Securing RTI Data Distribution Service with SELinux

February 4, 2009

Karl MacMillan and Chris PeBenito

Tresys Technology

Contents

1 Introduction

RTI Data Distribution Service (RTI's DDS) is communications middleware for distributed real-time applications. It is the most reliable, flexible and highest-performing implementation of Object Management Group's (OMG) Data Distribution Service (DDS) for Real-Time Systems standard available today. RTI's DDS is field proven and is used in many time-critical and data-critical applications such as financial transaction processing, national railways, air traffic control, traffic monitoring, combat systems and industrial automation.

Security Enhanced Linux (SELinux) is a flexible Mandatory Access Control (MAC) mechanism originally developed by the U.S. National Security Agency and available as a standard part of Linux and integrated into several Linux distributions. SELinux provides a new level of host security, data separation, and role-based access control. SELinux allows the creation of solutions that are secure even in the face of zero-day exploits and meet the most stringent commercial and government security requirements.

Securing RTI's DDS with SELinux provides a new level of security for applications and solutions created with the DDS standard, including:

- Strongly separating data distributed via DDS to prevent inappropriate data disclosure as a result of mis-configuration, software errors, or application vulnerabilities.
- Protecting DDS-enabled application processes, configuration files, data, and audit logs from other applications running on the same operating system instance.
- Constraining administrators, including limiting 'root' accounts, to create limited-privilege administrators and enabling separation of duties.
- Meeting the most stringent government and commercial security standards including DCID 6/3 PL4, NSSI 1253, DoD 8500.2, NIST 800-53, PCI, SOX, and HIPPA.

This white paper will provide technical details concerning leveraging SELinux to secure RTI -based applications and explore how SELinux and Linux can be used to create solutions with RTI's DDS that meet the most stringent government security requirements.

2 Overview of Security Enhanced Linux

SELinux provides a new level of security for Linux systems by providing the benefits of MAC, long used to create the most secure systems for government, with unmatched flexibility. The strength and flexibility of SELinux allows developers and administrators the ability to craft security policies that meet a broad range of security goals, including data confidentiality, system and application integrity, and administrator role separation.

SELinux has three main advantages over other security mechanisms: *type enforcement* based access control, mandatory security, and a consistent centralized policy controlling all system access. These advantages allow SELinux to provide the level of security

needed to face the hostile modern computing environment, even in the presence of vulnerable software and malicious users. These advantages also address the most serious shortcomings of Discretionary Access Control (DAC), the type of access control traditionally offered by Unix, Linux, and Windows[1].

2.1 Type Enforcement

SELinux controls access to system resources, such as files, based upon users, the current role of the user, and a new concept called *type enforcement.* Optionally, SELinux also offers multi-level security (MLS) sensitivity and categories. While the user, role, and MLS based controls offered are useful, it is the type enforcement access control model that offers an entirely new level of security. Type enforcement allows restrictions to be applied to processes and resources based upon their function or content. Unlike DAC, which only controls access based upon users and groups, type enforcement can apply access control separately to processes running on behalf of the same user. This allows the creation of *least-privilege* policies, which restricts processes to the minimum access required to correctly function.

Type enforcement allows a security policy writer to assign an abstract identifier, or *type*, to each process, file, or other resource. Access is then allowed in terms of the process type's access to a resource type. For example, consider a web server reading a web page stored on disk in order to serve a client request. Assuming the type "apache_t" is assigned to the web server process and the type "html_file_t" is assigned to the stored web page, access would be granted in the SELinux policy using the following rule:

```
allow apache_t html_file_t : file read;
```

This statement allows all processes with the type "apache_t" to read files with the type "html_file_t".

All processes and resources with the same type are security equivalent, and have the same access. Policy writers create as many, or as few, types as needed for a given system and can flexibly assign those types to applications based on many factors. Some examples include the type of the executable file, the type of the process executing the application, and the current user and role.

To further illustrate the advantages of type enforcement, consider a typical user session in which an email application, web browser, and word processor are running. Under DAC, each application, which is comprised of one or more processes, would have all of the access granted to the current user. This includes access to read or write all files owned or accessible to that user, send and receive data over the network, and access special devices such as sound cards or removable storage. A flaw in any of the applications would allow an attacker complete access, despite each application only requiring a subset of the total

[1] For the remainder of this paper, DAC will refer specifically to DAC as implemented in Unix and Linux. Other forms of DAC, including the role-based access control found in Microsoft Windows, suffers from similar shortcomings to the mechanism implemented in Linux.

access to function correctly. Under DAC, a vulnerable web browser could disclose confidential files. A malicious document could use a word processor as a platform for launching network attacks against an internal corporate network.

