Have You Driven an SELinux Lately?

An Update on the Security Enhanced Linux Project

James Morris
Red Hat Asia Pacific Pte Ltd
jmorris@redhat.com

Abstract

Security Enhanced Linux (SELinux) [18] has evolved rapidly over the last few years, with many enhancements made to both its core technology and higher-level tools.

Following integration into several Linux distributions, SELinux has become the first widely used Mandatory Access Control (MAC) scheme. It has helped Linux to receive the highest security certification likely possible for a mainstream off the shelf operating system.

SELinux has also proven its worth for general purpose use in mitigating several serious security flaws.

While SELinux has a reputation for being difficult to use, recent developments have helped significantly in this area, and user adoption is advancing rapidly.

This paper provides an informal update on the project, discussing key developments and challenges, with the aim of helping people to better understand current SELinux and to make more effective use of it in a wide variety of situations.

1 Introduction

The scope of this paper is to cover significant advances in the SELinux project since its initial integration with mainstream Linux distributions. For context, a brief technical overview of SELinux is provided, followed by project background information.

1.1 Technical Overview

SELinux is a flexible MAC framework for Linux, derived from the Flask research project [30], and integrated into the Linux kernel via the Linux Security Modules (LSM) API [34].

All security-relevant accesses between subjects and objects are controlled according to a dynamically loaded mandatory security policy. Clean separation of mechanism and policy provides considerable flexibility in the implementation of security goals for the system, while fine granularity of control ensures complete mediation.

An arbitrary number of different security models may be composed (or "stacked") by SELinux, with their combined effect being fully analyzable under a unified policy scheme.

Currently, the default SELinux implementation composes the following security models: Type Enforcement (TE) [7], Role Based Access Control (RBAC) [12], Muilti-level Security (MLS) [29], and Identity Based Access Control (IBAC). These complement the standard Linux Discretionary Access Control (DAC) scheme.

With these models, SELinux provides comprehensive mandatory enforcement of least privilege, confidentiality, and integrity. It is able to flexibly meet a wide range of general security requirements:

- Strong isolation of applications.
- Information flow control.
- Ensuring critical processing flow.
- Protection of system integrity.
- Containment of security vulnerabilities.

SELinux is also able to meet the requirements of and interoperate with traditional MLS systems.

1.2 Project Background

Based on the growing need for stronger security [19], SELinux was released as an open source project by the

US National Security Agency in December 2000. A vibrant community formed around the project, with Debian developers leading efforts to integrate SELinux into a mainstream distribution. Subsequent early integration efforts were also largely community-led, with individual developers undertaking SELinux packaging work for the Gentoo, SuSE, and Red Hat distributions.

In March 2001, the NSA presented SELinux to senior kernel developers at the Linux Kernel Summit. Feedback from the summit led to the formation of the LSM project—the aim of which was to allow different security mechanisms to be plugged into the core kernel.

SELinux helped to drive LSM requirements, while SELinux developers participated as core contributors in LSM development. SELinux was ported to the resulting LSM API, and then merged into the mainline Linux kernel for the kernel's v2.6 release in December 2003.

Concurrently, SELinux was fully integrated into the Fedora Core distribution, version 2 of which was released in May 2004. This marked the first release of an operating system with official support for SELinux. The security policy distributed with Fedora Core 2 was largely derived from the initial NSA example policy, as well as from ad-hoc community contributions. It had not yet been well-tuned to many general Linux deployment scenarios and was a noted source of frustration in causing many applications to fail unexpectedly due to policy violations.

In some cases, SELinux was blocking previously undetected programming errors (such as the leaking of open file descriptors), although the initial security policy was also generally regarded as being too strict. A solution was implemented during the development of Fedora Core 3 (released November 2004), whereby only critical applications were confined by policy. This policy was designated as *targeted*, and was considered to be a major improvement in usability. It allowed SELinux to be enabled by default, leading to what is believed to be the first ever release of a mainstream operating system with mandatory security enabled in a standard configuration.

Red Hat Enterprise Linux 4, based on Fedora Core 3, shipped a few months later in February 2005 as the first commercial distribution with full SELinux support.

These initial major distribution releases, while shipping with relatively few confined services by default, provided useful mandatory security out of the box. They were key milestones in the the transition of SELinux from a research project into a production-deployable system.

With SELinux now being distributed to the wider community, the project began to evolve rapidly in terms of both function and usability.

2 Policy

Early versions of targeted policy confined only a few critical and typically network-facing applications. Ongoing contributions from the community, in conjunction with continued efforts by core project developers, led to a steady increase in the number of applications confined by policy. As of April 2008, there was policy support for several hundred applications in the upstream repository.

2.1 Booleans

The *Booleans* facility[1] was developed to allow certain policy features to be selectively enabled or disabled without having to reload policy. This was originally designed with smaller systems in mind, but has proven to be significantly useful in the general case.

