Reprinted from the

Proceedings of the Ottawa Linux Symposium

June 26th–29th, 2002
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc.*
Stephanie Donovan, *Linux Symposium*
C. Craig Ross, *Linux Symposium*

Proceedings Formatting Team

John W. Lockhart, *Wild Open Source, Inc.*

A Distributed Security Infrastructure for Carrier Class Linux Clusters

Makan Pourzandi, Ibrahim Haddad, Charles Levert
Miroslaw Zakrzewski, Michel Dagenais
Open Systems Lab, Ericsson Research Canada
8400 Décarie Blvd, Town of Mount-Royal (QC) Canada H4P 2N2
Makan.Pourzandi@ericsson.ca, Ibrahim.Haddad@ericsson.com, Charles.Levert@ericsson.ca
Miroslaw.Zakrzewski@ericsson.ca, Michel.Dagenais@polymtl.ca

Abstract

Traditionally, the telecom industry has used clusters to meet its carrier-class requirements of high availability, reliability, and scalability, while relying on cost-effective hardware and software. Efficient cluster security is now an essential requirement and has not yet been addressed in a coherent fashion on clustered systems. This paper presents an approach for distributed security architecture that supports advanced security mechanisms for current and future security needs, targeted for carrier-class application servers running on clustered systems.

Keywords: Linux, Security, Carrier Class Clusters, Distributed Infrastructure, IPSec, LSM.

1 Introduction

The interest in clustering from the telecommunication industry originates from the fact that clusters address carrier-class characteristics such as guaranteed service availability, reliability, and scaled performance, using cost-effective hardware and software. There are several efforts on going to use Linux as basic block for building next generation telecom clusters [12, 7]. These carrier-class characteristics have evolved and now include requirements for advanced levels of security. However, there are few efforts to build a coherent distributed framework to provide advanced security levels in clustered systems.

Our work targets implementing security mechanisms for soft real-time distributed carrier-grade applications running on large-scale Linux clusters. These clusters are dedicated to run only one application. They must provide five nines availability (99.999% uptime) that includes hardware upgrade and maintenance and operating system and applications upgrades. In such clusters, software and hardware configurations are under the tight control of administrators. The communications between the nodes inside the cluster and to external computers are restricted.

In this paper, we present the rationale behind developing a new architecture, named Dis-

tributed Security Infrastructure (DSI). We describe the main elements of this architecture, and discuss our preliminary results. DSI supports different security mechanisms to address the needs for telecom application servers running on clustered systems. DSI provides applications running on clustered systems with distributed mechanisms for access control, authentication, integrity of communications, and auditing.

The paper is organized as follows: Section 2 illustrates the need for a new approach to security requirements for carrier-class clustered servers. Sections 3 and 4 discuss the DSI architecture and its characteristics. Sections 5 to 12 present the main elements of the design. Section 13 compares our approach to other related work. Section 14 presents some preliminary results. Section 15 concludes with our ongoing work and future plans.

2 The need for a new approach

There exist many security solutions for Linux clustered servers ranging from external to cluster solutions, such as firewalls, to internal solutions such as integrity checking software. However, there is no solution dedicated for clusters. The most commonly used security approach is to package several existing solutions. Nevertheless, the integration and management of these different packages is very complex, and often results in the absence of interoperability between different security mechanisms. Additional difficulties are also raised when integrating these many packages, such as the ease of system maintenance and upgrade, and the difficulty of keeping up with numerous security patches and upgrades.

Carrier-class clusters have very tight restrictions on performance and response time. Therefore, much pressure is put on the system designer while designing security solutions. In fact, many security solutions cannot be used due to their high resource consumption.

Currently implemented security mechanisms are based on user privileges and do not support authentication and authorization checks for interactions between two processes belonging to the same user on different processors. However, for carrier-class applications, there are only a few users running the same application for a long period without any interruption. Applying the above concept will grant the same security privileges to all processes created on different nodes for a long period of time. This is due to the fact that the granularity of the basic entity for the above security control is the user. For carrier-class applications, some classes of actions require fine-grained access control to some resources, or enforcement of specific security policies, or both. Therefore, the user-based granularity is not sufficient. By consequence, DSI is based on a more fine-grained basic entity: the individual process.