Contrastingly, in a least-privilege type-enforcement policy each application is assigned a unique type and is only granted the access required in order to achieve its functional purpose. For example, a web browser would be allowed to send and receive data over the network, render web pages on the display, and read its preferences. Access to read documents in the users home directory, run additional applications, or access special devices could be denied. With type-enforcement, a flaw in the web browser is contained by the least-privilege policy. Even the most severe vulnerability, one that allowed arbitrary code execution, would be prevented from causing damage such as disclosure of confidential data.

The concept of least-privilege can be extended to all system applications, including server oriented applications such as web servers, databases, mail servers, etc. Furthermore, applications can be specifically architected to take advantage of least-privilege policies by splitting their functionality into several cooperating processes. This strategy can produce applications that are secure even in the presence of flawed software.

2.2 Mandatory Security

SELinux access control is mandatory, meaning that all applications are confined by its security policy. This is in contrast to DAC, where applications are able to influence the outcome of access decisions. This distinction makes it possible to fully understand the security properties of a system and create a system that remains secure despite flawed software or malicious users.

To understand mandatory security, consider a user connecting to a shared file server. Under DAC, applications running on behalf of the user can set the access allowed to the files owned by that user. This allows applications to gain additional access, such as making read-only files writable, or granting access to other users. Under DAC, it is not possible to configure a system so that malicious applications or users cannot grant additional access over that desired by organizational policy.

Contrastingly, under SELinux, only privileged applications are allowed to change the security properties of resources. As a result, users and applications must work within the confines of the policy and cannot influence security results. Under SELinux, the security properties of a system are set at system creation time and remain fixed throughout the life of the system. As a result, this allows system architects the ability to design, develop, and deploy systems while eliminating the potential of allowing system integrity to be compromised.

2.3 Consistent and Centralized Policy

SELinux implements access control over all system resources including: files, directories, devices, networking, and inter-process communication, all using the same access control

model and centralized policy. As a result, all application access is controlled using the same mechanism and syntax, simplifying the design, implementation, and verification of the access control policy. Having a single system policy governing all access, the centralized policy allows analysis of a complete system including complex automated analysis to track the data flow throughout the system.

Alternatively, traditional Unix DAC only controls access to files and directories. Other types of access are controlled using other mechanisms, such as IPTables for network access. These other access control mechanisms often have differing means of specifying access and offer different types of control granularity. Furthermore, some access, such as binding to low ports, is controlled by implicit, hard coded rules that are inflexible. Leveraging DAC results in a applications being granted access that is extremely difficult to understand.

Finally, granting required access under DAC often requires security compromises. For example, many applications must run as root to bind to low ports or exercise other administrative privileges. This results in applications gaining broad privileges when only limited access is required. Such compromises are not required when SELinux is employed, as it is possible to grant the appropriate access directly to an application.

3 Securing RTI's DDS with SELinux

SELinux can add security to solutions created with RTI Data Distribution Service by securing the underlying platform, protecting RTI-based applications, and enforcing independent control over these applications. This section will examine the basic approach to securing RTI-based applications with SELinux and then examine, in more detail, several additional aspects of securing these applications.

3.1 Overview of RTI's DDS and SELinux

Fundamentally, RTI and SELinux are well suited due to the peer-to-peer, library based implementation of RTI's DDS. The design links DDS into user applications without the need for a single, central application connecting DDS clients and servers. This architecture is well suited to the process oriented access control that SELinux provides, thus allowing comprehensive control of DDS applications, including controlling which DDS applications are able to communicate.

To understand the interaction of DDS and SELinux, consider a simplified scenario of two applications, a publisher and a subscriber, communicating using DDS. In this simple scenario, each application will consist of a single process with multiple threads[2]. All of the functionality of DDS will be embedded in each of these processes, allowing them to communicate directly using Linux inter-process communication (IPC) mechanisms or standard networking protocols.

