SELinux: A New Approach to Secure Systems

by Chris Runge

Abstract
In this whitepaper, we will examine Security-Enhanced Linux (SELinux), the benefits it brings, and how Red Hat is working to make those benefits available to mainstream enterprise computing.

July 2004

Table of Contents

WHP001US 7/04

Introduction

One of the greatest challenges now facing CIOs and IT directors is the task of maintaining the security of their IT environments. The effects of a security breach can be catastrophic, including unplanned downtime and the resulting loss of service—a potentially significant financial impact—and the loss of sensitive and confidential information. This problem has been compounded by the proliferation of networked PCs and servers as well as the growing intelligence of malicious software that seeks to exploit and expand throughout the Internet infrastructure.

The software industry is releasing new tools to address the needs of system administrators responsible for managing the security of large numbers of geographically dispersed systems. For example, Red Hat Network, a key part of Red Hat Enterprise Linux and Red Hat's Open Source Architecture, offers system administrators a way to review information about security vulnerabilities and proactively apply relevant security and other updates to large numbers of Red Hat Enterprise Linux systems.[1]

Tools like Red Hat Network can help system administrators react to and take action to thwart security breaches. At the same time, new technologies are emerging at the core operating system level to protect against and contain the effects of a security breach until security updates can be applied. For example, NX technology (available in new processors from AMD and Intel) and exec-shield (a new technology developed by Red Hat's Ingo Molnar), help protect against buffer overflows, a tactic frequently employed by attackers to infiltrate and compromise flawed software programs. Another of these technologies—the focus of this whitepaper—is Security-Enhanced Linux, or SELinux as it is most commonly known. In this whitepaper we will examine what SELinux is, the benefits it brings, and how Red Hat is working to make those benefits available to mainstream enterprise computing.

Access Control Mechanisms

Multi-user operating systems such as Linux, UNIX, Windows, and Mac OS X use access control mechanisms to determine whether and how the users and programs on that system can access objects (e.g., files, directories, and sockets) on the system. Actions such as whether users or programs running on the system can read, write, create, or delete files (such as log files, configuration files, or data files), and whether they can start new

1 For more information on Red Hat Network, see http://www.redhat.com/software/rhn/.

programs are determined by an access control mechanism. Given this, the characteristics of the access control mechanism being used become very important for the security of a given system.

Most Linux and UNIX operating systems use an access control mechanism known as *discretionary access control*.[2] Under a discretionary access control (DAC) scheme, files and directories on the system are labeled with a set of permissions indicating which user and group the file belongs to and what that user, group, or others can do with that object, such as reading, writing, and/or executing it. This relatively simple yet largely powerful scheme allows multiple users and programs to coexist on the same system, and ensures that users have control over their objects but no others if the permissions are properly set.

This scheme gets its name because users and programs have discretion over the objects in their control. The root user, or a program running as root, has full discretion over everything on the system. It follows that if a user owns files containing confidential information (e.g., financial information, patient data, trade secrets), that user can intentionally or mistakenly share that data with other users or programs that should not have access.

Programs run with the level of privilege of the user starting them. Accordingly, a program started by the user “chris” would run as the user “chris” and have the same level of control and access as “chris.” That same program, if started by root, would run with full root permissions. Some programs may be marked “setuid root,” meaning that they run with full root permissions regardless of the user starting them. If a flawed or malicious program is run as root, the entire system can be compromised; setuid root programs are of particular concern since they can be started by non-root users.

Attackers who desire to bring a system down, repurpose it (by running programs the owner did not intend to be run), or gain access to sensitive information on a system, generally seek to exploit the all-powerful nature of the root user. By exploiting a program, such as a daemon, that runs with full root permissions, they can use that program as a launching pad for their desired malicious actions on the system.

Mandatory access control is an alternative (and far less frequently used) access control mechanism. Found on what are known as trusted operating systems, mandatory access control, or MAC, takes security decisions out of the hands of the user. While users and groups may still own files and directories on the system, permission is ultimately governed by a *security policy*, which defines which users and programs can access which objects on the system. This security policy is enforced over all

SELinux and Multi-Level Security

Multi-Level Security, or MLS, is a specific mandatory access control security model that is focused on confidentiality. Traditionally, the separation of multiple classifications of information (e.g., Top Secret, Secret, Confidential, Unrestricted, etc.) was enforced through physical separation—a dedicated system would be used for each security classification to ensure that, for example, Top Secret information was not mistakenly or intentionally made visible to users possessing only a Secret (or lower) security clearance. A Multi-Level Security operating system is able to enforce both the separation of multiple classifications of information as well as manage multiple users with varying levels of information clearance. This allows one system to be used where multiple systems might otherwise be required. Support for Multi-Level Security in SELinux is experimental but work remains ongoing.