3 DSI characteristics

As part of a carrier-class clusters, DSI must comply with carrier-class requirements such as reliability, scalability, and high availability. Furthermore, DSI to answer the needs explained in 2 supports the following requirements:

- Coherent framework: Security must be coherent through different layers of heterogeneous hardware, applications, middleware, operating systems, and networking technologies. All mechanisms must

fit together to prevent any exploitable security gap in the system. Therefore, DSI aims at integrating together different security solutions and adapting them to soft real-time applications.

- Process level approach: DSI is based on a fine-grained basic entity: the individual process.

- Maximum performance: The introduction of security features must not impose high performance penalties. Performance can be expected to degrade slightly during the first establishment of a security context; however, the impact on subsequent accesses must be negligible. It is possible to disable security by security administration decision.

- Pre-emptive security: Any changes in the security context will be reflected immediately on the running security services. Whenever the security context of a subject changes, the system will re-evaluate its current use of resources against this new security context.

- Dynamic security policy: It must be possible to support runtime changes in the distributed security policy. Carrier-class server nodes must provide continuous and long-term availability and thus it is impossible to interrupt the service to enforce a new security policy.

- Transparent key management: Cryptographic keys are generated in order to secure connections. This results in numerous keys that must be securely stored and managed.

- Framework supports fast detection and reaction to security incidents.

4 Architecture

DSI targets clusters and, in doing so, introduces original contributions to their security. Some of its parts, however, such as its Access Control Service and its use of security contexts and identifiers, owe much to existing propositions, such as Security Enhanced (SE) Linux [5].

4.1 Distributed architecture: Inside the cluster

DSI has two types of components: the management components and security service components. DSI management components define a thin layer of components that includes a security server, security managers, and a security communication channel (Figure 1). The service layer is a flexible layer, which can be modified or updated through adding, replacing, or removing services according to the needs of the cluster.

The security server is the central point of management in DSI, the entry point for secure operation and management, and alarms coming from the intrusion detection systems from outside the cluster. It is the central security authority for all the security components in the system. It is responsible for the distributed security policy. It also defines the dynamic security environment of the whole cluster by broadcasting changes in the distributed security policy to all security managers.

Security managers enforce security at each node of the cluster. They are responsible for locally enforcing changes in the security environment. Security managers only exchange security information with the security server. The secure communication channel provides encrypted and authenticated communications between the security server and the security

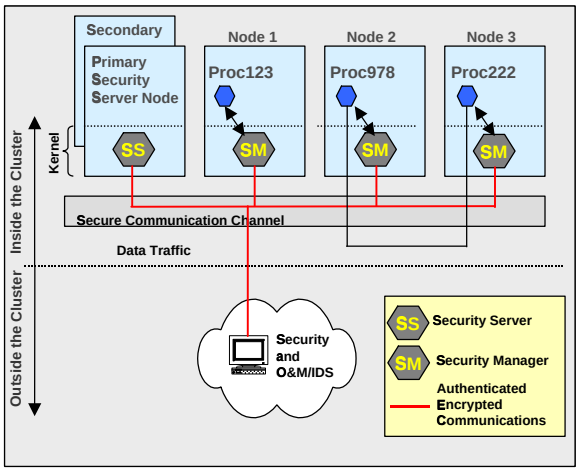

Figure 1: Distributed Architecture of DSI

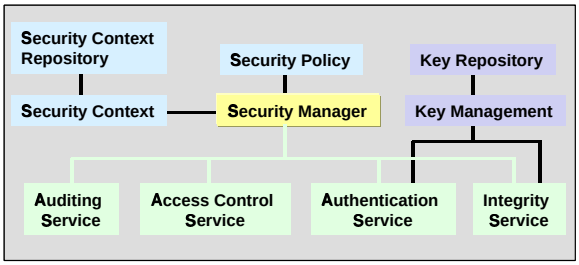

Figure 2: DSI Services

managers. All communications between the security server and the outside of the cluster take place through the secure communication channel. To avoid a single point of failure, the security services run on an equally hardened secondary security server as hot swappable services. These nodes are security hardened versions of Linux distributions to maximize security. All connections from and to these nodes are encrypted and authenticated.