Feedback from the community indicated that there were common patterns in the way applications would be customized. Booleans were able to provide a suitable high-level mechanism for administrators to effect limited yet useful customization of policy according to these patterns.

For example: policy for an FTP server might have a boolean which enables remote access to user home directories. Rather than having to modify or even know any SELinux policy, the administrator could instead control this behaviour by simply running a command to enable or disable the associated boolean.

2.2 Loadable Policy Modules

SELinux initially shipped with a monolithic policy scheme, requiring a full rebuild of policy from source for any modification. The Loadable Policy Module architecture [20] was introduced to provide a more robust and flexible approach to composing and deploying policy.

[1] also referred to as *Conditional Policy*

With this new architecture, binary policy modules could be loaded dynamically, eliminating the need for systems to carry a full policy source tree and associated build infrastructure in case customization was required.

The core operating system policy was separated into a base policy module, allowing it to be streamlined and further tailored for different use scenarios. Higher-level components could now be developed and managed independently, enabling the distribution of third-party policy, as well as the ability to incorporate policy modules with application packages.

Loadable Policy Modules were first shipped with Fedora Core 5 in March 2006.

2.3 Reference Policy

One of the most significant advances in the SELinux project has been the *Reference Policy* effort [28]. This was a fundamental reworking of the policy framework to provide a more structured basis for policy design and analysis. A critical goal was to improve the quality of policy and thus facilitate increased overall assurance.

Key elements of the Reference Policy effort included:

- Introducing the principles of layering and interfaces to the policy language to facilitate better abstraction and modularization.
- Leveraging the new Loadable Policy Module architecture discussed in Section 2.2.
- Building support for documentation into the infrastructure to encourage the practice of literate programming [17].
- Easier configuration of security models, so that features such as MLS may enabled without requiring separate policy source trees.
- Porting the NSA example policy to the new framework.

For policy developers, Reference Policy meant a greatly simplified view of policy. Much of the low-level complexity was abstracted away. Design could be performed in a modular fashion, utilizing well-defined and documented interfaces. This facilitated a greater focus on high-level security goals and better comprehensibility of policy.

Reference Policy was first shipped along with Loadable Policy Modules in Fedora Core 5.

2.4 Other Developments

The (somewhat experimental) MLS functionality supplied with the original SELinux release was significantly revised to meet certification requirements and to be more useful in the real world. This included policy enhancements and the ability to enable MLS dynamically. MLS was previously a kernel compile-time option and not enabled at all in mainstream distributions.

The MLS infrastructure was also adapted to a new security model, Multi-Category Security (MCS) [24], which utilized the *category* attribute of MLS labels and a simple policy to provide end users with a discretionary labeling scheme. Fedora Core 5 shipped with MCS enabled by default, allowing much of the MLS infrastructure to be exercised by general users. The future of MCS is currently unclear, as its wider adoption requires extension to applications, and it may be better to instead utilize TE for the same purpose.

The RBAC scheme was greatly improved, allowing roles to be defined and loaded as policy modules, rather than having to be managed as part of the previous monolithic policy scheme.

By the time of writing, targeted and strict versions of policy in the upstream repository had been merged into a single version. Targeted behavior is now selected by including the "unconfined" policy module. Strict behavior may also be re-selected incrementally by mapping users to confined roles.

3 Toolchain and Management

Major changes such as the Loadable Policy Modules and Reference Policy projects entailed reworking much of the low level SELinux infrastructure. This provided opportunities to improve the function and usability of the toolchain, and to implement a foundation for the development of high-level SELinux applications.

The following are highlights of recent advances made in SELinux toolchain and management technology.

3.1 libsemanage

The extensible *libsemanage* library was developed in conjunction with Reference Policy as the first programmatic API for applications which need to manipulate policy. Such applications range in scope from the core system management utilities and scripts through to graphical policy design and management tools.

3.2 semanage

The *semanage* command line tool was introduced as a means to unify low-level SELinux administrative tasks, many of which, prior to Reference Policy, involved editing disparate policy source files and rebuilding policy.

For the system administrator, *semanage* improved overall usability by providing a well documented, canonical utility for managing key aspects of an SELinux system. Examples of *semanage* use include configuring local file labeling rules and the labeling of system objects such as network ports.

Recently, much the core of *semanage* has been refactored into a Python module to enable re-use by other management tools such as *system-config-selinux*.

3.3 system-config-selinux

system-config-selinux is a GUI tool for comprehensive SELinux system management. It has been integrated into Fedora-based distributions, and serves as a graphical alternative to *semanage*.

A flexible underlying Python-based architecture should make it readily adaptable to the management schemes of other distributions and operating systems.

3.4 Loadable Policy Module Tools

- *checkmodule* is the policy module compiler. It verifies the correctness of a policy source module then converts it into a binary representation.
- *semodule_package* bundles a binary policy file as created by *checkmodule* with optional related data such as file labeling data into a format ready to be installed by *semodule*.
- *semodule* is the core policy module management tool. It is used for installing, upgrading, querying and deleting binary policy modules.

