TLA+ based access control model specification

A.V. Kozachok <a.kazachok@academ.mks.rsnet.ru>
Academy of the Federal Guard Service,
35, Priborostroitelnaya St., Oryol, 302034, Russia

Abstract. The article describes TLA+ access control model specification for computer systems, ensuring compliance with the mandatory integrity and confidentiality monitoring requirements with considering memory-based covert channels. The distinctive feature of the model is taking into account the characteristics of the lifecycle of electronic documents and their operating procedure. To specify the access control model, Lamport's Temporal Logic of Actions language was chosen (TLA+). Its notation seems to be the closest to generally accepted mathematical notation and its expressive capabilities and tools allow describing and verifying systems specified as finite automata. The following actions are defined in the model: create/delete a subject, read, write, append (blind write), create/delete an object, grant/remove access rights, include an object, exclude a nested object, approve an object (document), archive an object (document), cancel an approved object (document), copy an object (document). The following invariants are also defined: the type invariant (includes checking the compliance of all fields of the object, the compliance of the subject type, the uniqueness of the subject and object identifiers) and the safety invariant (includes checking the confidentiality and integrity labels of the interacting subjects and objects, the correctness of rights assignment procedures).

Keywords: security models; computer systems; verification; modelling; temporal logic; security policy; access control.

DOI: 10.15514/ISPRAS-2018-30(5)-9

For citation: Kozachok A.V. TLA+ based access control model specification. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 5, 2018, pp. 147-162. DOI: 10.15514/ISPRAS-2018-30(5)-9

1. Introduction

The problems of ensuring information security become more acute when information technologies are developing and penetrating in all spheres of life. The complexity and amount of software being developed and used are constantly increasing, which leads to the emergence of new threats and vulnerabilities.

It should be noted that some vulnerabilities are caused not by typical errors when programming, but by errors when designing software systems in general. Such defects are quite difficult to detect and to correct during the operation phase.

One of the possible solutions to solve this problem is the modeling and verification of the algorithms being developed for compliance with specified properties.

It is especially important to model the protection mechanisms of computer systems. For example, "The Information Security Requirements for Operating Systems" by FSTEC of Russia and developed on the basis of these requirements and according to GOST R ISO/IEC 15408 protection profiles and security targets contain the requirements of the ADV_SPM.1 functional component to present a formal security policy [1, 2]. In scientific studies, the formal description of security policies and access control models in operating systems is also given special attention [3-6].

There are a number of formal languages and relevant software tools providing the ability of the formalized description of a mathematical model [7]: Alloy [8], B [9], Event-B [10, 11], VDM [12], Z [13], TLA + [14-17].

2. Problem Formulation

Access control systems in computer systems provide mechanisms for controlling and restricting access for users or processes (subjects) to a variety of objects.

As part of the research, the task was to develop a model for controlling access to computer system resources, which would ensure that the requirements of mandatory integrity and confidentiality monitoring are met, taking into account information flows by memory [18].

The distinctive feature of the model is taking into account the characteristics of the lifecycle of electronic documents and their operating procedure.

To specify the access control model, Lamport's Temporal Logic of Actions language was chosen (TLA+). Its notation seems to be the closest to generally accepted mathematical notation and its expressive capabilities and tools allow describing and verifying systems specified as finite automata [19-21]. Also, in some research works, this notation was used to solve the problem of verifying access control models [22, 23].

3. Model Specification

The extension of temporal logic [24] is Lamport's Temporal Logic of Actions. It allows describing interacting and open-loop systems.

Unlike predicate logic, temporal logic of actions has the following operators [14]:

$\square$ "always in the future" operator;

$\blacksquare$ "always in the past" operator;

$\bigcirc$ "next-time" operator;

$\ominus$ "at one point in time" operator;

$\Diamond$ "once in the future" operator;

$\blacklozenge$ "once in the past" operator;

$\mathcal{U}$ "until" binary operator;

$\mathcal{S}$ "since" binary operator.

The basic relations between the operators can be represented as follows:

$$\Diamond F \equiv \neg\Box\neg F \qquad \blacklozenge F \equiv \neg\blacksquare\neg F$$

$$\Diamond F \equiv (F \vee \neg F)\mathcal{U}F \qquad \blacklozenge F \equiv F\mathcal{S}(F \vee \neg F)$$

Logical formulas in the proposed access control model are defined as follows (in the Backus-Naur form):

$$\langle\phi\rangle \models PredAction \mid p(t_1,\dots,t_n) \mid \neg\phi$$

$$\mid \phi\vee\phi \mid \phi\wedge\phi \mid \phi\to\phi \mid \forall x:\phi$$

$$\mid \exists x:\phi \mid \Box\phi \mid \Diamond\phi \mid \bigcirc\phi \mid \phi\mathcal{U}\phi$$

$$\mid \blacksquare\phi \mid \blacklozenge\phi \mid \ominus\phi \mid \phi\mathcal{S}\phi,$$

where $PredAction$ – actions, p – arity of a predicate n, $t_1,\dots,t_n$ – terms, x – variable.

