A Tool of Analysis and Implementation of Security Protocols on Distributed Systems

L. Mengual, C. de la Puente

Dpto. de Lenguajes y Sistemas Informáticos e Ingeniería de Software
Facultad de Informática (UPM), 28660 Boadilla del Monte, Madrid
lmengual@fi.upm.es, cpuenteg@yahoo.com
Tel: 913367397 Fax: 913367393

ABSTRACT

In this paper we present an analysis and automatic implementation tool of security protocols based on Techniques of Formal Description. A sufficiently complete and concise formal specification that has allowed us to define the state machine that corresponds to a cryptographic protocol has been designed to achieve our goals. This formal specification also makes it possible to incorporate in a flexible way the mechanisms and functions of security. Our system makes it possible to automatically translate formal specifications of protocols to a group of derivation Prolog rules. Our work improves the solution first proposed by J.K. Millen in [1]. We study the vulnerability of security protocols exploring, in an automatic way, all its possible behaviors in agreement both with their formal specification and with the information potentially recorded by an intruder. As a result of our work, for example, the vulnerability of security protocols that is already proven in a theoretical way (Needham-Schroeder protocol [2] or SSH or AKA protocols [3]) can be proven and analyzed automatically starting from the translation to rules of those protocols. Additionally, our tool incorporates a new concept of implementation of protocols based on the interpretation of formal specifications on distributed systems.

Keywords: security protocols, formal specification, Prolog Rules, automatic implementation

1. INTRODUCTION

The Formal Description Techniques are the base of the automated support in different development activities. The formal specification is an essential tool in the engineering of communications protocols. Using the Techniques of Formal Description significant improvements can be obtained in the quality of the product, availability in the market and cost of the lifecycle.

The formal models of security have evolved in parallel with the development of the computer systems (software, hardware, operating systems), as well as with the technology and extension of the data networks. Initially, the formal models have treated the problem of control access in individual systems. The criteria of security TCSEC (Trusted Computer System Evaluation Criteria) of the DoD (Department of Defense) of the government of USA [4] or the European ITSEC (Information Technology Security Evaluation Criteria) [5] criteria are an example of the formalization in the field of secure computer systems. Other examples are the models developed in [6], [7], [8].

The important improvements of the Local Area Networks (LAN), and their interconnection with remote networks has given birth to new and more sophisticated threats associated with the distribution of the information (eavesdropping, tampering, impersonation, etc)[9]. This has led to the development of new security mechanisms and functions to protect not only the information in traffic in open systems but also the problems of access control to resources in individual systems [10] [11]. In this case, the Techniques of Formal Description (TDF) have been used to specify security protocols and to evaluate the vulnerability of these protocols against different attacks. Examples of this type of formal analysis are given in [1], [12], [13].

The formal specification makes it possible both to verify and to validate communications protocols in an efficient way before the development of a software product. However, the use of Techniques of Formal Description in the specification of security protocols implies a series of additional considerations that are not taken into account in the conventional specifications of communications protocols [14]. Thus, a series of desirable aspects have to be considered:

- **Appropriate specification of potential attacks.** The chosen technique of formal description will contemplate the possibility of an intruder knowing dynamically partial aspects of the operation of the protocol. Taking into account this information, the formal specification will facilitate the task of analysis of the vulnerability and consistency of the protocol.

- **Exhaustive search of alternative behaviors.** The chosen technique of formal description will make it possible to analyze all the possibilities of behavior of the protocol (not only a subset) in function of the definition of the protocol and in function of the mechanisms specified.

- **Implementation Easiness.** The chosen formal specification will be able to derive a finite state machine to represent the exchanges, mechanisms and specified functions to facilitate the implementation tasks.

The techniques of formal description used for the formal specification of communications protocols (ESTELLE, LOTOS, SDL etc), although having some positive aspects, present certain limitations in the analysis and evaluation of security protocols [12]. Other techniques have been developed expressly to study the vulnerability of security protocols. One of these techniques is the Logic of Authentication of Burrows, Abadi and Needham (logic BAN) [15]. The logic BAN provides a formal basis for the analysis of protocols although it possesses certain limitations in the specification and study of protocols [16] and it does not make an automation of the analysis process possible.