3.5 Boolean Management

setsebool and *getsebool* are the standard command-line utilities for managing the state of policy Booleans (see Section 2.1). Booleans may also be managed via *system-config-selinux*.

3.6 restorecond

A common early issue for administrators was that some files were particularly susceptible to being mislabeled, such as the */etc/resolv.conf* file being recreated by certain system tools.

The *restorecond* utility was developed to automate relabeling of such files, reducing administrative burden.

Files to be monitored are listed in a configuration file (typically */etc/selinux/restorecond.conf*). *restorecond* utilizes the Inotify subsystem to detect changes to any of these files, then performs relabeling if needed.

Note that file labeling is usually handled transparently by SELinux policy, or by enabling system tools to preserve security labels on files. As such, *restorecond* is an optional SELinux component aimed at improving usability on general purpose systems.

3.7 setroubleshoot

setroubleshoot is a GNOME facility which triggers a user alert upon SELinux policy violations. The alert notifies the user of the problem and assists in resolving the issue or filing a report with SELinux developers. *setroubleshoot* may also be configured to send email alerts to an administrator for centralized management.

setroubleshoot utilizes a pluggable rule database and attempts to heuristically determine problem causes.

Community feedback indicates this facility has significantly improved the SELinux user experience, by helping to resolve issues encountered, and by giving users a clearer understanding of what is happening on their system when SELinux prevents an access.

setroubleshoot first shipped with Fedora Core 6 in October 2006.

3.8 SELinux Policy Management Server

The *SELinux Policy Management Server* [21] is an ongoing project which addresses several requirements in the management of policy on an SELinux system. Its aim is to provide a platform for installing, updating and querying SELinux policy on production systems, with the ability to safely delegate administration of policy to separate users.

Currently in a prototype phase, a notable planned feature of the policy management server is support for remotely managing a single policy across multiple machines.

4 Policy Development

Since the introduction of Reference Policy, there have been several advances in the area of authoring SELinux policy.

4.1 Command Line Tools

While SELinux policy is ideally shipped with the system and managed with high-level tools, administrators may still wish to develop their own enhancements to policy. This task has been simplified somewhat with modular policy and new or enhanced command-line tools.

4.1.1 audit2allow

The *audit2allow* utility parses the audit log and converts access denial records into security policy. It has proven to be a valuable tool in resolving local policy issues.

For example, in the case of an application not having any SELinux policy (e.g., locally developed or provided by a third party), a policy module may be developed for it with a few simple commands [31]. The system is configured in permissive mode, so that access denials will be logged but not be enforced, and then the audit log is passed to *audit2allow* to generate a binary policy module. The new module may then be loaded into the system via *semodule*. This is a form of "learning mode." It is always recommended that the administrator review the resulting policy, and to request further review from the community if necessary.

4.1.2 audit2why

The *audit2why* tool was developed to help administrators better understand SELinux audit messages. It takes raw audit logs as input and analyzes them to determine which policy component triggered a particular audit message.

Often, the audit messages alone do not provide enough information to trace an access denial back to the associated policy component. Access denials have several possible causes: missing TE rules, missing RBAC rules, and policy constraints. *audit2why* is able to pinpoint precisely which.

Recently, *audit2why* was extended to determine which policy Boolean, if applicable, may be modified to resolve the issue.

4.2 SLIDE

The *SELinux Policy IDE (SLIDE)* is a sophisticated GUI policy development environment, implemented as an Eclipse plugin. It is aimed at making policy development easier, and includes many developer-oriented features such as syntax highlighting, project exploration, auto-completion, wizards, and refactoring support.

SLIDE also includes support for testing, deploying, and remotely managing policy. It was recently integrated into the Fedora distribution as an official package.

4.3 Policy Druid

A policy generation druid is included in *system-config-selinux*. This is a simple graphical wizard which presents the user with a series of questions about an application based on common security traits. A policy module is then automatically generated for the application, which may then be further managed via *system-config-selinux*.

No knowledge of the SELinux policy language is required. This tool is useful for rapidly generating policy with broad confinement properties.

4.4 SEEdit

The *SELinux Policy Editor (SEEdit)* [27] is a graphical tool which allows users to develop policy in a simplified policy language. Introduced by Hitachi Software in

2005, *SEEdit* utilizes pathname-based configuration and is targeted at developing policy for embedded systems.

4.5 SETools

A powerful set of policy analysis tools has been developed by Tresys and packaged into the *SETools* [6] suite. Included in the suite are utilities for analyzing SELinux policy, analyzing audit messages and creating audit reports, and verifying and examining policy.

4.6 CDS Framework Toolkit

The *CDS Framework Toolkit* [1] is a high-level tool for designing SELinux policy for "cross-domain solutions" (CDSs). CDSs are specialized guard systems for controlling information flow between different security domains.

