embeddedworld2017
Exhibition&Conference
... it´s a smarter world

An approach to Separation of Duties validation for MILS security configurations

Semen Kort, Dmitry Kulagin, Ekaterina Rudina
Future Technologies
Kaspersky Lab
Russian Federation

***Abstract*— Separation of duties (SoD) is an important concept aimed to constrain the excessive powers of subjects regarding system assets and control functions. Ensuring the fact that SoD is properly implemented for the particular task may require the individual approach in every given case.**

This paper proposes an approach to SoD validation conducted by the analysis of the security configuration of MILS-based solution. The security policy based on object capabilities is considered for this purpose. For this security policy two basic issues should be met. The first issue is the enough expressivity of the security policy. It is addressed with demonstration of particular examples of usage scenarios. The second issue regards the conditions under which the security problem remains tractable. Solving this issue in context of specifically defined SoD criteria is at the core of this research.

The approach is implemented for the security configurations of Kaspersky Security System.

Keywords—security system; security policy; security configuration; separation of duties; validation.

I. INTRODUCTION

Let's consider the following example. Some system needs to run the executable files downloaded from the external network. This system trusts several sources; before running the file it should verify the integrity and authenticity of the executable file image. Running the file before its verification by the specialized tool should not be possible. The latter excludes the possibility of launching the executable immediately by the verifier. Hence, here we have an example of the separation of duties.

1. If a process has downloaded a file, it cannot run this file for execution. Moreover, it cannot verify this file by itself but shall provide the dedicated verifier with a file image and attributes for the checking instead. Otherwise, the falsification of the security verdict is possible.

2. If some process is allowed to verify the file, it cannot run this file for execution. In another case, nothing restricts the possibility of running this executable before all necessary checks are completed.

3. If some process is allowed to launch executables, it should not be authorized for both downloading files and their verification. The former may cause the compromise of the process with an execution of malicious payload, and the latter may cause the violation of the established SoD policy.

Hence, in terms of role-based access control (RBAC) what we have here is three different roles implementing the whole procedure controlled according to the history-based dynamic separation of duties policy [1]. We may implement this scheme with a more simple operational static separation of duties between three roles, Downloader, Verifier and Executor, statically authorizing these roles for the operations that comprise the whole procedure. In this case, the correctness of the sequence may be either provided at the system level or ensured by the trusted agent.

This is only the one of examples where SoD policy is essential for the system security. Cyberphysical systems often require the separation of operations comprising the whole procedure, for security and safety reasons [2], [3]. The main difficulty for such cases was that the checks are specified for the definition of the operations and not enforced in their implementation.

To illustrate this fact let’s return to the example. Eventually, the SoD policy is enforced for the operations such as file creation, opening, sharing of file identifier among the processes separating the duties. The validation of the proper policy implementation for every procedure may not take into account the implementation details of these operations. At the same time, the implementation of specific file operations generally is not considered as trustworthy. Therefore, the policy enforcement should not be a part of the operations implementation;

conversely, it is provided by the external mechanism controlling the explicitly defined operations. It means that the validation results make the difference for the system that guarantees the proper separation of components responsibilities.

This is why we consider the problem of SoD validation in the Multiple Independent Levels of Security (MILS) context [4]. The system based on MILS architecture allows making a conclusion about the behavior of domains without going into details of the implementation of particular operations.

II. Related Work

Gligor, Gavrila, and Ferraiolo [1] formally described a wide variety of separation of duties policies and established their relationships within a formal RBAC model. What characterizes this research is the orientation on the implementation, not on the validation, of every described policy. From this point of view, the composability of SoD policies is considered as a quite important property.

The original paper by Ferraiolo, Cugini, and Kuhn on RBAC [5] presented operational separation of duty as a supplement to the static and dynamic separation of duty. Operational separation of duty required that no role could contain the permissions for all the operations necessary to perform a process. This type of SoD will be mainly considered in this paper.

Simon and Zurko [6] also enumerated various forms of SoD and indicated how the SoD policies could be expressed in Adage, a general purpose access control policy editor.

There are many works aimed at the validation or enforcement of SoD property in various context. The paper of Ahn, Sandhu, Kang, and Park [7] describes proof-of-concept implementation to demonstrate the practical feasibility of specifying and enforcing role-based authorization models [8] (that potentially support the dynamic SoD) for web-based workflow systems. The works of Basin, Burri, and Karjoth [9], [10] concentrate on the runtime enforcement of the SoD requirements on workflows, thereby preventing fraud and errors. Both works use specifications in SoDA, separation of duty algebra, defined by Li and Wang [11], and bridge the gap between the specification of SoD constraints modeled in SoDA and their enforcement in a dynamic, service-oriented enterprise environment.

