



# **CLIP & Sentinel 3: Solid Foundation, Secure Messaging Solution**

**September, 2009**

Copyright ©2009 Tresys Technology, LLC. All Rights Reserved.

Other names and brands may be claimed as the property of others. Information regarding third party products is provided solely for educational purposes. Tresys Technology, LLC is not responsible for the performance or support of third party products and does not make any representations or warranties whatsoever regarding quality, reliability, functionality, or compatibility of these devices or products.

## 1 Introduction

Government and business are faced with an increased need to provide secure computer systems. The escalating number and sophistication of attacks, lack of strong security in most computer systems, and the increasing value of computer systems to organizations is creating a perfect storm requiring a new level of security. Many organizations are adopting a high-level approach to providing this increased security by the creation of separated networks, with each of these networks representing a distinct security domain with differing levels of access and security controls.

In the past this approach for many typical organizations involved the use of a single firewall to divide the internet from an internal network, perhaps with a ‘de-militarized zone’ or DMZ for hosting internet facing system. This simplistic division is no longer sufficient. Increasingly organizations are creating separated networks for a variety of security domains, such as credit card processing for financial institutions or control systems for public utilities. Within government, networks are typically divided along classification lines.

All of these approaches, while effective, typically suffer from a common weakness: strict separation without adequate, and sometimes critical data interchange. Separated networks, even those that have the most stringent security requirements, almost always require data to flow in or out. When this happens, those connection points between networks become the key point of attack and require computer systems with a higher level of security.

Systems designed to bridge networks representing different security domains are called Cross-Domain Solutions (CDS) or guards. CDSs are designed to facilitate the transfer of data between networks without compromising the security benefits of the separation. They typically inspect data and meta-data to enforce organizational policies about data confidentiality and integrity.

One such CDS is the Nexor family of guards designed to provide safe transfer of data between separated networks. The Nexor Sentinel is a high assurance mail guard, single-box appliance designed to protect an organization by validating that inbound and outbound electronic messages conform to the security policy of the protected domain. The Nexor Chat Guardian provides similar inspection of chat data using the XMPP protocol.

This paper describes how Tresys and Nexor combined the latest security technology, including Red Hat Enterprise Linux 5, Security Enhanced Linux (SELinux), the Certifiable Linux Integration Platform (CLIP), and the Nexor Watchman policy enforcer to create secure solutions for inter-domain communication. The development process used by Nexor and augmented by Tresys expertise allowed the creation of solutions with the highest security, best performance, and lowest development cost.

## 2 Challenges

CDSs, which are designed to bridge separated networks to allow data flow between security domains, face a particularly tough set of challenges. The systems must not compromise the security gained through the creation of separate networks while at the same time often meeting stringent performance or reliability requirements. Additionally, these systems, though secure, typically are held to similar standards as other systems in terms of cost, development time, and compatibility.

### 2.1 Secure Data Transport

The secure transport of data between security boundaries is achieved by first enforcing network separation. While this may seem counter-intuitive, it is similar to how hydro-electric dams are designed. In order to ensure that water flows through the power generating turbines first a dam is built. The dam, which is capable of stopping the flow of all water, is the key for allowing the controlled release of water through the turbines. If the dam were incapable of first halting the flow of water there would be no way to ensure that all of the water flowed through the turbines.

Similarly, CDSs are designed to block the flow of information between connected networks except through tightly controlled paths. These paths funnel the data through carefully designed data inspection components. The controlled flow ensures that the data inspection components have an opportunity to inspect all data in order to make a release decision.

The data inspection components are particularly challenged for secure data transport. These components must be capable of inspecting data safely and determining whether data should pass based upon an organizational policy. Real-world data formats tend to be complex, insufficiently specified, and difficult to fully inspect. Additionally, the data may not be able to be inspected easily in a single operation because it is either very large (e.g., for multi-gigabyte transfers) or streamed as a series of packets in real-time (e.g., chat or voice-over IP). Finally, the specification language or configuration for the organizational policy must be powerful and expressive while being configurable and maintainable by administrators.