The security mechanisms are on widely known, proved, and tested algorithms.

For the security mechanisms to be effective, users must not be able to bypass them. Hence, the best place to enforce security is at the kernel level; all security decisions, when necessary, are implemented at kernel level through DSI Security Module (DSM) [9]. This module is loaded on each node by the security manager upon its initialization.

4.2 Service based approach

The DSI architecture at each node is based on a set of loosely coupled services (Figure 2).

The security manager controls different security services on the node. This service-based architecture has the following advantages:

- The service implementation is separated from the rest of the system. By keeping the same API, the service implementation can be changed without affecting the application. However, an API for accessing security services is provided at user level for applications with special security needs (Section 10.1).

- It runs only predefined services according to the needs, performance issues, or security environment. In addition, services can be replaced on run time without major drawback on the running application. This enables the architecture to be modified and to resist changes throughout the system's lifetime.

- It is possible to add, remove, or update different services without administrative intervention. This reduces configuration errors due to the numerous security patches that need to be applied manually.

The security manager discovers the different services. Each service, upon its creation, sends a presence announcement to the local security manager, which registers these services and

provides their access mechanisms to the internal modules. There are two types of services: security services (access control, authentication, integration, auditing) and security service providers that provide services to security managers.

The security management is implemented at all levels of DSI. There is a complete chain of commands from security administrators (human beings) of the cluster to different DSI components inside each nodes kernel. The administartors access the Security Server through the SCC. The Security server interprets the commands and propagate them to the Security Managers. Security managers translate these to control settings for different security services. For example, security administrator detecting a back door on some software used indicates that some external IP can not accessed from cluster nodes. This is sent to the Security Server and translated as modification in DSP forbidding all connection to defined IP address. Propagated through SCC to security managers, this policy decision is enhanced at DSM level.

5 Security Server

The security server is the reference for all security managers and has the authority to declare a node as compromised. It subscribes to all updates to keep its cache of different security contexts up to date, which makes it the ideal candidate for running Intrusion Detection Systems (IDSs). It has a local certification authority[1] (CA). This last issues the certificates for secondary certification authorities run by the security managers. The primary tasks for security server include auditing, triggering alarms and warnings to inside and outside the cluster, managing the distributed security policy, receiving, interpreting and propagating security management operations to security managers.

[1]We mean that the CA is used for using inside the cluster.

6 Security Manager

The security manager enforces security on each node. It is primarily a lookup service to register different security services and service providers and connect them together. The security manager is instantiated at boot time with digital signatures to make certain that it is not replaced with a malicious security manager. Upon its creation, it joins the DSI framework and exchanges keys with the security server. Each security manager must publish any change to the security contexts of its local entities involved with remote entities and subscribe to changes in the security contexts of remote, related entities (see Section 8). The primary tasks for security managers include access control, process authentication, audit management, alarm publication, key management, as well as maintenance, and update of the locally stored distributed security policy.

7 Secure Communication Channel (SCC)

The secure communication channel provides secure communications for the security components inside and outside the cluster. Within the cluster, it provides with authenticated and encrypted communications among security components (Figure 3). It supports priority queuing to send and receive out-of-band alarms. It is coupled to the security manager by an event dispatching mechanism. For large-scale clusters, an event driven approach based