For our research, we need the formal representation of SoD security policy, which would help us to validate the implementation of this policy according to the configuration of rights and the scheme of their transferring between subjects for the initial state of the system. As will be shown below, in our case the most convenient model for this purpose is Schematic Protection Model described by Sandhu [12] to find the conditions under which the safety question for the access control model remains tractable.

III. Kaspersky Security System

A. Main Features

Kaspersky Security System (KSS) [13] was initially implemented as a part of KasperskyOS with a view to supporting diverse security models. Later it has evolved into a stand-alone project and can now be embedded into other systems demanding enhanced security.

One of the security models supported by KSS is capability-based access control. A capability is an object that embeds the resource reference and the set of access rights for this resource. The capability is always possessed by some entity that presents this capability for getting the access to the appropriate resource. The internals of the capability is available for the resource driver and transparent for usage by the application-level entities.

The entity may pass the capability to another entity if security policy allows. Passing the capability may be implemented in two ways: transferring the capability of its deriving. Capability transfer is like copying the capability; the entity that transferred the capability may be not allowed to bring the situation back. In case the capability was derived it may be revoked by the parent entity. Moreover, the access rights provided within this capability may be adjusted by the parent entity beforehand.

This research mainly refers the capability-based model described above and its configurations.

One of the main features of KSS is flexible security configuration. Configuration tools are used to adjust the security policies and deploy them in the system.

We distinguish two types of system configuration: the configuration of the security policy that is defined in JSON format, and the binding configuration, or CFG file. Security policy configuration usually contains the parameters of the security models. While it may also be important for providing the particular security aspects, our main interest is in addressing the issues of validation of binding configuration (CFG file). The application specific concerns may parametrize the system with CFG means in a way that will enforce the required security properties. This parametrization is a subject of our research.

B. Security Configurations

CFG is a declarative language that allows declaring the control rules for the following actions in the system: launching the entity (*execute* action); interaction of two entities (*call* action); asking about the security decision for the internal event (*security ask* action).

Control is determined by binding the actions with security policies using the references onto these policies provided by the KSS. Every reference contains the name of the policy implementation and may be provided with the optional policy configuration.

Every entity is accompanied with its definition and definitions of the interfaces. While the format of these definitions is outside the scope of this paper, it should be noted that the set of methods and parameters may be listed within the appropriate files and referred in the configuration.

C. Schematic Protection Model

Schematic Protection Model (SPM) is the meta-model used for determining whether the security problem is tractable for another model defined with SPM terms [9]. Tractability of the security problem means the existence of the algorithm capable of identifying the access rights leakage. Algorithms created for

the Schematic Protection Model may also be used for the validation of other significant properties of the security policy as will be shown below.

The key notion of SPM is the protection type or just type. The type is a label for the entity that determines its involvement in the distribution of the access rights among entities. The combination of the right and the objects of this right is referred as the ticket. If the entity possesses the ticket Z/r, it means this entity has the right r over object Z.

For describing the conditions of transferring the rights among entities, SPM uses two functional primitives: the link predicate and filter function. The link predicate determines whether there exists the direct link by which the rights may be potentially between two types of entities. The filter function specifies the types of tickets that may be passed via this link. The types of tickets are determined according to the types of entity for which the right is possessed.

The order in which the entities of one type may create entities of another is essential for the tractability of security problem. In SPM terms, the graph of creation must be acyclic to avoid the unsolvable security problem.

D. Modeling of KSS Security Configurations with SPM

1) Typification

Typification is one of the main concepts determining the expressiveness and tractability of the security problem in SPM. Also, this is one of the base concepts underlying KSS.

The set of subject types *TS* in SPM matches the set entities to which CFG file refers. The set of objects types *TO* is represented by the types of capabilities. Their unification comprises the whole set of types *T*.

In our system, entities may have rights over the objects obtained via the mechanism of capabilities transfer or capabilities derivation. Regarding SPM, it means that the subjects of appropriate type may demand the tickets that describe the rights over the objects. The capability represents the object, e.g. *Z* and every right *r* that is kept and transferred by this capability may be referred separately as *Z/r:c*.

The typification of capabilities allows us to keep the typification of rights. Semantically it means that the right, for example, of reading the file is not the same as the right of reading the network socket. The policies that applied for the regulation of transfer of these rights may be different because the types of the rights are different (even if the APIs for accessing the appropriate objects look similar).

Given that the CFG file defines the policies for the interface call separately for every kind of entities, and policies are applied for both communication parties, we have got the complete match with the SPM. The right is typed accordingly to the object type and controlled individually to the type of the subject using this right.