### 2.2 System Integrity

CDSs, because they are targets for attack, must be able to maintain system integrity even in the face of concentrated and sophisticated attacks. This is complicated by the fact that they also must inspect and process data, often in complex formats, provided from networks representing different security domains. Finally, any failure in system integrity must be detected quickly and securely audited.

## 2.3 Certification and Accreditation

In addition to the challenges of designing a secure system, CDS providers must be able to demonstrate the security of the systems that they build as part of a security certification and accreditation (C&A) process. This requires effort during system design and build in order to meet the specified requirements and during the C&A process. While C&A processes differ in the level and kind of scrutiny, they typically require a large amount of evidence. Designing systems to meet these requirements and generating the required evidence requires time and expertise in the C&A process, OS platform, and specific solution.

Ideally, a CDS will not reduce the security benefits provided by the separation between networks. This requires a strict control over the flow of data through the system. The enforcement of the data flow policy requires that a trustworthy component inspect all traffic flowing through the system. These two security goals combined – network separation and controlled data flow – are challenging to meet on traditional operating systems. Most access control mechanisms, such as those found by default on Windows and Linux, are concerned with limiting users' access to data and applications. But it is the control of system processes and resources (not just users and data) that is most meaningful to enabling network separation and controlled data flow.

## 3 Solution Approach

Nexor and Tresys used a multi-prong approach to solving both the general challenges inherent in CDSs and the specific challenges addressed by the Nexor solutions. Nexor chose SELinux as the primary OS security mechanism because it is designed to meet a variety of security goals, including the separation and controlled data flow requirements of CDSs. To effectively use SELinux, they relied on the Certifiable Linux Integration Platform (CLIP), an open source project sponsored by Tresys that provides a secure configuration of Red Hat Enterprise Linux designed to meet a variety of stringent security requirements. Finally, they integrated their secure applications on top of this platform to provide a complete and secure solution. This section will provide an overview of SELinux and CLIP.

### 3.1 Overview of Security Enhanced Linux

SELinux provides a new level of security for Linux systems by enabling the benefits of Mandatory Access Controls (MAC), long used to create the most secure systems for government, with unmatched flexibility. This flexibility allows developers and administrators 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<sup>1</sup>.

### 3.1.1 Type Enforcement

SELinux controls access to system resources, such as files, based upon users, as in DAC, but also controls access based upon the current role of the user, a new concept called *type enforcement*, and optionally multi-level security (MLS) sensitivity and categories. While the user, role, and MLS based controls offered by SELinux are useful, it is the type enforcement access control model that offers an entirely new level of security from other systems. 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 policy writer to assign an abstract identifier, or *type*, to each process, file, or other resource. Access is then allowed in terms of a process type's access to a resource type.

For example, consider a web server reading a web page stored on disk to serve a client request. Assuming the type “apache\_t” is assigned to the web server process and the type “html\_file\_t” assigned to the on-disk 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, including 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,

---

<sup>1</sup> 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.

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 access to function correctly. Under DAC, a vulnerable web browser could disclose confidential files or 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 for correct functioning. For example, the 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 the read documents in the user's home directory, run additional applications, or access special devices could be denied. Under a type-enforcement policy 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 applications on a system, including server oriented applications such as web servers, databases, mail servers, etc. Further, 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 face of flawed software.

### 3.1.2 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 to grant 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 able 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 constant through the life

of the system, making it possible for system designers to prevent the disclosure of confidential data or compromises of system and application integrity.

### 3.1.3 Consistent and Centralized Policy

SELinux implements access control over all system resources, including files, directories, devices, networking, inter-process communication, using the same access control model and centralized policy. All application access is controlled using the same mechanism and syntax, simplifying the design, implementation, and verification of the access control policy. Additionally, there is a single system policy governing all access on a system. The centralized policy allows analysis of a complete system, including complex automated analysis to track the flow of data through a system.

