

An Information Flow Based Security Model for Linux Clusters

Liguo Yu

*Computer Science and Informatics
Indiana University South Bend
ligyu@iusb.edu*

Xubin He

*Electrical and Computer Engineering
Tennessee Technological University
hexb@tntech.edu*

Abstract

With the increasing use of clusters and distributed networks in industry, research, and scientific discoveries, security issues are becoming more and more important. Most existing security models are based on the discretionary access control (DAC) methods, where each user is granted (or denied) access to resources depending on the policies of the system. These models introduce several difficulties when applied to a distributed computing environment, such as clusters. In a distributed network, each node may support users with different running processes and each system may maintain its own security policy. This makes it difficult for the management of the access to distributed resources. Therefore, there is a need to develop more coherent security mechanisms for distributed networks. In this paper, to address the distributed resource management and access control issues in distributed environment, we extend the mandatory access control (MAC) mechanisms used at a single node level to distributed networks. We present a security model based on the information flow between distributed processes and resources. The information flow based security model focuses on the data integrity across distributed networks. This model is initially proposed for Linux clusters and it can be further extended to other distributed networks.

1. Introduction

High speed networking has significantly changed the nature of computing. Clusters and distributed networks are becoming more and more common in research, scientific discoveries, and even industry. On the other hand, the intricate nature of distributed systems has fundamentally changed the requirement of system security and brought new security issues [1, 2]. The conventional security approach is based on discretionary access control (DAC) of the interactions between users and resources within a single system.

This approach has worked well in a single system, where the operating system knows who is responsible for each process. However, DAC model has difficulties to be applied to distributed networks, such as clusters.

A distributed network has its own properties that are different from a single system: (1) a distributed network could have more than one high requirement secure zones, such as kernels. Accordingly, all these high requirement secure zones need to be protected appropriately; (2) the distributed network is usually much larger and it may involve more users with different processes than a single system; (3) the distributed network contains nodes that may have different security policies [3]. These distinct properties of distributed network make the conventional security approach no longer flexible and secure enough for distributed applications.

Data integrity is an important security requirement. It refers to the consistency, accuracy, and correctness of data. In this paper, we extend the mandatory access control (MAC) model for single system to distributed networks. We present a security model based on the information flow between processes and resources of distributed networks. This security model focuses on the data integrity across the distributed networks.

The paper is organized as follows: Section 2 reviews the current security models. Section 3 discusses the information flow between processes and resources. In Section 4, we describe our security model for Linux clusters. Our conclusions are in Section 5.

2. Brief review of security models

Security deals with the control of unauthorized use and the access to hardware and software resources of a computer system. In most security models, there are three components [4]: *object*, *subject*, and *access type*. *Object* is an entity of a computer system to which access is to be controlled. *Subject* is an entity that accesses the object. *Access type* is the type of control

that a subject can access an object. Examples of access types are read, write, execute, etc [5].

The conventional security mechanism is based on the discretionary access control (DAC) model. In DAC model, the *subject* is a user and the *object* is system resources, such as files, programs, etc. Therefore, DAC is generally used to limit a user's access to system resources. In this type of access control, it is the owner of the resource who controls other users' accesses to the resource. When these rights are managed correctly, only those users approved by the owner may have access to the resources. Therefore, DAC is also called user-based security model.

A drawback of DAC model is it does not support authentication and authorization checks for interactions between two processes belonging to the same user. This makes it rather difficult for one user to restrict the access to some distributed resources from some processes or users of the same group. It is very likely that a vulnerable small component may compromise the whole system. For example, Trojan horses whereby a devious unauthorized user can trick an authorized user into disclosing sensitive data. Therefore, there is a need for finer granularity security mechanisms.

Mandatory access control (MAC) is a technique intends to replace DAC, which uses the concepts of users and groups. The most important feature of MAC is that the user cannot fully control the access to resources that they create. The system security policy (as set by the administrator) entirely determines the access that is to be granted and a user is not permitted to grant less restrictive access to their resources than the administrator specifies. Such a framework prevents an authenticated user at a specific classification or trust level to access information, processes, or devices in a different level.

Generally, in MAC model, a user is modeled as a *subject*. This works fine for a single system. However, for a distributed network, users on different nodes may be granted different levels of access right. Therefore, user-based MAC model does not work well for distributed networks. Some research has been done to provide security models for distributed systems based on processes [6] [7] [8]. In these refined MAC models, a process is modeled as a *subject*. Therefore, they are also referred as *process-based MAC models*.

However, most of these researches focus on the *subject* and policy determination, but ignore the differentiation of *access types*. In our research, we classify *access types* based on the information flow between *subjects* and *objects*. Because different *access types* have different impacts on *objects*, the

corresponding *subjects* that initiate the access request should be given different access rights. This is further discussed in Section 3.

