELinux enforces all given policies without any com-

promises. This enables full functionality and the maximum of security which can provided by SELinux. This mode is very restrictive and should only be used when the policy set used is proven 100 percent correct and working. During development this is a demand that can not be guaranteed. So another mode should be used, which is the permissive mode. The permissive mode indeed does regards the policy and still checks the policy for every access, but decisions whether an access should be granted or not, are not enforced. If an access is actually denied within the policy, permissive mode SELinux only generates an error message telling that the access should be denied according to the policy, but grants it anyway. So, no access is denied, but all accesses that actually should be not allowed by the policy generate error messages, helping in policy configuration. For more comfortable configuration this bootline parameter is generally passed to the Linux kernel by the boot loader. The Maemo boot loader is not capable of this, so a solution had to be found.

Another main problem is that SELinux decides access requests upon types. These types have to be attached to the files permanently in some way. Within the current SELinux implementation this is realized by writing this information into the Extended Attributes (xattr) [7] of a file or directory. These attributes are supported for the ext2, ext3, jfs and xfs file systems. Unfortunately the N770 is using the jffs2 file system which is officially not supported.

Furthermore, Maemo does not bring along all needed applications and application changes needed to run SELinux properly. For example, there are some new applications like the command line tool `newrole`, which will be required to change the role when using text shells.

Also some applications need source code changes to have SELinux-specific information available. For example, to see the types attached to the files mentioned before, the `ls` user land application is originally not able to show this information.

3. SOLUTION: PORTING

3.1 File system modifications

This section describes which modifications were made to the Maemo O/S 2006 development root file system which was used as a basis and expanded to have SELinux capabilities available on the N770. A list of packages that were created by downloading and recompiling the corresponding Debian (Etch) packages from the Debian homepage [2] follows. The packages were recompiled within the scratchbox [11], a cross compilation toolkit. The scratchbox was supplied with an appropriate compiler (cs2005q3.2-glibc-arm) and a rootstrap for the N770 (Maemo_Dev_Platform_v2.0_armel-rootstrap.tar [9]). Not all of the packages listed below are absolutely necessary to enable SELinux. They are only mentioned for a more convenient development process, so it is not the minimum set of packages needed. The packages are the following

- **bash:** To have a real "bash"-shell available on the device. Other packages required a real bash since included scripts did not work with the shell built in the busybox which is the standard in the Maemo root file system.
- **checkpolicy:** This is the SELinux policy compiler, which is needed to compile policies to the binary form used by the kernel. This is an essential package, since it is a needed part of the SELinux infrastructure.
- **coreutils:** As the name lets presume, a collection of core utilities for Linux. Contains programs like `ls, cp, mv, rm, echo`, etc. Some of them needed changes or enhancements for SELinux. For example `bin/ls` and `usr/bin/id` which both obtained the new option `-Z` to show SELinux specific information (for `id`: user context; for `ls`: attached file contexts).
- **find:** To have a real `find`-program available. The busybox-built-in `find` lacked of some parameters that where needed by scripts from the *checkpolicy* package to work correctly.
- **libacl:** Access control list shared library. Required by *coreutils*.
- **libattr:** Shared library to enable support for xattr in file system. Required by *libacl*.
- **libdb:** Berkeley v3 Database runtime Libraries.
- **libpam0g:** Required by *policycoreutils*, although PAM-Authentication is not used.
- **libpam-modules:** Pluggable Authentication Modules for PAM. Needed by *passwd*, although PAM- Athentication is not used.
- **libselinux1:** Essential SELinux shared libraries, providing an interface for security-aware applications.
- **libsemanage1:** Shared libraries, providing an interface for SELinux management. Used by SELinux policy manipulation tools.
- **libsepol1:** SELinux policy library for changing policy binaries.
- **libslang:** S-Lang programming library
- **login** Replacing the busybox-built-in login program by fully functional one.
- **m4:** Implementation of the traditional UNIX macro processor. SELinux policy compilation bases upon m4-macros.
- **make:** GNU version of the `make` utility. Used for SELinux policy compilation.
- **mc:** GNU Midnight Commander: a text-mode full-screen file manager for easier navigation. Not absolutely necessary, but very helpful.
- **passwd:** Programs to maintain password and group data; needed to have fully functional passwd programm available, which is aware of shadow password support.

