Paper

Model of User Access Control to Virtual Machines Based on *RT*-Family Trust Management Language with Temporal Validity Constraints – Practical Application

Krzysztof Lasota and Adam Kozakiewicz

Research and Academic Computer Network (NASK), Warsaw, Poland

Abstract—The paper presents an application of an *RT*-family trust management language as a basis for an access control model. The discussion concerns a secure workstation running multiple virtual machines used to process sensitive information from multiple security domains, providing strict separation of the domains. The users may act in several different roles, with different access rights. The inference mechanisms of the language are used to translate credentials allowing users to access different functional domains, and assigning virtual machines to these domains into clear rules, regulating the rights of a particular user to a particular machine, taking into account different periods of validity of different credentials. The paper also describes a prototype implementation of the model.

Keywords*—RT-*family languages, security model, user access control, virtual environment.

1. Introduction

The article presents issues related to granting secure user access to resources having different sensitivity levels. This subject is one of the most important aspects of the project called "Special workstation for special applications". The project is focused on building a secure system to work with documents from different security domains located in virtual environment. Separation of system resources (e.g., Processors, RAM Memory, etc.) between virtual machines is not part of presented security model and it will not be described in this article. However, it was included in the architecture of the prototype solution.

The rest of this section presents a short description of the "Secure workstation for special application" project and the RT-family trust management languages, one of which is used to describe the proposed model. Section 2 focuses on presentation of the most important functional requirements. The proposed security model is presented in Sections 3, 4, and 5. Section 6 describes the implemented prototype. The article concludes with a short summary in Section 7.

1.1. Secure Workstation for Special Application

The work presented in this paper is a part of the project called "Secure workstation for special applications" [1], which aim is to create a secure environment for processing of sensitive information based on virtualization technology. The documents belonging to different security domains (different sensitivity levels or functional domains) are processed in separate, isolated, virtual machines running special secure versions of guest systems. The expected result of the project, to be delivered later this year, is an advanced technology demonstrator. The products of the project will include:

- secure system platform, referenced as SSP – software component integrating a secure host operating system and virtual machine management tools, which is able to run several instances of guest operating system;
- special versions of guest operating systems – secure version of Linux and Windows systems prepared to run under control of the secure system platform;
- technical and operational documentation of the system, recommendations, procedures and templates;
- examples of cryptographic data protection and authentication mechanisms, e.g. biometrics.

The project consortium is led by the Military University of Technology in Warsaw and consists of Filbico Sp. zo.o., Military Telecommunications Institute and Research and Academic Computer Network (NASK).

1.2. RT-Family Trust Management Language

Role-based trust management (RT) languages were introduced in [2] and combine features of trust management [3] and Role Based Access Control [4]. They are used for representing security policies and credentials in centralized and distributed access control systems. A credential provides information about the user access privileges and the security policies issued by one or more trusted authorities. So far the family consists of: RT_0, RT_1, RT_2, RT^T, RT^D languages [2], [5]–[7] which are progressively increasing in expressive power and complexity. For language RT^T which is backwards compatible with RT_0, RT_1 and RT_2, in [8] time validity of credentials is proposed. The extended versions of these languages are referenced as RT^T_+,

RT_{2+}, RT_{1+}, and RT_{0+}. In [9] the general structure of proofs of soundness and completeness of inference systems for these languages are presented. The complete proof is still waiting for publication.

While legal regulations enforce only using mandatory access control (MAC [10], [11]), the proposed model of user access control to virtual machines will be based on a trust management language from the RT family with time validity, enabling more detailed specification of access rules. Language RT_{2+} fits our expectations by supporting parameterized roles and defining o-sets which are able to group objects representing resources in much the same way as roles group subjects. This enables completely abstract policies to be created where permissions are specified in terms of roles, and o-sets and actual relations between subjects and objects are established by assigning them to roles and o-sets.

2. Security Model – Requirements

The basic requirement for the defined model of access control to virtual machines is its validity. Additionally it has to comply with functional requirements of the project. The most important ones are described in this section.

2.1. Security Clauses

The security model must support various levels of security which are assigned to system users and documents located on virtual machines. Figure 1 shows a diagram of relations between security levels in Poland, European Union and NATO.

