The security protocols analyzer using extensions of SPi-calculus

Sergey S. Seleznev,
e-mail: sermyt@yandex.ru

Alexander S. Mikhailov,
e-mail: almikh@mail.ru

Department of Cybernetics
Moscow Engineering Physics Institute (State University)
31, Kashirskoe shosse, Moscow, 115409, Russia

***Abstract*—This paper is about the tool called Janus for automated analysis of security protocol models. The type checking method that is a modern extension of SPi-calculus was implemented in Janus. The tool is based on windows GUI application. It provides graphical schema of analysis process and textual reports. This paper contains brief description of type checking, sample of protocol modeling and analysis, tool's design and description. We also tried to compare Janus with some analogues.**

***Index Terms* — analysis of security protocols, protocol models, formal methods, SPi-calculus, type checking, software tools**

I. Introduction

Distributed systems that contain security protocols are widely used. A protocol model is a core of distributed system. Usual approach for development of these systems consists of three steps: modeling, implementation and testing; this process does not include protocol model analysis. Models should be analyzed otherwise protocols can be failed. There are several formal algebraic methods for analysis and verifications of security protocols. General idea of methods is to evolve the decision using reduction of models to prove feasibility of specified security properties. This paper is about the protocols analyzer using extensions of SPi-calculus [1, 2]. SPi-calculus [1] is a one of modern formal algebraic calculus.

There are amount of tools for analysis of protocol models. Two of them are most interesting and close to our work: STA (Symbolic Trace Analyzer) [3] and AVISPA (Automated Validation of Internet Security Protocols and Applications) [4]. The STA was created by Michele Boreale and Marzia Buscemi. This tool can be used for analysis of protocol models described in sort of SPi-calculus. STA converts SPi-calculus specified model to the terms of ML-language. Protocols are modeled as systems of concurrent processes. STA analyzes the execution traces of this system to detect possible failures of security properties. The core of STA method is an optimized exploration of the state-space. Currently the tool is provided command-prompt user interface. AVISPA is an international project for creation of universal tool for analysis of security protocols. Protocols can be described using original notation, then they can be analyzed. Variants of tool are provided for Linux and MacOS, WEB-access is allowed also. Project supports the library of examples. We can conclude that both tools are appropriated for researchers who want enough level of autoimmunization. Tools use enumerative reasoning to prove that protocol model is safe (so proving protocol safety can take lot of time); also tools don't have graphical user interface. The tool that we present in this paper provides graphic user interfaces for analysis. Janus works under MS Windows operation system.

II. Type checking method

Here we should say some more about the method Janus is based on. The main idea is a conception of how safe protocol should work. At first one can find moments when some procedures in protocol are starting and ending. For example we can determine moments when authenticity step of protocol is started and finished. Also there are some rules that can be used to check protocol if random numbers are used to check temporal precedence between events.

After applying this method to the part of the protocol model one can know conditions that guaranties this part of model to be safe. We will call such set of conditions for a part of the model as *effect* of this part.

Secure protocol written in SPi notation consists of protocol branches; each branch consists of SPi operations. Type checking analysis starts from the latest operation of each protocol branch; on first step effect of these operations is calculated. On second step effect of last 2 operations of each branch is calculated and so on. If the rest set of effects is empty then protocol is secure. Otherwise some issues are possible.

TABLE I
Some SPi operations used in this paper:

Operation	Description
In C (X)	Input a value from channel X to variable C
Out C N	Put a value of variable N into channel C
Cast N is N'	Set value of N' to N
New N	Set N to random value
Check N is X	Checks if N = X
Begin ProcLabel	Procedure labeled with ProcLabel was started.
End ProcLabel	Procedure labeled with ProcLabel was finished

Rules for analysis of protocol operation:

- Begin and end events
  - If end “label” is analyzed then end “label” effect should be added to the list of effects
  - If begin “label” is analyzed then end “label” effect should be removed from the list of effectse