While typically employed for sensitive government and military use, the underlying principles of have more general applications, such as in the case of a corporate Internet gateway which filters email and other user traffic. As SELinux is able to enforce critical processing flow,[2] security policy can be used to ensure, for example, that incoming email is always passed through a virus checker and a spam filter.

The *CDS Framework Toolkit* utilizes a GUI and abstract representation of the system, and does not require detailed knowledge of SELinux policy on the part of the user.

5 Networking

SELinux provides fine-grained controls over network accesses at several layers of the networking stack. At the socket layer, all socket system calls are mediated. Specialized controls are implemented for critical networking protocols (such as Unix domain sockets), the IP layer, and the interface layer.

Since the initial releases of SELinux, several aspects of networking support have been reworked or newly implemented.

[2] also referred to as an *assured pipeline*

5.1 Secmark

The first release of SELinux included rudimentary IP layer controls for packet flow. An effort called *Secmark* [25] was undertaken in 2006 to modernize the IP layer controls, leveraging the rich firewalling capabilities available in the Linux kernel.

With *Secmark*, *iptables* rules are used to label packets based on attributes which can be determined from the packet alone (e.g., destination port). Packet flow is then mediated using these labels according to SELinux policy.

This allows the utilization of all available iptables matches as selectors, as well as features such as connection tracking (or "stateful inspection"). Use of the latter leads to greatly simplified network packet policy. A single rule can be used, for example, to permit the flow of all traffic in a validated "established" or "related" state, eliminating the need to explicitly allow (or ignore) ephemeral port use, and to automatically handle multi-connection protocols such as FTP.

The code for this has been merged upstream, although distribution integration is not yet complete, as there is currently an outstanding issue of how best to perform the integration without clashing with current users of iptables.

5.2 Labeled Networking

While *Secmark* provides local labeling of network traffic, there is also a need for the ability to mediate traffic based upon remote characteristics such as the security context of the peer application. This is achieved by conveying security labels with the traffic as it transits the network.

SELinux takes two approaches to this:

- **Labeled IPSec**

  This essentially involves labeling IPSec security associations (SAs) to implicitly label traffic carried over those SAs. This was derived from earlier Flask research [9], implemented for SELinux by an IBM team [15], and further refined by Trusted Computer Solutions for better MLS support and improved usability.

- **NetLabel**

  Legacy trusted systems utilize IP options to convey security labels across the network. It is desirable for SELinux to interoperate with these systems. A flexible implementation based on the abandoned CIPSO [14] standard has been developed and integrated into the kernel by HP.

All of the above network labeling schemes were designed to be security framework agnostic, and are available for use by other security modules.

6 Memory Protection

Linux systems take a layered approach to security, to provide "defense in depth," ensuring that security measures are implemented at every possible level. No single measure can defeat all attacks, but many attacks can be defeated by a combination of measures.

SELinux utilizes kernel-based mechanisms to confine the behavior of userland applications. It is thus not designed to protect directly against kernel bugs, nor certain classes of application issues such as memory-based attacks, although it can help confine the damage done by such attacks.

Many distributions ship with mechanisms to protect against memory-based attacks, such as support for non-executable pages (NX) and glibc memory checks. In some cases, applications may wish to override memory checks (e.g., certain virtual machine interpreters), and SELinux has been extended to allow these overrides to be controlled by SELinux policy [11].

Enforcement of these memory checks via SELinux policy has led to the discovery of several applications which were inadvertently performing dangerous memory operations. While initially causing inconvenience to users, many of these applications were subsequently fixed, while developer awareness of the underlying issues was increased.

7 Security Evaluation and Accreditation

For some classes of government and military users, security certification is an important procurement requirement. The integration of SELinux made it possible to have Linux certified to the highest level currently achieved by mainstream operating systems.

In 2007, Red Hat Enterprise Linux version 5 in a server configuration was certified under the Common Criteria Evaluation and Validation Scheme (CCEVS) to Evaluation Assurance Level 4 Augmented (EAL4+), against the protection profiles:

- LSPP: Labeled Security Protection Profile
- RBACPP: Role Based Access Control Protection Profile
- CAPP: Controlled Access Protection Profile (Audit)

As CCEVS certifications include hardware, certification was performed for both HP and IBM platforms. A similar certification is currently underway for SGI hardware. EAL4+ is the highest assurance level likely achievable without a specially designed operating system, while LSPP is an updated equivalent of the earlier Trusted Computer System Evaluation Criteria (TCSEC or "Orange Book") B1 rating.

Several aspects of the certification are notable. It was the first time that Linux itself was known to be subjected to such a rigorous security validation process. Linux has now become directly competitive in the marketplace with existing "trusted" operating systems. An innovative approach was taken such that the Linux certifications were performed on the standard product line, rather than on separately maintained versions. Also innovative was the cooperative and open community certification effort, which led to a pooling of resources between several different companies and organizations.