Contrasting, 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 and granularity of control. Further, some access, such as binding to low ports, is controlled by implicit, hard coded rules that are inflexible. The result is that under DAC understanding the total access granted to an application is difficult or impossible.

Finally, granting required access under DAC often requires unfortunate security compromises. For example, many applications must run as root to bind to low ports or exercise other administrative privileges, resulting in applications gaining large amounts of privilege when only a small amount of additional access is required. Such compromises are not required with SELinux, as it is possible to directly grant the minimal required set of access directly to an application.

## 3.2 Certifiable Linux Integration Platform

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)<sup>2</sup> 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 a final decision process that balances threat, solution security posture, and mission need to determine if and how a system can be

---

<sup>2</sup> <http://iase.disa.mil/stigs/index.html>

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.

To meet these challenges with Linux, the Certifiable Linux Integration Platform (CLIP) was created<sup>3</sup>. 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 includes 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

### 3.2.1 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<sup>4</sup> 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

---

<sup>3</sup> <http://oss.tresys.com/projects/clip>

<sup>4</sup> [http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Installation\\_Guide-en-US/s1-kickstart2-what-is.html](http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Installation_Guide-en-US/s1-kickstart2-what-is.html)

complete solution, additional SELinux policy can be added to protect and control those applications.

## 4 Solution Details

In order to reach a broader customer set and to meet the dynamic requirements of a changing customer base the Nexor Sentinel was ported to a CLIP based RHEL 5 platform. The CLIP system allowed Nexor to begin with a base platform that had already been configured to meet many of the requirements their customers had as well as providing them a subset of the documentation the product would need for certification. The CLIP base allowed Nexor to focus on porting their applications and not having to address all of the nuances required to lock down the new operating system they were porting to.

### 4.1 Application Access Control and Information Integrity

Nexor Sentinel is a high assurance mail guard that allows mail to be sent between up to four possible different security domains. In order to meet the strict requirements detailing the separation of data between these domains Nexor built policy for their guarding application on top of the SELinux policy provided by CLIP.

In the simplest configuration the Nexor Sentinel provides bidirectional email between two domains A and B, a web based user interface to manage the guard configuration, and SNMP to query information about the state of the system. To ensure the integrity of the data and to limit the access control of each component SELinux policy was written to create a provable pipeline that could then be evaluated by a certification team.

The SELinux domain that the web based user interface is confined by can only read the configuration files and pass changes to a set of scripts that run in a separate confined domain. The SELinux domain confines the web based user interface so configuration files can only be read and changed by a set of scripts that run in a separately confined domain. The scripts verify the configuration data, update the configuration files, and restart the necessary components. The compartmentalization of this process protects the system if a vulnerability in the web server is exploited by an attacker.

Each security domain has its own set of processes for receiving and sending emails, processing the email through a set of filters, and re-grading the email before placing it in a queue for another domain. Each component of this chain runs in its own domain that is limited to only interact with the component directly preceding and following it in the path. A very simplified diagram of this process is shown in Figure 1.

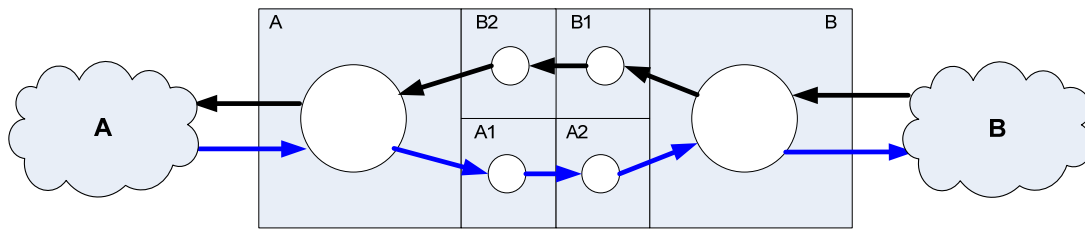


Figure 1 - Simplified email processing pipeline.