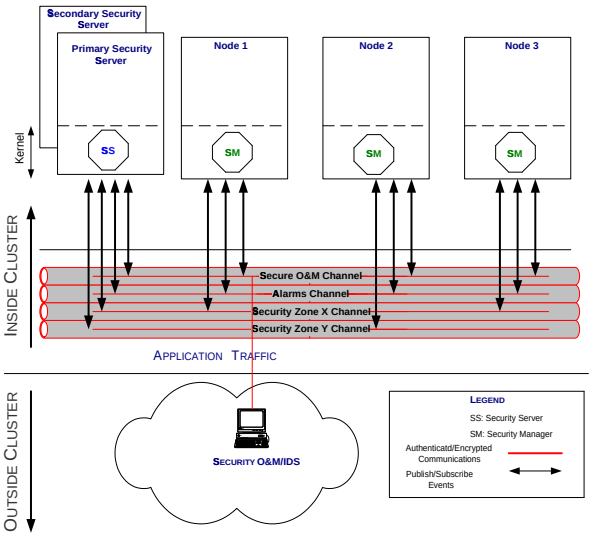

Figure 3: SCC is based on an event-driven logic and different channels

on subscription to events from defined channels reduces the system load compared to the polling mechanisms. Further more, the benefits of this approach are:

- It does not present a single point of failure.
- It gives the possibility of event filtering, therefore less bandwidth used, and less time used for treating irrelevant information before discarding it.

The secure communication channel provides channels for alarms and warnings, security management, service discovery, and distribution of the security policy. It also provides a portability layer to avoid dependency on the low-level communication mechanisms.

8 Security Context

For efficiency, a security identifier (SID) is defined as an integer that corresponds to a security context. All entities in the cluster have a SID. This SID is added at kernel level and cannot be tampered by users. For example, a structure containing SID is added to the structure presenting the process in kernel [9].

We define Cluster SID (CSID) as the pair of SID associated to the subject and the node where the subject belongs to. CSID is transferred across processors by security managers and interpreted through the whole cluster. Once the security context for a subject is needed outside of the local processor (for instance if this process accesses a remote object), its CSID is sent to the security manager of the node containing the object. The CSID propagation inside the cluster is based on SelOpt open source software implementation [8]. To avoid retransmissions, security managers rely on caching mechanisms.

To ensure the pre-emptive access control, the security manager of the node containing object subscribes through SCC to the event of a possible change in the security context of the access initiator entity.

9 A Coherent Vision: Security Contexts and the Distributed Security Policy (DSP)

Security configuration must be kept simple. Following this approach, DSI relies on a centralized security policy stored and managed on the security server. However, to maintain the cluster's scalability, read-only copies of the policy are pushed from the security server to

the individual security managers through the SCC. This Distributed Security Policy (DSP) is an explicit set of rules that governs the configurable behavior of DSI. Each node, at secure boot time, relies on a minimal security policy that is either stored in flash memory or downloaded along with its digital signature. As soon as the DSP becomes available on a node, it prevails.

DSP allows a configurable behavior for security services. The DSI administrator (a human being) manipulates the primary copy of the DSP that resides on the security server. Thus, it must be represented in a human readable format. The basic update mechanism for DSP is to push a full copy of each new version of the policy through the SCC. However, given the mere size that the policy can take, an incremental update mechanism will be made available.

There can be several possible originating sources for the security policy rules. Manual configuration by the DSI administrator allows the most flexibility, but it rapidly becomes cumbersome. Thus, default policy rules are inferred from the very nature of the various software packages that are installed and running on the system. These default rules codify good security practices. The DSP should only need to be updated because of events such as the installation of new software components, but it should not be updated whenever ordinary recurring events occur. A security session manager handles this kind of events by updating the security context repository. A security context defines privileges associated with each entity. It is defined uniquely through the whole cluster, but it is the responsibility of the security manager who created it.

10 Access Control Service (ACS)

Access control can be defined as the prevention of unauthorized use of a resource [4]. It relies on the notions of subject (or access request initiator), object (or target), environment, decision, and enforcement. The Access Control Service (ACS) assumes that the subjects have been properly authenticated (see Section 11). DSI allows verifying the access control privileges even when subjects and objects are located on different nodes in the cluster. In order to simplify, we handle the access control in two levels: local when subject and object are on the same node and remote when they are on different nodes.