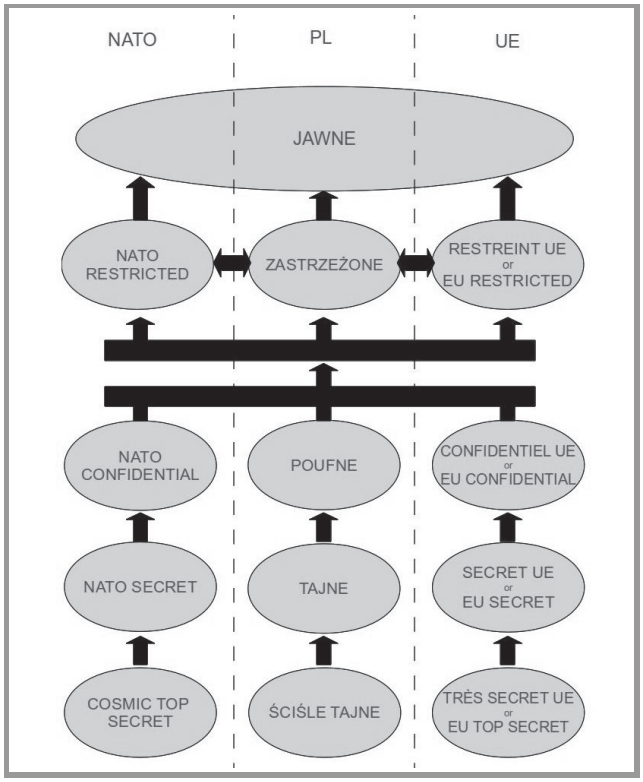

Fig. 1. Relations between security levels in Poland, European Union and NATO.

2.2. Protected Resources

Protected resources, e.g., documents, are located inside virtual systems. The security model for user access to virtual machines presumes that all protected resources are located on virtual machines. A permission is granted to access a particular virtual machine which holds documents with the same sensitivity level. A user in the system can have access to many virtual machines, as well as one virtual machine can be accessed by many users.

2.3. Functional Domains

The security model has to support various functional domains. Domain partitioning allows for more elastic management of system resources and user permissions. Examples of security domains might include: finance – this domain holds information related to payrolls, incomes of employees, etc., or projects – the domain holds information about running projects.

Additionally, the security model should support separation of resources located in the same functional domain but with different sensitivity levels. Security domain can hold resources with different sensitivity levels assigned to them, but a user can only access those with sensitivity level not exceeding his clearance.

2.4. Users of the SSP System

The security model has to differentiate users based on the potential system utilization by them. A basic system user can only use resources which he is allowed to access. From the system security point of view, the people responsible for granting other users rights to use the system are the most important group. Additionally, due to the accountability requirement for the secure workstation, it is required to include users responsible for auditing the system.

2.5. Time Validity

The security model should include information about validity period of the issued credentials. Implementing time frames of validity for granted permissions is necessary in the system. Security clearance and certificates of received trainings considering data protection are valid only for specified periods of time. The procedures for assigning access rights to protected resources for a user need to enforce a definition of their validity period, so that the model will be similar to a real system.

3. Security Model – Subjects and Objects

Security model is based on two types of entities – a secure workstation and system users – and the same number of objects corresponding to the resources of the system – virtual machines and shared storage areas.

Secure workstation is a central point of authority. Only roles and sets defined by the Secure Workstation and credentials issued by it are supported by the deductive system. The most important attribute of an entity is a *unique identifier* that is referenced as B later in the article.

System User is a key entity. Based on issued credentials, protected system resources are made available to it. User entity possesses a number of attributes that are derived from requirements for security model:

- *Unique identifier* – referenced as u_{id} – used to differentiate between users;
- *Security levels* – referenced as u_{lev} – contains a list of all sensitivity levels which the user has access to.

Additionally each clause has a defined time period of its validity. The validity depends on the clearance held by the user, e.g., in Polish law validity is determined by owned security credentials and completed trainings on protection of classified information. The time period of certificate validity is referenced as v_l, where l corresponds to the identifier of security level.