TABLE II
BEGIN AND END EVENTS

Events	Set of effects
begin L; end L	[]
end L; end L:	[end L, end L]
begin L; end L; end L	[end L]
begin L; begin L; end L; end L:	[]

- Communicating parallel processes
  - If In or Out commands are analyzed then list of effects should not be changed
  - If parallel command '|' is analyzed (for example, '(protocol branch 1)|(protocol branch 1)') then result effect is a sum of effect of protocol branch 1 and effect of protocol branch 2
- Freshness of random numbers
  - If cast operation is analyzed then some effect should be added to the list of effects (effect type depends on operation parameters)
  - If check operation is analyzed then effect is removed and new check effect is added to the list of effects
  - If random operation is analyzed then check effect is removed from the list of effects

III. SIMPLE PROTOCOL MODELING AND ANALYSIS

For example we can take system of two participants: A and B. They are communicated using the simple protocol for start connection. The protocol:

1) Random number N generation. Both A and B know N
2) A sends N to B using channel C
3) B receives some X from channel C
4) If X = N then B ensures he is communicating with A

The SPi-model of the above protocol is following:

1) System A(N) = out C N
2) System B(N) = in C(X); check N is X
3) SYSTEM = new N; (System A(N) | System B(N))

We want proof that an authenticity property of protocol is valid. For this goal we will add 'begin' and 'end' labels. The modified model is following:

1) System A(N) = begin m; out C N
2) System B(N) = in C(X); check N is X; end m
3) SYSTEM = new N; (System A(N) | System B(N))

Then we also need specify types of variables.

TABLE III
TYPES OF VARIABLES

Variable	Type	Description
N	Un	Random type
X	Nonce [end m]	Random number is used for checking of m event
C	Ch((Nonce [end m]))	Channel type
N’	Nonce [end m]	Additional variable for passing data using a typed channel

For analysis we will try to calculate effects of above protocol. There are two sorts of affects are interested:

1) “end m” label – we should meet in previous operators “begin m” or “check X is Y”. In first case the rest set of effects will be empty ([]). In second, if type of Y = Nonce [end m] then effect is changed to [check X]

2) check N – we should ensure variable N is random (by using New operator)

Main protocol analysis steps:

For system A:

out C N' – empty effect ([])
out C N'; cast N is N’ – effect is ([end m])
out C N'; cast N is N’; begin m – empty effect (begin + end = [])

For system B:

end m – effect is ([end m])
end m; check N is X – effect is ([check N])
end m; check N is X; in C (X) – effect is still the same ([check N])

For the whole system:

(System A(N) | System B(N)) - effect is ([check N])
(System A(N) | System B(N)); new N – empty effect ([])

The result effect is empty. Thus we can conclude that described protocol is secure.

IV. ANALYSIS AUTOMATION

Janus consists of four parts. First (and main) part is analyzer core where all protocol model data are stored; next part is analyzer itself; also it contains a module for loading protocol model from text file and a module for painting of protocol tree.

Object-oriented approach was used for developing analyzer core. For example, an abstract protocol operation is represented by abstract class; operation groups are inherited from abstract operation (of course, some fields were added and methods were override needed to represent operation behavior). Each operation is inherited from corresponding operation group. So any protocol model can be represented using this core.