The certification efforts also led to the development of several enhancements to SELinux and Linux itself, such as an overhaul of the Audit subsystem, and the introduction of polyinstantiated directories via Linux namespaces and the Pluggable Authentication Modules (PAM) subsystem. Many of these certification-related developments have proven generally useful, with a notable example being the introduction of *Kiosk Mode* (see Section 12).

SELinux-based systems have also been accredited on a per-system basis, independently of CCEVS. Details of such accreditations are often not public, although a case

study was presented at the 2007 SELinux Symposium where an SELinux-based system was developed to allow US Coast Guard Intelligence to consolidate access to separate classified networks [16].

The Certifiable Linux Integration Platform (CLIP) [2] project consists of specialized SELinux policy and system configuration packages aimed at meeting rigorous security requirements. For example, CLIP has been used to help SELinux systems meet *Director of Central Intelligence Directive 6/3 at Protection Level 4*, a common requirement for government and military systems which handle Sensitive Compartmented Information (SCI).

8 Performance and Scalability

The initial SELinux code release was not tuned for performance. Increased SELinux adoption saw the contribution of several enhancements aimed at improving performance, scalability, and resource utilization.

Examples include:

- *Utilizing Read Copy Update (RCU) [23] to remove locking and atomic operations from critical performance paths.* In January 2005, patches were merged upstream which dramatically increased the scalability of the core SELinux kernel code. Benchmarks run on 4-node 16-way NUMA system indicated a 50% increase in memory bandwidth and near-linear scheduler scalability.
- *Calculating information on non-critical paths as needed rather than ahead of time.* Patches were developed to perform more kernel access decision calculations on the fly and also reduce the size of related kernel structures. These changes, merged in September 2005, resulted in savings of around 8 MB of kernel memory for targeted policy and 16MB for strict policy. Under Linux, kernel memory is not pagable and thus a particularly precious resource.
- *Better utilization of the Linux slab allocator [13].* The kernel objects which hold SELinux policy rules were originally allocated via a generically-sized slab class. In August 2004, a patch was merged which implemented a custom slab class for policy rules, leading to memory savings of 37% on 64-bit systems.
- *Caching information rather than calculating it in performance critical paths.* Recent patches from HP have introduced label caches for virtual entities such as network addresses and ports, to avoid consulting the kernel policy database for labels at each access.
- *Optimized revalidation.* Under SELinux, read and write permissions for an open file are revalidated on each access. This is to ensure correct mediation even if there have been labeling or policy changes since the initial access. In September 2007, patches were merged which ensured that such revalidation would only occur if it was known that labels or policy had actually changed. Benchmarking indicates that overhead for read and write was reduced by a factor of five on Pentium-based systems and a factor of ten on the SuperH platform.

Many of the performance and resource utilization enhancements were contributed by developers from the embedded community.

9 Mitigation

A significant goal of SELinux is to mitigate threats arising from flaws in applications. Programming flaws are relatively common and effectively impossible to prevent entirely. Some flaws, especially those present in network-facing services and in privileged applications, may be exploited by malicious attacks. Such attacks can lead to disclosure of private information, corruption or destruction of valuable data, abuse of resources (e.g., hijacking a system to send spam), and the staging of more sophisticated attacks.

SELinux policy may be used to confine an application to ensure that it is capable of performing only the accesses needed for normal operation. This is an application of the principle of least privilege, which ensures that if an application is compromised or even malfunctions, that its actions will be limited to those it was supposed to be performing.

For example, if a web server was vulnerable to remote attack, an exploit may attempt to publish sensitive information from user directories or to send spam. With an appropriate SELinux policy, such actions would not be permitted, and these threats would be mitigated.

Several cases of successful threat mitigation with SELinux have been documented, where systems running with SELinux enabled were protected against real vulnerabilities and exploits [22].

These cases have demonstrated the value of deploying SELinux for general use.

10 Extending SELinux

The SELinux architecture has been extended beyond the Linux kernel to other areas of the system, and to other operating systems. Such efforts benefit from the ability to re-use existing SELinux components, such as the policy framework, code, and tools.

10.1 Desktop

Extending SELinux to the desktop environment is a significant undertaking. The modern desktop is made up of many layers and components, all of which need to be analyzed and understood in terms of information flow. At each layer, an SELinux policy model needs to be developed, and mediation hooks inserted into the code. This area has seen steady progress in parallel with the integration of SELinux into the base operating system.

The X Access Control Extension (XACE) [32] is a pluggable mandatory security framework for the X server, developed by the NSA, and merged into the upstream X.org tree. A Flask/TE module for XACE called XSELinux was also developed and merged upstream.

Work has also begun on securing the GNOME desktop environment by extending the SELinux architecture to GConf (the GNOME configuration system) [8] and D-BUS (the freedesktop.org messaging system). The D-BUS work has been merged upstream, while the GConf extensions are expected to remain in a prototype stage until further core desktop security infrastructure is in place.