[2] SELinux views all threads within the same process as equivalent as there is no safe way to allow separate access to individual threads on a Unix system.

SELinux easily controls the access of the DDS publisher or subscriber processes in the above example, including access that represents DDS communication with standard type-enforcement policy statements. Assigning a unique type to each process allows SELinux to flexibly control communication, contain those applications using a least-privilege policy, and to protect the application integrity.

3.2 DDS Domain Separation

DDS Domains represent logical groupings of DDS applications used to control communication; DDS applications must participate in the same DDS Domain in order to communicate. DDS Domains are a core concept to DDS and can act as a fundamental building block to secure systems built with DDS. However, the separation offered by DDS Domains is, by design, enforced solely by the DDS libraries and is intended as a functional rather than secure separation mechanism. Therefore, an exploitable flaw in one DDS application could allow an attacker to break the DDS Domain isolation and attack DDS applications participating in other DDS Domains.

The access control, authentication, and transport layer security features integrated into RTI's DDS have important security benefits and offer additional levels of separation from DDS Domains. However, these security measures alone cannot achieve full DDS Domain separation that can survive flawed applications. In particular, threat models that assume cooperating malicious applications cannot be addressed without a separate, independent layer of control such as SELinux.

To understand the security added by SELinux to DDS Domain separation, consider two DDS Domains, each with multiple publisher and subscriber applications. Enforcing DDS Domain separation using SELinux can be achieved by creating four SELinux types: a publisher and subscriber type for each DDS Domain. An SELinux policy can then be crafted that only allows communication between processes within the same domain. For local communications using Linux IPC, the SELinux policy directly allows the required access. For network communications SELinux controls DDS network communications in two ways: network access control and IPSec network labeling.

Network access control leverages the predictable and standardized network port usage of DDS. SELinux allows applications to control a specific processes access to the network in terms of ports, IP addresses, network interfaces, and protocols using either the legacy SELinux network controls or the newer IPTables-based packet labeling. Regardless of which controls are used, DDS Domain separation is enforced by strictly controlling network access to those ports associated with a particular DDS Domain.

Leveraging the network access control features of SELinux brings strong, independent control over the communication of DDS based applications. Even if multiple DDS applications in separate DDS Domains are exploited, they can be prevented from communicating as they will not be allowed to communicate over the same network ports. Further, SELinux can restrict the ability of applications to generate raw IP packets to circumvent access controls. Of course, these controls are still built upon the overall security of the connected network.

IPsec labeled network adds an additional level of security by leveraging IPsec to reliably convey a process type across the network. The IPsec labeling is transparent to the application and allows a remote system to determine the type of remote processes without relying on the security of the network transport[3].

3.3 DDS Application Integrity

Strong protection of application integrity is one of the benefits of SELinux over other access control mechanisms. Type enforcement can be used to protect configuration files, executables, libraries, audit logs, and other application data from corruption or tampering by users or other applications running on a system. This protection extends to administrative 'root' users and other privileged processes.

Protection of application integrity can reinforce other security measures utilized in DDS-based applications. For example, by only allowing the correct DDS applications to read certificates and private keys used for the DTLS authentication and transport layer security features of RTI's DDS, SELinux can prevent rogue software from posing as a trusted publisher or subscriber. Similarly, SELinux can provide an important component of the runtime security when creating tamper resistant applications by preventing access to software executables and libraries.

4 RTI's DDS and SELinux Performance

A security architecture is not viable unless it can be enabled in all situations, including those that require the high performance offered by RTI. Through careful design and implementation, the performance impact of SELinux has been reduced so as to achieve near native Linux performance. The minimal performance impact of an active SELinux security policy allows it to be enabled by default as a part of Red Hat Enterprise Linux 4 and 5.

SELinux introduces a small performance overhead on each system call, which has several performance implications:

- Performance of code that does not involve system calls, including computationally intensive floating or fixed-point math, is in no way effected by SELinux.
- The performance impact of SELinux for a specific application depends on system call usage. For example, the performance impact would be smaller for an application that makes infrequent write calls with a large amount of data compared to an application that makes frequent write calls with a small amount of data.

[3] More information on SELinux network security, including information on IPsec labeling can be found at http://securityblog.org/brindle/2007/05/28/secure-networking-with-selinux/.