---

[1] This work is supported by Science and Technology Ministry of Spain under the proyect: "A Multi-Agent System Architecture for the automatic and dynamic Implementation of security protocols (TIC2001-3376)"
In this field, the translation of a security protocol to a group of rules Prolog although seems to be a complex task presents certain benefits. The first attempts [17, 1] represent a remarkable evolution in the analysis and verification of security protocols. In the first case [17], an authentication logic (based on the logic BAN) is implemented in Prolog. In the second case [1], the conditions that regulate the exchange of messages of the security protocol, the functions of security, as well as the initial knowledge of an intruder are specified formally by means of rules. The idea behind this is that exploring all the possible combinations of variable fields of a message and all the possible messages to exchange (according to the definition of the protocol) it can be verified if an intruder can access to sensitive information. Undoubtedly, the property of backtracking of a Prolog specification is essential in the exploration of all the possible alternatives and of unquestionable value in the verification of the security. A Prolog specification matches several of the requirements mentioned before for the specification of security protocols, however, other problems have to be solved:

- It is not easy to translate the group of exchanges, functions and mechanisms of security to a group of Prolog rules.
- It is not easy to derive a finite state machine that makes the implementation of a protocol possible starting from a Prolog specification.

In [1] a partial solution is given to these problems creating an initial specification of the protocol. Starting from this initial specification the Prolog specification can be derived. However, the automation of the process of translation and of implementation of the protocols has not been already solved. As a consequence, the conception of a more elaborated initial specification of the protocol is needed. The Techniques of Formal Description usually have associated with them tools that make the automatic implementation of communications protocols possible. The final tests should guarantee the adaptation from the code to the requirements of the specification. The application of techniques of formal description to the field of the security should contemplate the necessity to specify the exchanges that appropriately correspond to the security protocols, as well as the functions and mechanisms implied in order to be interpreted in the implementation phase.

A great effort has been carried out with unquestionable success in the development of Techniques of Formal Description (TDF) for the verification, validation and implementation of communications protocols. However, a parallel effort has not been carried out in the development of formal techniques that make the analysis and verification of protocols that implement security services (security protocols) possible. Thus we can point out some of the lacks of the existent models of analysis of security:

- A complete and integrated model that makes it possible, on the one hand, the automatic verification of security protocols based on the exhaustive analysis of attacks, starting from a formal specification, and on the other hand, the automatic generation of a prototype has not been developed. This is due partly to the fact that the specifications used for the analysis of security were neither complete enough nor developed for the generation of implementations.
- It has not been developed a dynamic and versatile system able to implement multiple and varied security services generated from formal specifications. This largely limits the use of the specifications when implementing security protocols.

In this work we present the results obtained from the development of the Analyzer of Attacks (A.N.A.). This system makes it possible to automate the analysis, verification and implementation of security protocols, starting from formal specifications and with it, the versatility in the incorporation of different security services in a distributed environment. In definitive we have developed a system that has the following features:

- It specifies in an appropriate way the necessary exchanges among the entities that correspond to a security protocol. We have conceived a formal specification of a security protocol that is sufficiently clear and concise to allow you to derive the finite state machine that constitutes the security protocol. The objective is twofold: the formal specification is the base for the analysis of a group of simulated attacks and it supports the generation of a real implementation of the protocol.
- It allows you to analyze specified protocols based on the realization of a group of tests that verify its resistance against a group of simulated attacks. Thus a technique has to be chosen for the analysis of attacks from the formal specification. In this field we decided to translate the security protocol to a group of Prolog clauses according to the works begun in [1]. Our contribution has been the incorporation of new clauses that make it possible to analyze in an exhaustive way the attacks, as well as to automate completely the translation process.
- It generates a real implementation of the specified protocol starting from its formal specification. This prototype is the base to derive real implementations, as well as for the implementation of possible attacks with the goal of proving the vulnerability of the protocol.