- **policycoreutils:** Contains the core policy utilities that are required for basic operation of an SELinux system. These utilities include `load_policy` to load policies, `setfiles` to label filesystems, `newrole` to switch roles, and `run_init` to run /etc/init.d scripts in the proper context. Requires *python* and the according selinux/semanage extensions.
- **procps:** Utilities to browse the `/proc` filesystem; like `ps`, which was extended to show selinux contexts via option `-Z`.
- **python2.4-selinux:** Python bindings to SELinux shared libraries. Required by *policycoreutils*.
- **python2.4-semanage:** Python bindings for the manipulation of SELinux binary policies, used for policy compilation and file system relabeling. Required by *policycoreutils*.
- **selinux-policies:** A basic policy providing ruleset for minimal functionality of SELinux-enabled N770. This package was modified like described in section 3.3 - *SELinux policy changes for N770*.
- **ssh:** To have a fully functional and SELinux-aware SSH-daemon available.
- **sysvinit:** Important package since a SELinux aware init program is needed to bring started programs to correct context.
- **tar:** To have a fully functional tar available on device.

These packages were all cross-compiled within the scratchbox from Debian (Etch) sources. No changes were made to these sources, except when mentioned.

Things look different on kernel source code. There were modifications that were necessary and these will be described next.

3.2 Kernel changes

There were two things to do to make the kernel SELinux ready. First, the support for xattr had to be compiled into the kernel and second, SELinux options hat to be switched on.

Since the main SELinux capabilities are already included in the official 2.6.16 kernel sources, the changes that need to be made to the kernel sources are changes of the jffs2 file system drivers. SELinux needs to have xattr enabled for the used filesystem. This capability is not provided for the jffs2 file system in the official kernel sources. To have xattr available for the file system of the N770, these changes were derived from a patch [5].

The kernel was installed within the scratchbox as described under [8]. Then the patch was applied and the kernel was recompiled with altered kernel configuration as described in figure 1. It shows the differences between the kernel configuration `n770_defconfig` in the original maemo kernel sources and the configuration used to enable SELinux. '<' marks the old line within the kernel configuration file, '>' marks the new line, '>'-lines without corresponding '<'-line were not present in the original configuration. Recompiling the kernel after patching the jffs2 driver changes and using the altered configuration is sufficient to make the kernel SELinux ready.

3.3 SELinux policy changes for N770

This section describes the SELinux policy changes used within the device to show the proof of concept. The starting point of this policy is the Debian package 'selinux-policy-default'. It brings along sample policy files for a basic configuration and many commonly used programs where each program has its own configuration files.

3.3.1 Reducing policy size

The first step that was required for an embedded device was the reduction of the amount of provided configuration files. It was checked which programs are running on the device and which configuration files were definetely not needed. In case of doubt the configuration files were left untouched, since they do not disturb and to ensure nothing important is missing. These files were removed to reduce the total amount of rules and so the total size and computiation complexity of the later compiled binary policy. The motivation for this step is that embedded and mobile devices in general do not sport large amounts of memory and disk space available. So, on the one hand reducing the amount of configurations reduces the amount of rules created by the SELinux compiler and so the size of the binary representation of the policy and on the other hand it provides a clearer view to the files used for the policy.

3.3.2 Reworking the policy rules

The second step was the elimination of denied messages that appeared during accesses when working with the N770. These accesses were not covered by the policy. The compiled packet *policycoreutils* contains a script named `audit2allow` that can be used to convert denying SELinux audit messages to rules that will allow the operation that caused the error when the created rule will be included in the policy. This script was very useful to eliminate errors during bootup, runtime and shutdown. This may be a comfortable procedure, but should be used with caution and implementing rules without more precise analysis what is allowed with this rule can lead to security holes.

3.3.3 Telling SELinux jff2s xattr support is available

Until now it was only possible to assign a static context to the whole jffs2 file system at mount time as possible with all mountpoint of unsupported file systems. So, another important thing to do was to tell SELinux that it should use xattr for jffs2 since it did not know yet, that xattr support is now also available for jffs2. To do so, the lines marked with '>' had to be added in `usr/share/selinux/policy/beta1.0/fs_use`, lines marked with a leading '<' then had to be removed:

```
  fs_use_xattr jfs system_u:object_r:fs_t;
> fs_use_xattr jffs2 system_u:object_r:fs_t;
  fs_use_xattr reiserfs system_u:object_r:fs_t;
```