- Benchmark data created using micro-benchmarks, such as lmbench[4], will typically show worst case performance with typical application workloads experiencing much lower performance impact[5].
- The performance impact tends to be fixed and predictable both in terms of CPU utilization and latency. Careful design allows SELinux access checks to avoid long or unpredictable impact on system call latency.

Initial measurements of latency generated by running the standard RTI latency measurement utility with and without SELinux enabled show small increases in latency. The test setup utilized Red Hat Enterprise Linux 5 with the legacy SELinux network access controls and RTI Data Distribution Service 4.3d, both 32bit versions. These performance measurements are useful primarily for comparing relative performance rather than absolute performance, as little overall performance tuning was performed. Further, the network access controls used show worst case performance and later versions of the Linux kernel beyond that found in Red Hat Enterprise Linux 5 include many SELinux performance enhancements that will improve these results[6].

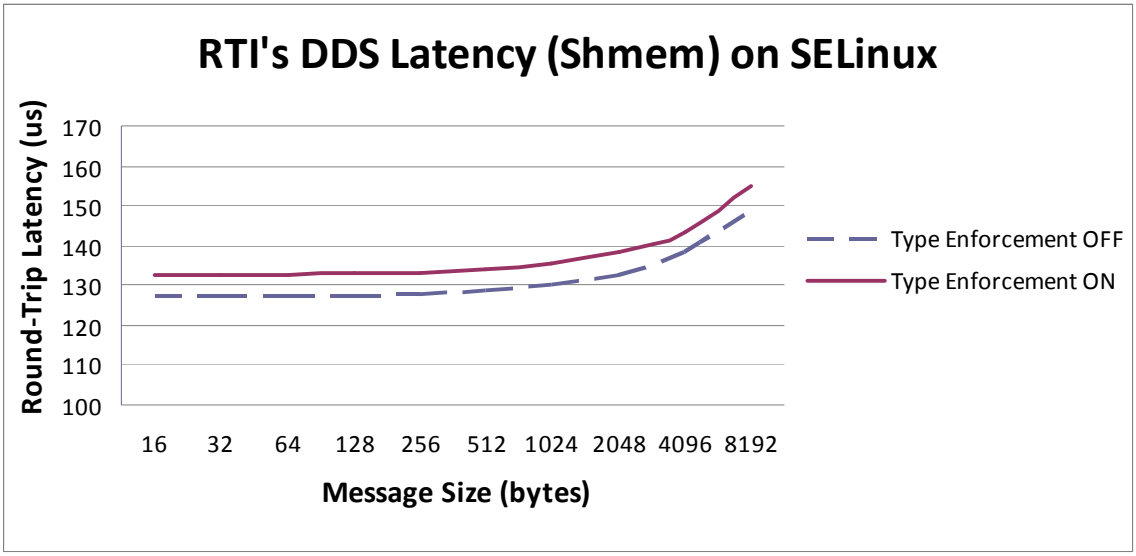

Figure 1 - Comparison of latency with and without SELinux enabled for the shared memory transport.

Figure 1 shows the relative latency achieved using the shared memory transport with and without SELinux. Figure 2 shows the same measurement using the IPv4 transport. Both

[4] http://www.bitmover.com/lmbench/

[5] Some early performance measurements showed this clearly – see http://www.nsa.gov/selinux/papers/freenix01/freenix01.html.

[6] Summary of some recent performance improvements can be found at http://james-morris.livejournal.com/2153.html and http://james-morris.livejournal.com/31714.html.

tests confirm that the SELinux introduces only a small, linear increase in latency of approximately 2%.