Virtual machine and **shared storage area**. All resources of the secure workstation to which access is determined by the defined model are contained in the described objects. The main purpose of virtual machines is to allow for working with documents of different security level. Shared storage areas provide access to resources that are located directly on the secure workstation. They enable different access to be assigned to directories or hard drives for different users. In particular, a separate hard drive destined for collecting logs is intended and available only for the safety auditor. The most important attributes of an object are:

- *Unique identifier* – referenced as mv_{id} and ss_{id} – used to differentiate between objects.
- *Validation period* – referenced as v_{mv} and v_{ss} – period of time when particular object is available for systems users.
- *Security level* – referenced as mv_{lev} and ss_{lev} – attribute specifies security level of all resources contained in a particular object.

4. Security Model – Roles and O-Sets

The RT_1 is an extention of RT_0 and introduces parameterized roles, while the RT_2 language extends RT_1 with o-sets. In the following section all roles and o-sets used by the access control model are described.

4.1. Parameters

All defined roles for grouping subjects, and also all object sets can depend only on the following parameters:

User role parameter – referenced as *rol* – has a slightly different meaning depending on the context where it is used, either referencing security model roles or o-sets. In case of security model roles, it defines a role for the system user. Second case is when it describes a user's role necessary to access protected resources (either a virtual machine or shared storage area). The parameter can be assigned the following values:

- *USER* – references users with basic rights in the system which can get access only to resources assigned to them;
- *SPEC* – references Chief Information Security Officer (security specialist) with function of granting users access rights to resources, but does not have an access to these resources himself. This is the only role which can add new credentials interpreted by the system. In fact, a user with this role can be viewed as the host system's administrator;
- *AUDIT* – references Chief Audit Executive (security auditor) whose role is assuring system accountability by validating system event log files;
- *ADMIN* – additional role for virtual machine administrators responsible for the configuration of virtual machines.

Functional domain parameter – referenced as *dom* – corresponds to system partitioning into separate domains allowing an easy management of users and resources. A user gains access to resources by acquiring proper credentials to access a domain. A particular resource is accessible only if it is assigned to a domain. The parameter does not have a defined set of allowed values. The only restriction is a unique identifier of a newly added domain.

Type of access parameter – referenced as *rig* – the security model provides means to determine the access type granted for a user to a resource. Two basic types of access rights are distinguished: a right to read referenced as *R* and a right to read and write referenced as *RW*. The first type is applicable in case of shared storage areas containing log files of system events. A system administrator should have an access to it, but should not have a right to modify it. An auditor has the ability to create a backup of secured data and delete the old one by using an additional software. In case of resources on virtual machines, only the second type of access rights is used.

Security level parameter – referenced as *lev* – The security model requires specifying a sensitivity level for each

resource. The resource may not be accessed by users without a security clearance for this – or higher – level of security. The following security level identifiers are used to reference proper security clearances used in Poland, EU and NATO:

- *J* – public level, common for security clearance in Poland, EU and NATO;
- *Z* – restricted level, references: Polish *'ZASTRZEZONE'*, and *'EU RESTREINT'* or *'EU RESTRICTED'* and *'NATO RESTRICTED'*;
- *P-PL* – confidential level, references Polish *'POUFNE'* clearance;
- *P-EU* – confidential level, references *'EU CONFIDENTIAL'* or *'CONFIDENTIEL UE'* security clearance;
- *P-NA* – confidential level, references *'NATO CONFIDENTIAL'* security clearance;
- *T-PL* – secret level, references Polish *'TAJNE'* clearance;
- *T-EU* – secret level, references *'SECRET UE'* or *'EU SECRET'* clearance;
- *T-NA* – secret level, references *'NATO SECRET'* clearance;
- *S-PL* – top secret level, references Polish *'SCISLE TAJNE'* clearance;
- *S-EU* – top secret level, references European *'EU TOP SECRET'* or *'TRES SECRET UE'* clearance;
- *S-NA* – top secret level, references *'NATO COSMIC TOP SECRET'* clearance.

4.2. User Roles

The security model allows for grouping of the entities related to users in three roles. Role *B.ide* of the security model contains information about all system roles that can be assigned to users, and is defined as follows:

$$B.ide(?rol) \text{ in } V, \tag{1}$$

where:
rol – may take all defined values.