A proof of concept project called *Imsep* [33] demonstrated a promising approach to protecting desktop applications by separating image processing functions into a separate security domain.

10.2 Database

SE-PostgreSQL [5], an extension of the SELinux architecture to the PostgreSQL relational database system, was released in September 2007. It features security labeling of data at the row and column levels, with enforcement of mandatory access control for authorized clients.

This importantly allows security policy to be uniformly applied to data at the OS level and within the database, where previously, fine-grained mandatory control was lost once information entered the database.

10.3 Virtualization

Efforts have been made by the NSA to integrate a flexible MAC scheme into the Xen hypervisor. Currently, the Xen Security Modules (XSM) project [10] implements a pluggable hook framework within the hypervisor, allowing different security models to be selected. An existing MAC scheme for Xen called Access Control Module (ACM) has been ported to XSM, and a Flask/TE module has been developed based on SELinux principles.

XSM removes security model logic from the core Xen code, and provides security model configuration flexibility. Hooks are implemented to allow mediation of privileged hypercalls, inter-domain communication (e.g., event channels and grant tables), and access to system resources by domains. An aim of this work is to increase robustness and assurance by decomposing the highly privileged Dom0 into separate domains. Interactions between these domains may then be controlled with fine-grained mandatory policy.

XSM and the Flask/TE module were merged into version 3.2 of the mainline Xen release.

10.4 Storage

While many local filesystems support SELinux via extended attributes, support for remote filesystems is rudimentary. When a remote filesystem is mounted, a policy-defined default security label may be assigned to all files on the mount; or in the case of NFS, the label may be specified by the administrator as a mount option. This provides coarse protection, although fine-grained labeling is not available; remote labels if they exist are

ignored; and there is no way to remotely set labels on files. There is also no awareness of whether the remote system is performing any SELinux enforcement.

The Labeled NFS project [3] was started in 2007 to address these issues. Thus far, detailed requirements based on previous similar projects have been gathered, while proof of concept code has been published and reviewed by Linux NFS developers. A key challenge of the project is to accommodate the needs of multiple upstream groups including the IETF, Linux NFS developers and core kernel maintainers.

Labeled NFS was presented by the NSA at the IETF 71 meeting in March 2008. RFC documents are being developed with the aim of establishing Labeled NFS as an Internet standard, while work is continuing on the prototype Linux code.

10.5 Beyond Linux

Several non-Linux operating systems have adopted, or are in the process of adopting, Flask/TE technology.

FreeBSD led the way in this area with the SE-BSD project, which ported the SELinux architecture to its TrustedBSD framework. This work was then ported to Apple's Darwin operating system, as SE-Darwin. These projects are not currently incorporated into their respective mainline trees.

Recently, the OpenSolaris project announced Flexible Mandatory Access Control (FMAC) [4], an effort to incorporate the Flask/TE architecture into their mainline operating system. This project is also expected to leverage the Flask/TE port to Xen, and to implement Labeled NFS and Labeled IPSec.

The FMAC project presents an exciting opportunity for Linux and OpenSolaris to offer compatible mandatory security to users. A significant potential benefit is increased overall adoption of mandatory security.

11 Community

As discussed in Section 1.2, the SELinux project emerged from academic and government research efforts. As SELinux has been integrated into Linux distributions and made more generally usable, the SELinux community has expanded in both size and scope.

11.1 SELinux Symposium

The SELinux Symposium (2005–2007) has been of profound importance to the project, bringing together core developers, security researchers and significant early adopters.

A small, invite-only developer summit, arising from earlier informal meetings, was typically held after the main symposium, driving much of the direction of development for each year. The developer summit is now planned to be a separate, open event aimed at engaging the wider community.

11.2 Online Resources

The nsa.gov site remains the primary point of focus for the project, hosting many critical documents, the main mailing list, and historical core code releases.

Other SELinux sites have been deployed in recent years:

- `selinuxnews.org` – project news and events, and an SELinux community blog aggregator.
- `oss.tresys.com` – hosts several open source SELinux projects developed by Tresys.
- `selinuxproject.org` – a wiki hosting miscellaneous developer and project resources.

Distributions with SELinux support typically also utilize mailing lists and web sites to provide resources to developers and users. Links to these may be found at the SELinux for Distributions site: `selinux.sourceforge.net`.

Many seminal SELinux papers and presentations are archived at the SELinux Symposium web site: `selinux-symposium.org`.

SELinux kernel development is now hosted in public repositories on `kernel.org`.

11.3 Distributions

While Fedora-based distributions have been at the forefront of SELinux development in recent years, SELinux integration efforts have continued in other distributions.

Gentoo, Debian, and Ubuntu all have security hardening efforts involving SELinux integration into their mainline distribution releases.

SELinux was incorporated into the mainline release of Debian 4.0 ("Etch"). Previously, SELinux packages for Debian were only available from separate repositories.

As of version 8.04, Ubuntu includes full support for SELinux in its server release.