In this example a message from a sender in the A domain is received on the A side SMTP daemon. This daemon that passes the message to filters A1 which then passes the message to A2, and finally the regrading process places the message into the queue for the B side SMTP server to be sent out to the recipient. The MAC system has been configured so that the message cannot bypass any part of this pipeline, nor can the message be read by any previous part of the pipeline once passed on.

Nexor Sentinel is a high assurance mail guard, single-box appliance designed to protect an organization by validating that inbound and outbound electronic messages conform to the security policy of the protected domain.

Nexor Sentinel is supplied on two secure platform options, one evaluated to Common Criteria EAL4+, the other to EAL5+. The underlying secure platform ensures network separation of the connected domains, whilst the mail guard application is based upon an EAL4 design, which ensures only policy conformant messages can pass between these connected networks. Nexor Sentinel products have been deployed worldwide by national defense forces and organizations such as NATO.

Designed with security in mind, Nexor Sentinel appliances make use of the evaluated security functionality provided by the underlying platforms to ensure an electronic message can only pass from one domain to the other via a trusted path. The trusted path includes ensuring all messages are scanned by the Nexor Watchman policy enforcement module which validates the messages conform to the defined security policy. Non-conformant messages are rejected preventing the potential damage caused by outbound data loss or inbound malicious content.

To meet the interoperability and security needs of the world's most demanding organizations, Nexor separates content and protocol conversions from message security. Nexor Sentinel provides message security, while Nexor Border Gateway provides content and protocol conversion. This approach reduces solution assurance and accreditation costs.

Sentinel 3 is designed to work on HP server hardware using Red Hat Linux 5.0. This combination is evaluated to EAL4+ and takes advantage of the latest hardware and

SELinux's security model. This meets the customer mainstream and support requirements.

While developing Sentinel 3 the Nexor team took the guard design back to first principles. A redesign of the architecture was conducted to enhance the flexibility to allow the handling of more than just the SMTP and X400 messaging protocols. This new generic guard architecture has the capability to handle any data protocol, and this was recently demonstrated by the production of an Instant messaging guard based on the XMPP protocol known as Nexor Chat Guardian.

Sentinel 3 and Chat Guardian represent the generic guard architecture that will form the base of Nexor's new family of guarding products.

The Sentinel 3 product has been developed with Nexor's latest development methods, which incorporated a number of radical changes to ensure the security and quality of our products. Nexor's staff has been trained in the latest security techniques and vulnerabilities and hold certifications such as CSSLP (Certified Secure Software Lifecycle professional) and CISSP (Certified Information Systems Security Professional), and CEH (Certified Ethical Hacker). In addition to this the development cycle has incorporated a number of concepts such as Threat Analysis and the introduction of Static Code Analysis tools to ensure there is an objective review of the code and system to ensure security.

## 5 Meeting Real World Threats

In order to understand the security value of the solution approach chosen by Nexor and Tresys it is helpful to understand how the platform can mitigate real world vulnerabilities. This section examines two vulnerabilities and shows how they are mitigated by CLIP and the Nexor guards. While it is important that these vulnerabilities are mitigated, what is particularly interesting is that both of the vulnerabilities, and any zero-day exploits based upon them, were mitigated without any system changes.

Both vulnerabilities are mitigated by SELinux. The first exploit was mitigated by the default SELinux policy shipped in Red Hat Enterprise Linux for all processes (version 4 at the time of the vulnerability of the release). The second vulnerability was mitigated by the default SELinux policy for most confined processes. Since the release of the vulnerability the default SELinux policies have been extensively reviewed and can mitigate the vulnerability for more processes.

### 5.1 Proc Filesystem Race Condition

Linux 2.6 kernel versions before 2.6.17.6 are vulnerable to a local root privilege escalation due to a race condition in the procfs pseudo-filesystem. The procfs filesystem, typically mount at "/proc", is used by the Linux kernel to expose information about running processes and the system.

