A New Architecture for Secure Carrier-Class Clusters

Makan Pourzandi Ibrahim Haddad Charles Levert Miroslaw Zakrzewski
Michel Dagenais
Open Systems Laboratory, Ericsson Research Canada
8400 Decarie Blvd, Town of Mont-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.

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. 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.

This research targets soft real-time distributed applications running on large-scale carrier class 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 the Distributed 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.

2. The need for a new approach

There exist many security solutions for clustered servers ranging from external 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.

The 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 (even if the processes are created on remote 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. This would lead to no security checks for many actions through the distributed system. The granularity of the basic entity for the above security control is the user. For carrier-class applications, this granularity is not sufficient. Therefore, 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 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.

- 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.

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 [2].

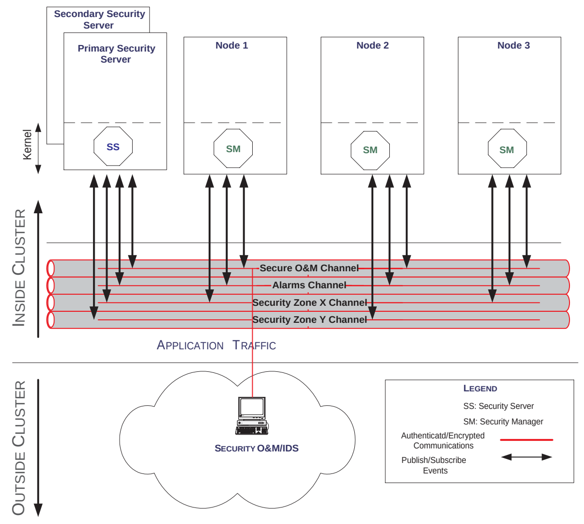

Figure 1. Distributed Architecture of DSI

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 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 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 managers. All communications between the security server and the outside of the cluster take place through the secure communication channel. Two nodes (to avoid a single point of failure) host the security server and different security service providers, such as the certification authority, are hardened to maximize security. All connections from and to these nodes are encrypted and authenticated.

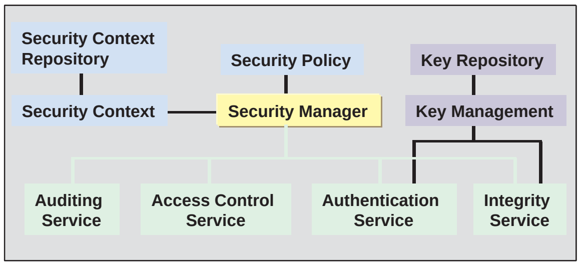

Figure 2. DSI Services

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, same as for the main security manager component, which has stubs into the kernel. In the Linux operating system, these stubs are implemented through load modules.

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.

- 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.

There are two types of services: security services (access control, authentication, integration, auditing) and security service providers that run at user level and provide services to security managers.

4.3. Secure Communication Channel (SCC)

The secure communication channel provides secure communications for the security components inside and outside the cluster (Figure 1). It provides with authenticated and encrypted communications among security components. 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 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 and 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.

5. 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.

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 [3]. 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.

6. A Coherent Vision: 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.

7. 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 DSM: a security module based on Linux Security Module (LSM) that enforces the security policy as part of the DSI access control service [1]. 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.

We implemented a secure communication channel based on OmniORB, an open-source implementation of Corba. 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.

At this time, we are implementing the distributed security policy. In order to ease administration and maintenance of this policy, we plan 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. Specification of the exact language used to express the policy and of the compilation and loading mechanisms remains to be completed.

In our early prototype, we use DSM in collaboration with IPSec for defining integrity and confidentiality for communications inside and outside of the cluster at process level. This is defined according to the CSID assigned to the process initiating the communication and DSP. We are currently integrating this work to DSI.

8. Conclusions and future works

In this paper, we presented the need for a new security approach for carrier-class applications running on telecom-clustered servers. 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 clusters with carrier-class needs.

References

[1] Linux security-module (lsm) framework. *http://lsm.immunix.org*.

[2] P. Loscocco. Security enhanced linux. *Linux 2.5 Kernel Summit, San Jose (Ca) USA*, 2001.

[3] J. Morris. Selopt: Labeled ipv4 networking for se linux. *http://www.intercode.com.au/jmorris/selopt*.