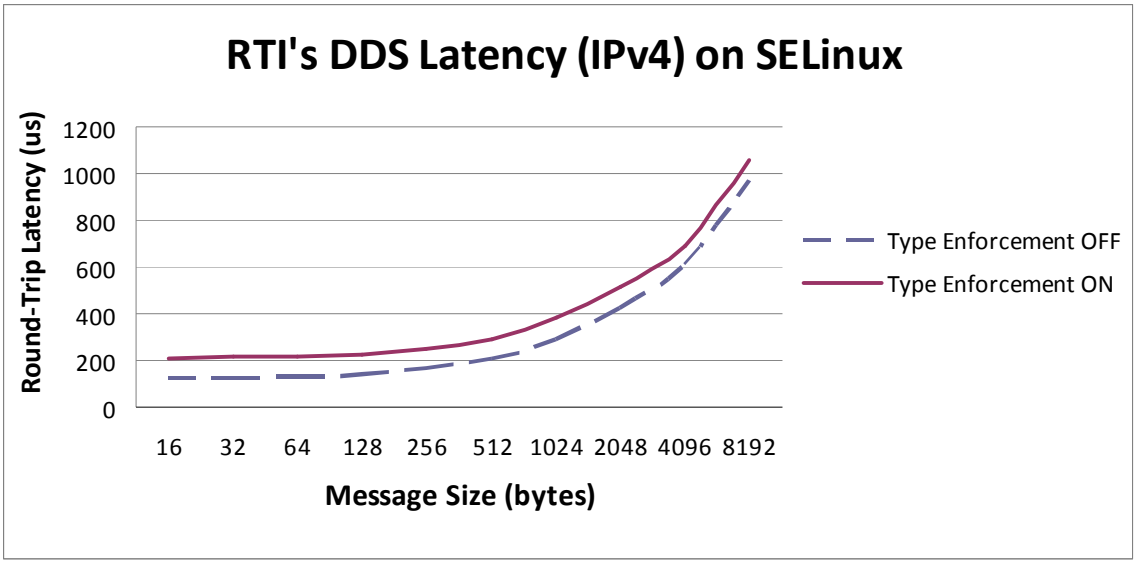

Figure 2 - Comparison of latency with and without SELinux enabled for the IPv4 transport.

5 Meeting Government Security Requirements

Solutions targeting the US Government are often required to meet stringent security requirements and configuration guidance checklists. These range from the baseline configuration mandated by the Defense Information Systems Agency (DISA) Security Technical Implementation Guides (STIGS)[7] to requirements for classified, multi-level systems outlined in the Director of Central Intelligence Directive 6/3 (DCID 6/3) and National Security Systems (NSS) Instruction 1253. Meeting these requirements is often a challenge from a technical, documentation, and developer knowledge point-of-view.

Meeting government security requirements impacts all phases of solution development and deployment. The requirements must be met technically, requiring careful design, implementation, and testing. Then the solution is evaluated for compliance, typically requiring documentation, testing, and final decision process that balances threats, solution security posture, and mission need to determine if and how a system can be fielded. Documentation includes information on likely threats, intended usage, design, and operation of the solution. Finally, a fielded solution must be maintained and monitored in accordance to the documented procedures.

[7] http://iase.disa.mil/stigs/index.html

To meet these challenges with Linux, the Certifiable Linux Integration Platform (CLIP) was created[8]. CLIP provides a security hardened operating system platform to host secure applications. CLIP defines a specific configuration of Red Hat Enterprise Linux 4 and 5, including SELinux policy, designed to provide the foundation for hosting secure applications. This configuration consist of a separation of roles, mandatory access control (MAC), discretionary access control (DAC), and data separation. With this foundation in place, the hosted application needs only to concern itself with the specific security details of its task and not necessarily those associated with these overhead functions. By using CLIP, implementers can provide evidence of compliance with established operating system security requirements. These established operating system security requirements are:

- Director of Central Intelligence Directive 6/3 "Protecting Sensitive Compartmented Information within Information Systems" (DCID 6/3) Protection Level 4 (PL4)
- National Security Systems (NSS) Instruction 1253 "Security Controls Catalog for National Security Systems" High Impact requirements
- Department of Defense (DoD) Instruction Number 8500.2 "Information Assurance (IA) Implementation" MAC I Classified requirements
- Defense Information System Agency (DISA) Information Assurance Support Environment (IASE) Security Technical Implementation Guides (STIG) Unix V5R1

This section will provide a brief technical overview of these requirement sets and describe how CLIP can be used to meet requirements with RTI's DDS.

5.1 Technical Overview of Security Requirements

Most security requirement sets, including those referenced above, cover a variety of subject areas including:

- **Confidentiality:** SELinux policy is used in CLIP to guarantee that only those entities with sufficient access approval may process sensitive data. The extensible nature of SELinux policy enables a developer to manage sensitive data, and create a security policy that exposes this data on a need-to-know basis. An example of a secure application which would benefit using CLIP is a Cross Domain Solution, which needs to have fine-grained control over the disclosure of information, most of which could be managed by proper configuration of SELinux policy.
- **Integrity:** A secure system must protect against unauthorized modification of data. Data integrity need not be limited to system security relevant information, but all information contained on the system. The mandatory access controls provided by SELinux ensures the integrity of the data.