3. Information flow between subjects and objects

Our proposed distributed security model is an extended MAC model with a process as the *subject*. The resource that a subject wants to access is modeled as the *object*. We classify the resources that need to be protected as two types, service and data. The interactions between processes and resources of distributed network are modeled as remote method call, which is shown in Figure 1.

The access request from a process to a resource can be divided into two types, *definition* and *use*. An access to the resource without changing the content or the state of the resource is classified as *use*. This includes "read" access to a file, database, or invocation of a method without changing the state of the method. The *use* access does not change the content of the resource and accordingly does not affect other processes that may access the same resource. Therefore, the *use* access can keep the data integrity of resources.

An access to the resource that results in the modification of the content or the state of the resource is classified as *definition*. This includes "write/update" access to a file, database, or invocation of a method that results state change of the method. The *definition* access may affect the data integrity of resources.

Definition and *use* of resources are related with the information flow between a subject and an object. *Definition* means the information is transferred from *subject* to *object*. *Use* means the information is transferred from *object* to *subject*. That's why we call our model *information flow based security model*.

4. Information flow based security model

4.1 Distributed system structure

Our distributed security model is initially designed for Linux clusters. For convenience, we define two levels of security, the kernel level, which is represented as K and the non-kernel level, which is represented as NK. The kernel level has higher security requirement than non-kernel level. Therefore, a multiple nodes Linux cluster may have different kernel level and non-kernel level security requirement. We refer to this kind of distributed structure as a multi-kernel system as shown in Figure 2.

In this multi-kernel system, a process (*subject*) and a resource (*object*) can reside either in kernel or non-

kernel of the same node or different nodes across the network.

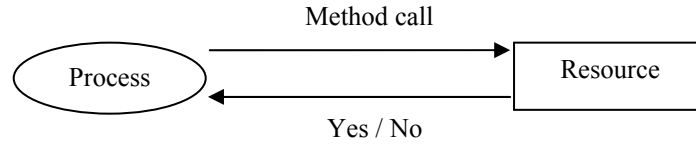


Figure 1: The access to resource from a process via method call.

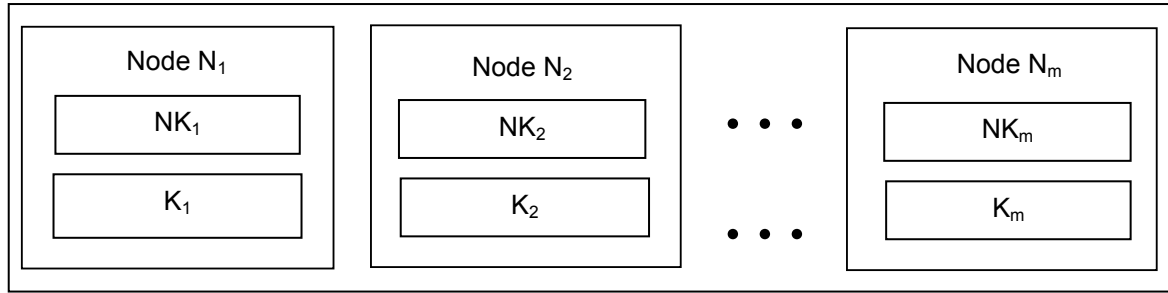


Figure 2: Depiction of a distributed multi-kernel system.

4.2 Component identifications

In our model, we modified the process level granularity proposed by Pourzandi [9] and Zakrezewski [10]. The process level granularity means the process is modeled as the *subject*. Because in a distributed network, there are different processes (*subjects*) and different resources (*objects*) on different nodes, we need to identify the different security contexts for individual processes and resources within the network. Each node within the network is assigned a security node identifier (N_ID). This identifier is thereby associated with each process (*subject*) and resource (*object*) in the node.

Each process and resource is also assigned a security level identifier (S_ID). In our distributed multi-kernel system (Figure 2), we define two levels of security. Therefore, S_ID could be either K (kernel level) or NK (non-kernel level). The S_ID (security level) of a process or resource is determined by the system administrator. Any newly created process is assigned a N_ID and S_ID which is based on the N_ID and S_ID of its parent process [9].

The remote access to resources from processes is achieved through a remote method call. Any remote method call is pre-assigned an access type ID (A_ID). The value of A_ID could be either *definition* or *use*. If

a remote method call can modify the data or the status of the object, its A_ID is *definition*, otherwise, its A_ID is *use*.

In our information flow based security model, the decision of grant or deny an access to a resource is determined by the combination of process N_ID, S_ID, the resource N_ID, S_ID, and the remote method call A_ID.

4.3 Data integrity

Data integrity is an important security requirement. It generally refers to the consistency, accuracy, and correctness of data stored in a database. It also refers to the condition that data is not accidentally or maliciously modified, altered, or destroyed. In our study, the data include not only the application information in database, or file, but also system information, such as system state variables.

