UDC 681.3.06
DOI: 10.15587/2706-5448.2021.229539
Article type «Reports on Research Projects»

Victor Zhora,
Oleksandr Synetskyi

USE OF THE PVS FORMAL LOGIC SYSTEM IN THE METHOD OF FORMAL PROOF OF SECURITY IN THE CONSTRUCTION OF INFORMATION SECURITY SYSTEMS

The object of research is the information and telecommunication system (ITS) and ensuring the protection of information stored, processed and circulating in it. One of the most problematic areas in the creation of secure ITS is the logical inconsistency and incompleteness of the information security policy. That is, a set of laws, rules, restrictions, recommendations, etc., which regulate the procedure for processing information and are aimed at protecting information from a certain set of threats. The reason for such problems is usually the absence of pre-design modeling of the information security system as a component of the information and telecommunications system, which in the end causes the latter to be vulnerable.

An important prerequisite for the creation of a secure ITS is the construction of a subject-object model of the system, which makes it possible to determine the connections between objects, their features, to model information flows and types of access to information and infrastructure resources. According to the existing clear, complete and consistent subject-object model of the ITS, it becomes possible to apply mathematical methods to modeling the processes of its functioning, including for solving the problem of formal proof of security.

The paper considers the main idea of the method of formal proof of security, which can be used when building information security systems or assessing the security of the created information and telecommunications system. It is shown that for its implementation it is possible to use the methodology of automatic theorem proving. One of the ways to solve this problem, which is proposed in the work, is the use of the PVS (Prototype Verification System) formal logic system, which is widely used for writing specifications and constructing proofs. The main components of this system are considered, as well as the possibilities of its use for automatic proof of statements about the impossibility of unauthorized access under the conditions of a certain security policy. An example of the use of the PVS system for the formal proof of the security of the system in the framework of the Bella-LaPadula security policy is given.

Keywords: *information and telecommunication system, information security system, security policy, method of formal proof of security, specification language.*

Received date: 16.12.2020
Accepted date: 22.02.2021
Published date: 30.04.2021

How to cite
Zhora, V., Synetskyi, O. (2021). Use of the PVS formal logic system in the method of formal proof of security in the construction of information security systems. Technology Audit and Production Reserves, 2 (2 (58)), 41–45. doi: http://doi.org/10.15587/2706-5448.2021.229539

1. Introduction

The issues of protecting information and telecommunication systems (hereinafter – ITS) from threats implemented by attacks and/or activation of destabilizing factors [1] has not lost its relevance over the past decades. A significant set of algorithms for the implementation of threats is associated with the exploitation of vulnerabilities that can arise both in the configuration of information protection mechanisms in the ITS, and in the ITS security policy. Let's understand the security policy as a set of laws, rules, restrictions, recommendations, etc., which regulate the procedure for processing information and aimed at protecting information from a certain set of threats.

The mandatory elements of the information protection system in the ITS are security policy and a set of hardware and software components for information protection, the use of which is conditioned and regulated by the security policy.

The likelihood of the implementation of threats in the ITS in the opposite way depends on the completeness, correctness and reliability of the information protection system, which can be formally verified before the attack or the emergence of a destabilizing factor. The main stages of the formal method of checking the information security system for completeness and correctness are:

1) definition of objects and objectives of protection;
2) development of a security policy;

ISSN 2664-9969

3) evidence that, subject to the security policy, the implementation of threats is impossible;

4) definition of a set of security functions (services) to support the security policy;

5) evidence that the set of security functions (services) enforces the security policy.

The basis of the proofs, which are carried out in sections 3 and 5, is usually a set of theorems. However, carrying out such an analysis for each system is difficult, expensive, and requires highly qualified specialists.

Therefore, the issue of simplifying this procedure is relevant, especially in the context of widespread industrial use of ITS modeling and the above-mentioned approach to formal proof of security. One of the possibilities for solving this problem is the use of means of automating theorem proving, the theoretical possibility of which has long been known [2, 3]. Automatic proof of the statement about the impossibility of unauthorized access to the ITS would greatly facilitate the process of building information security systems.

There are various approaches to the implementation of automatic theorem proving, among which the PVS system of formal logic is popular and widespread [4–6].

Thus, *the object of the study* is an information and telecommunication system and ensuring the protection of information stored, processed and circulated in it. *The purpose of the work* is to demonstrate the applicability of the PVS system within the framework of the method of formal proof of ITS security.