[8] http://oss.tresys.com/projects/clip

- **Availability:** SELinux policy isolates data into separate security domains. CLIP provides a utility to backup file security labels, allowing overall filesystem backup to occur without affecting the security relevant state of the filesystem.
- **Accountability:** In any type of secure system, it is essential to maintain accountability for security relevant events. CLIP uses system call auditing, combined with the auditing and user authentication capabilities of SELinux, to provide administrators with detailed information about all security relevant changes to a system's state.

5.2 Creating Solutions with CLIP

The CLIP package includes several software packages, a full SELinux policy, and a set of repeatable and customizable configurations in the form of Kickstart[9] scripts. CLIP also includes extensive documentation available by request that is suitable as the starting point for the documentation required by certification and accreditation.

Using the provided Kickstart script, which automates the installation and configuration of Red Hat Enterprise Linux, a developer can quickly create a hardened CLIP system. The installed system will meet all of the target requirement sets of CLIP and can be used as a development platform. During the development of the applications that form the complete solution, additional SELinux policy can be added to protect and control those applications.

5.3 Designing Applications to Meet Government Requirements

A solution designed to meet government security requirements requires both a secure platform, such as CLIP, and secure applications. Secure application design, particularly secure applications designed to maximize the benefits of MAC, is a large topic. However, there are several guiding principles that can form a useful starting point:

- Place the minimal amount of trust possible in each application on the system. This design principle is accomplished through careful design to produce an architecture where the security of the system relies on a small number of carefully designed components.
- Grant each component the least amount of privilege necessary to correctly function. This principle is the tactical complement of the reduced trust goal. After architecturally reducing the trust in each component, grant the component the minimal necessary privilege via system security policies. This limits the damage that can be accomplished in the event of an exploit.
- Divide applications into multiple, cooperating processes rather than a single monolithic process. This allows further reduction in trust of each process and

[9] http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Installation_Guide-en-US/s1-kickstart2-whatis.html

yields functional benefits, including the efficient use of multiple core processors as well as the ability to distribute workloads across several systems.

- Rely on operating system security functions before application security functions. Rather than enforce security in the applications, where possible shift the security "heavy lifting" to the operating system functions to gain the advantage of the careful design and extensive testing of these features. Example functions include the Pluggable Authentication Modules (PAM), Linux Audit Subsystem, and SELinux.
- Enforce the flow of information through the system with SELinux. SELinux allows the creation of assured pipelines[10], where the flow of information through the system components is enforced by the SELinux policy. The careful control of information flow within a system allows a wide variety of security goals to be achieved, including data separation and confidentiality.

By providing a high-performance middleware layer, RTI simplifies the creation of applications that follow these design principles. The peer-to-peer nature of RTI'S DDS simplifies the creation of cooperating, least-privilege processes that can easily be secured by SELinux.

6 Summary

RTI Data Distribution Service is a high-performance, distributed middleware layer for real-time applications. SELinux is a flexible Mandatory Access Control mechanism available as a part of Linux, including Red Hat Enterprise Linux. SELinux provides a new level of host security, data separation, and role-based access control and allows the creation of solutions that are secure even in the face of zero-day exploits and meet the most stringent commercial and government security requirements. This new level of security is achieved by the use of *type-enforcement*, which applies unique access control policies to individual processes.

RTI and SELinux are well suited because of the peer-to-peer, library based implementation of RTI's DDS. This design embeds DDS into third-party applications without the need for a single, central application connecting DDS clients and servers. This design fits well with the process oriented access control model of SELinux, allowing comprehensive control of DDS applications, including controlling which DDS applications are able to communicate. Combining RTI and SELinux allows the creation of solutions that meet the most stringent security requirements, including those mandated by the US Government.

For an overview of DDS and its applicability to your needs, you may be interested in the RTI whitepaper "Is DDS for You": https://www.rti.com/mk/DDS.html

[10] http://cs.unomaha.edu/~stanw/papers/csci8920/loscocco.pdf