11.4 Adoption

SELinux was always intended to be adaptable to a wide variety of usage scenarios, and to provide useful protection in the general case. Initially, serious adoption appeared to be focused around traditional higher-assurance users such as government and military organizations.

With the integration of SELinux into general purpose, mainstream distributions, wider and more general adoption is now being seen.

There has been growing interest from industry sectors with critical security needs, such as finance and healthcare.

Consumer electronics is significant area of adoption which was not perhaps originally anticipated. As consumer devices become more sophisticated and connected, the nature and scope of their security requirements is markedly increased. A flexible MAC scheme such as SELinux often proves useful in this area, as security policy can be tailored to the specific function of each product, leading to tighter security than is typically possible for a general purpose system. There is potential to increase the time to patch if a vulnerability is discovered which is mitigated by SELinux policy, perhaps delaying the update until another critical update is required. Updating software in fielded devices can be expensive and risky.

In terms of general purpose adoption, statistics gathered by the Fedora project since the release of Fedora 8 tentatively indicate that a significant majority of users have SELinux enabled. Recent advances in usability and increased awareness of the value of mandatory security are likely factors here.

12 Current Developments

At the time of writing, an area of active development is in confining users. Under targeted policy, general users on an SELinux system are unconfined, with a focus on protecting against external threats. With the release of Fedora 8, a facility called *Kiosk Mode* [26] (or *xguest*) was introduced. This allows an anonymous user to access a desktop session in limited but useful manner, such as to only allow web browsing. The security goal in this case is to protect the system from the user. *Kiosk Mode* may be useful for providing public access to desktop and Internet applications in varied settings, such as libraries, cafes, conferences, product demonstrations, and training sessions.

Another current development is *permissive domains*, a mechanism to allow permissive mode to be invoked only for specific applications. SELinux enforcement may be disabled for a selected application, say, to debug its policy, while maintaining SELinux enforcement for the rest of the system.

13 Future Work

Areas of ongoing and future work in the SELinux project include:

- Continued extension of SELinux architecture to the desktop infrastructure and major applications. The Imsep work mentioned in Section 10.1 looks to be a promising model for general separation of security domains within applications.
- Working with the IETF to standardize Labeled NFS, and with the Linux community to have it accepted into the mainline kernel.
- Ongoing performance improvement, and efforts to further reduce the memory footprint of SELinux.
- Further simplification of policy, perhaps through the development of a higher-level policy language with idioms more familiar to Linux administrators.
- Support for more virtualization models, including Linux as hypervisor (e.g., KVM) and containers.
- Improved support for third party distribution of policy modules, such as the case of cross-building RPMs on systems with a conflicting host policy.

- Continued usability improvements for end users, administrators and developers.
- Better documentation.

14 Conclusion

Following the transition of SELinux from a research project into a deployable, community-driven technology, there has been rapid and intense growth in its function, usability, and adoption.

The SELinux project has pioneered the practical application of mandatory security in general purpose computing, utilizing a flexible, open approach to both the technology and the execution of the project itself.

While still a work in progress, SELinux has matured to the point where it is able to provide useful mandatory protection to general users, while also meeting higher assurance goals in critical security environments.

Other operating systems have begun adopting concepts innovated by the SELinux project, such as incorporating mandatory security into mainline products and making it a standard feature enabled by default.

It is hoped that SELinux will continue to both provide and foster stronger computer security for users of the continually evolving globally networked environment.

15 Acknowledgements

Thanks to Stephen Smalley, Dan Walsh, Karl MacMillan, and Christopher PeBenito for providing valuable feedback during the preparation of this paper.

References

[1] CDS Framework Toolkit. `http://oss.tresys.com/projects/cdsframework`.

[2] Certifiable Linux Integration Platform. `http://oss.tresys.com/projects/clip`.

[3] Labeled NFS. `http://www.selinuxproject.org/page/Labeled_NFS`.

[4] OpenSolaris Project: Flexible Mandatory Access Control. `http://www.opensolaris.org/os/project/fmac/`.

[5] Security Enhanced PostgreSQL. `http://code.google.com/p/sepgsql/`.

[6] SETools. `http://oss.tresys.com/projects/setools`.

[7] W. E. Boebert and R. Y. Kain. A Practical Alternative to Hierarchical Integrity Policies. *Proceedings of the 8th National Computer Security Conference*, 1985.

[8] J. Carter. Using GConf as an Example of How to Create an Userspace Object Manager. *Proceedings of the 3rd Annual Security Enhanced Linux Symposium*, March 2007.

[9] Ajaya Chitturi. Implementing Mandatory Network Security in a Policy-flexible System. Master's thesis. `http://www.cs.utah.edu/flux/papers/ajay-thesis-abs.html`.

[10] G. Coker. Xen Security Modules (slides). `http://xen.org/files/xensummit_4/xsm-summit-041707_Coker.pdf`, April 2007.