It is very important in the context of real system operations. Taking into account that a user can be assigned to different roles, identifying a particular one is done by user 'Identity' (denoted as ide in role names). The Identity is defined as a combination of user and assigned role, and typically corresponds to an account in the system – a user may have more than one account and change roles by switching between them. From the security point of view, assigning to the same user *AUDIT* role and any other is forbidden. Next two roles, defined as:

$$B.ide\_dom\_rig(?rol, ?dom, ?rig) \text{ in } V \tag{2}$$

and

$$B.ide\_dom\_lev(?rol, ?dom, ?lev) \text{ in } V, \tag{3}$$

where:
rol – may take values from set {*ADMIN,USER,AUDIT*},
dom – may take all defined domains,
rig – may take all defined values {*R,RW*},
lev – may take all defined values {*J, Z, P-PL, P-EU,* {*P-NA, T-PL, T-EU, T-NA, S-PL, S-EU S-NA*},

are holding information about an access of user identities to functional domains. Role (2) determines the type of an access to domain resources and role (3) defines restriction on the highest sensitivity level of resources which a given identity is allowed to access. User identity can only access those resources with sensitivity level not exceeding clearance.

4.3. Virtual Machine O-Sets

The security model allows to group objects related to virtual machines in two ways. The first one (4) defines membership of a virtual machine to resources of a particular functional domain. Sensitivity level of data located on a virtual machine is an attribute of an object representing the virtual machine. The assignment is performed with a specific security level which may not differ from the sensitivity level assigned to the virtual machine itself.
The second one (5) describes a system role which a user has to be assigned to gain access to a particular virtual machine.

$$B.mv\_dom(?dom, ?lev) \text{ in } V \tag{4}$$

where:
dom – may take all defined domains,
lev – may take all defined values.

$$B.mv\_rol(?rol) \text{ in } V \tag{5}$$

where:
rol – may take values from set {*ADMIN,USER,AUDIT*}.

4.4. Shared Storage Area O-Sets

Similarly as for virtual machines, the security model allows two groupings of objects related to shared storage areas. The first one (6) defines functional domain with a particular sensitivity level that has the shared storage area in its resources. The second one (7), aside from defining a system role a particular user has to be granted with, defines the type of access. There is no possibility to assign shared storage area as a resource designed for users in *USER* system role.

$$B.ss\_dom(?dom, ?lev) \text{ in } V \tag{6}$$

where:
dom – may take all defined domains,
lev – may take all defined values.

$$B.ss\_rol(?rol) \text{ in } V \tag{7}$$

where:
rol – role – may take values from set {*ADMIN,AUDIT*},
rig – rig – may take all defined values.

4.5. Main Role

The most important role (8) of the security model can provide information about specific time when user is allowed to access a particular resource, based on user credentials and deductive system.

$$B.main(?rol, ?dom, ?rig, ?lev) \text{ in } V \tag{8}$$

where:
rol – may take values from set {*ADMIN,USER,AUDIT*},
dom – may take all defined domains,
right – may take all defined values,
lev – may take all defined values.