The local access control at each node is based on SID added to the structure in the kernel which, represents each entity (e.g., process, socket...). For local access control, the access rights are the functions of the security IDs of the subject (SSID) and the object (TSID). They are enhanced through DSI Secure Module, which we implemented [9].

10.1 Remote access control

For remote access control, we extend the local access control mechanisms by adding a new parameter: the security node ID. Therefore, the access rights are no more just the functions of the subject and target security IDs, but as well, the function of the security node ID (NSID). The SSID along with the NSID are sent to the node containing the object added to each IP packet as IP options (Figure 4).

A first level of access control based on SSID is done at this level, by the security manager on the source node. This is completely transparent to the process initiating the communication.

The second level of security check is done by the security manager on the target node based on SSID and NSID. This is also transparent to the process receiving the communication. Till this point, the access control decision is transparent. It is granted or denied based on SSID and NSID (i.e., CSID). This level of access control is enough for majority of the applications.

For applications with needs for finer grained access control, DSI provides an API allowing to take into account the SSID and NSID when making access decision to resource on target node.

We believe that it is not possible to implement an enough flexible and usable fine grained access control based only on platform. For fine grained security the application needs to collaborate with the security mechanisms provided by the platform. The application through DSI API asks to be set a new SID based on the SSID and NSID. Notice that the SSID and NSID are not revealed to the application. The application asks to change the SID based on the communication mechanism. For example, for a server process it means to pass the socket as an argument to DSI API. The security manager sets the new SID for the server process based on the original SID of the server (TPSID), SSID and NSID contained in each IP packet.

Therefore the final access control decision is based on SSID, NSID, TPSID, and TSID.

Remark that DSI targets the carrier class platforms, with software environment under tight control with few applications running on the cluster. Therefore, even if the support of an additional API for security is a burden for application developers, we believe that only a small percentage of applications need to be modified. As for the majority of applications, transparent

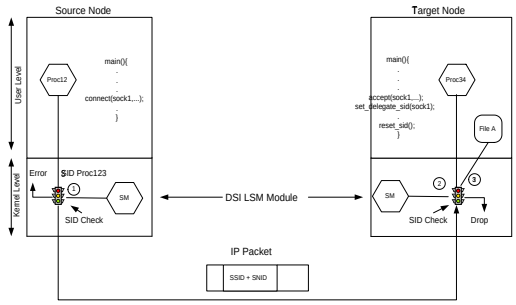

Figure 4: **Secure Remote access control:** Following security checks are done: 1) local check by the security manager (SM), if access agreed the Node ID (NSID) and Process SID (SSID) are added to each IP packet sent to the node 2) Application transparent check by the SM based on NSID and SSID, if access agreed the connection is established 3) For some applications with special security needs, the application can choose to further enhance the security by taking into account the SSID and NSID. To do this, the application needs to use DSI API (i.e., *set_delegate_sid)*.

support of grant/deny based on SSID and NSID provided by the security managers is enough.

10.2 ACS architecture

The ACS that runs on the cluster's processors is comprised of two parts:

- A kernel-space part: This part is responsible for implementing both the enforcement and the decision-making tasks of access control. These two responsibilities are separated, as advocated by [3]. The kernel-space part maintains an internal representation of the information upon which it bases its decisions. This part is implemented as a Linux Security Module (LSM): DSI Security Module (DSM) [6].

- A user-space part: This part has many responsibilities. It takes the information from the Distributed Security Policy and from the Security Context Repository, combines them together, and feeds them to the DSM in an easily usable form. It also takes care of propagating back alarms from the kernel space part to the security manger, which will feed them to the Auditing and Logging Service and if necessary propagate to the security server through SCC.

Both parts are started and monitored by the local Security Manager (SM). The SM also introduces them to other services and subsystems of the infrastructure with which they need to interact.

10.3 ACS principles of operation

The ACS aims to provide fine-grained access control (at a sub-system call level). It respects the minimization principles of least privilege to limit the propagation and damage caused by eventual security breaches. As such, it provides defense in depth. The ACS that is running on a processor must make as little assumptions as possible about other processors, including whether they have been compromised. For that reason, an ACS instance is always the one making access decisions about resources that are local to its processor. For the initial design of the ACS, only grant/deny decision will be considered. Other more involved decisions would involve rate limiting and total usage limiting. Actions other than access control decision, such as interposition and active reactions, are not implemented either.

