Extending SELinux to meet LSPP data import/export requirements

Janak Desai[1], George Wilson[1], Chad Sellers[2]
janak@us.ibm.com, ltcgcw@us.ibm.com, csellers@tresys.com

[1]*Linux Technology Center, IBM Corporation*
[2]*Tresys Technology*

Abstract

Common Criteria certification of SELinux at Evaluation Assurance Level 4 against the Labeled Security Protection Profile(LSPP)[1] and Role-Based Access Control Protection Profile(RBACPP)[2] is intended to advance its acceptance and deployment in the federal sector. SELinux already provides a flexible security policy infrastructure upon which systems that conform to hierarchical Multi-level Security (MLS), as required by LSPP, and role-based access control security policies may be built. However, some RBACPP and LSPP requirements in the user data protection category and their effects on usability make support for features such as polyinstantiated directories, multi-context aware cron, and data import/export restrictions based on device security attributes desirable. This paper presents proposed extensions to SELinux to satisfy some of the LSPP data import/export and RBACPP requirements for Common Criteria certification at EAL4, while maintaining a functional and usable system.

1 Introduction

SELinux, with its flexible and powerful Mandatory Access Control (MAC) architecture, provides a full set of features to create systems that implement a Bell-La-Padula(BLP)[3] hierarchical Multilevel Security(MLS) policy. The Common Criteria Labeled Security Protection Profile (LSPP) largely focuses on the BLP policy. However there are certain specific LSPP requirements that SELinux currently does not meet. For example, LSPP requires that terminal devices have sensitivity labels, and that system enforce MAC policy when a user, with a certain sensitivity label, attempts to login on a particular terminal device. In addition, there are LSPP requirements that, when enforced, detract from the usability of SELinux systems. For example, the BLP *-Property adversely affects usability of common public use directories such as */tmp* and */var/tmp*, and user home directories. SELinux must be extended both to certify it at Common Criteria Evaluation Assurance Level 4 with respect to LSPP and RBACPP, and to improve its usability.

To address the problem outlined above, the authors propose extensions to three SELinux components. The first satisfies an LSPP functionality currently missing from SELinux, while the other two improve the reduced usability incurred by meeting the BLP *-Property. The proposed extensions are to the SELinux PAM module, support for polyinstantiated directories, and a multi-context aware cron subsystem. Changes to the SELinux PAM module prevent a user from logging into a terminal whose sensitivity label range falls outside the sensitivity label with which the user is trying to login. A polyinstantiated directory provides an instance of a directory at an appropriate security context as defined by the security policy. Polyinstantiated directories allow processes with different security contexts transparent access to common directories. An extension to the cron subsystem allows a user to create and execute cron jobs from different security contexts. Such a multi-context aware cron subsystem allows a user to submit cron jobs after changing role or sensitivity level.

These proposed extensions represent about 15% of the development work required to meet LSPP and RBACPP functionality. This paper does not describe other ongoing LSPP and RBACPP related development work for subsystems such as audit, network, device allocation, and print.

Section 2, describes the problems that these three proposed extensions attempt to solve. Section 3 provides high level design of the extensions and Section 4 provides implementation details. The final section summarizes findings.

2 Problem

This section defines the problem to be solved, namely extending SELinux to meet certain LSPP and RBACPP requirements while maintaining a functional and usable system. It first describes the LSPP data import/export, the BLP *-Property, and the RBACPP access control requirements. Then it examines the impact of the *-Property on usability and outlines likely SELinux extensions that should be made to lessen that impact.

2.1 LSPP data import/export requirement

LSPP states that input/output devices such as login terminals must be included in the list of objects that must have sensitivity labels associated with them. Data import/export requirements, FDP_ETC and FDP_ITC, require that MAC policy is enforced on all subjects and objects. Thus a user should not be able to login to a terminal with a sensitivity label that falls outside the range of the terminal sensitivity label. Current login on SELinux changes the sensitivity label of the terminal to match the sensitivity label with which the user is logging in, thus ignoring the original sensitivity label of the terminal. This paper presents a proposed extension to the SELinux PAM module that prevents login from changing the label of the terminal if the login sensitivity label falls outside the allowed terminal sensitivity label range.