2 This generally applies to Windows as well, although the particulars of how the Dicretionary Access Control scheme is implemented may be different than on a Linux or UNIX-based operating system.

processes and objects on the system, not just a subset. In addition, policy decisions are based on all security-relevant information, not just the user identity. The security policy trumps the intentions of the user and is the final determiner of what takes place on the system. The mandatory nature of the security policy is what gives this access control mechanism its name.

Mandatory access control systems lack the necessary core concept of an all-powerful root user;[3] instead, the defining principle of MAC is the *principle of least privilege*. This principle dictates that programs are granted only the minimum amount of privilege in order to function, rather than inheriting the privileges of the user starting them. This configuration is done via the security policy present on the system. The end result is a higher level of security. If a program is compromised under a DAC scheme, the attacker gains the ability to run programs and access objects on the system at the same level of privilege as the user account the program is running under; however, under a MAC scheme the attacker is limited to the actions allowed by the system's security policy.

Mandatory access control offers clear security benefits, so why does it only have limited adoption? There are several reasons. First, the trusted operating systems that provide mandatory access control have largely been separate offerings from their more mainstream counterparts. Due to a traditionally limited potential audience consisting of largely specialized military and intelligence applications, these trusted systems have received less focus from operating system vendors and third-party vendors alike. They tend to lag behind their commercial cousins in terms of functionality as well as hardware and application support. Moreover, MAC systems can be difficult and time-consuming to implement and use. All of this results in a small community of use, which further serves as a disincentive for vendors to pour additional resources into the product. Today, trusted operating system use is largely confined to specific applications and deployments in the military and intelligence communities. Even then security features may be disabled or weakened, with users having a natural preference for functionality and ease-of-use over increased security.

Security-Enhanced Linux

Enter into this backdrop Security-Enhanced Linux, or SELinux, an open source research project sponsored by the National

3 Depending on the configuration of the security policy, the system administrator domain (sysadm_t) on an SELinux system may have a wide variance in the processes and objects that are placed under its control. Accordingly, sysadm_t could be quite powerful, approaching the level of control of the root user under a discretionary access control model. At the same time, sysadm_t could be restricted to just actual authenticated administrator sessions, and not be used as a domain for any daemons or other privileged programs. The key point here is that an omnipotent superuser is not a fundamental assumption of mandatory access control.

Security Agency (NSA) to implement mandatory access control in Linux. The NSA chose Linux as the basis for this project because of its large community of use and its open source development model. This would allow the NSA to receive valuable feedback during the development process. At the same time, it provided an opportunity to improve the security of a rapidly growing and increasingly popular operating system.[4]

Recognizing that secure applications require a secure operating system[5], SELinux provides security at the kernel level through the added implementation of a security policy and a separate enforcement mechanism. By adopting this approach, SELinux is able to support most commonly used applications, generally without modification, while largely retaining full application functionality. The security policy on the system, rather than the configuration of a particular application, will ultimately dictate the boundaries of the application's capability.

SELinux provides several benefits. By leveraging the principle of least privilege and through the institution of a security policy on the system, SELinux prevents the compromise of an entire system due to the compromise of a single application running with what would otherwise be elevated privileges. Programs are placed into individual sandboxes, isolating them from one another and from the underlying operating system (see Figure 1). A second benefit is that SELinux protects the confidentiality and integrity of data. By removing discretion from users over how data may be manipulated, sensitive data can be restricted from willful or inadvertent sharing, modification, or deletion.

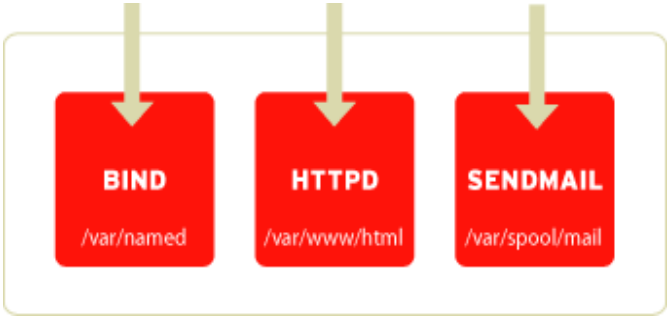

Figure 1. SELinux isolates programs from the operating system and from one another.