11 Authentication and communication integrity services

The authentication standard for now is the authentication by assertion. It means that the program accessing resources on remote processors asserts that it does this in behalf of a user. Neither the user schema nor the assertion only can be trusted seriously in an environment exposed to external attacks.

Local authentication in DSI is based on local verification by the DSM of each subject at node level.

The remote authentication of a process is the result of the local authentication of the process at the source node by DSM and the authentication of the node containing the subject to the target node.

IPSec is used for authenticating each node inside the cluster.

Developing for carrier class clusters, we have strong constraints on performance. IPSec has the advantage of covering both TCP and UDP[2]. To avoid applying the same policy to all IP traffic between two nodes (in particular, to avoid encrypting all data between two nodes), three IP addresses corresponding to three different subnetworks are assigned to each node. Each subnetwork defines a security policy: No security, authenticated only (IPSec AH mode), authenticated and encrypted (IPSec ESP mode). Filtering rules are further more used at networks elements (switches...) and at network interfaces of each node to enhance further the security rules.

The security manager at each node, based on

[2]The necessity of support of efficient security protocol for UDP is one of the main reasons why we have chosen IPSec.

SID of the sending process and the target node ID, and the target SID[3] according to the DSP transparently defines the security policy (e.g., subnetwork) to use: No Security, AH, or ESP.

Integrity and confidentiality of communications between two nodes is supported by use of IPSec ESP mode when necessary.

FreeS/WAN IPSec implementation has been used between nodes [2], however the support for IPSec AH mode is an issue. FreeS/WAN uses opportunistic encryption, which means that ESP mode is used even when AH mode is asked for. somehow in some cases for performance issues, it is though preferable to support AH mode without encryption load. At the end, we hope that the support for certificates will integrate the FreeS/WAN and will be not only supported as a patch.

12 Auditing Service

The auditing services are responsible for monitoring and auditing data and reporting security related information. This information may be used for several different purposes: intrusion and denial of service attacks detections, providing evidence in case of litigation or inquiries.

Auditing service for each node is responsible for analyzing the logs, detect the possible attack patterns, trigger the alarms, and propagate them through SCC. This service is responsible for functionality related to the lawful intercept.

This service has increase functionality on security server. It also monitors the internal network for the cluster and the distributed logs in order to detect attacks using Snort IDS [13]. This service on security server is related to external IDS through SCC.

The auditing service is connected to external log servers when needed. The connection between the auditing service and external loggers is not through SCC for performance reasons.

The requirements for this service are currently being defined.

13 Related work

This work distinguishes itself by being focused on the design of a security infrastructure targeted for clustered servers as compared to previous work that is focused on single computers or on clusters of general-purpose Linux machines. In addition, DSI takes into account all the issues related to security management starting at the design level. Some of the related work includes CorbaSec, the CORBA security service that handles the security issues regarding access control and authentication for interactions between different objects. CorbaSec does not take into account all aspects of security for example detection and reaction mechanisms like DSI and guarantees the security at middleware level independently from platform considerations.

On the other hand, Security Enhanced (SE) Linux from the National Security Agency (NSA) [5] or the Linux Security Module [6] (LSM) effort run on a single computer; they do not extend to a cluster.

Finally, Grid Security Infrastructure (GSI) was subsequently developed, based on existing standards, to address the security requirements that arise in Grid environments [1]. The DSI

[3]Target SID is defined by the port number on the target node.

approach is more fine-grained and is based on modifying the OS to enhance security mechanism (as explained in Section 10.1). The approach of DSI is possible because the software and hardware configuration in the cluster is under tight control. In practice, DSI supports a coherent vision of security throughout the whole cluster as GSI supports secure interoperable mechanisms between different trust domains for multiple users.