In general, the specification of the access control model in TLA+ is as follows

$$Spec \triangleq Init \wedge \Box[Next]_{vars}, \tag{1}$$

where $Init$ – initialization procedure of initial values of model variables, $Next$ – action predicate that changes the state of the model and the values of variables, $vars$ – variables of the model.

3.1 Definition of Model Variables

Variable values may change after the execution of action predicates:

$$\text{VARIABLES } A, O, S,$$

$$vars \triangleq \langle A, O, S\rangle.$$

where A – set of current (happened) access events, O – set of objects, S – set of subjects, $\triangleq$ – "equal by definition" symbol.

3.2 Creating Data Types Describing Objects and Subjects of the Model

TLA + does not have strong typing (only embedded types are checked by default); however, checking of invariants of types is an integral part of the specification, because the verification is performed by the ModelChecking method [25]. A number of values specified in the model are model for reducing the resource-intensiveness of the verification process, but they do not affect the generality and adequacy of the model as a whole.

Description of the type that specifies the objects:

$$\begin{aligned} Objects \triangleq \big[& oid : ObjectIDs, meta : ObjectMeta, body : ObjectBody, \\ & owner : SubjectIDs, grantm : GrantedRights, \\ & grantb : GrantedRights, incl : ObjectIDs, \\ & st : ObjectStates \big]. \end{aligned}$$

Description of the type that specifies the subjects:

$$\begin{aligned} Subjects \triangleq \big[& sid : SubjectIDs, cnfl : ConfidLevels, \\ & \mathrm{int}\, l : IntegrLevels, cat : \mathrm{SUBSET}\, Categories, \\ & owner : SubjectIDs \big]. \end{aligned}$$

Sets of identifiers of subjects and objects (model values):

$$SubjectIDs \triangleq 0 \ldots 5,$$

$$ObjectIDs \triangleq 0 \ldots 5.$$

Sets of labels for the levels of confidentiality, integrity, and categories (model values):

$$ConfidLevels \triangleq 0 \ldots 1,$$

$$IntegrLevels \triangleq 0 \ldots 1,$$

$$Categories \triangleq \{\text{"c1"}, \text{"c2"}, \text{"c3"}\}.$$

Set of object states ("work", "approved", "archived", "cancelled"):

$$ObjectStates \triangleq \{\text{"work"}, \text{"approved"}, \text{"archived"}, \text{"cancelled"}\}.$$

Set of types of access and tuple of rights assignment

$$Rights \triangleq \{\text{"read"}, \text{"write"}\},$$

$$GrantedRights \triangleq \langle sid : SubjectIDs, r : Rights \rangle.$$

For electronic documents in electronic document management systems, there is the separation of rights of access to meta information and the content of a document [26]. This opportunity was also taken into account in the developed model:

$$ObjectParts \triangleq \{\text{"meta"}, \text{"body"}\},$$

$$ObjectMeta \triangleq [cnfl : ConfidLevels, intl : IntegrLevels],$$

$$ObjectBody \triangleq [cnfl : ConfidLevels, intl : IntegrLevels].$$

Auxiliary operators and functions were also identified for the selection: child element of an object ($sc(o)$), set of child elements of an object ($scs(o)$), set of copies of an object ($scp(o)$), set of child objects of a subject ($sw(s)$) and update the owner of a subject ($UpdateOwner(sh, sp)$).

3.3 Initialization of Initial Values

The set of current access events is empty at the initialization stage. The set objects can also be empty at this stage. However, the set of subjects in this case must contain at least one subject. For example, the values of the sets of subjects and objects are initialized with model values:

$$
\begin{aligned}
& s0 \triangleq [sid \mapsto 0, cnfl \mapsto 1, intl \mapsto 1, \\
& \qquad cat \mapsto \{"\mathrm{c1}","\mathrm{c2}"\}, owner \mapsto 0], \\
& s1 \triangleq [sid \mapsto 1, cnfl \mapsto 1, intl \mapsto 0, \\
& \qquad cat \mapsto \{"\mathrm{c2}","\mathrm{c3}"\}, owner \mapsto 1], \\
& o0 \triangleq [oid \mapsto 0, \\
& \qquad meta \mapsto [cnfl \mapsto 0, intl \mapsto 0], \\
& \qquad body \mapsto [cnfl \mapsto 0, intl \mapsto 0], \\
& \qquad cat \mapsto \{"\mathrm{c1}","\mathrm{c2}"\}, \\
& \qquad owner \mapsto 1, \\
& \qquad grantm \mapsto \{\langle 0,"\mathrm{write}"\rangle, \langle 0,"\mathrm{read}"\rangle\}, \\
& \qquad grantb \mapsto \{\langle 0,"\mathrm{read}"\rangle\}, \\
& \qquad incl \mapsto \{\}, \\
& \qquad copy \mapsto \{\}, \\
& \qquad st \mapsto "\mathrm{work}", \\
& Init \triangleq \land A = \{\} \\
& \qquad \land S = \{s0, s1\} \\
& \qquad \land O = \{o0\}.
\end{aligned}
\tag{2}
$$