2. Methods of research

To model ITS, the object-subject approach is often used, the essence of which is to select in the ITS sets of passive objects O and subjects S (which, in turn, includes a set of process objects O_p and users U). After the formation of these sets, connections are established between them and the rules of the security policy are formulated.

Let a security policy P be given. Let's formulate the following statement [7]: a protection system is considered effective (good) if it reliably supports policy P, and ineffective (bad) if it does not reliably support a certain policy P. However, it is not defined here what a reliable policy support. To clarify this definition, we use a hierarchical scheme.

Let the policy P be expressed in some language M_1, the formulas of which are defined in terms of the services $U_1,...,U_k$. For simplicity, let's split the set of all subjects of the system S of the system into two subsets S_1 and S_2, and $S_1 \cup S_2 = S$, $S_1 \cap S_2 = \varnothing$. Let also all objects that can be accessed be divided into two classes O_1 and O_2, and $O_1 \cup O_2 = O$, $O_1 \cap O_2 = \varnothing$.

Within the framework of such a distribution, it is easy to formulate the following simple security policy P: subject S can have access $\alpha \in R$ to the object O if and only if $S \in S_i$, $O \in O_i$, $i = 1, 2$. For each request from subject S to access to an object in the security system must be able to calculate membership functions:

$$I_x(A) = \begin{cases} 1, x \in A, \\ 0, x \notin A, \end{cases}$$

for all subjects and objects: $I_s(S_1)$, $I_s(S_2)$, $I_o(O_1)$, $I_o(O_2)$. Then the Boolean expression is evaluated:

$$(I_s(S_1) \wedge I_o(O_1)) \vee (I_s(S_2) \wedge I_o(O_2)).$$

If the value received is 1 (true), then access is allowed. If 0 (false), then – unresolved. Thus, it is clear that the language M_1 language, in which the security policy P is expressed, relies on the services:

– calculation of membership functions $I_x(A)$;
– calculation of a logical expression;
– calculation of the operator «if x=l, then S gets access to B, if x=0, then it does not».

To support services in the M_1 language, another M_2 language is needed, in which the basic expressions for the provision of services to the upper-level language are defined. It may happen that the M_2 functions need to be implemented relying on the M_3 language of a lower level, etc.

Suppose we can guarantee the services described in the M_2 language. Then the reliability of the policy P is determined by the completeness of its description in terms of services $U_1,...,U_k$. If model P is formal, that is, language M_1 formally defines the rules of policy P, then it is possible to prove or disprove the assertion that the set of services provided completely and unambiguously determines policy P. This means that guarantees of the performance of these services are equivalent to guarantees of compliance with the policy. This, in turn, means that the more difficult task is reduced to a simple one or to proving that these services are sufficient to fulfill the policy. All this provides a proof of protection from the point of view of mathematics, or a guarantee in terms of confidence in the support of the policy from the side of simpler functions.

3. Research results and discussion

PVS (Prototype Verification System) is an automated system for the construction of a formal specification and its verification, the main components of which are the specification language and the theorem proving subsystem [8]. In addition, PVS contains syntax checkers, type correctness, parsers, and several predefined theories. Let's consider each component of PVS in more detail.

The base of the PVS specification language is the classical typed logic of the first order [9, 10]. The main types of this logic are:

– uninterpretations that can be specified by the user;
– built-in (e. g. real numbers and Booleans);
– are interpreted to allow constraints on uninterpretations or built-in types, in particular by means of predicates.

Predicative subtypes are dependent types that can be applied to introduce additional constraints, such as defining the type of prime numbers as a subset of integers. Such restricted types can create additional proof assertions (called Type-Correctness Conditions, or TCCs), but they greatly increase the expressiveness and clarity of specifications. However, in practice, most TCCs are accounted for automatically. Specifications in PVS are grouped into parameterized theories, they can contain assumptions, definitions, axioms and theorems. Each theory contains a series of statements called declarations. They define the types, constants, variables, axioms, and formulas that will be used to prove theorems. Theories can be reused, and some of the standard ones are included in PVS. They allow to use bit vectors, lambda calculus, graphs, and the like in specifications.

