СПЕЦИФИКАЦИЯ МОДЕЛИ УПРАВЛЕНИЯ ДОСТУПОМ К РАЗНОКАТЕГОРИЙНЫМ РЕСУРСАМ КОМПЬЮТЕРНЫХ СИСТЕМ

Козачок А.В.[1]

Цель статьи: *разработка модели управлениям доступом к разнокатегорийным ресурсам компьютерных систем, обеспечивающей выполнение требований мандатного контроля целостности и конфиденциальности без учета информационных потоков по времени, и ее верификация на соответствие инвариантам безопасности методом Model Checking.*

Метод: *моделирование на языке 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 автоматическом режиме методом Model Checking.*

Полученный результат: *задана модель управления доступом к разнокатегорийным электронным документам, отличительной особенностью которой является учет особенностей жизненного цикла электронных документов и порядка работы с ними. 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).*

Ключевые слова: *модели безопасности, верификация, моделирование, темпоральная логика, политика безопасности, электронные документы.*

DOI: 10.21681/2311-3456-2018-4-2-8

1. Введение

По мере развития и широкого проникновения информационных технологий во все сферы жизнедеятельности все более острыми становятся проблемы обеспечения информационной безопасности. Сложность и объемы разрабатываемого и используемого программного обеспечения постоянно растут, что приводит к появлению новых угроз и уязвимостей.

Следует отметить, что некоторые уязвимости обусловлены не типовыми ошибками при написании программ, а ошибками при проектировании программных систем в целом. Такие дефекты достаточно трудно выявляются и исправляются на этапе эксплуатации.

Одним из возможных решений данной проблемы является моделирование и верификация разрабатываемых алгоритмов на предмет соответствия заданным свойствам.

Особенно важно производить моделирование механизмов защиты компьютерных систем. Так, например, «Требования безопасности информации к операционным системам» ФСТЭК России и разработанные на их основе в соответствии с ГОСТ Р ИСО/МЭК 15408 профили защиты и задания по безопасности содержат требования функциональной компоненты ADV_SPM.1 по представлению формальной политики безопасности [1, 2]. В научных исследованиях вопросам формального описания политик безопасности и моделей управления доступом в операционных системах также уделяется особое внимание [3-6].

Существует ряд формальных языков и соответствующих программных инструментов, предоставляющих возможность формализованного описания математической модели [7]: Alloy [8], B [9], Event-B [10, 11], VDM [12], Z [13], TLA+ [14-17].

2. Постановка задачи

Системы управления доступом в компьютерных системах предоставляют механизмы контроля и ограничения доступа пользователей или процессов (субъектов) к множеству объектов.

В рамках проводимого исследования была поставлена задача по разработке модели управления доступом к разнокатегорийным ресурсам, обеспечивающей выполнение требований мандатного контроля целостности и конфиденциальности без учета информационных потоков по времени [18]

Отличительной особенностью разрабатываемой модели является учет особенностей жизненного цикла электронных документов и порядка работы с ними.

1 Козачок Александр Васильевич, кандидат технических наук, сотрудник Академия ФСО России, г. Орёл, Россия. E-mail: a.kozachok@academ.msk.rsnet.ru

Автором для задания модели управления доступом был выбран язык темпоральной логики действий Лэмпорта (TLA+), поскольку его нотация представляется наиболее близкой к общепринятой математической и выразительные возможности и инструментальные средства позволяют описывать и верифицировать системы, заданные в виде конечных автоматов [19-21]. Также в ряде исследовательских работ данная нотация применялась для решения задачи верификации моделей управления доступом [22].

3. Спецификация модели

Расширенной формой темпоральной логики является темпоральная логика действий Лэмпорта [14]. Она позволяет описывать взаимодействующие и незамкнутые системы.

В отличие от логики предикатов темпоральная логика действий содержит следующие операторы [14]:

□ – оператор «всегда в будущем»;
■ – оператор «всегда в прошлом»;
○ – оператор «в следующий момент времени»;
⊖ – оператор «в предыдущий момент времени»;
◊ – оператор «однажды в будущем»;
♦ – оператор «однажды в прошлом»;
$\mathcal{U}$ – бинарный оператор «до тех пор пока»;
S – бинарный оператор «с тех пор как».