where oid – object identifier, $meta$ – object meta information, $body$ – object content, $cnfl$ – level of confidentiality, $intl$ – level of integrity, cat – category, st – object state, $owner$ – owner, $incl$ – set of included documents, $grantm$ – access rights assigned to meta information, $grantb$ – access rights assigned to the content of an object, sid – subject identifier.

The model provides actions to create and delete subjects and objects; therefore, the values presented in (2) only allow modeling possible states and finding errors faster.

3.4 Predicates of Actions

The following possible actions are specified in the model:

$$\begin{aligned} Next \triangleq\ & \lor CreateSubjectD \lor DeleteSubjectD \\ & \lor ReadD \qquad\qquad \lor CreateObjectD \\ & \lor WriteD \qquad\qquad \lor AppendWD \\ & \lor DeleteObjectD \quad \lor GrantRightsD \\ & \lor RemoveRightsD \lor IncludeObjectD \\ & \lor ExludeObjectD \quad \lor ApproveObjectD \\ & \lor ArchiveObjectD \lor CancelObjectD \\ & \lor CopyObjectD, \end{aligned}$$

where $CreateSubjectD$ – create a subject, $DeleteSubjectD$ – delete a subject, $ReadD$ – read, $WriteD$ – write, $AppendWD$ – write at the end ("blind" writing), $CreateObjectD$ – create an object, $DeleteObjectD$ – delete an object, $GrantRightsD$ – assign access rights, $RemoveRightsD$ – remove access rights, $IncludeObjectD$ – include an object to an object, $ExludeObjectD$ – exclude an included object, $ApproveObjectD$ – approve an object (document), $ArchiveObjectD$ – archive an object (document), $CancelObjectD$ – cancel the action of an approved object (document), $CopyObjectD$ – copy an object (document).

Taking the example of the action presented in (3), one can consider the order in which the necessary pre-conditions and post-conditions are specified. Pre-conditions are predicates, the execution of which is necessary to ensure the safety of an action. Post-conditions determine how model variables may be changed according to the results of an action.

The $Read(s, o, r, op)$ action is performed by s (subject) in relation to o (object) with the $r = "read"$ right and the specific component of the meta information document ($op = "meta"$) or its content ($op = "body"$). The action in the model defines the following post-conditions: the current access to the set of access events is added ($A' = A \cup \{\langle s.sid, o.oid, r, op \rangle\}$) and sets of subjects and objects remain as unchanged ($\mathrm{UNCHANGED}\langle \mathrm{S}, \mathrm{O} \rangle$).

The $ReadD$ action imposes requirements to account all possible states of the model ($\exists r \in R : \exists s \in S : \exists o \in O : \exists op \in ObjectParts$) and to verify the necessary safety predicates of the execution of the action (pre-conditions).

$$
\begin{aligned}
ReadD \triangleq \exists r \in Rights :\ & \\
& \exists s \in S : \\
& \exists o \in O : \\
& \exists op \in ObjectParts : \\
& \qquad \wedge r = \text{"read"} \\
& \qquad \wedge o.cat \subseteq s.cat \\
& \qquad \wedge \vee \wedge op = \text{"meta"} \\
& \qquad\qquad \wedge s.cnfl \geq o.meta.cnfl \\
& \qquad\qquad \wedge \vee \{\langle s.sid, r\rangle\} \subseteq o.grantm \\
& \qquad\qquad\quad \vee o.owner = s.sid \\
& \qquad\quad \vee \wedge op = \text{"body"} \\
& \qquad\qquad \wedge s.cnfl \geq o.body.cnfl \\
& \qquad\qquad \wedge \vee \{\langle s.sid, r\rangle\} \subseteq o.grantb \\
& \qquad\qquad\quad \vee o.owner = s.sid \\
& \qquad \wedge Read(s, o, r, op)
\end{aligned}
\tag{3}
$$

Important ones are checks of confidentiality labels ($s.cnfl \geq o.meta.cnfl$ and $s.cnfl \geq o.body.cnfl$), as well as checks of categories as a part of mandatory confidentiality monitoring ($o.cat \subseteq s.cat$). It then checks the condition if s (subject) has the r access right to o (object) depending on op ($\langle ssid, r\rangle \subseteq o.grantm$ or $\langle ssid, r\rangle \subseteq o.grantb$).