The PVS theorem proving subsystem is a set of elementary inference rules applied by the user interactively within the sequent calculus. These rules can be combined

into a proof strategy. The proof subsystem also allows proofs to be restarted and checks all conditions (e. g. TCC). Examples of elementary PVS withdrawal rules:

– propositional rules – for example, the antecedent rule;
– quantifier rules, allowing, in particular, to replace variables in the existence quantifier with terms;
– equality rules, such as replacing one part of an identity with another.

Other rules allow lemmas, axioms, type constraints, and the like to be introduced.

The proof of the theorem begins with a conclusion and consists in applying the rules chosen by the user one by one to divide the proof into subgoals. The process is repeated until the subgoals become trivial.

Let's consider an example of PVS application for formal proof of security within the Bell-LaPadula policy [11]. The Bella-LaPadula policy is a type of mandatory security policy, its basis is mandatory access control, based on the following principles:

– all subjects and objects of the system must be uniquely identified;
– a linearly ordered set of security labels must be specified;
– each object of the system is assigned a secrecy label, which determines the value of the information contained in it;
– each subject of the system is assigned a secrecy label, which determines the level of confidence in him in the ITS, that is, the maximum value of the secrecy label of objects to which the subject has access; the security label of a subject is also called its access level.

The main goal of the Bella-LaPadula policy is to prevent information leakage from objects with a high level of access to objects with a low level of access, that is, to counter the emergence of information channels in the ITS from top to bottom. For this, within the framework of the policy, two rules of access control are formulated:

1. Simple security property: a subject can read information from an object if and only if the subject's access level is not lower than the object's secrecy level.

2. *-property: a subject can record information to an object if and only if the subject's access level is not higher than the object's secrecy level.

Let's formalize the above reasoning.

Let the system define a set of objects O and subjects S, $S \subset O$ set, a set of types of access $R = \{r, w\}$ (here r – write access, w – read) and security levels $L = \{U, SU, S, TS\}$. On the set L, let's introduce the relation «$\le$» and the operators of the smallest upper and lowest boundaries «$\bullet$» and «$\otimes$». The «$\le$» relation has the following properties:

1) reflexivity:

$$\forall a \in L : a \le a.$$

It means the possibility of transmitting information about the same level;

2) antisymmetry:

$$\forall a_1, a_2 \in L : \left((a_1 \le a_2) \wedge (a_2 \le a_1)\right) \to (a_1 = a_2);$$

3) transitivity:

$$\forall a_1, a_2, a_3 \in L : \left((a_1 \le a_2) \wedge (a_2 \le a_3)\right) \to (a_1 \le a_3).$$

Thus, the relation «$\le$» is a loose order relation.

Operators of the smallest upper and lower bounds «$\bullet$» and «$\otimes$» are defined as follows:

$$a = a_1 \cdot a_2 \leftrightarrow (a_1, a_2 \le a) \wedge$$
$$\wedge \left(\forall a_3 \in L : (a_3 \le a) \to \left((a_3 \le a_1) \wedge (a_3 \le a_2)\right)\right);$$

$$a = a_1 \otimes a_2 \leftrightarrow (a \le a_1, a_2) \wedge$$
$$\wedge \left(\forall a_3 \in L : \left((a_3 \le a_1) \wedge (a_3 \le a_2)\right) \to (a_3 \le a)\right).$$

Then the structure $\Lambda = \{L, \le, \bullet, \otimes\}$ is called a lattice.

Let's also set a function $C : S \cup O \to L$ that for each object of the system define the corresponding privacy label and the Request: $S \cup O \cup R \to \{0,1\}$ function, which will show whether the subject is allowed or denied access to the object by a certain method.

Now, within the framework of the above concepts, let's formulate the specified access control rules:

1. Simple property of safety:

$$\forall s \in S, o \in O : \ Request(s, o, r) = 1 \Leftrightarrow \left(C(s) \le C(o)\right).$$

2. *-property:

$$\forall s \in S, o \in O : \ Request(s, o, w) = 1 \Leftrightarrow \left(C(o) \le C(s)\right).$$

As it is easy to see, the theorem about the impossibility of leakage of confidential information will have the form:

$$\forall s \in S, o \in O : \neg\left(C(s) \le C(o)\right) \to Request(s, o, r) = 0.$$

Let's formulate the above provisions in the PVS specification language.