Data integrity is more important in distributed applications, which enable processes to access resources across networks. Because different processes from different nodes may have different security levels, some processes should be allowed to access the resource, others should be denied. Therefore, we need a cluster wide policy to (1) ensure that the data are identically maintained during any

operation; (2) preserve data for their intended use; and (3) prevent accidental or deliberate but unauthorized modification or destruction of data.

4.4 Distributed security policy

Similar to the models proposed in [9] and [10], in our model, the security rules are also collected in the distributed security policy, which is enforced to all nodes. This guarantees a unique, homogeneous policy through the entire networks. The distributed security policy is automatically distributed to all nodes when the system is initialized. If any changes are made to the policy, the update will be broadcasted to all nodes of the cluster.

Table 1. The access policy within the same node

A_ID	K→K	K→NK	NK→K	NK→NK
<i>Use</i>	allowed	allowed	allowed	allowed
<i>Definition</i>	allowed	allowed	denied	allowed

Table 1 shows a suggested access policy within the same node. That is, both the process and the resource have the same N_ID. We use the symbol → to represent the access request from a process to a resource. For example, K→NK represents the request of a kernel level process to a non-kernel level resource, and NK→K represents the request of non-kernel level process to a kernel level resource. A_ID is the access type of the request.

Table 2. The access policy between different nodes

A_ID	K→K	K→NK	NK→K	NK→NK
<i>Use</i>	allowed	allowed	denied	allowed
<i>Definition</i>	denied	allowed	denied	denied

Table 2 shows the suggested access policy between different nodes across the cluster. The policies shown in Table 1 and Table 2 emphasize the integrity of kernel resources. As can be seen, it only allows kernel processes within the same node to modify the kernel level data. It does not allow processes from other different nodes to change kernel resource. Therefore, kernel level data integrity is strictly controlled.

These suggested policies also prevent non-kernel level process from using kernel level resources of different nodes. As we mentioned before, the policies are easy to be changed and propagated to every node within the cluster. Therefore, different kinds of policies can be enforced according to the requirement. For example, a distributed database may have different

security requirement, therefore, they may have different policies.

4.5 Discussions

Our information flow based security model is initially proposed for Linux clusters with two security levels. For other distributed systems that might have different levels of securities, the S_ID for processes and resources should have different values. Accordingly, different security policy should be defined.

One property of our information flow based security model is its coherency and homogeneity. Different nodes across the network apply the same policy. Therefore, our model can be generalized and applied to other peer-to-peer networks, such as a peer-to-peer file system.

There is one constraint about our model. The remote method call between process and resource needs to be designed specific for our model, because we need to identify the A_ID for every remote method call. We can not reuse remote method calls from other models directly without modifying them.

5. Conclusions

In this paper, we presented a distributed security model based on information flow between *subjects* and *objects*. This model focuses on the data integrity requirement of distributed applications. Our model was initially proposed for Linux clusters. It can be generalized and applied other distributed systems with multi-levels of security requirement.

Acknowledgements

This work was partially supported by Research Office under a Faculty Research Grant and Center for Manufacturing Research at Tennessee Technological University.

References

- [1] S. Greenwald, "A new security policy for distributed resource management and access control," *Proceedings of the 1996 workshop on new security paradigms*, pp. 74–86, 1996.
- [2] A. Abdul-Rahman and S. Hailes, "A distributed trust model," *Proceedings of the 1997 workshop on New security paradigms*, pp. 48–60, 1998.

- [3] W. A. Wulf, C. Wang, and D. Kienzle, "A new model of security for distributed systems," *Proceedings of the 1996 workshop on New security paradigms*, pp. 34–43, 1996.
- [4] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Communications of the ACM*, vol. 19, no. 8, pp. 461–471, 1976.
- [5] C. E. Landwehr, "Formal models of computer security," *ACM Computing Surveys*, vol. 13, no. 3, pp. 247–278, 1981.
- [6] I. K. Georgiev and I. I. Georgiev, "A security model for distributed computing," *Journal of Computing Sciences in Colleges*, vol. 17, no. 1, pp. 178–186, 2001.
- [7] G. S. Benson, I. F. Akyildiz, and W. F. Appelbe, "A formal protection model of security in centralized, parallel, and distributed systems," *ACM Transactions on Computer Systems*, vol. 8, no. 3, pp. 183–213, 1990.
- [8] Z. Li and Z. Qiu, "A New type of Security and Safety Architecture for Distributed System: Models and Implementation," *Proceedings of the 3rd international conference on Information security*, Shanghai, China, pp. 107–114, 2004.
- [9] M. Pourzandi, "A new distributed security model for Linux clusters," *Proceedings of USENIX 2004 Annual Technical Conference*, pp. 231–236, 2004.
- [10] M. Zakrezewski, "Mandatory access control for Linux clustered servers," *Proceedings of Ottawa Linux Symposium 2002*, pp. 618–631, 2002.