It is also possible that the subject s is the owner of the object o ; in this case, the requirement to have a right in the set of access rights to the object is not imposed; the owner has full rights ($o.owner = s.sid$).

Consider also the action to create a copy of the $CopyObjectD$ object in (4).

$$
\begin{aligned}
CopyObject(s, o, id) \triangleq\ & \land O' = O \cup \{[oid \mapsto id, \\
& \quad meta \mapsto [cnfl \mapsto o.meta.cnfl, \\
& \quad intl \mapsto o.meta.intl], \\
& \quad body \mapsto [cnfl \mapsto o.body.cnfl, \\
& \quad intl \mapsto o.body.intl], \\
& \quad owner \mapsto s.sid, \\
& \quad grantm \mapsto o.grantm, \\
& \quad grantb \mapsto o.grantb, \\
& \quad cat \mapsto o.cat, \\
& \quad incl \mapsto o.incl, \\
& \quad st \mapsto \text{"approved"}, \\
& \quad copy \mapsto \{o.oid\}]\} \\
& \land A' = A \cup \{\langle s.sid, o.oid, \text{"copy"}, id\rangle\} \\
& \land \text{UNCHANGED}\langle S\rangle
\end{aligned}
$$

$$
\begin{aligned}
CopyObject \triangleq\ & \\
& \exists s \in S : \\
& \land O \neq \{\} \\
& \land \exists o \in O : \land o.owner = s.sid \\
& \qquad \land o.incl = \{\} \\
& \qquad \land o.st = \text{"approved"} \\
& \qquad \land s.cat \subseteq o.cat \\
& \qquad \land s.cnfl = o.meta.cnfl \\
& \qquad \land s.intl \geq o.meta.intl \\
& \qquad \land s.cnfl = o.body.cnfl \\
& \qquad \land s.intl \geq o.body.intl \\
& \qquad \land Cardinality(scp(o)) < 2 \\
& \qquad \land \exists id \in ObjectIDs : \\
& \qquad\quad \land \forall oo \in O : \\
& \qquad\qquad \land id \neq oo.oid \\
& \qquad\qquad \land CopyObject(s, o, id)
\end{aligned}
\tag{4}
$$

The $CopyObject(s, o, id)$ is performed by s (subject) in relation to o (object). As a parameter, the identifier is also passed for the object being created, the selection of which is carried out taking into account the requirement for the unique identifiers of all objects ($\exists id \in ObjectIDs : \forall oo \in O : id \neq oo.oid$). In this action, a new object is added to a set of objects possessing the attributes of the o original object with the exception of the $copy$ field, in which it is indicated which object is a copy of this object. It also adds the current access to the set of access events and indicates that the set of subjects does not change when performing this action.

The $CopyObjectD$ imposes requirements to account all possible states of the model ($\exists s \in S : O \neq \{\} \wedge \exists o \in O$) and to check the necessary pre-conditions:

- the subject is the owner of the object ($o.owner = s.sid$);
- the object has no included objects ($o.incl = \{\}$);
- the object is in an "approved" state ($o.st = "approved"$);
- subject categories are a subset of object categories ($s.cat \subseteq o.cat$);
- the privacy level of the subject is equal to the privacy level of the object ($s.cnfl = o.meta.cnfl \wedge s.cnfl = o.body.cnfl$);
- the level of integrity of the subject is greater than or equal to the level of integrity of the object ($s.intl \geq o.meta.intl \wedge s.intl \geq o.body.intl$);
- the number of copies of the object does not exceed the specified value ($Cardinality(scp(o)) < 2$);
- the created object must have a unique identifier.

3.5 Model invariants

In addition to specifying the pre- and post-conditions in the model, it is possible to set invariants for global properties, which are mandatory for all states of the model. The basic and generally accepted invariant is an invariant of types in (5).