2.2 *-Property and public directories

The BLP *-Property as specified in LSPP section FDP_IFF.2 states that, if the sensitivity label of a subject is greater than the sensitivity label of the object, only a read operation is permitted. On SELinux, there are public directories, such as */tmp* and */var/tmp*, to which processes running at different sensitivity labels expect to be able to write. Similarly, if a user home directory is at a particular sensitivity level and the user logs in at a higher sensitivity level (which is allowed so long as the level is dominated by the user clearance) will not be able to write in his or her home directory. This effect of the *-Property results in diminished usability vis-a-vis a traditional SELinux system. A new extension to SELinux adds support for polyinstantiated directories to improve MLS usability. A polyinstantiated directory provides an instance of the directory at a suitable sensitivity level that transparently allows write access to that directory.

2.3 RBACPP and multi-context cron

RBACPP section FIA_USB.1 states that the system must associate appropriate user security attributes with subjects acting on behalf of the user. One such security attribute is the role assumed by the user. Currently, cron under SELinux allows users to submit jobs while assuming different roles. However, the cron job is executed in a derived context controlled by the system security policy. An extension to cron gives it the new ability to execute cron jobs for a user while assuming the role under which the user submitted the job.

3 Design

This section describes the high level design of the tree proposed extensions to SELinux.

3.1 Data import/export on terminals

Input/output devices such as interactive terminals are assigned sensitivity label range by a system administrator. LSPP import/export restrictions specify that a user cannot login to a terminal with a sensitivity label that falls outside the terminal sensitivity label range. Currently login and other session-establishing programs set the sensitivity label of the login terminal to match the sensitivity label at which the user is logging in via Pluggable Authentication Modules (PAM). The proposed approach to restrict a user from logging in with a sensitivity label outside the terminal sensitivity label range enforces the restriction by preventing the SELinux PAM module from setting the sensitivity label of the terminal. The SELinux PAM module calls `getfilecon()` to extract the security context of the login terminal and relabels the terminal with the security context returned by the `security_compute_relabel()`. With the proposed extension, the SELinux PAM module extracts the sensitivity label range from the security context of the terminal and ensures that the sensitivity label with which the user is trying to login, falls within the terminal sensitivity label range. If the login sensitivity label is outside the terminal sensitivity label range, the SELinux PAM module returns an error and prevents login from succeeding.

3.2 Polyinstantiated directories

A polyinstantiated directory presents an instance of a directory based on process and directory security contexts. Legacy MLS operating systems implemented polyinstantiated directories by making intrusive changes to kernel path name translation code. The SELinux kernel provides per-process namespace instances[4]. Therefore it is feasible to implement polyinstantiated directories using per-process namespaces as proposed on the SELinux mailing list by Stephen Smalley.

The main concept is to create polyinstantiated member directories with different security contexts depending on the process security context, the polyinstantiated directory security context and the security policy, and bind mount an appropriate member directory on top of the polyinstantiated directory. This gives the process a transparent access to a directory where it can create/delete directory entries, while preventing it from viewing directory entries with a different context to which it does not have access. The bind mount process changes the process namespace. Hence, each process that accesses polyinstantiated directories must possess its own namespace. A new system call, unshare, allows a process to disassociate from namespace shared with other processes. The mechanism operates as follows:

1. A new PAM session module, pam_namespace isolates the polyinstantiation mechanism to a single location. Session- establishing programs such as login and sshd are configured to use this PAM module.
2. The pam_namespace PAM module consults the module configuration file to get the list of polyinstantiated directories on the system. If polyinstantiated directories are configured, the module invokes the `unshare()` system call to disassociate from any shared namespace.
3. For each directory to be polyinstantiated, the PAM module then invokes `security_compute_member()` to obtain security context of the member directory from policy, computes its name by generating a hash of the security context, and creates the member directory, with appropriate security context, if it doesn't exist.
4. Bind mounts the polyinstantiated directory under a different name to allow future access, as allowed by policy, to security-aware applications.
5. Bind mounts the appropriate member directory on top of the original polyinstantiated directory.

Whenever a process security context is changed (for example through su or newrole), the command accesses the polyinstantiated directory from its alternative location and rebinds the original location to the appropriate member directory based on the new process security context.