Основные соотношения между операторами можно представить следующим образом:

$$\Diamond F \equiv \neg\Box\neg F \qquad \blacklozenge F \equiv \neg\blacksquare\neg F$$
$$\Diamond F \equiv (F \vee \neg F)\mathcal{U}F \qquad \blacklozenge F \equiv FS(F \vee \neg F)$$

Логические формулы в рамках предлагаемой модели управления доступа задаются следующим образом (в форме Бэкуса-Наура):

$$\langle\phi\rangle|= PredAction|p(t_1,\dots,t_n)|\neg\phi$$
$$|\phi \vee \phi|\phi \wedge \phi|\phi \rightarrow \phi|\forall x{:}\,\phi$$
$$|\exists x{:}\,\phi|\Box\phi|\Diamond\phi|\bigcirc\phi|\phi\mathcal{U}\phi$$
$$|\blacksquare\phi|\blacklozenge\phi|\ominus\phi|\phi S\phi,$$

где *PredAction* действия, *p* –предикат арности *n*, t_1... t_n, -- термы, *x* переменная.

В общем виде спецификация модели управления доступом на языке TLA+ имеет следующий вид:

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

где *Init* – процедура инициализации начальных значений переменных модели, *Next* – предикат действия, изменяющий состояние модели и значения переменных, *vars* – переменные модели.

3.1 Определение переменных модели

Значения переменных могут изменяться после выполнения предикатов действий:

$$\text{VARIABLES } A, O, S,$$
$$vars \triangleq \langle A, O, S\rangle.$$

где *A* – множество текущих (произошедших) доступов, *O* множество объектов, *S* – множество субъектов, ≜ – символ «равно по определению».

3.2 Задание типов данных, описывающих объекты и субъекты модели

В языке TLA+ отсутствует строгая типизация (по умолчанию производится проверка только встроенных типов), однако, проверка инвариантов типов является неотъемлемой частью спецификации, поскольку верификация производится методом ModelChecking [23]. Ряд значений, заданных в модели, являются модельными для снижения ресурсоемкости процесса верификации, но не влияют на общность и адекватность модели в целом.

Описание типа, задающего объекты:

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

Описание типа, задающего субъекты:

$$Subjects \triangleq [sid{:}\,SubjectIDs, cnfl{:}\,ConfidLevels,$$
$$intl{:}\,IntegrLevels, cat{:}\,\text{SUBSET } Categories,$$
$$owner{:}\,SubjectIDs].$$

Множества идентификаторов субъектов и объектов (значения модельные):

$$SubjectIDs \triangleq 0 \dots 5,$$
$$ObjectIDs \triangleq 0 \dots 5.$$

Множества меток уровней конфиденциальности, целостности и категорий (значения модельные):

$$ConfidLevels \triangleq 0 \dots 1,$$
$$IntegrLevels \triangleq 0 \dots 1,$$
$$Categories \triangleq \{\text{"c1"},\text{"c2"},\text{"c3"}\}.$$

Множество состояний объекта («в работе», «утвержден», «помещен в архив», «отменен»):

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

Множество видов доступа и кортеж задания прав доступа:

$$Rights \triangleq \{\text{"read"},\text{"write"}\},$$
$$GrantedRights \triangleq \langle sid{:}\,SubjectIDs, r{:}\,Rights\rangle.$$

Для электронных документов в системах электронного документооборота характерно разделение прав доступа к метаинформации и содержимому документа [24]. В разработанной модели была также учтена данная возможность:

$$ObjectParts \triangleq \{\text{"meta"},\text{"body"}\},$$
$$ObjectMeta \triangleq [cnfl{:}\,ConfidLevels, intl{:}\,IntegrLevels],$$
$$ObjectBody \triangleq [cnfl{:}\,ConfidLevels, intl{:}\,IntegrLevels].$$

Также были определены вспомогательные операторы и функции для выбора: дочернего элемента объекта (*sc(o)*), множества дочерних элементов объекта (*scs(o)*), множества копий объекта (*scp(o)*, множества дочерних объектов субъекта (*sw(s)*) и обновления владельца субъекта (*UpdateOwner(sh,sp)*).