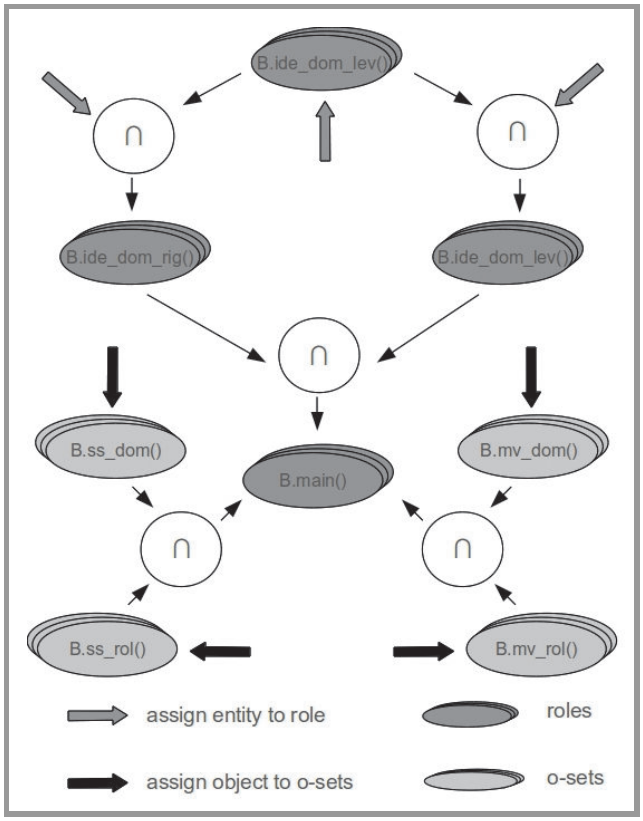

Fig. 2. Visualization of user access control model.

5. Security Model – Credentials

A security model based on the RT_{2+} language featuring time validity of credentials is presented in Fig. 2. It is composed of roles and o-sets, as defined in the previous section. Furthermore, it presents all possible credentials the system is able to process. Two types of credentials were distinguished: credentials created by an Information Security Officer, and credentials obtained from the deductive system process.
The first type of credentials includes:

- a user role attribution, creating a user identity (9);
- an identity access type to resources of a particular functional domain (10);
- an identity access to resources of a functional domain with particular sensitivity level (11);
- a virtual machine inclusion as a resource of a particular domain with specific sensitivity level (13);
- a virtual machine attribution to users holding specific system role (14);
- a shared storage area inclusion as a resource of a particular domain with specific sensitivity level (16);
- a shared storage area attribution to users holding specific system role and type of access (17).

The second type of credentials include:

- a type of access allowed to resources of a domain with particular sensitivity level for a user holding specific role (12);
- a virtual machine inclusion as a resource of a domain with a particular sensitivity level, with access type available for a user holding a certain role (15);
- a shared storage area inclusion as a resource of a domain with a particular sensitivity level, with specific access type available for a user holding a certain role (18).

5.1. User Credentials

$$\begin{gathered} B.ide(rol = X) \text{ in } V \\ \longleftarrow \\ B.USER(u\_id = I) \text{ in } v \end{gathered} \tag{9}$$

where:
X – role value which is assigned,
I – value of user unique identifier,
v – base time validity of credential,
V – active time validity of credential.

$$\begin{gathered} B.ide\_dom\_rig(rol = X, dom = Y, rig = Z) \text{ in } V \\ \longleftarrow \\ B.ide(rol = X) \\ \cap \\ B.USER(u\_id = I) \text{ in } v_{ide} \cap v \end{gathered} \tag{10}$$

where:
X – role value which is assigned,
Y – domain value which is assigned,
Z – access type value which is assigned,
I – value of user unique identifier,
v_{ide} – active time validity of component credential (9),

v – base time validity of credential,
V – active time validity of credential.

$$\begin{gathered} B.ide\_dom\_lev(rol = X, dom = Y, lev = W) \ \text{ in } V \\ \longleftarrow \\ B.ide(rol = X) \\ \cap \\ B.USER(u\_id = I, u\_lev = L : W \in L) \\ \text{in } v_{ide} \cap v_{WL} \cap v \end{gathered} \tag{11}$$

where:
X – role value which is assigned,
Y – domain value which is assigned,
W – sensitivity level which is assigned,
L – set of all security levels accessible to a particular user,
v_{ide} – active time validity of component credential (9),
v_{WL} – time validity of user sensitivity level,
v – base time validity of credential,
V – active time validity of credential.

$$\begin{gathered} B.main(role = X, dom = ?Y, rig = ?Z, lev = W) \\ \text{in } V \\ \longleftarrow \\ B.ide\_dom\_rig(role = X, dom = ?Y, rig = ?Z) \\ \cap \\ B.ide\_dom\_lev(role = X, dom = ?Y, lev = W) \\ \text{in } v_{rig} \cap v_{lev} \end{gathered} \tag{12}$$

where:
X – role value which is assigned,
Y – set of domain values which are assigned,
Z – set of access type values which are assigned,
W – security level value which is assigned,
v_{rig} – active time validity of component credential (10),
v_{lev} – active time validity of component credential (11),
V – active time validity of credential,