3.3 Multi-context aware `cron`

The cron subsystem, as it exists on SELinux, creates a per user cron job entry in the */var/spool/cron* directory. The cron daemon traverses the */var/spool/cron* directory and processes jobs submitted by different users. Before performing a job on behalf of a user the daemon transitions to a context for the user that is reachable from the daemon's own context. Thus, the cron security policy can run cron jobs in a derived domain, giving it different permissions than a user session. This existing scheme does not provide a way to differentiate sensitivity label or role (although roles are tightly coupled to types that can be controlled through the system security policy). A new extension to the cron protocol captures the user security context, which includes the role and sensitivity label in addition to user ID when creating cron jobs. With the availability of the full security context, the cron daemon sets the appropriate security context while processing a cron job for a user.

4 Implementation

This section provides implementation details, some of which are still being finalized in collaboration with the SELinux open source development community, for the proposed three extensions to SELinux to meet certain LSPP and RBAC protection profile requirements while maintaining a function usable system.

4.1 Data import/export on terminals

Proposed extension to data import/export on terminals consists of two changes to the SELinux PAM module. The first one changes the `security_label_tty()` to return `NULL` when the login sensitivity label falls outside the terminal sensitivity label range and the second one changes `pam_sm_open_session()` to return error if `security_label_tty()` is unable to change the terminal security context.

4.1.1 Modification to `security_label_tty()`

SELinux PAM module calls `getseuserbyname()` and `get_ordered_context_list_with_level()` to setup the security context with which to login a user. If appropriate security context is not found from the configuration files, `manual_context()` is called to obtain the context by interacting with the user. `security_label_tty()` is then invoked with this security context to appropriately label the terminal. The proposed extension modifies `security_label_tty()` to return NULL if login sensitivity label does not fall within the terminal sensitivity label range.

4.1.2 Modification to `pam_sm_open_session()`

Current version of the SELinux PAM module continues with the login process even if `security_label_tty()` returns NULL. With the proposed extension, the `pam_sm_open_session()` returns an error to the calling program if `security_label_tty()` is unable to label the login terminal, thus preventing login from succeeding.

4.2 Polyinstantiated directories

Polyinstantiated directories consists of three different components. A new system call, unshare, a new PAM module, pam_namespace, and modifications to the SELinux command, newrole.

4.2.1 New system call, `unshare`

unshare system call allows a process to selectively disassociate parts of its context that are currently being shared with other processes. That is, unshare reverses process context sharing that was setup using `fork()` or `clone()` system call. Parts of the process context to unshare are specified as an unsigned integer argument computed by performing bitwise or operation on their corresponding clone flags. At a high level, unshare follows the following algorithm:

- Check system call argument. Silently force unsharing of implied context. For example, if `CLONE_NEWNS` is specified to unshare namespace, then force `CLONE_FS` since unsharing of namespace requires unsharing of file system information as well.
- For each context structure that is being unshared, allocate the corresponding new structure and initialize it with values from the current shared version of the context.
- Lock the current task structure, associate newly allocated and duplicated context structures with appropriate fields of the current task structure, and release the lock on the current task structure.
- Appropriately release older, shared, context structures by decrementing reference counts and freeing memory, if reference count goes to zero.

4.2.2 New PAM module, `pam_namespace`

pam_namespace module sets up a private namespace with polyinstantiated directories for a session managed by PAM. pam_namespace recognizes two arguments, namely debug and unmnt_first. debug provides extensive logging through syslog and unmnt_first is used by programs such as su and newrole that have to undo previous polyinstantiation before polyinstantiating based on the new user ID and/or security context. pam_namespace module consults */etc/security/namespace.conf* configuration file and sets up a private namespace with polyinstantiated directories. Each line in the namespace configuration file describes a directory to polyinstantiate in the form:

```
<polydir> <mem_path> <reset_path> <method> <id>
```

Where:
`<polydir>` is the absolute pathname of the directory to polyinstantiate. Special entry `$HOME` is supported to designate user's home directory. This field cannot be blank.

`<mem_path>` is the absolute pathname of the parent directory where an instantiation of `<polydir>` is to be created. This instance is bind mounted on the `<polydir>` to provide an instance of `<polydir>` based on the `<method>` column. Two special entries `$HOME` and `$HOME_PARENT` are supported to designate user's home directory and parent of user's home directory. This field cannot be blank.

