available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

An intruder model with message inspection for model checking security protocols

Stylianos Basagiannis, Panagiotis Katsaros, Andrew Pombortsis*

Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece

ARTICLE INFO

Article history:
Received 13 February 2009
Received in revised form
10 July 2009
Accepted 2 August 2009

Keywords:
Security protocols
Verification
Model checking
Intrusion attacks
Formal analysis

ABSTRACT

Model checking security protocols is based on an intruder model that represents the eavesdropping or interception of the exchanged messages, while at the same time performs attack actions against the ongoing protocol session(s). Any attempt to enumerate all messages that can be deduced by the intruder and the possible actions in all protocol steps results in an enormous branching of the model's state-space. In current work, we introduce a new intruder model that can be exploited for state-space reduction, optionally in combination with known techniques, such as partial order and symmetry reduction. The proposed intruder modeling approach called Message Inspection (MI) is based on enhancing the intruder's knowledge with metadata for the exchanged messages. In a preliminary simulation run, the intruder tags the analyzed messages with protocol-specific values for a set of predefined parameters. This metadata is used to identify possible attack actions, for which it is a priori known that they cannot cause a security violation. The MI algorithm selects attack actions that can be discarded, from an open-ended base of primitive attack actions. Thus, model checking focuses only on attack actions that may disclose a security violation. The most interesting consequence is a non-negligible state-space pruning, but at the same time our approach also allows customizing the behavior of the intruder model, in order e.g. to make it appropriate for model checking problems that involve liveness. We provide experimental results obtained with the SPIN model checker, for the Needham–Schroeder security protocol.

1. Introduction

Analyses of existing cryptographic protocols have shown that even when cryptographic primitives are considered perfectly secure (e.g. perfect encryption by key-based cryptographic schemes, infeasible inversion of hash functions, nonce values that cannot be predicted) the protocol itself may have flaws, which can be exploited by an intruder. In the related bibliography (Burrows et al., 1990; Lowe, 1995) there are examples of protocols that were published with errors, which remained undiscovered for many years. Thus, formal waysof reasoning (Lopez et al., 2002) for whether a given protocol meets its security goals is an absolute necessity.

Model checking is a widespread fully automatic formal analysis that has been successful in discovering flaws in protocols considered to be correct. However, ongoing research has not stopped to look for new ways to tackle the problem of state-space explosion, which still prevents analyses of complex protocols and protocol configurations (e.g. higher bounds in the number of ongoing protocol sessions).

* *Corresponding author*. Tel.: +30 2310 998532; fax: +30 2310 998419.
E-mail addresses: basags@csd.auth.gr (S. Basagiannis), katsaros@csd.auth.gr (P. Katsaros), apombo@csd.auth.gr (A. Pombortsis).

0167-4048/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2009.08.003

In general-purpose model checking (Clarke et al., 1999), state-space explosion comes from the asynchronous composition of the modeled concurrent processes and the inherent symmetry redundancy of models in many different problem domains. In security model checking, one additional factor that makes the problem harder is the complexity of the assumed intruder behavior.

Model checking security guarantees such as secrecy and authentication are based on the hardest possible assumptions for the dominance of the intruder over the communication between the protocol participants. These assumptions represent the general Dolev–Yao intruder model (Dolev and Yao, 1983): the intruder can intercept any message transmitted on a public communication channel and can also replace it with a message constructed from his initial knowledge and parts of the messages sent by the participants in the same or in other protocol sessions (intruder's knowledge base). The new messages are created by applying one or more out of four (4) basic operations: *encryption*, *decryption*, *concatenation* and *projection*. Also, a typical Dolev–Yao intruder model includes additional assumptions, such as the un-breakability of the encryption used and the possibility the intruder to prevent an original message from reaching its destination.

