Modelling of security properties in Alloy

Lilia Georgieva
School of Mathematical and
Computer Sciences
Heriot-Watt University, Riccarton
Edinburgh, EH14 4AS
Email: lilia@macs.hw.ac.uk

***Abstract*—We study the problem of verification of security properties of Session Initiation Protocol (SIP) using the model analyser Alloy. We propose a novel approach to model analysis and demonstrating robustness of protocol models in first-order logic.**

I. Introduction

Wireless networks have become pervasive since evolving towards the 3G and 4G architectures and strive to provide faster and more reliable service, better integration of services and enhanced protection of users against malicious attacks [12]. One of the key considerations is to integrate security in the model design in order to incorporate better defences to protect against malicious attacks [10], [17].

Current challenges of wireless networks include optimal protocol design, protocol adaptation, consistent quality and reliability of service [2], [15], [7].

In this paper we investigate a logic-based approach to modelling of a Session Initiation Protocol (SIP). We model the protocols in first-order logic and use the model analyser Alloy [14] to specify properties of the message exchange using SIP.

We study the first-order logic model of SIP [8] as developed in [8]. Verification of SIP using Alloy involves the creation and analysis of a protocol model in typed first-order logic. The novelty of our approach is in using a light-weight modelling tool for the analysis of the model to determine its security properties. We test the model, specify assertions about the model, and explore its state-space using the model analyser Alloy in order to assure ourselves that the protocol under verification has the desired properties. Our model intuitively relates to the underlying multi-agent design framework of SIP. First-order models of multi-agent systems have been analysed in a different context using Alloy [16].

Using a lightweight formal method has the advantage of having an elegant but powerful language with simple notation [14] giving a simple and robust meaning to the message exchange using SIP. The Alloy tool-set allows us to have a visual representation and analysis of the model developed in [8].

II. Session Initiation Protocol

The Session Initiation Protocol (SIP) [13] handles signaling such as call setup, routing, and authentication to endpoints within an IP domain [5].

A SIP network consists of a user agent who can create new SIP requests (User Agent Client (UAC)) and can also respond to SIP requests (User Agent Server (UAS)), that is, a user agent is a logical entity that can act as both a UAC and a UAS [13], [15].

The SIP Proxy server is used in forwarding SIP requests received from UAs or other proxies to other locations (proxies, UAs or external networks). It is also used to authenticate and authorize users which are already registered their current locations with the SIP Registrar [5]. The base SIP specifications defines six SIP requests which are also know as methods and they are INVITE, ACK, OPTIONS, BYE, CANCEL and REGISTER [13]. SIP is based on HTTP and it maintains a standard format which is made up of the start line, header fields and the message body [15].

Security flaws of SIP have been identified [8]. Security strategies for enhancing the overall security level have been proposed [9] [4].

III. Alloy

The lightweight model analyser Alloy [14] uses typed first-order logic. The specifications written in Aloy are declarative and based on relations. The notation is concise and intuitive and allows for formulation of a range of useful properties.

Alloy has been used for the modelling and analysis of crucial design properties in various applications [16], [4], [3]. It provides a fully automatic analysis for checking properties of the specifications.

The analysis performed by Alloy is within a limited range and uses selected propositional satisfiability (SAT) solvers [14]. Given an Alloy formula and a scope, i.e., a bound on the universe of discourse, the analyzer translates the Alloy formula into a Boolean formula in conjunctive normal form (CNF), and solves it using an off-the-shelf SAT solver [14]. The Alloy tool-set has been used successfully to check designs of various applications [6], [1], [11].

IV. SIP model in Alloy

Consider, for example, the beginning of our Alloy specification of the SIP protocol.

```
module protocol/sip

sig SIPMessage{header: FirstLine,
body: SIPBody, status: SIPStatus}
```

```
abstract sig SIPStatus{}

...

one sig Transmitted, Authenticated,
Timed extends SIPStatus{}

...

fun statusChange(): SIPStatus-> SIPStatus{
SIPStatus<: iden+ Transmitted->Authenticated+
Transmitted->Authenticated +
Authenticated-> Timed
}
```

The keyword *module* names a model. A *sig* declaration introduces a set of (indivisible) atoms; the signatures SIPMessage and SIPStatus respectively declare a set of atoms indicating the components of the message and a set of atoms identifying the status of the message. The fields of a signature declare relations. The field *status* in our specification defines a relationship of type *SIPMessage* $\times$ *SIPStatus* indicating that the SIPMessage will have a specific status. The set of possible states that the status of the message can be in is declared by the signatures *Transmitted, Authenticated, Timed* which are defined to be subsets of *SIPStatus*. In our model, *statusChange* is declared as a function updating the message status in a sequence of steps.