It verifies that all fields match all objects and also checks the conformity of the type of all subjects, as well as it checks the uniqueness of all identifiers of subjects and objects.

$$
\begin{aligned}
& ObjTypeInv \triangleq \\
& \qquad \wedge \forall o \in O : \wedge\, o.oid \in ObjectIDs \\
& \qquad \wedge\, o.meta \in ObjectMeta \\
& \qquad \wedge\, o.body \in ObjectBody \\
& \qquad \wedge\, o.owner \in SujectIDs \\
& \qquad \wedge \{o.incl\} \subseteq \text{SUBSET}\, ObjectIDs \\
& \qquad \wedge \{o.copy\} \subseteq \text{SUBSET}\, ObjectIDs \\
& \qquad \wedge\, o.st \in ObjectStates \\
& \qquad \wedge\, o.cat \subseteq \text{SUBSET}\, Categories
\end{aligned}
$$

$$
\begin{aligned}
TypeInv \triangleq\ & \wedge S \subseteq Subjects \\
& \wedge ObjTypeInv \\
& \wedge \forall sn \in S : \text{IF}\, \exists\, sm \in S : \wedge\, sm \neq sn \\
& \qquad\qquad\qquad\qquad\quad \wedge\, sn.sid = sm.sid \\
& \qquad \text{THEN FALSE} \\
& \qquad \text{ELSE TRUE} \\
& \wedge \forall on \in O : \text{IF}\, \exists\, om \in O : \wedge\, om \neq on \\
& \qquad\qquad\qquad\qquad\quad \wedge\, on.oid = om.oid \\
& \qquad \text{THEN FALSE} \\
& \qquad \text{ELSE TRUE}
\end{aligned}
\tag{5}
$$

The second invariant specified in the model is the safety invariant in (6). It checks the following conditions:

- the level of confidentiality of object meta information does not exceed the level of confidentiality of the content of the object ($o.meta.cnfl \leq o.body.cnfl$);
- the integrity level of object meta information is equal to the integrity level of object content ($o.meta.intl = o.body.intl$);
- if the object contains an included object, then the set of access rights to the parent object is a subset of access rights to the child object, and the state of the parent object is equal to the state of the child one ($o.grantm \subseteq oi.grantm \wedge o.grantb \subseteq oi.grantb \wedge o.st = o.ist$);

- the number of included objects does not exceed the specified value ($Cardinality(scs(o)) \leq 1$);
- the number of copies of the object does not exceed the specified value ($Cardinality(scp(o)) \leq 2$);
- if the subject is the owner of the object, the rights for it are not assigned, because the subject has full rights ($\neg o.grantm \subseteq (s.sid \times Rights) \wedge \neg o.grantb \subseteq (s.sid \times Rights)$);
- if the object is in the "archived" or "canceled" state, then it is forbidden to assign the right to "record" to any of the subjects ($o.grantm \cap (SubjectIDs \times \{\text{"write"}\}) \neq \{\} \vee o.grantb \cap (SubjectIDs \times \{\text{"write"}\}) \neq \{\}$).

The execution of invariants for all states of the model provides the proof of the following theorem (7) regarding the specification of the model (1) and the invariants (5), (6):

Theorem. $Spec \Rightarrow \Box(TypeInv \wedge Safety)$. (7)

3.6 Model verification

The significant limitation of the approach to the verification based on the ModelChecking method is the need to check all possible model states. That is, if one specifies any conditions for countable sets, for example, $sid \in Nat$ or $oid \in Nat$, the verification process does not end, because the number of model states will also be countable. Therefore, in the specification, model values were used to reduce the time required for model verification.

The verification of the developed model was performed using the TLC2 tool version 2.13 [27]. Thus, the time spent on verification was about 2835 minutes (more than 47 hours) on the server with the Ubuntu 16.04 operating system, Intel Xeon E5-2620 v2 24 cores 2.10 GHz and 32 GB of RAM. 16,284,800,554 states were verified with the average system performance of 5,743,616 states per minute.

4. Conclusion

The developed model can be used to set policies for controlling access to computer system resources, where information of various confidentiality categories, as well as various levels of confidentiality and integrity, circulates. The TLA + language notation used in the model has sufficient flexibility and expressive capabilities for solving a wide range of modeling problems in computer security. We should note that the requirements for the absence of covert timing channels in this model were not taken into account; that is the direction for further research.

```
$Safety \triangleq \land \forall o \in O : \land o.meta.cnfl \leq o.body.cnfl$
                   $\land o.meta.intl = o.body.intl$
                   $\land \text{IF } o.incl \neq \{\}$
                     $\text{THEN } \forall i \in o.incl :$
                            $\land \exists oi \in O :$
                              $\land oi.oid \neq o.oid$
                              $\land o.oid = i$
                              $\land o.grantm \subseteq oi.grantm$
                              $\land o.grantb \subseteq oi.grantb$
                              $\land o.st = oi.st$
                     $\text{ELSE TRUE}$
                   $\land Cardinality(scs(o)) \leq 1$
                   $\land Cardinality(scp(o)) \leq 2$                         (6)
                   $\land \exists s \in S :$
                     $\land o.owner = s.sid$
                     $\land \text{IF } o.grantm \neq \{\}$
                       $\text{THEN } \neg o.grantm \subseteq (\{s.sid\} \times Rights)$
                       $\text{ELSE TRUE}$
                     $\land \text{IF } o.grantb \neq \{\}$
                       $\text{THEN } \neg o.grantb \subseteq (\{s.sid\} \times Rights)$
                       $\text{ELSE TRUE}$
                   $\land \neg \exists o \in O : \land \lor o.st = \text{"archived"}$
                                 $\lor o.st = \text{"cancelled"}$
                               $\land \lor o.grantm \cap (SubjectIDs \times \{\text{"write"}\}) \neq \{\}$
                               $\land \lor o.grantb \cap (SubjectIDs \times \{\text{"write"}\}) \neq \{\}$
```