3.3 Инициализация начальных значений

Множество текущих доступов *A* на этапе инициализации является пустым. Множество объектов *O* также может быть пустым на этапе инициализации. Однако

множество субъектов S при этом, обязательно должно содержать хотя бы один субъект. Для примера значения множеств субъектов и объектов инициализированы модельными значениями:

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

где *oid* – идентификатор объекта, *meta* – метаинформация объекта, *body* содержимое объекта, *cnfl* – уровень конфиденциальности, *intl* – уровень целостности, *cat* категория, *st* – состояние объекта, *owner* владелец, *incl* – множество вложенных документов, *grantm* – права доступа назначенные к метаинформации, *grantb* – права доступа назначенные к содержимому объекта, *sid* – идентификатор субъекта.

В модели предусмотрены действия по созданию и удалению субъектов и объектов, поэтому представленные в выражении (2) значения лишь позволяют быстрее моделировать возможные состояния и находить ошибки.

3.4 Предикаты действий

В модели были определены следующие возможные действия:

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

где *CreateSubjectD* создание субъекта, *DeleteSubjectD* удаление субъекта, *readD* чтение, *writeD* запись, *AppendWD* дозапись («слепая» запись), *CreateOjectD* создание объекта, *DeleteOjectD* удаление объекта, *GrantRightsD* назначение прав доступа, *RemoveRightsD* удаление прав доступа, *IncludeOjectD* вложение объекта в объект, *ExcludeOjectD* исключение вложенного объекта, *ApproveOjectD* утверждение объекта (документа), *ArchiveOjectD* отправка объекта (документа) в архив, *CancelOjectD* отмена действия утвержденного объекта (документа), *CopyOjectD* копирование объекта (документа).

На примере действия *ReadD* представленного в выражении (3), рассмотрим порядок задания необходимых пред- и постусловий. Предусловиями являются предикаты, выполнение которых необходимо для обеспечения безопасности выполнения действия. Постусловия определяют как по результатам выполнения действия изменяются переменные модели.

Действие *Read (s, o, r, op)* осуществляется субъектом в отношении объекта *r=«read»* с правом и определенной составляющей документа – метаинформации (*op=«meta»*) или его содержимого (*op=«body»*). Само действие в рамках модели определяет следующие постусловия добавляется текущий доступ ко множеству доступов $(A' = A \cup \{\langle s.sid, o.oid, r, op\rangle\})$ и остаются неизменными множества субъектов и объектов $\text{UNCHANGED}\langle S, O\rangle)$.

Действие предъявляет требования по учету всех возможных состояний модели $(\exists r \in R : \exists s \in S : \exists o \in O : \exists op \in ObjectParts)$ и проверке необходимых предикатов безопасности выполнения действия (предусловий).

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

Ключевыми являются проверки меток конфиденциальности ($s.cnfl \geq o.meta.cnfl$ и $s.cnfl \geq o.body.cnfl$), а также категорий как составляющей мандатного контроля конфиденциальности ($o.cat \subseteq s.cat$). Затем производится проверка условия наличия права доступа у субъекта к объекту в зависимости от $\langle ssid, r\rangle \subseteq o.grantm$ или $\langle ssid, r\rangle \subseteq o.grantb$).

Также возможна ситуация, когда субъект является владельцем объекта o , в этом случае требование по наличию права во множестве прав доступа к объекту не предъявляется владелец обладает полными правами (*o.owner=s.sid*).

Рассмотрим также действие по создании копии объекта *CopyObjectD* выражение (4).