SELinux architecture and operation

SELinux is made up of both kernel and user-space components. With respect to the kernel, SELinux adds a security server containing the security policy. It also adds a separate enforcement mechanism that receives and applies the policy decision. Because the policy and enforcement mechanism are

4 See http://www.nsa.gov/selinux

5 See "The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments," by Peter A. Loscocco, Stephen D. Smalley, et. al. http://www.nsa.gov/selinux/papers/inevit-abs.cfm

independent, policy changes do not require changes to the enforcement mechanism. SELinux also modifies or adds several user-space components for recognizing and handling security contexts, role changes, policy development, and other tasks necessary for implementing mandatory access control on the system.

SELinux combines two security mechanisms to achieve a flexible security policy model: *type enforcement* (TE) and *role-based access control* (RBAC). Type enforcement implements a mechanism whereby processes (running programs) are placed in what are known as domains, which govern what these processes can do. For example, BIND, running in the named_t[6] domain, can access its zone files under /var/named but not the system's webpages under /var/www/html (which would be the province of the Apache HTTP server, running in the httpd_t domain). Each domain is an isolated sandbox on the operating system where an application resides. Applications are only allowed to play in their own individual sandboxes; they are constrained by the security policy on the system from interfering with other applications or with the underlying operating system. In a similar fashion, types are assigned to objects (files, directories, sockets, etc.) on the system. Types determine who gets to access the object and can be used to protect the integrity and confidentiality of data on the system.

Role-based access control helps to simplify policy creation and management. Instead of creating rules for every user as it relates to every object on the system, *roles* are used to determine what domains can be used. Common roles include user_r[7] (the standard unprivileged user role) and sysadm_r (the system administrator role). Those roles are then in turn assigned to users. Users can belong to multiple roles, with access dependent on the current role being used.

How does all of this work in actual operation? Files and processes are labeled with a *security context*, which is made up of the user, the role, and the domain or type. For example, the security context of the Apache HTTPD daemon is

```
system_u: system_r: httpd_t
```

while the security context of the Apache HTTPD configuration file is

```
system_u: object_r: httpd_config_t
```

When access is requested, the traditional UNIX permission system used by a discretionary access control scheme is referenced first. If the action is denied by that mechanism, the request immediately ends. If access is permitted, the mandatory

6 The _t suffix indicates a domain or type.
7 The _r suffix indicates a role.

access control scheme implemented by SELinux is used. The security server in the kernel checks the security context of the requesting user or program with the security context of the requested object in light of the security policy on the system. The resulting policy decision is received and applied by the enforcement mechanism, which either allows access or denies it (see Figure 2). All access requests and their corresponding policy decisions may be audited, if required.

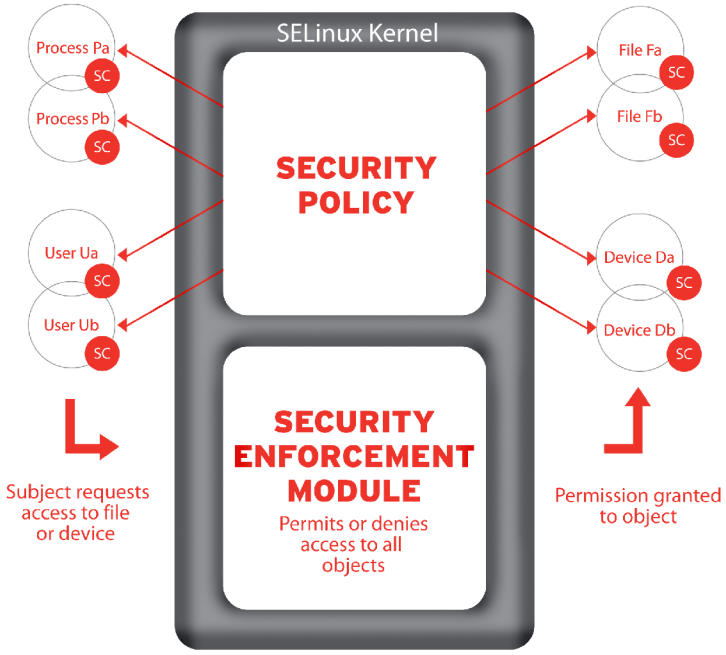

Figure 2. SELinux access control mechanism

Examples of SELinux in action

The best way to understand the benefits that SELinux provides is to examine two real-world usage scenarios: an Apache HTTP server and a BIND DNS server. Because these are popular network-based applications they are particularly susceptible to attack, and consequently stand to benefit from SELinux protection.