2. STRUCTURE AND SCOPE OF THE ANALYZER OF ATTACKS

The Analyzer of Attacks (A.N.A.) is the name of the model of analysis, verification and implementation of security protocols proposed in this work. On the one hand, this model is able to generate in an automatic way a group of derivation rules in Prolog that represent a security protocol starting from its formal specification. The rules make it possible to carry out an exhaustive analysis of the vulnerability of the protocol. On the other hand, the Analyzer of attacks is able to generate in an automatic way a real implementation of the protocol. We have designed a system that makes it possible to incorporate in a dynamic and flexible way security services in the TCP/IP protocol suite.

The Analyzer of Attacks is structured in two independent logic software elements: the Logical Element of Analysis and the Logical Element of Implementation (see figure 1). The Formal Specification I is the description of the security protocol containing the exchanges, mechanisms and functions that satisfy the requirements defined in the security policy. This will be the starting point of our model. The Logical Element of Analysis carries out the functions of syntactic analysis of the Formal Specification I and the translation of the protocol to a group of derivation rules in Prolog (Formal Specification II) that will allow us the analysis and verification of the protocol. This leads to (once completed with success the process of syntactic analysis) the generation of the protocol implementation.
2. **The function of the following modules:** appears in [18]. The design of this specification, carried out the formal specification. The syntax and semantics consists of several modules that in turn contain a series of its vulnerability. The "Prolog Rules Generator" translates in an automatic way derivation rules (Formal Specification I) to a group of encryption patterns. An encryption pattern represents a group of address, key, numeric or temporary identifiers encrypted by an identifier of key type. An encryption pattern is represented by the formal identifier "encrypt(KX,[Id1, Id2, ...])" where "KX" represent the key of the encryption algorithm and "Id1, Id2, ..." represent the identifiers to encrypt.

3. **LOGICAL ELEMENT OF ANALYSIS**

The Logical Element of Analysis translates in an automatic way a formal specification of a security protocol according to some predefined syntactic rules (Formal Specification I) to a group of Prolog derivation rules (Formal Specification II) that represent the protocol and that makes it possible an exhaustive analysis of its vulnerability. The Logical Element of Analysis of our model consists of several modules that in turn contain a series of associate procedures whose description is detailed in [18]. These modules represented in figure 2 are the following ones: "Syntactic Analyzer", "Encryption Patterns Locator" and "Prolog Rules Generator".

4. **Declaration of Rules associated with relationships.** This module contains a set of constant identifiers associated with the relationships defined in the previous element. These identifiers try to represent the concrete values that we would obtain by applying the relationships or functions of security specified in the previous element.

5. **Declaration of the information known by the intruder.** This module contains the information that an intruder has before beginning a session of the security protocol. It is assumed that this information can come from recordings made of that protocol in previous sessions.

An example of a Formal Specification I is represented in figure 3. In this case we have specified a variant of the well-known protocol of Needham-Schroeder [2].

The function of the Syntactic Analyzer is to verify the syntax of the formal specification. The syntax and semantics contemplated in the definition of the Formal Specification I appears in [18]. The design of this specification, carried out according to the requirements exposed in [18] contains the following modules:

1. **Identifier of the beginning of the specification.** This module determines the beginning of the formal specification of a security protocol.

2. **Constants Declaration.** This module contains the chains of characters that have been called constant identifiers that will be used in the definition of the security protocol. Five types of identifiers have been defined: Address, Key, Numeric, Temporal and Data.

3. **Messages of the security protocol.** This module contains the exchanges of the security protocol. An integer number identifies the actual message. Each message contains constant or variable identifiers and/or encryption patterns.

4. **Rules associated with messages.** This module contains a group of procedures or conditions that an pair entity should apply when generating a message. In this module the mechanisms and functions of security implied in the encryption protocols are specified. For example, symmetric and asymmetric cryptography, nonce identifiers, timestamps generation etc. To each one of these functions corresponds a formal identifier: "session_key()", "public_key()", "private_key()", "secret_key()", "random()", "function_x()".