$$
\begin{aligned}
&CopyObject(s,o,id) \triangleq \wedge\, O' \cup \{[oid \mapsto id,\\
&\qquad meta \mapsto [cnfl \mapsto o.meta.cnfl,\\
&\qquad intl \mapsto o.meta.intl],\\
&\qquad body \mapsto [cnfl \mapsto o.body.cnfl,\\
&\qquad owner \mapsto s.sid,\\
&\qquad grantm \mapsto o.grantm,\\
&\qquad grantb \mapsto o.grantb,\\
&\qquad cat \mapsto o.cat,\\
&\qquad incl \mapsto o.incl,\\
&\qquad st \mapsto \text{"approved"},\\
&\qquad copy \mapsto \{o.oid\}]\}\\
&\qquad \wedge A' = A \cup \{\langle s.sid, o.oid, \text{"copy"}, id\rangle\},\\
&\qquad \wedge UNCHANGED\langle S\rangle\\
&CopyObject \triangleq\\
&\quad \exists s \in S:\\
&\quad \wedge O \neq \{\},\\
&\quad \wedge \exists\, o \in O: \wedge o.owner = s.sid\\
&\qquad \wedge o.incl = \{\}\\
&\qquad \wedge o.st = \text{"approved"}\\
&\qquad \wedge s.cat \subseteq o.cat\\
&\qquad \wedge s.cnfl = o.meta.cnfl\\
&\qquad \wedge s.intl \geq o.meta.intl\\
&\qquad \wedge s.cnfl = o.body.cnfl\\
&\qquad \wedge s.intl \geq o.body.intl\\
&\qquad \wedge Cardinality(scp(o)) < 2\\
&\qquad \wedge \exists id \in ObjectIDs:\\
&\qquad\quad \wedge \forall oo \in O:\\
&\qquad\qquad \wedge id \neq oo.oid\\
&\qquad\qquad \wedge CopyObject(s,o,id).
\end{aligned}
\tag{4}
$$

Действие *CopyObject(s,o,id)* осуществляется субъектом в отношении объекта *o*. В качестве параметра также передается идентификатор для создаваемого объекта *id*, выбор которого осуществляется с учетом требования по уникальности идентификаторов всех объектов ($\exists id \in ObjectIDs : \forall oo \in O : id \neq oo.oid$). В рамках указанного действия производится добавление нового объекта ко множеству объектов, обладающего атрибутами исходного объекта *o*, за исключением поля *copy*, в котором указывается копией какого объекта является данный объект. Также производится добавление текущего доступа ко множеству доступов *A* и указание на то, что множество субъектов при выполнении этого действия не изменяется.

Действие *CopyObjectD* предъявляет требования по учету всех возможных состояний модели ($\exists s \in S : O \neq \{\} \wedge \exists o \in O$) и проверке необходимых предусловий:

– субъект является владельцем объекта (*o.owner=s.sid*);

– объект не имеет вложенных объектов (*o.incl={}*);

– объект находится в состоянии «утвержден» (*o.st=«approved»*);

категории субъекта являются подмножеством категорий объекта ($s.cat \subseteq o.cat$);

– уровень конфиденциальности субъекта равен уровню конфиденциальности объекта ($s.cnfl = o.meta.cnfl \wedge s.cnfl = o.body.cnfl$);

– уровень целостности субъекта больше или равен уровню целостности объекта ($s.intl \geq o.meta.intl \wedge s.intl \geq o.body.intl$);

– число копий объекта не превышает указанного значения ($Cardinality(scp(o)) < 2$);

– создаваемый объект должен иметь уникальный идентификатор.

3.5 Инварианты модели

Помимо указания пред- и постусловий в модели имеется возможность задания инвариантов глобальных свойств, выполнение которых обязательно для всех состояний модели.

Базовым и общепринятым инвариантом является инвариант типов выражение (5).

$$
\begin{aligned}
&ObjTypeInv \triangleq\\
&\qquad \wedge \forall o \in O: \wedge o.oid \in ObjectIDs\\
&\qquad \wedge o.meta \in ObjectMeta\\
&\qquad \wedge o.body \in ObjectBody\\
&\qquad \wedge o.owner \in SbjectIDs\\
&\qquad \wedge \{o.incl\} \subseteq \text{SUBSET } ObjectIDs\\
&\qquad \wedge \{o.copy\} \subseteq \text{SUBSET } ObjectIDs\\
&\qquad \wedge o.st \in ObjectStates\\
&\qquad o.cat \subseteq \text{SUBSET } Categories\\
&TypeInv \triangleq \wedge S \subseteq Subjects\\
&\qquad \wedge ObjTypeInv\\
&\qquad \wedge \forall sn \in S: \text{IF } \exists sm \in S: \wedge sm \neq sn\\
&\qquad\qquad\qquad\qquad \wedge sn.sin = sm.sid\\
&\qquad\qquad \text{THEN FALSE}\\
&\qquad\qquad \text{ELSE TRUE}\\
&\qquad \wedge \forall on \in O: \text{IF } \exists om \in O: \wedge om \neq on\\
&\qquad\qquad\qquad\qquad \wedge on.oin = om.oid\\
&\qquad\qquad \text{THEN FALSE}\\
&\qquad\qquad \text{ELSE TRUE}
\end{aligned}
\tag{5}
$$

Он включает в себя проверку соответствия всех полей для всех объектов, также проверку соответствия типу всех субъектов и проверку уникальности всех идентификаторов субъектов и объектов.