Example: Apache HTTP Server

The term “secure Web server” can have multiple meanings. It commonly refers to the security that a Web server may provide by encrypting incoming and outgoing traffic. However, that encryption process alone does not ensure the underlying security of the Web server software in its own right. An attacker can still

exploit a flaw or misconfiguration of the Web server to gain access to the data it manages, allowing the attacker to deny service, deface websites, or capture confidential information. The ability of a Web server, if so configured, to execute scripts and other program code can exacerbate the situation.

Web server administrators can improve security by configuring the Web server to run as a user other than root, and by reducing functionality such as disabling the ability of the Web server to execute program code. However, these steps do not necessarily provide the level of containment desired if the Web server is exploited. In addition, it may be desirable in some situations to allow the Web server to run scripts and use other functionality-enhancing capabilities. This is where SELinux can help.

An Apache HTTP server using SELinux carries a couple of principle advantages from an implementation standpoint. First, the standard Apache HTTP Server software (such as that provided with the operating system) can be used; no specially compiled binaries are required. Second, the full functionality of the Web server software, including running scripts and program code, is available. By placing the Web server in a domain, typically called httpd_t, the system's security policy will provide a box around the software and what it can do, even if the Web server is compromised.

SELinux's flexible policy model allows a system administrator or security officer to implement a security policy tuned to the requirements of the application. SELinux's approach to policy is that everything not expressly allowed is denied. Using the principle of least privilege, an example policy might therefore look as follows:

- The Web server is allowed to bind to port 80
- The Web server is allowed to read its configuration files in /etc/httpd/conf and read/write its log files
- The Web server is allowed to execute the programs and libraries it needs in order to function
- Read-only access to the system's webpages in /var/www/html is allowed
- User pages may be read; in addition user scripts may be executed that read from and write to those pages

If a Web server running with this policy were compromised, the potential damage would be confined. The system's webpages in /var/www/html could not be defaced. The attacker could not use the Web server to read files on the system other than webpages, the Web server's log files, and the Web server's configuration files. Only user scripts could be run, and only user pages modifiable by those scripts could be defaced. Finally, the underlying operating system as well as other applications running on the system would remain unaffected.

Example: BIND

BIND is a popular DNS server used on a number of Linux and UNIX operating systems. BIND has been a popular target because of its ubiquity, and because it typically runs with superuser privileges on the system, enabling attackers who compromise it to execute commands with superuser privileges.

The approach for using SELinux to lock-down BIND is similar to the approach taken above to secure the Apache HTTP server. The standard BIND software can be used and a chroot jail[8]—a commonly used tactic to improve the security of BIND—is not required. Again, the security policy on the system will provide the boundaries around the software and what an attacker who potentially exploits it can do on the system.

An example policy for BIND would be to create a domain, named_t, with the following characteristics:

- BIND can bind to port 53 on the system's external network interface.
- BIND can read its configuration file, /etc/named.conf, and read/write its log files
- BIND can read and write its dynamic zone files in /var/named
- BIND can execute the programs and libraries it needs in order to function

Everything else would be implicitly denied.

If BIND were compromised, once again the potential for the attacker to execute malicious action would be limited. In this case the only thing the attacker could corrupt or modify would be the dynamic zone files in /var/named. The attacker could not install new binaries (such as a root-kit), and is prevented from altering or removing existing system files. Moreover, the security policy would prevent the attacker from attaching to the system's internal network interface, blocking any attempts to use the compromised software to begin attacks on the internal network. The underlying operating system and any other applications on the system would remain untouched.

Deciding where to deploy SELinux

As seen above, Internet-facing edge servers running daemons such as the Apache HTTP server and the BIND DNS server stand to gain immediate benefits from the use of SELinux. Other

8 A chroot jail provides an increased level of security by moving the binaries, libraries, and other files used by BIND to a particular area on the filesystem, which in turn is treated as though it were the root directory of the operating system for the running BIND server. The idea is to contain a successful attacker to this limited filesystem, and in doing so prevent the attacker from compromising the operating system or other applications. A chroot jail may not provide complete security, however. See http://www.stahl.bau.tu-bs.de/~hildeb/bind/chroot.shtml for more information.

mission-critical infrastructure systems—mail servers, firewalls, authentication and authorization servers—are also good candidates.

SELinux also offers valuable information assurance, by protecting the integrity and confidentiality of sensitive information. This includes not only classified information in a more traditional sense that is hosted on systems used by the military and intelligence community, but also any confidential or sensitive information that needs to be protected. Examples include human resource data (e.g. payroll and personnel files), patient data (e.g. that protected by HIPAA), and financial data (e.g. credit card information).