A. The fist-order model in Alloy

We define signature (objects), to model the SIP message and the header of the message, which correspond to the unary predicates $SIP_{Message}(x)$ and SIP_Header in the first-order model.

```
abstract sig SIPHeader extends SIPMessage{
identifier: one FirstLine}

pred show(m: SIPMessage){
#m.header>=3}
```

The predicate emphshow expresses that any SIP message requires existence of a first line and at least three SIP headers.

The properties that a first line should be a request or response, and the consistency check that a request should not be a response and vice versa are modelled easy in first-order logic.

```
sig FirstLine{
first: SIPHeader,
status: SIPStatus}


sig Act{}

sig Request extends Act
{request: lone  Request}
sig Response extends Act
{response: lone Response}

abstract sig SIPHeader extends
SIPMessage{identifier: one FirstLine}
...

assert noSelfResponse{no m: Response| m=m.response}
assert noSelfSend{no n: Request| n=n.request}
```

We define the methods for authentication which are *Invite, Register, Options.*

```
abstract sig Method{}

one sig Invite, Register, Options extends Method{}
sig Authenticate{
mode: Method}
```

Message authentication, which is SIP is identified by, for example, using specific method can be defined as follows:

```
sig messageAuth{authentication:
SIPBody-> one Authenticate}
one sig Mechanism1, Mechanism2
extends Authenticate{}
```

Using Alloy we can perform standard checks, such as *no message is a response to itself* and *no message was sent by itself*

```
check noSelfResponse
check noSelfSend
```

Assertions in Alloy allow us to check constraints of our model. For example, we assert that a message has its first line in the header.

```
assert{some f: FirstLine,
m: SIPMessage| f in m.header}
```

We can vary the scope, forces different upper bounds of our model. In the case below the upper bound is set to 3 nodes. The but keyword specifies a separate bound for a signature whose name follows the keyword. Thus we restrict a generated example to 1 Transmission, i.e. message exchange.

```
run show for 3 but 1 Transmission
```

Independent of each other attacks on SIP can be modelled in first-order logic [8]. For example, the Alloy specification below models a malformed SIP attack which is complement to SIP message.

```
module protocol/sip
open util/relation as rel

sig SIPMessage{header: FirstLine, body: SIPBody,
status: SIPStatus, action: Action}
abstract sig FirstLine{}
abstract sig SIPBody{}

abstract sig SIPStatus{}


abstract sig SIPAttack{ currentAttack: SIPAttack}
one sig Transmitted, Authenticated,
Timed extends SIPStatus{}

one sig Malformed, Signalling, Flood
extends SIPAttack{

malformed: Malformed,
signalling: Signalling,
flood: Flood}

abstract sig Action{}
one sig Transmit, Signal extends Action{}
```

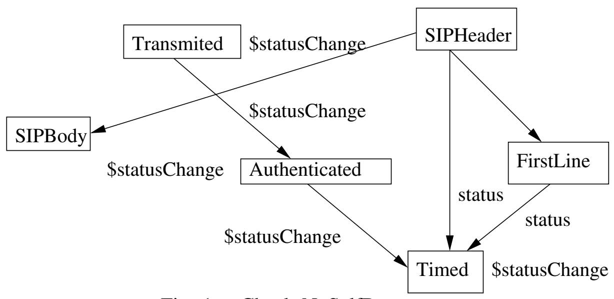

Fig. 1. Check NoSelfResponse

```
sig Time{}

assert noSignalling{all m: SIPAttack |
m in Malformed
}


check noSignalling
```

By using the signature *Time* we can model the existence of two or more identical SIP messages within different time frames which is occurrence of signalling attack, and the existence of any not authenticated message which is also considered as a signalling attack.

B. *A first-order security model*

The first-order model of SIP as designed in [8] uses first-order logic to model properties of SIP. The SIP message structure is formalised using unary and binary predicates, which define specific properties based on SIP grammar specifications, the properties that can be specified are, for example that any SIP message requires the existence of a first line and at least three SIP headers, a first line should be a request or response, whereas, a request should not be a response and vice versa.