Вторым заданным в модели инвариантом является инвариант безопасности выражение (6). Он осуществляет проверку следующих условий:

– уровень конфиденциальности метаинформации объекта не превышает уровня конфиденциальности содержимого объекта ($o.meta.cnfl \leq o.body.cnfl$);

– уровень целостности метаинформации объекта равен уровню целостности содержимого объекта ($o.meta.intl = o.body.intl$);

– если объект содержит вложенный объект, то множество прав доступа к родительскому объекту является подмножеством прав доступа к дочернему объекту и состояние родительского объекта равно состоянию дочернего ($o.grantm \subseteq oi.grantm \wedge o.grantb \subseteq oi.grantb \wedge o.st = oi.st$);

– число вложенных объектов не превышает указанного значения ($Cardinality(scs(o)) \leq 1$);

– число копий объекта не превышает указанного значения ($Cardinality(scp(o)) \leq 2$);

– если субъект является владельцем объекта, права для него не назначаются, так как он обладает полными правами ($\neg o.grantm \subseteq (s.sid \times Rights) \wedge \neg o.grantb \subseteq (s.sid \times Rights)$);

– если объект находится в состоянии «помещен в архив» или «отменен», то запрещено назначать право «записи» кому-либо из субъектов ($o.grantm \cap (SubjectIDs \times \{\text{"write"}\}) \neq \{\} \vee o.grantb \cap (SubjectIDs \times \{\text{"write"}\}) \neq \{\}$).

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

Выполнение инвариантов для всех состояний модели обеспечивает доказательство следующей теоремы (7) относительно спецификации модели (1) и инвариантов (5), (6):

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

3.6 Верификация модели

Существенным ограничением подхода к верификации на основе метода ModelChecking является необходимость проверки всех возможных состояний модели. То есть при задании каких-либо условий для счетных множеств, например $sid \in Nat$ или $oid \in Nat$, процесс верификации не завершится, так как количество состояний модели также будет счетным. Поэтому в спецификации использовались модельные значения для сокращения времени, требуемого на верификацию модели.

Верификация разработанной модели производилась с помощью инструментального средства TLC2 версии 2.13 [25]. При этом время, затраченное на верификацию, составило порядка 2835 минут (более 47 часов) на сервере с операционной системой Ubuntu 16.04, 24 ядрами Intel Xeon E5-2620 v2 2.10 ГГц и 32 Гб оперативной памяти. Было проверено 16 284 800 554 состояний при средней производительности системы 5 743 616 состояний в минуту.

4. Выводы

Разработанная модель может применяться для задания политики управления доступом к разнокатегорийный ресурсам компьютерных систем, где циркулирует информация различных категорий конфиденциальности, а также различных уровней конфиденциальности и целостности.

Примененная в модели нотация языка TLA+ обладает достаточной гибкостью и выразительными возможностями для решения широкого круга задач по моделированию в сфере компьютерной безопасности.

Следует отметить, что требования по отсутствию скрытых каналов по времени в данной модели учтены не были, что является направлением дальнейших исследований.

Рецензент: *Цирлов Валентин Леонидович, кандидат технических наук, доцент Московского государственного технического университета им. Н.Э. Баумана, г. Москва, Россия. E-mail: v.tsirlov@bmstu.ru*