First, let's define the sets of objects, subjects, access types and security labels as follows:

```
Object: TYPE.
Subject: TYPE FROM Object.
Access: TYPE={r, w}.
Label: TYPE={U, SU, S, TS}.
```

Let's introduce the mappings *C* and *Request*, which describe, respectively, the value function and the request for access:

```
C: [Object->Label].
Request: [Subject, Object, Access->bool].
```

Taking into account the above, the rules for differentiating access will take the form:

```
simple_property: AXIOM FORALL obj,
subj: (Request(subj, obj, r)=TRUE)
<=>(C(obj)<=C(subj)).
star_property: AXIOM FORALL obj,
subj: (Request(subj, obj, w)=TRUE)
<=>(C(subj)<=C(obj)).
```

Finally, the theorem on the impossibility of leakage of confidential information can be formulated as follows:

```
unauthorized_access_impossibility:
THEOREM FORALL obj,
subj: (NOT (C(obj)<=C(subj)))
=>(NOT (Request(subj, obj, r)=TRUE)).
```

ISSN 2664-9969

Now let's look at an automated proof of this theorem using PVS tools:

```
|-------
{1} FORALL obj, subj:

(NOT (C(obj)<=C(subj)))
=>(NOT (Request(subj, obj, r)=TRUE)).
```

Let us reduce the formula to Skolemi normal form, depriving it of the universal quantifier due to the introduction of the constants «obj!1» and «subj!1». To do this, apply the rule (skolem!). There is:

```
|-------

{1} (NOT (C(obj!1)<=C(subj!1)))
=>(NOT Request(subj!1, obj!1, r)).
```

Let's add the «simple_property» axiom to the antecedent, which is a formal expression of a simple security property in the Bell-LaPadula model.

As a result, let's obtain:

```
{–1} FORALL (obj, subj):

(Request(subj, obj, r)=TRUE)
<=>(C(obj)<=C(subj))
|-------

[1] (NOT (C(obj!1)<=C(subj!1)))
=>(NOT Request(subj!1, obj!1, r)).
```

Now, to remove the added axiom from the quantifier, replace the quantized variables with the constants «obj!1» and «subj!1», thus achieving syntactic correspondence to the consequent formula:

```
{–1} Request(subj!1, obj!1,
r)<=>(C(obj!1)<=C(subj!1))
|-------
[1] (NOT (C(obj!1)<=C(subj!1)))
=>(NOT Request(subj!1, obj!1, r)).
```

Let's rewrite the formula of the antecedent form $A \leftrightarrow B$ as $(A \rightarrow B) \wedge (B \rightarrow A)$:

```
{–1} Request(subj!1, obj!1, r)
IMPLIES (C(obj!1)<=C(subj!1))
{–2} (C(obj!1)<=C(subj!1))
IMPLIES Request(subj!1, obj!1, r)
|-------

[1] (NOT (C(obj!1)<=C(subj!1)))
=>(NOT Request(subj!1, obj!1, r)).
```

Let's divide the antecedent formula, denoted as [–1], into clauses, reducing it in form $A \rightarrow B$ to $\neg B \vee A$:

```
[–1] Request(subj!1, obj!1, r)
IMPLIES (C(obj!1)<=C(subj!1))

[–2] (C(obj!1)<=C(subj!1))
IMPLIES Request(subj!1, obj!1, r)

{–3} Request(subj!1, obj!1, r)
|-------
{1} (C(obj!1)<=C(subj!1))
```

After the previous step, the proof of the theorem is split into two sub-theorems, each of which can be performed using the inference rule $\frac{\Gamma \mapsto \phi,\ \phi \in \Gamma}{}$:

```
1:
{–1} (C(obj!1)<=C(subj!1))

[–2] (C(obj!1)<=C(subj!1))
IMPLIES Request(subj!1, obj!1, r)

[–3] Request(subj!1, obj!1, r)
|-------

[1] (C(obj!1)<=C(subj!1))

2:
[–1] (C(obj!1)<=C(subj!1))
IMPLIES Request(subj!1, obj!1, r)

[–2] Request(subj!1, obj!1, r)

|-------

{1} Request(subj!1, obj!1, r)

[2] (C(obj!1)<=C(subj!1))
```

Having completed both sub-theorems, let's show the truth of the original theorem on the impossibility of leakage of confidential information in the Bell-LaPadula model.