5.2. Virtual Machine Credentials

$$\begin{gathered} B.mv\_dom(dom = Y, lev = W) \text{in } V \\ \longleftarrow \\ B.MV(mv\_id = M, mv\_lev = W) \ \text{ in } v_{mv} \cap v \end{gathered} \tag{13}$$

where:
Y – domain value which is assigned,
W – security level value which is assigned,
M – value of virtual machine unique identifier,
v_{mv}– time validity of virtual machine,
v – base time validity of credential,
V – active time validity of credential.

$$\begin{gathered} B.mv\_rol(role = X) \\ \text{in } V \\ \longleftarrow \\ B.MV(mv\_id = M) \ \text{ in } v_{mv} \cap v \end{gathered} \tag{14}$$

where:
X – role value which is assigned,
M – value of virtual machine unique identifier,
v_{mv}– time validity of virtual machine,
v – base time validity of credential,
V – active time validity of credential.

$$\begin{gathered} B.main(role = ?X, dom = ?Y, rig = ' RW', lev = W) \\ \text{in } V \\ \longleftarrow \\ B.mv\_dom(dom = ?Y, lev = W) \\ \cap \\ B.mv\_rol(role = ?X) \\ \text{in } v_{dom} \cap v_{rol} \end{gathered} \tag{15}$$

where:
X – set of role values which are assigned,
Y – set of domain values which are assigned,
W – security level value which is assigned,
v_{dom} – active time validity of component credential (13),
v_{rol} – active time validity of component credential (14),
V – active time validity of credential.

5.3. Shared Storage Area Credentials

$$\begin{gathered} B.ss\_dom(dom = Y, lev = W) \text{ in } V \\ \longleftarrow \\ B.SS(ss\_id = S, ss\_lev = W) \ \text{ in } v_{ss} \cap v \end{gathered} \tag{16}$$

where:
Y – domain value which is assigned,
W – security level value which is assigned,
S – value of shared storage area unique identifier,
v_{ss} – time validity of shared storage area,
v – base time validity of credential,
V – active time validity of credential.

$$\begin{gathered} B.ss\_rol(role = X, rig = Z) \\ \text{in } V \\ \longleftarrow \\ B.SS(ss\_id = S) \ \text{ in } v_{ss} \cap v \end{gathered} \tag{17}$$

where:
X – role value which is assigned,
Z – access type value which is assigned,
S – value of shared storage area unique identifier,
v_{ss} – time validity of shared storage area,
v – base time validity of credential,
V – active time validity of credential.

$$\begin{gathered} B.main(role = ?X, dom = ?Y, rig = ?Z, lev = W) \\ \text{in } V \\ \longleftarrow \\ B.ss\_dom(dom = ?Y, lev = W) \\ \cap \\ B.ss\_rol(role = ?X, rig = ?Z) \\ \text{in } v_{dom} \cap v_{rol} \end{gathered} \tag{18}$$

where:

X – set of role values which are assigned,
Y – set of domain values which are assigned,
Z – set of access type values which are assigned,
W – security level value which is assigned,
v_{dom} – active time validity of component credential (16),
v_{rol} – active time validity of component credential (17),
V – active time validity of credential.

6. Prototype

The software implementing the adopted user access control model can be divided in two parts: server and client side. The architecture of the software divided into functional modules is presented in Fig. 3. The server part of

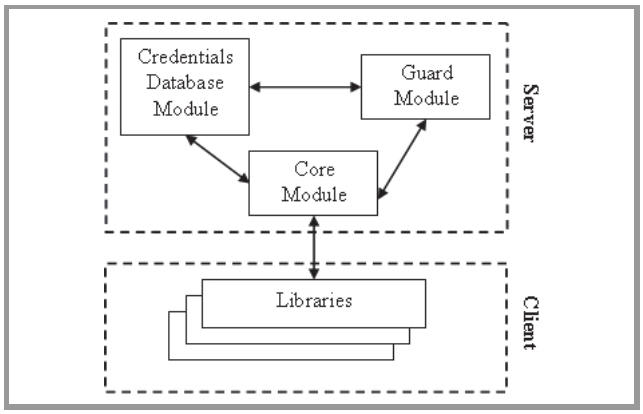

Fig. 3. Architecture of prototype solution.

the software is responsible for handling client requests according to adopted security model, and it is divided into three modules:

- *The Core* – responsible for handling communication with clients and other server modules.
- *The Credential Database* – provides functionality of knowledge base about possible actions of the SSP system users.
- *The Guard* – responsible for translating existing credentials located in the knowledge base into SELinux rules, effectively granting actual access to resources for running applications to users of the secure workstation.