2) Link predicate and filter function

In SPM, the possibility of the right transfer is determined by two functional primitives: link predicate and filter function. Link predicate refers the existence of a kind of connection between two types of entities via which the capability may be passed. The filter function is applied to this capability according to the types of entities to adjust the set of typed rights according to the fine-grained policy for every pair of subject types.

The link predicate is constructed by the analysis of the CFG file. For every policy defined within the CFG file, the validator builds the relationships between entities according to the definition of policies for the *call* actions. The validator goes through the definition of entity analyzing its *call* policies.

Then the filter function is created. For every transferred capability the set of rights is constrained according to the policy definition and mandatory restrictions set for the entity.

3) Can-create Predicate and Create Rules

The next important concept is the entities creation. The SPM requires the creation of entities to fit with the so-called acyclic attenuating scheme. The acyclic nature of the scheme is formally defined by the *can-create* predicate determining entities of which types can create entities of other types. The graph of creation must be acyclic.

This property may be guaranteed via the *execute* action policy binding that allows configuring the execution of entities according to the type of parent entity and parameters passed by this entity during execution. For example, every parent that is allowed to run the entity of particular type may be configured using the individual policy binding, with the obligatory verification of matching the type with the policy specifically configured for this type.

More straightforward approach presumes the general awareness of the validator about the entities creation scheme.

The attenuation of privilege that is also the essential requirement that means the set of rights passed to the newly created entity must not exceed the set of rights of the parent entity.

Taking into account the nature of capability-based access control, we may certainly state that creating the entities will meet the requirement of the attenuation of privilege. If the entity does not have the right or capability, it cannot pass this right or capability to the newly created one.

IV. SoD Validation for Security Configurations

A. The Formal Definition of SoD in SPM

The operational static separation of duties criteria is originally formulated for the RBAC model as follows [1]: for any state, there is no such subset of roles that would be authorized for the whole set of operations comprising the transaction (or another type of sensitive sequence of action in the system).

We assume that

- the states in SPM correspond the states in RBAC

- the set of subject types in SPM corresponds the set of roles in RBAC

- the types of tickets in SPM correspond the operations RBAC

- the typification of subjects in SPM (in relational form) corresponds the role members relation in RBAC

- the membership of the ticket of particular type in the domain of entity in SPM corresponds the authorization of subject for the right in respect to object in RBAC

Thus, the operational static SoD criteria are formulated as follows. For the set of tickets with different types and related to the same object, the operational static separation of duties is enforced if there is no subset of subject types TS for which the appropriate set of subjects could have all tickets from this set.

In case the operational static SoD is applied to the set of two types of tickets the definition is simpler. Two tickets of different types for the object shall not be possessed by the entities of the same type.

B. The Validation Process

The validation of the operational static separation of duties takes as input the names of types of capabilities and rights, provided by these capabilities, for which SoD is applied.

Several statements proven for the SPM facilitate the checking whether the current system configuration interpreted with SPM terms satisfies the operational static SoD criteria. The detailed description and proof of these statements go beyond the scope of this paper.

The validation process requires only the polynomial computations, particularly:

1. Building the SPM scheme according to the CFG file describing the security configuration of KSS.
2. Checking whether the scheme is acyclic.
3. Computing the special fully unfolded state for the SPM [10]. This is the state for which entities of all possible types are created, and for any entity that may be additionally created another entity of the same type, created in the same way already exists. It is provable that such a state is reached for the acyclic attenuating scheme by a polynomial algorithm.
4. Checking whether the fully unfolded state satisfies the following condition: for any type of entities separating the duties there is the type of the ticket that doesn't present in the domains of the subjects of other types.

C. Enhancing the Scheme with the Linear Rights

The most used case within the concept of operational SoD is providing the monopoly access to the critical resource for any operation at any moment of time. The monopoly may also restrict the sharing of the right to the subjects of the same type. In this case the monopoly access is a particular case of the operational SoD described above.

Unfortunately, this case is the most difficult to implement if the scheme is monotonic and attenuation of privilege principle is upheld. The entity can't transfer the right for the monopoly access to the resource (if it had transferred the right, the right would not have been a monopoly right), and the right can't be removed. To address this issue, we suggest using so-called linear rights.

The idea of linear rights allows addressing the requirements to the monopoly access to resources. For example, any subject may read the audit log but simultaneous writing this log is not allowed.

In case the right is defined as linear only one entity may possess this right at every moment of time. If the capability with the linear right is revoked, then this linear right will be given back to the parent.