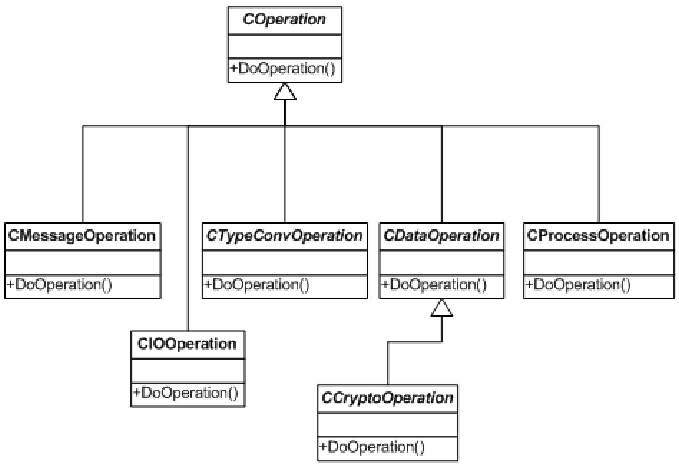


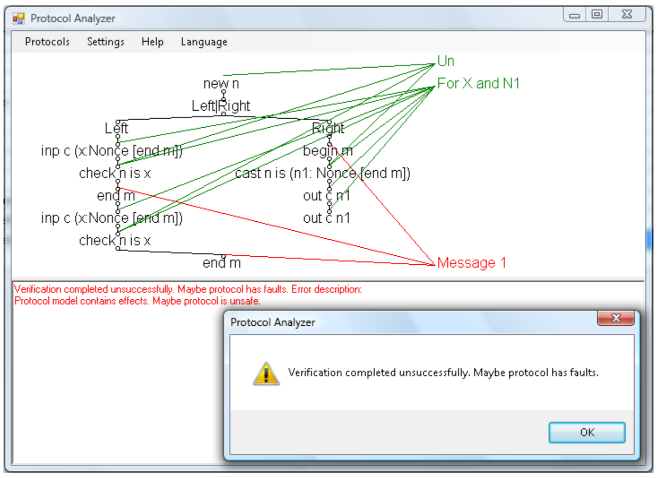
Figure *1. Operation groups*

Protocol analyzer came to be very simple due to considered core structure. One of the problems related to analysis process was branch associating. Protocol analyzer works from the bottom of the protocol tree to it's top, so to calculate effect for «branch» operation one must know effects for all branches, as described in sections II and III. Also some efforts were made to locate place in model which causes model to be unsafe. Protocol analyzer generates messages each time it sees that model structure differs from it's supposition.

Reading absolutely any protocol model from text file was too hard task for this version of the project (because of potential infinity of variable data type description), so model was restricted in some aspects. For example, nested data types and nested branches were not allowed. Reading from file was realized in this limitations.

Protocol model drawing appeared to be very simple task. In analyzer one can paint protocol tree as far as used data types and used begin/end messages.

V. The Analyzer Tool



Verification completed unsuccessfully. Maybe protocol has faults. Error description:
Protocol model contains effects. Maybe protocol is unsafe.

Protocol Analyzer

Verification completed unsuccessfully. Maybe protocol has faults.

Figure 2. Janus user interface

Protocol operations tree is situated at the main form's center. Types list (green) and events list (red) are situated to the right of it. One can disable types and events painting using «Settings» menu item. Protocol tree, types and events are painted just after protocol loads. Log window is situated at the bottom of the form. Analyzer adds messages to this window during analysis process. After analysis finishes a message window appears with information about analysis result.

One can use «Settings» menu to prepare program for advanced users and for newbie. For newbie it is useful to hide log window and disable types and events painting. For advanced users it better to show all messages and enable types and events painting.

VI. Conclusion

Janus tool uses comparatively new approach, based on type checking and graphical representation of analysis process. Janus is MS Windows compatible tool that has user-friendly interface. Proposed approach and tool allow analyzing protocol models quickly because approach doesn't use enumerative reasoning. At the same time modeling of protocols depends on human proficiency. The issue is that sometimes secure model can be wrongly considered as insecure. At the same time an insecure models will be exactly disclosed. We think current version of Janus can be appropriated as additional checking tool of protocol models during their development.

References

[1] Abadi M. Calculus for Cryptographic Protocols: The SPi Calculus. / M. Abadi, A. A.Gordon. // Information and Computation – 1999 – V148(1) – P. 1-70

[2] Gordon A. Authenticity by Typing for Security Protocols. // A. Gordon, A. Jeffrey. //Journal of Computer Security – 2003 – V.11(4) – P 451-520.

[3] Experimenting with STA, a Tool for Automatic Analysis of Security Protocols. M. Boreale, M. Buscemi. A shorter version appears in Proc. of SAC '02, ACM Press, 2002.

[4] The AVISPA project. http://www.avispa-project.org/.