Литература

1. Девянин П. Н. О проблеме представления формальной модели политики безопасности операционных систем // Труды ИСП РАН. 2017. Т. 29, № 3. С. 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 // Прикладная дискретная математика. Приложение. 2017. № 10. С 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 // Прикладная дискретная математика. Приложение. 2013. № 4. С. 22–40.
4. Девянин П. Н. Необходимые условия нарушения безопасности информационных потоков по времени в рамках МРОСЛ ДП-модели // Прикладная дискретная математика. Приложение. 2015. № 8. С. 81–83. DOI:10.17223/2226308X/8/30.
5. Девянин П. Н. О результатах формирования иерархического представления МРОСЛ ДП-модели // Прикладная дискретная математика. Приложение. 2016. № 9. С. 83–87. DOI: 10.17223/2226308X/9/32.
6. Девянин П. Н. Уровень запрещающих ролей иерархического представления МРОСЛ ДП-модели // Прикладная дискретная математика. 2018. № 39. С. 58–71. DOI: 10.17223/20710410/39/5.
7. Моделирование и верификация политик безопасности управления доступом в операционных системах / П. Н. Девянин [и др.]. – Институт системного программирования им. В. П. Иванникова РАН, 2018. 181 с. URL: http://www.ispras.ru/publications/security_policy_modeling_and_verification.pdf.
8. Jackson D. Software Abstractions: Logic, Language, and Analysis. The MIT Press, 2012. 376 pp.
9. A Comparison of the Declarative Modelling Languages B, Dash, and TLA+ / A. Abbassi [et al.] // 2018 IEEE 8th International Model-Driven Requirements Engineering Workshop (MoDRE). – IEEE. 2018. – P. 11–20.
10. Boiten E. Modeling in Event-B System and Software Engineering Abrial Jean-Raymond Cambridge University Press, May 2010 ISBN-10:0521895561 // Journal of Functional Programming. 2012. Vol. 22, no. 2. P. 217–219.
11. Comparison of specification decomposition methods in Event-B /P. N. Devyanin

[et al.] // Programming and Computer Software. 2016. July. Vol. 42, no. 4. P. 198-205. DOI: 10.1134/S0361768816040022. URL: https://doi.org/10.1134/S0361768816040022.
12. Fitzgerald J., Larsen P. G. Modelling Systems: Practical Tools and Techniques in Software Development. – 2nd. – New York, NY, USA : Cambridge University Press, 2009. – 304 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 // Proceedings of International Conference on ICT for Sustainable Development. Singapore: Springer Singapore, 2016. P. 25–38. DOI: 10.1007/978-981-10-0135-2_3.
14. Lamport L. The Temporal Logic of Actions // ACM Trans. Program. Lang. Syst. 1994. Vol. 16, no. 3. P. 872–923. DOI: 10.1145/177492.177726. URL: http://doi.acm.org/10.1145/ 177492.177726.
15. Lamport L. Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley, 2002.
16. Hansen D., Leuschel M. Translating B to TLA+ for validation with TLC // Science of Computer Programming. – 2016. – Vol. 131. – P. 109–125.
17. Wayne H. Practical TLA+ Planning Driven Development. – Apress, 2018. – 332 pp. – DOI: 10.1007/978-1-4842-3829-5.
18. Девянин П. Н. Условия безопасности информационных потоков по памяти в рамках МРОСЛ ДП-модели // Прикладная дискретная математика. Приложение. 2014. № 7. С. 82–85.
19. TLA + Proofs / D. Cousineau [et al.] // FM 2012: Formal Methods - 18th International Symposium, Paris, France, August 27-31, 2012. Proceedings. 2012. P. 147–154. DOI: 10.1007/978-3-642-32759-9_14. URL: https://doi.org/10.1007/978-3-642-32759-9%5C_14.
20. Wayne H. PlusCal // Practical TLA+. – Springer, 2018. – P. 23–42.
21. Merz S., Vanzetto H. Encoding TLA+ into Many-Sorted First-Order Logic // Abstract State Machines, Alloy, B, TLA, VDM, and Z / ed. by M. Butler [et al.]. Cham : Springer International Publishing, 2016. P. 54-69.
22. Gouglidis A., Grompanopoulos C., Mavridou A. Formal Verification of Usage Control Models: A Case Study of UseCON Using TLA+ // Proceedings of the 1st International Workshop on Methods and Tools for Rigorous System Design, MeTRiD@ ETAPS 2018, Thessaloniki, Greece, 15th April 2018. 2018. P. 52-64. DOI: 10.4204/EPTCS.272.5. URL: https://doi.org/10.4204/EPTCS.272.5.
23. McMillan K. L. Eager Abstraction for Symbolic Model Checking // Computer Aided Verification / ed. by G. Chockler Hana and Weissenbacher. Cham : Springer International Publishing, 2018. P. 191-208. DOI: 10.1007/978-3-319-96145-3_11.
24. Storey V. C., Song I.-Y. Big data technologies and Management: What conceptual modeling can do // Data & Knowledge Engineering. 2017. Vol. 108. P. 50-67. DOI: 10.1016/j.datak.2017.01.001. URL: http://www.sciencedirect.com/science/article/pii/ S0169023X17300277.
25. How AmazonWeb Services Uses Formal Methods / C. Newcombe [et al.] // Commun. ACM. New York, NY, USA, 2015. Mar. Vol. 58, no. 4. P. 66-73. DOI: 10.1145/2699417. URL: http://doi.acm.org/10.1145/2699417.