References

[1]. Devyanin P.N. On the problem of representation of the formal model of security policy for operating systems. Trudy ISP RAN/Proc. ISP RAS. vol. 29, issue 3, 2017, pp. 7-16 (in Russian). DOI: 10.15514/ISPRAS-2017-29(3)-1.

[2]. Devyanin P. N. Approaches to formal modelling access control in postgresql within framework of the mrosl DP-mode. PDM. Prilozhenie/Applied Discrete Mathematics. Supplement, no. 10, 2017, pp. 111-114 (in Russian.) DOI: 10.17223/2226308X/10/44.

[3]. Devyanin P. N. System administration in MROSL DP-model. . PDM/Applied Discrete Mathematics. Supplement, 2013, no 4, pp. 22-40 (in Russian).

[4]. Devyanin P. N. Security violation necessary conditions for time information flows in MROSL DP-model. . PDM. Prilozhenie/Applied Discrete Mathematics. Supplement, no 8, 2015, pp. 81-83 (in Russian) DOI: 10.17223/2226308X/8/30.

[5]. Devyanin P. N. About results of designing hierarchical representation of mrosl DP-model. PDM. Prilozhenie/Applied Discrete Mathematics. Supplement, 2016, no 9. pp. 83-87 (in Russian) DOI: 10.17223/2226308X/9/32.

[6]. Devyanin P. N. The level of negative roles of the hierarchical representation of MROSL DP-model. PDM/Applied Discrete Mathematics, 2018, no 39, pp. 58-71 (in Russian) DOI: 10.17223/20710410/39/5.

[7]. P.N. Devyanin et al. Modeling and verification of security policies for access management in operating systems. ISP RAN, 2018, 181 p. Available at: http://www.ispras.ru/publications/security_policy_modeling_and_verification.pdf (in Russian)

[8]. Jackson D. Software Abstractions: Logic, Language, and Analysis. The MIT Press, 2012, 376 p.

[9]. Abrial J.-R. The B-book: Assigning Programs to Meanings. Cambridge University Press, 1996, 779 p.

[10]. Jean-Raymond Abrial. Modeling in Event-B. System and Software Engineering. Cambridge University Press, 2010.

[11]. P.N. Devyanin, V. V. Kulyamin, A.K. Petrenko, A.V. Khoroshilov, I.V. Shchepetkov. Comparison of specification decomposition methods in Event-B. Programming and Computer Software, vol. 42, no. 4, 2016, pp. 198-205. DOI: 10.1134/S0361768816040022.

[12]. Jones C. B. Systematic Software Development Using VDM (2nd Ed.). Prentice-Hall, Inc., 1990, 333 p.

[13]. Singh M., Sharma A. K., Saxena R. Formal Transformation of UML Diagram: Use Case, Class, Sequence Diagram with Z Notation for Representing the Static and Dynamic Perspectives of System. Proc. of the International Conference on ICT for Sustainable Developmen, 2016, pp. 25-38, DOI: 10.1007/978-981-10-0135-2_3.

[14]. Lamport L. The Temporal Logic of Actions. ACM Trans. Program. Lang. Syst, vol. 16, no. 3, 1994, pp. 872-923.

[15]. Lamport L. Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley, 2002.

[16]. L. Lamport et al. Specifying and verifying systems with TLA+. Proc. of the 10th ACM SIGOPS European Workshop, 2002, pp. 45-48.

[17]. Lamport L. The PlusCAL Algorithm Language. Lecture Notes in Computer Science, vol. 4229, 2006, pp. 23-23. DOI: 10.1007/11888116_2.