Examples of specification of a SIP message properties, as defined in [8] in first-order logic are:

$$\text{SIP}_{\text{Message}}(x) \quad (1)$$
$$\forall x \text{SIP}_{\text{Message}}(x) \leftrightarrow \exists f \text{FirstLine(f)} \wedge \quad (2)$$
$$\exists^{<=3} h \text{SIP}_{\text{header}}(h) \quad (3)$$
$$\forall m \text{Method}(m) = \{\text{INVITE} \vee \text{REGISTER} \vee \text{OPTIONS}\} \quad (4)$$

Fomula (1) specifies a the SIP message as a unary protocol, formula (2) states that all SIP message has a first line, formula (3) that each SIP message must have at least 3 headers; formula (4) defines the methods as *Invite, Register*, or *Options*.

There is direct correspondence between these properties and our specification. We translate the model of [8] in typed first-order logic and automatically analyse and visualise it using Alloy. For example, in Figure 1 we show an abstraction of the Alloy model of assertion checking of *NoSelfResponse*. The model allows us to visualise the model states where the status changes, on the figure indicated by (*$ statusChange*. Projections with respect to time, authentication method or transmission are also possible.

V. Conclusion

In this paper we model the security properties of SIP and use Alloy to analyse the model.

Modelling in Alloy is structural, logical, and intuitive. The specification language is based on typed first-order logic and as such expressive enough to allow us to capture complex behaviour and specify and analyse interesting security properties.

References

[1] M. Auguston. *Software Architecture Built from Behavior Models* ACM SIGSOFT Software Engineering Notes, September 2009, Vol.34: 5, 2009
[2] Th. Bessis and V. Gurbani and A. Rana, *Session Initiation Protocol firewall for the IP Multimedia Subsystem core*, Bell Labs Technical Journal, Volume 15: 4, 2011
[3] B. Braga, J. Almeida, G. Guizzardi, A. Benevides. *Transforming OntoUML into Alloy: Towards Conceptual Model Validation using a Lightweight Formal Method*. Innovations in System and Software Engineering (ISSE), Springer, 2010.
[4] M. Frappier, B. Fraikin, R. Chossart, R. Chane-Yack-Fa, M. Ouenzar. *Comparison of Model Checking Tools for Information Systems* In Proceedings of ICFEM, 2010.
[5] G. Camarillo. SIP demystified. McGraw-Hill, New York, 2002.
[6] J. Galeotti, N. Rosner, C. Pombo, M. Frias. *Analysis of Invariants for Efficient Bounded Verification*. International Symposium on Software Testing and Analysis, 2010.
[7] V. Cortellessa, V. Grassi. *Reliability Modeling and Analysis of Service-Oriented Architectures*. Test and Analysis of Web Services, 2007.
[8] D. Geneiatakis and C. Lambrinoudakis and G. Kambourakis and A. Kafkalas. A First Order Logic Security Verification Model for SIP. Proceedings of ICCI 2009.
[9] D. Geneiatakis, T. Dagiuklas, G. Kambourakis, C. Lambrinoudakis, S. Gritzalis, K. Ehlert, D. Sisalem, *Survey of security vulnerabilities in session initiation protocol*, IEEE , Vol.8:3, 2006.
[10] M. Feredj, F. Boulanger, A. Mbobi. *A model of domain-polymorph component for heterogeneous system design*. In Journal of Systems and Software, Volumen 82:1, 2009.
[11] W. Hassan, L. Logrippo. *A Governance Requirements Extraction Model for Legal Compliance Validation*. IEEE International Requirements Engineering Conference, 2009.
[12] L. Buttyán and J. Hubaux. *Security and Cooperation in Wireless Networks: Thwarting Malicious and Selfish Behavior in the Age of Ubiquitous Computing*, Cambridge University Press, 2007.
[13] IETF RFC 3261: SIP: Session Initiation Protocol.
[14] D. Jackson. Software Abstractions: Logic, Language, and Analysis. MIT Press, 2006.
[15] A. Oredope and A. Liotta. *A Performance Based Evaluation of SIP Signalling across Converged Networks*. Proceedings of the World Congress on Engineering 2007 Vol II WCE, 2007.
[16] R. Podorozhny, S. Khurshid, D. Perry, and X. Zhang. *Verification of Multi-agent Negotiations Using the Alloy Analyzer*, In Proceedings of IFM, 2007.
[17] Ch. Wolter, P. Miseldine, Ch. Meinel. *Verification of Business Process Entailment Constraints Using SPIN*. In Proceedings ESSOS, 2009.