SPECIFICATION OF THE ACCESS CONTROL MODEL FOR COMPUTER SYSTEM RESOURCES OF VARIOUS CATEGORIES

Kozachok A.[1]

Purpose: Developing an access control model for computer system resources of various categories to ensure compliance with the mandatory integrity and confidentiality monitoring requirements, without time accounting of information flows, and its verification for meeting security invariants by the Model Checking method.

Research methods: Modeling using the language of Lamport's temporal logic of actions, as its notation seems to be closest to the generally accepted mathematical one; expressive capabilities and tools help describe and verify systems specified as finite automata in the automatic mode using the Model Checking method.

Results: A model was set up to control access to electronic documents of various categories, which takes into account life cycle features of electronic documents and their handling procedure. The model implements the following operations: subject creating/deleting, reading, writing, adding ("blind" writing), object creating/deleting, assigning/deleting access rights, attaching an object to an object, excluding an attached object, approving an object (document), sending an object (document) to the archive, canceling an approved object (document), copying an object (document). The following invariants were also defined: type checks (include checking compliance of all objects' fields, checking compliance with the subject type and checking the uniqueness of subject/object identifiers)

1 Alexander Kozachok, Ph. D., official for Academy of Federal Guard Service, Oryol, Russia. E-mail: a.kozachok@academ.msk.rsnet.ru

and security checks (include checking confidentiality/integrity marks of interacting subjects and objects, and validity of the rights assignment procedure).

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

References:

1. Devyanin P. N. O probleme predstavleniya formal'noj modeli politiki bezopasnosti operacionnyh system. Trudy ISP RAN [Proceedings of ISP RAS]. 2017. Vol. 29, no 3. Pp. 7-16. DOI: 10.15514/ISPRAS-2017-29(3)-1. [In Russ.]
2. Devyanin P. N. Realizaciya nevyrozhdennoj reshyotki urovnej celostnosti v ramkah ierarhicheskogo predstavleniya MROSL DP-modeli. PDM. Prilozhenie [Applied Discrete Mathematics. Supplement]. 2017. No 10. Pp. 111-114. DOI: 10.17223/2226308X/10/44. [In Russ.]
3. Devyanin P. N. Administrirovanie sistemy v ramkah mandatnoj sushchnostno-rolevoj DP-modeli upravleniya dostupom i informacionnymi potokami v OS semejstva Linux. PDM. Prilozhenie [Applied Discrete Mathematics. Supplement]. 2013. No 4. Pp. 22-40. [In Russ.]
4. Devyanin P. N. Neobhodimye usloviya narusheniya bezopasnosti informacionnyh potokov po vremeni v ramkah MROSL DP-modeli. PDM. Prilozhenie [Applied Discrete Mathematics. Supplement]. 2015. No 8. Pp. 81-83. DOI:10.17223/2226308X/8/30. [In Russ.]
5. Devyanin P. N. O rezul'tatah formirovaniya ierarhicheskogo predstavleniya MROSL DP-modeli. PDM. Prilozhenie [Applied Discrete Mathematics. Supplement]. 2016. No 9. Pp. 83-87. DOI: 10.17223/2226308X/9/32. [In Russ.]
6. Devyanin P. N. Uroven' zapreshchayushchih rolej ierarhicheskogo predstavleniya MROSL DP-modeli. PDM [Applied Discrete Mathematics]. 2018. No 39. Pp. 58-71. DOI: 10.17223/20710410/39/5. [In Russ.]
7. Modelirovanie i verifikaciya politik bezopasnosti upravleniya dostupom v operacionnyh sistemah / P. N. Devyanin [i dr.]. – Institut sistemnogo programmirovaniya im. V.P. Ivannikova RAN, 2018. 181 p. URL: http://www.ispras.ru/publications/security_policy_modeling_and_verification.pdf.. [In Russ.]
8. Jackson D. Software Abstractions: Logic, Language, and Analysis. The MIT Press, 2012. 376 p.
9. A Comparison of the Declarative Modelling Languages B, Dash, and TLA+ / A. Abbassi [et al.]. 2018 IEEE 8th International Model-Driven Requirements Engineering Workshop (MoDRE). – IEEE. 2018. – P. 11–20.
10. Boiten E. Modeling in Event-B{System and Software Engineering Abrial Jean-Raymond Cambridge University Press, May 2010 ISBN-10:0521895561. Journal of Functional Programming. 2012. Vol. 22, no. 2. P. 217-219.
11. Comparison of specification decomposition methods in Event-B /P. N. Devyanin [et al.]. Programming and Computer Software. 2016. July. Vol. 42, no. 4. P. 198-205. DOI: 10.1134/S0361768816040022. URL: https://doi.org/10.1134/S0361768816040022.
12. Fitzgerald J., Larsen P. G. Modelling Systems: Practical Tools and Techniques in Software Development. – 2nd. – New York, NY, USA : Cambridge University Press, 2009. – 304 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. Proceedings of International Conference on ICT for Sustainable Development. Singapore : Springer Singapore, 2016. P. 25-38. DOI: 10.1007/978-981-10-0135-2_3.
14. Lamport L. The Temporal Logic of Actions. ACM Trans. Program. Lang. Syst. 1994. Vol. 16, no. 3. P. 872-923. DOI: 10.1145/177492.177726. URL: http://doi.acm.org/10.1145/177492.177726.
15. Lamport L. Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley, 2002.
16. Hansen D., Leuschel M. Translating B to TLA+ for validation with TLC. Science of Computer Programming. – 2016. – Vol. 131. – P. 109–125.
17. Wayne H. Practical TLA+ Planning Driven Development. – Apress, 2018. – 332 pp. – DOI: 10.1007/978-1-4842-3829-5.
18. Devyanin P. N. Usloviya bezopasnosti informacionnyh potokov po pamyati v ramkah MROSL DP-modeli. PDM. Prilozhenie [Applied Discrete Mathematics. Supplement]. 2014. No 7. Pp. 82-85. [In Russ.]
19. TLA + Proofs / D. Cousineau [et al.] // FM 2012: Formal Methods - 18th International Symposium, Paris, France, August 27-31, 2012. Proceedings. 2012. P. 147-154. DOI: 10.1007/978-3-642-32759-9_14. URL: https://doi.org/10.1007/978-3-642-32759-9%5C_14.
20. Wayne H. PlusCal // Practical TLA+. – Springer, 2018. – P. 23–42.
21. Merz S., Vanzetto H. Encoding TLA+ into Many-Sorted First-Order Logic // Abstract State Machines, Alloy, B, TLA, VDM, and Z / ed. by M. Butler [et al.]. Cham : Springer International Publishing, 2016. P. 54-69.
22. Gouglidis A., Grompanopoulos C., Mavridou A. Formal Verification of Usage Control Models: A Case Study of UseCON Using TLA+ // Proceedings of the 1st International Workshop on Methods and Tools for Rigorous System Design, MeTRiD@ETAPS 2018, Thessaloniki, Greece, 15th April 2018. 2018. P. 52-64. DOI: 10.4204/EPTCS.272.5. URL: https://doi.org/10.4204/EPTCS.272.5.
23. McMillan K. L. Eager Abstraction for Symbolic Model Checking // Computer Aided Verification / ed. by G. Chockler Hana and Weissenbacher. Cham : Springer International Publishing, 2018. P. 191-208. DOI: 10.1007/978-3-319-96145-3_11.
24. Storey V. C., Song I.-Y. Big data technologies and Management: What conceptual modeling can do // Data & Knowledge Engineering. 2017. Vol. 108. P. 50-67. DOI: 10.1016/j.datak.2017.01.001. URL: http://www.sciencedirect.com/science/article/pii/S0169023X17300277.
25. How AmazonWeb Services Uses Formal Methods / C. Newcombe [et al.] // Commun. ACM. New York, NY, USA, 2015. Mar. Vol. 58, no. 4. P. 66-73. DOI: 10.1145/2699417. URL: http://doi.acm.org/10.1145/2699417.