[18]. Devyanin P. N. Security conditions for information flows by memory within the mrosl DP-model. PDM. Prilozhenie/Applied Discrete Mathematics. Supplement, issue 7, 2014, pp. 82-85 (in Russian).
[19]. Denis Cousineau et al. TLA + Proofs. Lecture Notes in Computer, vol. 7436, 2012, pp. 147-154. DOI: 10.1007/978-3-642-32759-9_14.
[20]. Kaustuv Chaudhuri et al. A TLA+ Proof System. In Proc. of the Combined KEAPPA - IWIL Workshops, 2008, pp. 17-37, URL: http://ceur-ws.org/Vol-418/paper2.pdf.
[21]. Merz S., Vanzetto H. Encoding TLA+ into Many-Sorted First-Order Logic. In Proc. of the 5th International Conference on Abstract State Machines, 2016, pp. 54-69.
[22]. Xinwen Zhang et al. Formal Model and Policy Specification of Usage Control. ACM Transactions on Information and System Security, vol. 8, no. 4, 2005, pp. 351-387. DOI: 10.1145/1108906.1108908.
[23]. Gouglidis A., Grompanopoulos C., Mavridou A. Formal Verification of Usage Control Models: A Case Study of UseCON Using TLA+. In Proc. of the 1st International Workshop on Methods and Tools for Rigorous System Design, 2018, pp. 52-64.
[24]. Stirling C. Modal and temporal logics. LFCS, Department of Computer Science, University of Edinburgh, 1991.
[25]. McMillan K. L. Eager Abstraction for Symbolic Model Checking. Lecture Notes in Computer Science, vol. 10981, 2018, pp. 191-208. DOI: 10.1007/978-3-319-96145-3_11.
[26]. Storey V. C., Song I.-Y. Big data technologies and Management: What conceptual modeling can do. Data & Knowledge Engineering, vol. 108, 2017, pp. 50-67. DOI: 10.1016/j.datak.2017.01.001.
[27]. Chris Newcombe et al. How AmazonWeb Services Uses Formal. Communications of the ACM, vol. 58, no. 4, 2015, pp. 66-73.

Спецификация модели управления доступом на языке темпоральной логики действий Лэмпорта

А. В. Козачок <a.kazachok@academ.mks.rsnet.ru>
Академия Федеральной службы охраны Российской федерации,
302015, Россия, г. Орёл, ул. Приборостроительная, д. 35

Аннотация. В статье представлено описание модели управления доступом на языке темпоральной логики действий Лэмпорта, обеспечивающей выполнение требований мандатного контроля целостности и конфиденциальности с учетом информационных потоков по памяти. Отличительной особенностью модели является учет особенностей жизненного цикла электронных документов (задания прав к метаинформации и содержимому документа отдельно, ограничение числа копий документа). Для задания модели управления доступом был выбран язык темпоральной логики действий Лэмпорта, поскольку его нотация представляется наиболее близкой к общепринятой математической, выразительные возможности и инструментальные средства позволяют описывать и верифицировать системы, заданные в виде конечных автоматов. В модели определены следующие действия: создание/удаление субъекта, чтение, запись, дозапись ("слепая" запись), создание/удаление объекта, назначение/удаление прав доступа, вложение объекта в объект, исключение вложенного объекта, утверждение объекта (документа), отправка объекта (документа) в архив, отмена действия утвержденного объекта (документа), копирование объекта (документа). Также определены следующие инварианты: проверки типов (включает в себя проверку соответствия всех полей объектов, также проверку соответствия типу субъектов и проверку уникальности идентификаторов субъектов и объектов) и проверки безопасности (включает в себя проверку меток конфиденциальности и целостности взаимодействующих субъектов и объектов, а также корректность процедуры назначения прав доступа).

Ключевые слова: модели безопасности, компьютерные системы, верификация, моделирование, темпоральная логика, политика безопасности, управление доступом.

DOI: 10.15514/ISPRAS-2018-30(5)-9

Для цитирования: Козачок А.В. Спецификация модели управления доступом на языке темпоральной логики действий Лэмпорта. Труды ИСП РАН, том 30, вып. 5, 2018 г., стр. 147-162 (на английском языке). DOI: 10.15514/ISPRAS-2018-30(5)-9

Список литературы

[1]. Девянин П.Н. О проблеме представления формальной модели политики безопасности операционных систем. Труды ИСП РАН, том 29, вып. 3, 2017 г., стр. 7-16. DOI: 10.15514/ISPRAS-2017-29(3)-1.

[2]. Девянин П.Н. Реализация невырожденной решётки уровней целостности в рамках иерархического представления МРОСЛ ДП-модели. ПДМ. Приложение, № 10, 2017 г., стр. 111-114. DOI: 10.17223/2226308X/10/44.

[3]. Девянин П.Н. Администрирование системы в рамках мандатной сущностно-ролевой ДП-модели управления доступом и информационными потоками в ОС семейства Linux. ПДМ. Приложение, № 4, 2013 г., стр. 22-40.