14 Results

We performed preliminary experiments on a cluster of Linux nodes. The fact that the source code of the Linux kernel is available and well documented is a major advantage for developing DSI on Linux based clusters.

So far, a secure boot mechanism for a diskless Linux system was implemented. Using secure boot with digital signatures, a distributed trusted computing base (DTCB) will be available as of the boot of the cluster nodes. The kernel at secure boot is small enough to be thoroughly tested for vulnerabilities. Furthermore, the use of digital signatures for binaries and a local certification authority will prevent malicious modifications to the DTCB.

We also implemented a security module based on Linux Security Module (LSM) that enforces the security policy as part of the DSI access control service [6]. This module is integrated with SCC and provides distributed access control mechanisms. DSI currently supports preemptive and dynamic security policy at the process level throughout the whole cluster for some operations. As future work, we will extend these capabilities to all operations on the cluster.

At this time, we are implementing the distributed security policy. In order to ease administration and maintenance of this policy, we completed a study to devise methods to reuse information already contained in package management systems (such as RPM for Linux) in order to generate part of the security policy, or to push such information to the package [10]. Specification of the exact language used to express the policy and of the compilation and loading mechanisms remains to be completed.

We implemented a secure communication channel based on OmniORB, an open-source implementation of Corba [11]. The implementation of SCC is independent from the communication middleware used. As mentioned in Section 7, SCC logics are implemented on top of a portability layer. This makes the implementation independent of any communication middleware used. The choice of CORBA as communication middleware for SCC was motivated by the following factors:

- Support from CORBA standard and implementations for distributed real-time and embedded systems,
- Support for advanced security mechanisms by CorbaSec,
- Interoperability.

15 Conclusions and future works

In this paper, we presented the need for a new security approach for carrier-class applications running on Linux clusters. Based on our motivations to develop a coherent solution addressing the security needs of carrier-class servers, we proposed a new design for a secure distributed infrastructure. We presented the main

elements of this design and discussed some of the preliminary results. We believe that this design is a practical approach to enhance security for large-scale Linux clusters with carrier-class needs.

To complete DSI, we plan to collaborate with Open Source projects and initiatives, and other organizations on the design and development of this secure infrastructure.

Acknowledgements

We thank David Gordon and Dominic Pellerin for their contributions to DSI. Also, we thank Marc Chatel and Bruno Hivert for their sanity checks on our design and implementation.

References

[1] *Foster I., Kesselman C., Tsudik G., Tuecke G., A Security Architecture for Computational Grids*, 5th ACM Conference on Computer and Communication Security.

[2] *Linux FreeS/WAN*, `http://www.freeswan.org`.

[3] *ISO 10181-3: Security Frameworks for Open Systems: Access Control Framework*, ISO, (1996).

[4] *ITU-U Recommendation X.800: Security Architecture for Open Systems Interconnection for CCITT Applications*, ITU-T (then CCITT), (1991).

[5] *Loscocco P.: Security-Enhanced Linux*, Linux 2.5 Kernel Summit, San Jose (Ca) USA, (2001), `http://www.nsa.gov/selinux/docs.html`.

[6] *Linux Security-Module (LSM) framework*, (2001), `http://lsm.immunix.org`.

[7] *MontaVista Linux Carrier Grade Edition 2.1*, `http://www.mvista.com/products/mvl_cge/mvlcge_overview.html`

[8] *Morris, J. Selopt: Labeled IPv4 networking for SE Linux*, `http://www.intercode.com.au/jmorris/selopt`

[9] *M. Zakrzewski. Mandatory Access Control for Linux Clustered Servers*, In Proceedings of Ottawa Linux Symposium, June 2002.

[10] *C. Levert, M. Dagenais, Security Policy Generation through Package Management*, In Proceedings of Ottawa Linux Symposium, June 2002.

[11] *omniORB*, `http://www.uk.research.att.com/omniORB/`

[12] *Open Source Development Lab, Carrier-Grade Linux Working Group*, `http://www.osdl.org/projects/cgl/`

[13] *SNORT*, `http://www.snort.org/`