The vulnerability, identified as CVE-2006-3626, allows an unprivileged application to cause a file in `procfs` to become `setuid` with the user and group owner set to root. Files that are `setuid` cause processes to change their effective user id to the file owner, root in this case, upon executing the file.

In addition to allowing the modification of the permission bits on the file, this vulnerability allows unprivileged applications to control the contents of the file. By allowing an unprivileged application to cause a file to become `setuid` and root owned, this vulnerability opens the door for privilege escalation. Combined with the ability to control the contents of the file, this vulnerability allows unprivileged applications to execute arbitrary code as root.

SELinux mitigated this vulnerability by preventing the change of the file to `setuid`, thereby stopping the first step in the vulnerability. For most confined applications, SELinux also prevents the execution of the file or limits the privilege of the process after execution.

## 5.2 SMM Cache Poisoning Vulnerability

One key part of CLIP is the protection of security-relevant system resources. This protection actually extends from protecting the system from software vulnerabilities to protecting the system from a newly relevant class of vulnerabilities: hardware vulnerabilities. One example of this class is the Intel System Management Mode (SMM) cache-poisoning attack.

SMM is a highly-privileged mode supported by the CPU and BIOS. Typically it handles activities such as addressing thermal events (turning on a fan), or adjusting display brightness. Where operating system kernels typically run in a privileged mode, the SMM is actually a privilege-level above the operating systems. The operating systems cannot prevent SMM from being entered. The SMM code itself is typically provided by the mainboard manufacturers, however a vulnerability in the way CPUs cache memory allows an attacker to run arbitrary code in this highly-privileged mode.

The attack could be used on Windows and Linux operating systems. The Linux attack once again used the `procfs` pseudo-filesystem that not only exposes software resources of the kernel and processes but also exposes some of the internals of the hardware. The vulnerability lies in the Intel x86 hardware itself and has been present since the 386-class of processors. By manipulating certain registers, specifically the memory type range registers (MTRR), a malicious user or application can cause SMM memory to be loaded into the cache on the processor. The user can then manipulate the memory inserting malicious code in the cache. The user then causes the hardware to execute the malicious code and since the processor has the memory cached, it uses the cached version instead of the version provided by the mainboard manufacturers.

SELinux, and more specifically the configuration of SELinux provided by CLIP, successfully mitigates the exposure to this vulnerability by restricting access to the file used to configure the MTRRs. In a traditional system supporting discretionary access control, Linux or Windows, any privileged process can manipulate the MTRR thus exposing the system to this cache poisoning attack. While this file is used by certain applications for system configuration, such as the X Window System, SELinux prevents other processes – even privileged processes – from manipulating this security-relevant file.

## 6 Summary

Government and business are faced with an ever increasing need to provide secure systems that can guarantee that personal and sensitive data is held securely, while at the same time allowing necessary data flow between different security domains to take place. The points at which data flows are traditionally the weak points in secure systems. It is critical to ensure that the data in transit is offered the same protection as the data held at rest within the secured domain.

In developing Sentinel 3, Nexor and Tresys have looked to offer this protection and have validated a number of key perspectives relevant to cross domain and high security solutions.

First, that the foundational security delivered with SELinux supports the development (and successful certification) of cross domain solutions that are critical in meeting evolving data and information sharing demands, even when they span disparate security domains.

Second, that the use of the open source CLIP development platform greatly facilitates the development of certifiable cross domain solutions like Sentinel. By building on CLIP, developers can avoid the costs associated with building and proving that level of security. Perhaps more importantly, they can remain focused on building applications that meet critical performance requirements, confident that underlying security is built in.

Finally, that the application of MAC and type enforcement make a difference in preventing real world attacks. Both the SSM cache poisoning and the Proc Filesystem Race Condition vulnerabilities would be successfully mitigated by the strong security embodied in this solution.

The combination of SELinux, CLIP and Nexor ensures that Nexor Sentinel offers trust to organizations looking to ensure their sensitive information is secured while in transit between different security domains.