`<reset_path>` is the absolute pathname of the parent directory where the "reset" version of the `<polydir>` is to be available. "reset" version of `<polydir>` provides original content of `<polydir>` to authorized programs. Special entry `$HOME_PARENT` is supported to designate parent of user's home directory. This field cannot be blank and this directory cannot be a subdirectory of `<polydir>`.

`<method>` is the method used for polyinstantiation. It can take 3 different values; "user" for polyinstantiation based on user name, "context" for polyinstantiation based on process security context, and "both" for polyinstantiation based on both user name and security context. Methods "context" and "both" are only available with SELinux. This field cannot be blank.

`<id>` is a comma separated list of user names for whom the polyinstantiation is not performed. If left blank, polyinstantiation will be performed for all users.

To create polyinstantiated private namespace for commands that establish a new user session, such as login, ssh, su, gdm, and newrole, the following line is inserted in their corresponding PAM configuration file:

```
session required /lib/security/pam_namespace.so
[unmnt_first]
```

4.2.3 Modifications to `newrole`

Current version of newrole does not call PAM session management functions. The proposed extension to newrole invokes PAM session management functions from the child process after the new security context is set and before executing the shell. Intervention by PAM session management functions allow the system administrator to appropriately setup polyinstantiation for desired user ID and/or security contexts that are honored both by newrole and commands such as login, ssh, and gdm.

4.3 Multi-context aware `cron`

The current cron subsystem creates a per-user cron job entry as */var/spool/cron/<user_name>*. When the job is processed, the job file name, which is the name of the user that submitted the job, is used to construct the security context with which the job is executed. Under the current scheme, only the user name is used to create a cron job, making it impossible for the cron daemon to duplicate user's role and sensitivity level under which the cron job was submitted. With the proposed extension, user's security context is appended to the user name and the cron job is created as */var/spool/cron/<user_name+user_security_context>*. When the cron daemon processes the cron job, user's name and security context is obtained from the cron job filename and appropriate security context, as allowed by the security policy, is set for the process which executes the cron job.

Multi-context aware cron consists of two components. Changes to the SELinux command crontab and modifications to the SELinux command cron.

4.3.1 Modifications to `crontab`

Proposed extensions to crontab includes a new option '-c' which appends user's security context to the user name when creating a cron job file in */var/spool/cron* directory. Users that wish to execute cron jobs with their current role and sensitivity level use this new option when submitting a cron job.

4.3.2 Modifications to `cron`

The cron daemon processes cron jobs in */var/spool/cron* directory. Proposed extensions to cron include support for processing cron jobs that contain a user name and a security context. The cron daemon, as it builds the database of cron jobs, obtains the security context with which to execute the job from the security context in the cron job filename instead of obtaining it using the SELinux library function get_default_context_with_level.

5 Conclusions

SELinux provides a complete set of features to setup a fully functional Mandatory Access Control Based system. However, implementation of BLP style hierarchical MLS security policy for Common Criteria certification at EAL4 against LSPP and RBACPP requires extensions to core SELinux. These extensions meet specific LSPP and RBACPP requirements or improve usability of SELinux impaired by configuration of MLS security policy and implementation of some LSPP and RBACPP requirements. Polyinstantiated directories, multi-context aware cron and data import/export restrictions based on security attributes of terminal devices represent about 15% of the total development work required to meet LSPP and RBACPP. With these extensions and other ongoing work on audit, network, device allocation and print subsystems, a usable SELinux based system can be created that is functionally ready for Common Criteria certification at EAL 4 against the LSPP and RBACPP.

Acknowledgements

Authors acknowledge Stephen Smalley for his conceptual design of polyinstantiated directories, Stephen Smalley and Dan Walsh for solution suggestions for data import/export restrictions on terminals, Chad Hansen for TCS multilevel cron patch, and contributors to redhat-lspp and selinux open source mailing lists for their guidance.

References

1. Information System Security Organization, *Labeled Security Protection Profile*, NSA

2. Jim Reynolds and Ramaswamy Chandramouli, *Role Based Access Control Protection Profile*, National Institute of Standards and Testing

3. D. E. Bell and L. J. LaPadula ,*Secure Computer Systems*. The Mitre Corporation

4. Rob Pike, Dave Presotto, Ken Thompson, Howard Trickey and Phil Winterbottom, *The Use of Name Spaces in Plan 9*, Bell Laboratories