[4]. Девянин П.Н. Необходимые условия нарушения безопасности информационных потоков по времени в рамках МРОСЛ ДП-модели. ПДМ. Приложение, № 8, 2015 г., стр. 81-83. DOI:10.17223/2226308X/8/30.

[5]. Девянин П.Н. О результатах формирования иерархического представления МРОСЛ ДП-модели. ПДМ. Приложение, № 9, 2016 г., стр. 83-87. DOI: 10.17223/2226308X/9/32.

[6]. Девянин П.Н. Уровень запрещающих ролей иерархического представления МРОСЛ ДП-модели. ПДМ, № 39, 2018 г., стр. 58-71. DOI: 10.17223/20710410/39/5.

[7]. П.Н. Девянин и др. Моделирование и верификация политик безопасности управления доступом в операционных системах. Институт системного программирования им. В. П. Иванникова РАН, 2018, 181 с. URL: http://www.ispras.ru /publications/2018/security_policy_modeling_and_verification.

[8]. Jackson D. Software Abstractions: Logic, Language, and Analysis. The MIT Press, 2012, 376 p.

[9]. Abrial J.-R. The B-book: Assigning Programs to Meanings. Cambridge University Press, 1996, 779 p.

[10]. Jean-Raymond Abrial. Modeling in Event-B. System and Software Engineering. Cambridge University Press, 2010.

[11]. P.N. Devyanin, V. V. Kulyamin, A.K. Petrenko, A.V. Khoroshilov, I.V. Shchepetkov. Comparison of specification decomposition methods in Event-B. Programming and Computer Software, vol. 42, no. 4, 2016, pp. 198-205. DOI: 10.1134/S0361768816040022.

[12]. Jones C. B. Systematic Software Development Using VDM (2nd Ed.). Prentice-Hall, Inc., 1990, 333 p.

[13]. Singh M., Sharma A. K., Saxena R. Formal Transformation of UML Diagram: Use Case, Class, Sequence Diagram with Z Notation for Representing the Static and Dynamic Perspectives of System. Proc. of the International Conference on ICT for Sustainable Developmen, 2016, pp. 25-38, DOI: 10.1007/978-981-10-0135-2_3.

[14]. Lamport L. The Temporal Logic of Actions. ACM Trans. Program. Lang. Syst, vol. 16, no. 3, 1994, pp. 872-923.

[15]. Lamport L. Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley, 2002.

[16]. L. Lamport et al. Specifying and verifying systems with TLA+. Proc. of the 10th ACM SIGOPS European Workshop, 2002, pp. 45-48.

[17]. Lamport L. The PlusCAL Algorithm Language. Lecture Notes in Computer Science, vol. 4229, 2006, pp. 23-23. DOI: 10.1007/11888116_2.

[18]. Девянин П. Н. Условия безопасности информационных потоков по памяти в рамках МРОСЛ ДП-модели. ПДМ. Приложение, № 7, 2014 г., стр. 82-85.

[19]. Denis Cousineau et al. TLA + Proofs. Lecture Notes in Computer, vol. 7436, 2012, pp. 147-154. DOI: 10.1007/978-3-642-32759-9_14.

[20]. Kaustuv Chaudhuri et al. A TLA+ Proof System. In Proc. of the Combined KEAPPA - IWIL Workshops, 2008, pp. 17-37, URL: http://ceur-ws.org/Vol-418/paper2.pdf.

[21]. Merz S., Vanzetto H. Encoding TLA+ into Many-Sorted First-Order Logic. In Proc. of the 5th International Conference on Abstract State Machines, 2016, pp. 54-69.

[22]. Xinwen Zhang et al. Formal Model and Policy Specification of Usage Control. ACM Transactions on Information and System Security, vol. 8, no. 4, 2005, pp. 351-387. DOI: 10.1145/1108906.1108908.

[23]. Gouglidis A., Grompanopoulos C., Mavridou A. Formal Verification of Usage Control Models: A Case Study of UseCON Using TLA+. In Proc. of the 1st International Workshop on Methods and Tools for Rigorous System Design, 2018, pp. 52-64.

[24]. Stirling C. Modal and temporal logics. LFCS, Department of Computer Science, University of Edinburgh, 1991.

[25]. McMillan K. L. Eager Abstraction for Symbolic Model Checking. Lecture Notes in Computer Science, vol. 10981, 2018, pp. 191-208. DOI: 10.1007/978-3-319-96145-3_11.

[26]. Storey V. C., Song I.-Y. Big data technologies and Management: What conceptual modeling can do. Data & Knowledge Engineering, vol. 108, 2017, pp. 50-67. DOI: 10.1016/j.datak.2017.01.001.

[27]. Chris Newcombe et al. How AmazonWeb Services Uses Formal. Communications of the ACM, vol. 58, no. 4, 2015, pp. 66-73.