Let's consider the example given in the introduction. The set of subject types (entities) contains

- File System agent allowing to read and write files
- Downloader downloading the file from the Internet
- Verifier checking the digital signature of the file
- Application allowed working only with correctly signed file.

Linear rights allow us to implement the scheme meeting all requirements by the following way:

- The Downloader entity demands the capability with writelinear and read rights from the File System entity
- The File System entity creates the file and transfers the requested capability to the Downloader entity
- The Downloader entity writes the file using the capability
- The Downloader transfers the capability to the Verifier entity. The right for writing is linear, hence, the Downloader entity is unable to change the file since this moment.
- The Verifier entity safely checks the digital signature
- In case the signature is correct, the Verifier entity transfers (by the derivation mechanism) the capability to the application without writelinear right
- The Application entity can safely read the file and transfer the capability to other entities.

As shown above, we may also need a simple read and write, except the appropriate linear rights. However, we shall avoid the situation where the same entity possess both the simple and linear right of the same kind (for example, both write and writelinear). These rights will be mutually exclusive for any entity (static SoD). The appropriate checks may be implemented at runtime.

The single linear right may be added to the scheme without the violation of already verified SoD properties. This fact allows combining the SoD aspect proven for the types of entities and fine-grained monopoly access to the resources where needed.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed two approaches to the implementation and validation of SoD policy for the configurations of the MILS-based system. These approaches cover substantially different cases: the operational SoD for the acyclic attenuating schemes of rights transfer, and operational SoD guaranteeing the monopoly access to the resource at any

moment of time. These approaches may be successfully combined if required.

We continue to investigate various MILS applications to find the relevant use cases and discover the new scenarios that may require separation of duties. Particularly, other types of SoD policies except the operational SoD are of our interest.

REFERENCES

[1] V. D. Gligor, S. I. Gavrila, and D. Ferraiolo, "On the formal definition of separation of duty policies and their composition," Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, May 36, 1998.

[2] A. Cardenas, S.Amin, B.Sinopoli, A.Giani, A.Perrig, and S.Sastry, "Challenges for securing cyber physical systems," Workshop on Future Directions in Cyber-physical Systems Security, DHS, Newark, NJ, July 23, 2009.

[3] C.Codella, A.Hampapur, C.Li, D.Pendarakis, and J.R.Rao, "Continuous Assurance for Cyber Physical System Security," Workshop on Future Directions in Cyber-Physical Systems Security, Newark, NJ, July 2009.

[4] H.Blasum, S.Tverdyshev, B.Langenstein, J.Maebe, B.De Sutter, B.Leconte, B.Triquet, K.Müller, M.Paulitsch, A.SödingFreiherr von Blomberg, A.Tillequin, "MILS Architecture," EURO-MILS: Secure European Virtualisation for Trustworthy Applications in Critical Domains, Whitepaper, 2014.

[5] D. Ferraiolo, J. Cugini, and D. R. Kuhn, "Role-based access control (RBAC): features and motivations," Proceedings of the 1995 Computer Security Applications Conference, December 1995.

[6] R. Simon and M. E. Zurko, "Separation of duty in role-based environments," Proceedings of the 10th Computer Security Foundation Workshop, Rockport, MA, June 1012, 1997.

[7] G.-J. Ahn, R. S. Sandhu, M. Kang, and J. Park, "Injecting RBAC to Secure a Web-Based Workflow System," Proceedings of the 5th ACM Workshop on Role-Based Access Control, Berlin, July 2628, 2000.

[8] R. Sandhu, "Separation of Duties in Computerized Information Systems," Proceedings of IFIP WG11.3 Workshop on Database Security, September 1990.

[9] D. Basin, Samuel J. Burri, and G.Karjoth, "Separation of Duties as a Service," Proc. of the 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS 11). Hong Kong, China, March 22-24, 2011.

[10] D. Basin, Samuel J. Burri, and G.Karjoth, "Dynamic enforcement of abstract separation of duty constraints," ACM Transactions on Information and System Security (TISSEC), Volume 15, Issue 3, November 2012.

[11] N. Li and Q. Wang, "Beyond separation of duty: an algebra for specifying high-level security policies," Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006.

[12] R. S. Sandhu, "The schematic protection model: its definition and analysis for acyclic attenuating schemes," Journal of the ACM (JACM), Volume 35 Issue 2, April 1988, Pages 404-432.

[13] S.Tverdyshev, H.Blasum, E.Rudina, D.Kulagin, P.Dyakin, S.Moiseev, "Security Architecture and Specification Framework for Safe and Secure Industrial Automation," Critical Information Infrastructures Security, 10th International Conference, CRITIS 2015, Berlin, Germany, October 5-7, 2015, Revised Selected Papers, Pages 3-14.