The client part is implemented as a set of dedicated libraries used by all applications, requiring communication with the security model provider.

6.1. Server – Core Module

The Core module is responsible for communication with client applications and validation of incoming requests for access and modification of user access rights to resources. Its purpose is also to confirm a connecting users right to access, and to modify data located in the Credential Database. It coordinates cooperation between the rest of the server side modules implementing the access control model: the Guard and the Credential Database. It was implemented in C++ language using the functionality of an XML-RPC server library.

6.2. Server – Guard Module

The Guard Module is responsible for implementing rules of granting access to resources for applications and users of the secure workstation. The rules are derived from credentials located in the credential database and implemented with functionality of SELinux. The temporal validity of credentials allows the Guard module to use Linux cron daemon for issuing and revoking access rights at specific points of time. The SELinux policy used by SSP is based on a modified and extended Multi Level Security (MLS) policy.

6.3. Server – Credential Database Module

The Credential Database module represents the adopted security model. It provides the functionality of a knowledge base of allowed actions in the SSP system performed by the users. It consists of two elements: API and a database. Communication is implemented via an API implemented in C++. Database used in the system is PostgreSQL. Access to information in the database is restricted by the control mechanism available in PostgreSQL, based on the model Role Base Access Control (RBAC). Requests coming from different versions of the library, prepared for specific applications are handled by the Core Module on a separate restricted connection.

6.4. Client – Libraries

The client part of the software is composed from several versions of libraries designed for applications requiring communication with the Core Module of the server. The libraries provide interface for reading and modifying credentials in the Credential Database. The functionality shared by the libraries corresponds to the API of the Credential Database module. The libraries were implemented in C++ with the same XML-RPC library as the Core Module. Keys required for digital signing of messages are obtained from the key database protected by the Trusted Platform Module (TPM). The set of shared functions of a particular library corresponds to the actual purpose of the library. There are five versions of the library:

- *Basic version* – intended for the application used to login, allows validating the logged-in user role.
- *User version* – intended for applications with active USER role. It only provides ability to check which resources are accessible by the logged-in user.

- *Admin version* – intended for applications dedicated for administrators of virtual machines. It allows checking which resources are accessible by the logged-in user and obtaining information required for correct configuration of the system, e.g., which users have access to the virtual machine.
- *Spec version* – intended for Information Security Officers. It allows for unrestricted reading and modification of data in the Credential Database.
- *Audit version* – intended for Chief Audit Executives, it allows for unrestricted read-only access to data in the Credential Database and has access to history of changes.

7. Conclusion

The article presents a security model of user access control to virtual machines, which complies with functional requirements of the "Special workstation for special applications" project. Additionally, it includes new functionality, like introduction of a new SSP user role – Administrator of virtual machines or adding shared storage area as a separate resource with controlled access. A presentation of a fully operational prototype software, implementing presented security model underlines the model's usability in real applications. The paper presents the access control module's architecture and a short description of all defined modules.

The Guard module is of particular interest, as it shows how complex access models can be mapped in a practical way into simple rules for the well known SELinux solution. *RT*-family languages can be even more powerful and complex (see, e.g., RT_+^T), and they are very well suited for large, distributed systems with many independent centers of authority providing users with credentials, with complex trust relations between them. The presented architecture is adaptable to this setting, showing that local security mechanisms, such as SELinux, are still applicable in such distributed systems. The necessary approach is to infer simple access rights from the *RT* credentials. These rights can be applied in an automatic way, even taking into account limited periods of validity of credentials.

Acknowledgements

This work is part of the project called "Secure workstation for special applications" and is funded by a grant number OR00014011 from the National Center for Research and Development – science funding for years 2010-2012.