4. Conclusions

The possibility of modeling and formal proof of the security of the ITS, in which the Bella-LaPadula security policy is implemented using PVS, has been demonstrated.

The use of the method of formal proof of security in the construction of secure ITS makes it possible to minimize the number and significance of possible errors in design due to in-depth analysis of the security of the system using the example of the formal model of ITS at the early stages of its creation. In addition, this method can also be applied to check the adequacy of an already built information security system.

The combination of the method of formal proof of security with formal logic systems, in particular PVS, makes it possible to automate the process of bringing security, reducing the task to correct modeling of the system in terms of object-subject interaction and the development of a security policy. The possibility of using previously defined theories in the proof allows to formulate in terms of the specification language the necessary mathematical apparatus for each of the common types of security policies: discretionary, mandated or role-based. Such a device can be further used as a standard module, turning the system simulation into the formation of properties, objects and access types specific for a given ITS.

References

1. Antoniuk, A. A., Zhora, V. V. (2005). Obobshchenye modely uhroz v ynformatsyonno-telekommunykatsyonnoi systeme. *Pravove,*

normatyvne ta metrolohichne zabezpechennia systemy zakhystu informatsii v Ukraini, *11*, 50–54.
2. Glushkov, V. M. (1980). *Sistema avtomatizatsii dokazatelstv. Avtomatizatsiia obrabotki matematicheskikh tekstov*. Kyiv: IK AN USSR, 3–30.
3. Bishop, M. (2018). *Computer Security Art and Science*. Addison Wesley Professional, 2865.
4. Owre, S., Shankar, N., Rushby, J. M., Stringer-Calvert, D. W. J. (2020). *PVS System Guide Version 7.1. SRI International*. Available at: https://pvs.csl.sri.com/doc/pvs-system-guide.pdf
5. Owre, S., Shankar, N., Rushby, J. M., Stringer-Calvert, D. W. J. (2020). *PVS Language Reference Version 7.1. SRI International*. Available at: https://pvs.csl.sri.com/doc/pvs-language-reference.pdf
6. Owre, S., Shankar, N., Rushby, J. M., Stringer-Calvert, D. W. J. (2020). *PVS Prover Guide Version 7.1. SRI International*. Available at: https://pvs.csl.sri.com/doc/pvs-prover-guide.pdf
7. Antoniuk, A. O., Zhora, V. V. (2007). Vikoristannia dokazovogo metodu dlia proektuvannia ta otsinki rivnia zakhischenosti informatsiino-telekomunikatsiinoï sistemi. *Problemi programuvannia, 3*, 88–96.
8. Muñoz, C. A., Demasi, R. A.; Meyer, B., Nordio, M. (Eds.) (2012). Advanced Theorem Proving Techniques in PVS and Applications. Tools for Practical Software Verification. LASER 2011. *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 7682. doi: http://doi.org/10.1007/978-3-642-35746-6_4
9. McGuinness, D. L.; Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., Patel-Schneider, P. F. (Eds.) (2010). *The Description Logic Handbook. Theory, Implementation and Applications*. Cambridge: University Press, 624. doi: http://doi.org/10.1017/cbo9780511711787
10. Turhan, A. Y.; Rudolph, S., Gottlob, G., Horrocks, I., van Harmelen, F. (Eds.) (2013). Introductions to Description Logics – A Guided Tour. Reasoning Web. Semantic Technologies for Intelligent Data Access. Reasoning Web 2013. *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 8067. doi: http://doi.org/10.1007/978-3-642-39784-4_3
11. Bell, D. E., La Padula, L. J. (1976). *Secure Computer Systems: Mathematical foundations and model*. Report ESD-TR-73-278. Mitre Corp. Bedford. Available at: http://www-personal.umich.edu/~cja/LPS12b/refs/belllapadula1.pdf

Victor Zhora, *Junior Researcher, Research department No. 11 «Automated Information Systems», Institute of Software Systems of National Academy of Science of Ukraine, Kyiv, Ukraine, ORCID: https://orcid.org/0000-0003-2679-3056, e-mail: victor.zhora@gmail.com*

Oleksandr Synetskyi, *Postgraduate Student, Institute of Software Systems of National Academy of Science of Ukraine, Kyiv, Ukraine, ORCID: https://orcid.org/0000-0002-5672-9269, e-mail: alexander.sinetskiy@gmail.com*