[11] U. Drepper. SELinux Memory Protection Tests. `http://people.redhat.com/drepper/selinux-mem.html`, April 2006.

[12] D. Ferraiolo and R. Kuhn. Role-Based Access Controls. *Proceedings of the 15th National Computer Security Conference*, October 1992.

[13] Brad Fitzgibbons. The Linux Slab Allocator, October 2000.

[14] IETF CIPSO Working Group. Commercial IP Security Option (CIPSO 2.2), July 1992.

[15] Trent Jaeger, Kevin Butler, David H. King, Serge Hallyn, Joy Latten, and Xiaolan Zhang. Leveraging IPsec for Mandatory Access Control of Across Systems. *Proceedings of the 2nd International Conference on Security and Privacy in Communication Networks*, August 2006.

[16] G. Kamis. US Coast Guard / NetTop2 - Thin Client Implementation (slides). `http://selinux-symposium.org/2007/slides/08-tcs.pdf`, March 2007.

[17] Donald E. Knuth. *Literate programming*. Center for the Study of Language and Information, Stanford, CA, USA, 1992.

[18] P. Loscocco and S. Smalley. Meeting Critical Security Objectives with Security-Enhanced Linux. *Proceedings of the 2001 Ottawa Linux Symposium*, July 2001.

[19] P. Loscocco, S. Smalley, P. Muckelbauer, R. Taylor, S. Turner, and J. Farrell. The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments. *Proceedings of the 21st National Information Systems Security Conference*, October 1998. http://www.cs.utah.edu/flux/papers/ajay-thesis-abs.html.

[20] Karl MacMillan. Core Policy Management Infrastructure for SELinux, March 2005.

[21] Karl MacMillan, Joshua Brindle, Frank Mayer, Dave Caplan, and Jason Tang. Design and Implementation of the SELinux Policy Management Server. *Proceedings of the 2nd Annual Security Enhanced Linux Symposium*, March 2006.

[22] D. Marti. A seatbelt for server software: SELinux blocks real-world exploits, February 2008. http://www.linuxworld.com/news/2008/022408-selinux.html.

[23] Paul E. McKenney, Jonathan Appavoo, Andi Kleen, Orran Krieger, Rusty Russell, Dipankar Sarma, and Maneesh Soni. Read-Copy Update. In *Ottawa Linux Symposium*, July 2001.

[24] J. Morris. A Brief Introduction to Multi-Category Security. http://james-morris.livejournal.com/5583.html, September 2005.

[25] J. Morris. New Secmark-based Controls for SELinux. http://james-morris.livejournal.com/11010.html, May 2006.

[26] J. Morris. Using SELinux Kiosk Mode in Fedora 8. http://james-morris.livejournal.com/25640.html, February 2008.

[27] Y. Nakamura. Simplifying Policy Management with SELinux Policy Editor. March 2005.

[28] C. PeBenito, F. Mayer, and K. MacMillan. Reference Policy for Security Enhanced Linux. *Proceedings of the 2nd Annual Security Enhanced Linux Symposium*, February 2006.

[29] R. Smith. Introduction to Multilevel Security. http://www.cs.stthomas.edu/faculty/resmith/r/mls/index.html, 2005.

[30] R. Spencer, S. Smalley, P. Loscocco, M. Hibler, D. Andersen, and J. Lepreau. The Flask Security Architecture: System Support for Diverse Security Policies. *Proceedings of the Eighth USENIX Security Symposium*, August 1999.

[31] D. Walsh. Creating Loadable Modules with Audit2Allow. http://fedoraproject.org/wiki/SELinux/LoadableModules/Audit2allow, February 2006.

[32] E. Walsh. Application of the Flask Architecture to the X Window System Server. *Proceedings of the 3rd Annual Security Enhanced Linux Symposium*, March 2007.

[33] C. Walters. Towards a Least Privilege Desktop (presentation slides), March 2005.

[34] C. Wright, C. Cowan, J. Morris, S. Smalley, and G. Kroah-Hartman. Linux Security Modules: General Security Support for the Linux Kernel. *USENIX Security Conference*, August 2002.

Proceedings of the Linux Symposium

Volume Two

July 23rd–26th, 2008
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc., Linux Symposium, Thin Lines Mountaineering*
C. Craig Ross, *Linux Symposium*

Review Committee

Andrew J. Hutton, *Steamballoon, Inc., Linux Symposium, Thin Lines Mountaineering*
Dirk Hohndel, *Intel*
Gerrit Huizenga, *IBM*
Dave Jones, *Red Hat, Inc.*
Matthew Wilson, *rPath*
C. Craig Ross, *Linux Symposium*

Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*
Gurhan Ozen, *Red Hat, Inc.*
Eugene Teo, *Red Hat, Inc.*
Kyle McMartin, *Red Hat, Inc.*
Jake Edge, *LWN.net*
Robyn Bergeron
Dave Boutcher, *IBM*
Mats Wichmann, *Intel*