With the mentioned assumptions, any attempt to enumerate all possible attacks in all protocol steps results in an enormous branching of the state space. In the general case, for a given set of eavesdropped messages, the Dolev–Yao operations may be combined recursively, thus producing infinitely many possible fake messages. In explicit state model checking, analysts bound the size of fake messages, in order to set their models finite. However, memory space becomes crucial, due to the need to store information for each state, including the local states of all protocol participants and the accumulated knowledge of the intruder, for the protocol execution. An additional problem is that under the described assumptions the involved protocol parties interact asynchronously through the same communication channel. The interleaving and concurrency among them may easily result into state-space explosion. Analysts observed that the size of the state space increases exponentially with the number of protocol sessions.

Let us consider a protocol execution with two parties *A* and *B* acting as initiators of two separate protocol sessions. The state where initiator *A* has started the protocol and *B* is idle is symmetric to the state where *A* is idle and *B* has started the protocol. Symmetry reductions partition the state space into various equivalence classes, which are exploited by taking into account only one state from each partition. Symmetry reductions for security protocol verification have been first implemented in Brutus (Clarke et al., 2000a). In another work (Clarke et al., 2000b), the same authors address the complications of applying partial order reduction, due to tracking the accumulated knowledge of the intruder. Partial order reduction avoids creating states that cannot be affected by interleaving the execution of the model's processes. Results from model checking experiments with partial order reduction pruned the state space by a factor of 10–1877, depending on the examined protocol and the numbers of initiators and responders. Model checking experiments with symmetry reductions that were applied together with partial order reduction resulted in reductions of the state spaces by a factor of up to 58, with a more significant effect in experiments with four to five initiators and four to five responders.

Techniques that delimit the branching of the state space, due to the intruder's fake messages – without excluding possible attacks – have been implemented in specialized security model checkers (Gritzalis et al., 1999). Some techniques (Shmatikov and Stern, 2000) exploit certain properties that have been identified as characteristics of security protocols, but there is also one recent work (Kim et al., 2003) that proposes a "divide-and-conquer" approach for reducing the amount of memory needed. A broad family of state-space reduction techniques adopts a symbolic representation of the state space, in order to avoid explicitly enumerating all possible messages that the intruder can generate. In general, most techniques can be exploited, only if the analyst will adopt the model checking tool that implements the respective technique. We provide a detailed review of related work in Section 5. However, we believe that any new proposal for state-space reduction still contributes into improving the efficiency and the feasible size of model checking tasks.

In current article, we introduce the Message Inspection (MI) intruder model, which is essentially a Dolev–Yao style man-in-the-middle intruder based on the idea of improving his knowledge with protocol-specific metadata that provide information for the exchanged messages. In a preliminary simulation run, the intruder tags the eavesdropped messages with specific metadata parameters enabling him to validate all possible attack actions. The MI algorithm then decides based on this enhanced knowledge, which of the attacks will certainly fail and the simulation run terminates with a report of the attack actions that can be discarded.

This approach guides the pruning of the model's state-space, since the intruder avoids performing attacks for which it is a priori known that they cannot uncover a protocol flaw. The described two-stage procedure does not limit the overall model checking effectiveness, because the overall analysis can still capture security violations that are encoded as safety guarantees (secrecy and authentication) and at the same time allows customizing the intruder model for capturing security violations that involve liveness (e.g. non-repudiation).

Section 2 introduces basic terminology and describes in detail the general Dolev–Yao intruder model. Section 3 presents the Message Inspection intruder model. The model structure is formally defined and subsequently we introduce the MI algorithm that decides, which attack actions will be performed against the analyzed protocol. In Section 4, we provide experimental results for a MI intruder model in the SPIN model checker (http://spinroot.com), when compared with a generic Dolev–Yao intruder model applied upon the Needham–Schroeder security protocol (NSPK) (Needham and Schroeder, 1978). Section 5 reviews related work on intruder modeling and state-space reduction techniques, in order to point out the differences from the proposed intruder model and eventually to discuss its strengths and its weaknesses. Finally, conclusions and future work prospects are discussed in Section 6.