The ultimate goal is to make SELinux capabilities valuable for every system. Due to its flexible policy model, SELinux has the potential to provide security for other classes of systems, such as an end-user desktop. For example, an SELinux policy could be created that would largely allow the end-user to work on the desktop as they normally would under a discretionary access control model. However, any network daemons (such as openssh, portmap, etc.) would be isolated into their own domains, as well as any applications (such as Mozilla and Evolution) that might potentially be misused as an agent for sending and receiving malicious software such as viruses and trojans.

Red Hat and SELinux

Red Hat recognizes the immediate and potential security benefits that SELinux brings to the open source community. As part of its goal to enable higher levels of security for all systems, Red Hat has made a strategic decision to adopt, incorporate, and further the development of the SELinux project.

As a core part of this effort, Red Hat will commercialize and provide the necessary commercial support for SELinux in its Red Hat Enterprise Linux product family. Until recently, deploying an SELinux-enabled system required patching the kernel and several user-space tools from an existing Linux distribution using code provided from the NSA's SELinux web site. With the release of the 2.6 Linux kernel, the kernel components of SELinux are now available in the upstream open source kernel, a key step in gaining wider adoption and long-term supportability in the open source community. Nonetheless, many organizations, especially those requiring commercial support for mission-critical systems, will wait until these capabilities are included in a generally available, enterprise-class Linux distribution, such as Red Hat Enterprise Linux.

Red Hat understands that to release SELinux only as a part of a separate trusted operating system would be to repeat the mistakes of the past. In order for SELinux—and the security benefits that it provides—to be more widely adopted, SELinux will need to be integrated into the mainstream operating system. As a result, those who require systems with mandatory access control will inherit the same operating system capabilities, commercial support offerings, and support from third-party vendors as those deploying systems without it.

System administrators will have several options around the configuration of SELinux on their systems. For those who desire to adhere to the traditional discretionary access control model, SELinux can be completely disabled. On systems where SELinux is enabled, administrators will be able to take advantage of its flexible policy model and tunable configuration options, enabling them to loosen or tighten the security on the system to meet their specific requirements.

Red Hat Enterprise Linux v.4, to be released in the first part of 2005, will feature the 2.6 Linux kernel and SELinux as two of its major features. With this release, organizations will be able to deploy a commercially supported SELinux-enabled operating system for production and mission-critical use. In the interim, in order to enable system engineers and administrators to begin development and testing, Red Hat is introducing SELinux into the Fedora Project, which serves as an important proving ground for new technologies targeted for inclusion in Red Hat Enterprise Linux. Fedora Core 2, released in May 2004, includes SELinux.[9]

Conclusion

SELinux introduces a new security paradigm—mandatory access control—to Linux. Mandatory access control uses a security policy to isolate applications from the operating system and from one another and to protect the integrity and confidentiality of information. While mandatory access control provides many benefits for security, it has traditionally been limited to trusted systems that have had limited adoption. Red Hat is incorporating SELinux in its future operating system releases, believing that this can provide better security for all systems.

SELinux development is continually ongoing. The incorporation of SELinux into the 2.6 kernel and into Fedora Core 2 is expected to accelerate this development process as SELinux gains increasing exposure in the open source community. Eventually, SELinux could evolve to the point where it is enabled by default, providing immediate benefits to all users.

9 For more information on the Fedora Project, visit http://fedora.redhat.com. For more information on getting started with SELinux in Fedora Core 2, see the Fedora Core 2 SELinux FAQ in the *Resources* section of this paper.

Ultimately SELinux should be viewed as a component of a larger set of security tools, technologies, processes, and procedures. Used in combination with these best practices, SELinux can enable system administrators to isolate and minimize the effects of an increasing number of malicious attacks.

www.redhat.com
1-888-REDHAT1 or +1-919-754-3700

Resources

1.NSA SELinux web site.
http://www.nsa.gov/selinux

2.NSA SELinux FAQ.
http://www.nsa.gov/selinux/info/faq.cfm

3.Red Hat SELinux web page.
http://www.redhat.com/solutions/security/SELinux.html

4.Fedora Core 2 SELinux FAQ.
http://people.redhat.com/kwade/selinux/selinux-faq/selinux-faq-en

5.Mailing list: fedora-selinux-list.
http://www.redhat.com/mailman/listinfo/fedora-selinux-list

6.The Unofficial SELinux FAQ.
http://sourceforge.net/docman/display_doc.php?docid=14882&group_id=21266

7.Getting Started with SELinux HOWTO: The New SELinux.
http://sourceforge.net/docman/display_doc.php?docid=20372&group_id=21266

8.Writing SELinux Policy HOWTO.
http://sourceforge.net/docman/display_doc.php?docid=21959&group_id=21266