and in `usr/share/selinux/policy/beta1.0/Makefile` a "jffs2 | " has to be added to the regular expression of the

```
< # CONFIG_AUDIT is not set
> CONFIG_AUDIT=y

< CONFIG_OMAP_RESET_CLOCKS=y
> # CONFIG_OMAP_RESET_CLOCKS is not set

< CONFIG_CMDLINE="root=1f03
   rootfstype=jffs2 time"
> CONFIG_CMDLINE="root=1f03
   rootfstype=jffs2 time selinux=1
   enforcing=0 audit=1"

< CONFIG_EXT2_FS=m
> CONFIG_EXT2_FS=y

< CONFIG_EXT3_FS=m
> CONFIG_EXT3_FS=y

< CONFIG_JBD=m
> CONFIG_JBD=y

< CONFIG_FS_MBCACHE=m
> CONFIG_FS_MBCACHE=y

< # CONFIG_FS_POSIX_ACL is not set
> CONFIG_FS_POSIX_ACL=y

< CONFIG_JFFS2_SUMMARY=y
<
<
<
> # CONFIG_JFFS2_SUMMARY is not set
> CONFIG_JFFS2_FS_XATTR=y
> CONFIG_JFFS2_FS_POSIX_ACL=y
> CONFIG_JFFS2_FS_SECURITY=y

< # CONFIG_DEBUG_FS is not set
> CONFIG_DEBUG_FS=y

< # CONFIG_DEBUG_LL is not set
> CONFIG_DEBUG_LL=y

< # CONFIG_SECURITY_NETWORK is not set
> CONFIG_SECURITY_NETWORK=y

> CONFIG_SECURITY_SELINUX=y
> CONFIG_SECURITY_SELINUX_BOOTPARAM=y
> CONFIG_SECURITY_SELINUX_BOOTPARAM_
  VALUE=1
> CONFIG_SECURITY_SELINUX_DEVELOP=y
> CONFIG_SECURITY_SELINUX_AVC_STATS=y
> CONFIG_SECURITY_SELINUX_CHECKREQPROT_
  VALUE=1
```

Figure 1: Kernel configuration differences

file system list:

```
< FILESYSTEMS=‘mount | grep -v "context=" |
   egrep -v ’\(((|.*,)bind(,.*|)\)’ |
   awk ’/(ext[23]| xfs| jfs| reiserfs).
   *rw/{print $$3}’;‘
> FILESYSTEMS=‘mount | grep -v "context=" |
   egrep -v ’\(((|.*,)bind(,.*|)\)’ |
   awk ’/(ext[23]| xfs| jfs| jffs2| reiserfs).
   *rw/{print $$3}’;‘
```

With this ruleset a basic operation of SELinux on a N770 is realized.

4. EVALUATION

The goal of this work was to provide a more fine grained security mechanism on a Linux based Internet Tablet. With having SELinux available on the device it is now possible to specify more detailed policies for access operations. A main difference is for example the possibility of enhancing a programs rights beyond the ones of the user. In contrast, in classical Linux a program launched by the user under normal conditions usually has exactly the same rights as the user has. There a several scenarios imaginable, where these mechanisms aren't sufficient any more. For example, it is usually not eligible that any program the user is launching should have access to *all* of the files the way the user has; why should an internet browser be able to read a user's telephone contacts stored somewhere where the user has access to?!

Another scenario would be a program provided by e.g. a service provider using a secret encryption key the user should not know. Under classical Linux it is not possible to for a user to launch an application that can read the encryption key stored in some file as long as the user is also not able to do so.

Since the standard Linux security mechanisms where not touched, SELinux provides an additional protection against unauthorized operations. The problems of the two scenarios described above can be elegantly handled using SELinux. Very private data can be protected by simply assigning a special type to these files and granting access only to some programs of the correct domain (type). Since in SELinux it is possible to transfer applications launched by a user to a special domain (type), the access rights to files with sensitive content can be different to the user and the domain the application is running in. So the program can access files the user does not have access to.

These mechanisms introduce some overhead in access, computation and reaction time. A performance impact on the Nokia 770 could subjectively not be noticed. More detailed evaluations are yet to be done. The last measures made concerning influence of SELinux on system behaviour showed that there was a performance loss of 7% on a system running SELinux [15], [16]. This number is heavily depending on system tuning and usage. It may be better or worse in some cases but may be also reduced by lean policies. Several changes have been made since last measures and further

performance optimization and new measures should be done as future work.