References

[1] A. Kozakiewicz, A. Felkner, J. Furtak, Z. Zieliński, M. Brudka, and M. Małowidzki, "Secure workstation for special applications", in *Secure and Trust Computing, Data Management, and Applications*, C. Lee, J.-M. Seigneur, J. J. Park, R. R. Wagner, Eds., Communications in Computer and Information Science, vol. 187. Berlin: Springer, 2011, pp. 174–181.

[2] N. Li, J. Mitchell, and W. Winsborough, "Design of a role-based trust-management framework", in *Proc. IEEE Symp. Secur. Priv.*, Oakland, CA, USA, 2002, pp. 114–130.

[3] A. Felkner, "Modeling trust management in computer systems", in *Proc. IX Int PhD Worksh OWD 2007, Conf Archives PTETiS*, Wisła, Poland, 2007, vol. 23, pp. 65–70.

[4] D. Ferraiolo and D. Kuhn, "Role-based access control", in *Proc. 15th Nat. Comp. Secur. Conf.*, Barltimore, USA, 1992, pp. 554–563.

[5] N. Li and J. Mitchell, "RT: A role-based trust-management framework", in *Proc. 3rd DARPA Inform. Survivability Conf. Exp.*, Washington, DC, USA, 2003, pp. 201–212.

[6] N. Li, W. Winsborough, and J. Mitchell, "Distributed credential chain discovery in trust management", *J. Comput. Secur*, vol. 1, pp. 35–86, 2003.

[7] A. Felkner and K. Sacha, "Deriving RT^T credentials for role based trust management", *e-Informatica Softw. Engin. J. (ISEJ)*, vol. 4, pp. 9–19, 2010.

[8] A. Felkner and A. Kozakiewicz, "Time validity in role-based trust management inference system", in *Secure and Trust Computing, Data Management, and Applications*, C. Lee, J.-M. Seigneur, J. J. Park, and R. R. Wagner, Eds., Communications in Computer and Information Science, vol. 187. Berlin: Springer, 2011, pp. 7–15.

[9] A. Felkner and A. Kozakiewicz, "Czasowa ważność poświadczeń języka RT_+^T", *Studia Informatica*, vol. 32, pp. 145–154, 2011 (in Polish).

[10] D. D. Bell and L. J. La Padula, "Secure Computer System: Unified Exposition and Multics Interpretation", ESDTR-75-306, Bedford, MA: ESD/AFSC, Hanscom AFB, 1974 [Online]. Available: http://csrc.nist.gov/publications/history/bell76.pdf

[11] D. E. Bell, "Looking back at the Bell-La Padula model", in *Proc. 21st Ann. Comp. Secur. Appl. Conf. ACSAC 2005*, Tucson, AZ, USA, 2005, pp. 337–351.

Krzysztof Lasota works as Research Associate at Network and Information Security Methods Team in the Research Division of NASK. He received his B.Sc. (2010) and M.Sc. (2012) degrees in Telecommunications from Warsaw University of Technology, Faculty of Electronics and Information Technology. Currently he is a Ph.D. student there. He is co-author of several publications. He participated in security-related projects at NASK, including FISHA and HoneySpider Network. Currently he participates in the project "Secure workstation for special applications", aiming to create a workstation using virtualization to separate sensitive information from different security domains. The project scope also includes access control issues (access control models, biometric and non-biometric identification) and audit support. Furthermore, his research aimes at developing new heuristic methods for threat detection. The study focuses on the possibilities of using lexical properties of domain names for detection of malicious WWW sites.

E-mail: krzysztof.lasota@nask.pl
Research and Academic Computer Network (NASK)
Wąwozowa st 18
02-796 Warszawa, Poland

Adam Kozakiewicz got his M.Sc. in Information Technology and Ph.D. in Telecommunications at the Faculty of Electronics and Information Technology of Warsaw University of Technology, Poland. Currently he works at NASK as Assistant Professor and Manager of the Network and Information Security Methods Team, also as part-time Assistant Professor at the Institute of Control and Computation Engineering at the Warsaw University of Technology. His main scientific interests include security of information systems, parallel computation, optimization methods and network traffic modeling and control.
E-mail: adam.kozakiewicz@nask.pl
Research and Academic Computer Network (NASK)
Wąwozowa st 18
02-796 Warsaw, Poland