Additional efforts should also be spent in patching all user space and X-Window programs running on the device against SELinux mechanisms to provide full protection in all use cases. This means also to rework the policies for all applications.

Another open point is the detailled analysis of the Nokia 770's special architecture to find out how the problem with the DSME tool can be solved.

4.1 Open issues and further open work

A main problem is the special architecture of the Nokia 770 Internet Tablet. A tool called DSME (Device State Management Entity) is responsible for the launch of several daemons on startup. Since this program could not be pulled off the kernel context due to architectural situation, no correct rules could be created for daemons that need to be started by this tool. The tool is interweaved deeply with the architecture of the N770. Since it's not currently released under a free license, no further information on how to get it into correct context could be provided. The tool is always launched in kernel context and all daemons launched by this tool are also.

Having these daemons running in kernel context makes the policy definition very insecure. The kernel and - even worse - all programs also running in kernel context, would get all the rights of the mentioned daemon running in this context. E.g. if a program is accidently running in kernel context and has to access a file of context `system_u:object_r:very_confident_t` a rule has to be, that a process of `system_u:system_r:kernel_t` context is allowed to access a file of context `system_u:object_r:very_confident_t`. This would give the kernel and all other programs accidently running in this context, access to files of type `system_u:object_r:very_confident_t`, what is not eligible. Eliminating this architectural characteristic with the DSME tool would simplify rule creation and increase security.

5. CONCLUSIONS

A solution was presented to make a more fine grained access control possible on an embedded device running Linux. SELinux was used as an enabler to overcome the limits of the standard Linux security mechanisms and to provide a more flexible way to define access rights to users and applications.

The boot line was hardcompiled into the kernel and so the enabling and run mode of SELinux is not changeable any more but by flashing a new kernel. The initual problem of not being able to pass boot line parameters to the kernel is even an advantage in regards of security aspects since it is not possible to disable SELinux but by flashing a whole new kernel.

The xattr functionality was ported to support the jffs2 file system of the device to make assignment of types to files and directories possible. As this assignment is fundamentially required for file access operations to provide SELinux functionality, this was a major important task to solve.

Also the SELinux specific libraries and userspace program modifications (like for `ls` and `id`) needed to be ported to the Nokia 770 Internet Tablet. They were cross-compiled for the device to provide the needed SELinux functionalities.

The SELinux policies provided for Debian systems were slightly modified to fit the device's peculiarities.

This work provides the means and describes the specific solution to realize a protected multiple domain security concept with fine grained access control on the embedded Nokia 770 Internet Tablet.

Acknowledgment

This work was performed in project E2R II [3] which has received research funding from the Community's Sixth Framework programme. This paper reflects only the authors' views and the Community is not liable for any use that may be made of the information contained therein. The contributions of colleagues from E2R II consortium are hereby acknowledged.

6. REFERENCES

[1] Debian linux website, http://www.debian.org/.
[2] Debian package website, http://www.debian.org/distrib/packages.
[3] E^{2}R2 End-to-End Reconfigurability II, http://www.e2r2.motlabs.com/.
[4] Gnu grub website, http://www.gnu.org/software/grub/.
[5] Kaigai.gr.jp xattr-patch, http://www.kaigai.gr.jp/pub/jffs2-xattr-v6-backport-into-2.6.16.patch.
[6] Lilo websitem, http://lilo.go.dyndns.org.
[7] Linux extended attributes and acls website, http://acl.bestbits.at/.
[8] Maemo kernel compilation how-to, http://maemo.org/maemowiki/howto_kernelcompilation.
[9] Maemo repository, http://repository.maemo.org/stable/2.0/armel/.
[10] Maemo website, http://www.maemo.org.
[11] scratchbox.org website, http://www.scratchbox.org/.
[12] Security-enhanced linux website, http://www.nsa.gov/selinux/.
[13] S. Smalley. Flask: Flux advanced security kernel, 2000.
[14] R. Spencer, S. Smalley, P. Loscocco, M. Hibler, D. Andersen, and J. Lepreau. The Flask security architecture: System support for diverse security policies. pages 123–139.
[15] The unofficial selinux faq, http://sf.net/docman/display_doc.php?docid=14882&group_id=21266#w ww.14, 2005.
[16] K. Wade, C. Sellers, and F. Tombolini. Fedora core 5 selinux faq, http://fedora.redhat.com/docs/selinux-faq-fc5/#id2965028, 2006.