5. **Declaration of Rules associated with relationships.** This module contains the information that represents those contents of the security protocol that are encrypted for its transmission in a data network. The Encryption Patterns Locator will carry out a specific study of those elements of the specification that are encryption patterns and will notify it to the Syntactic Analyzer. Finally, the Rules Generator will translate the formal specification to a group of Prolog derivation rules (Formal Specification II).
These rules make it possible to generate all the possible alternatives of messages exchanged in the protocol based on the specification of the protocol defined in the element MESSAGES, on the relationships and rules defined in the elements RELATIONS and RULES, and on the knowledge of a potential intruder defined in the rules KNOWS. These rules also make it possible to query if there is any possibility of an intruder knowing some component of the exchanged messages. To solve this kind of query, an exhaustive search of behaviors of the protocol will be made according to the formal specification of the same one and to the property of backtracking of the Prolog rules that define the protocol.

In the definition of the Prolog rules that specify our security protocol lists will be managed. A list is an ordered sequence of elements that can have any length. We will consider that a message is a list. The components of this list will be either identifiers or encryption patterns. A security protocol will be also represented as a list or group of messages. To take the possible attacks of an intruder into account, we will consider that in this list of messages there is, for each message sent by an entity, the corresponding received message for the other entity. In general, the list of messages of the security protocol in a given instant can be represented as \([M_1, M_2, ..., M_n]\) where \(M_n\) is the message "n" sent by an entity of the other entity of the security protocol, and \(M_n\) is the message received by the other entity of the security protocol after a possible attack of an intruder. The group of messages that constitutes an alternative of behavior of a protocol according to their formal specification and to the initial knowledge of an intruder has been denominated "Messages History" of the protocol.

Figure 4: Reglas Prolog para la generación de mensajes del protocolo de ejemplo

Starting from the Formal Specification I, our system automatically generates a group of Prolog rules that us allow to build the exchanges that correspond to the security protocol. Thus, in the case of the protocol defined in figure 3, the subset of generated rules in automatic way are the ones shown in the figure 4.

It is important to remark that some of these rules contemplate the possibility of an intruder modifies the contents of the messages from an initial knowledge. It is necessary to generate automatically all the possible combinations of contents of encryption patterns that the protocol specification contemplates, so that the Prolog interpreter explores all the possibilities regarding the contents of the messages. As a consequence in the example of the figure 3, our system will generate automatically the set of axioms that appear represented in the figure 5.

```prolog
encrypt_field0(encrypt(ksa,x)).
encrypt_field1(encrypt(ksa,encrypt(ksa,x))).
encrypt_field2(encrypt(ksa,encrypt(ksa,encrypt(ksa,x)))).
```

Figure 5 Axiomas generados automáticamente relativos a patrones de cifrado de la especificación de ejemplo

Finally our system generate automatically a sets of rules that make it possible to analyze exhaustively the protocol, in function of all the possible combinations of variable identifiers and encryption patterns (see figure 6). These rules allow you to determine if an intruder can access to sensitive fields of the specified protocol. A complete description of the Prolog rules associated to the example protocol can be found in [19]

```prolog
p_gets(X,M,PH,H,N):- encrypt_field0(encrypt(K0,W)),
member(X,W).
p_gets(X,M,PH,H,N):- encrypt_field1(encrypt(K0,W)),
member(X,W).
p_gets(X,M,PH,H,N):- encrypt_field2(encrypt(K0,W)),
member(X,W).
```

Figure 6: Reglas Prolog para el análisis del protocolo

3.1 Example of Analysis using the Analyzer of Attacks

In what follows an example of analysis of security protocol in function of the exhaustive search of alternatives provided by the Prolog specification is described. In this example, we begin with a specification of a security protocol made up of four exchanges (see figure 3). It has to be taken into account, that in the specification of the protocol the initial knowledge on the part of an intruder of the fields "kx" (old key) and "encrypt(ksa,x)" (encryption pattern) has been defined. In the following queries the Prolog interpreter behavior regarding the generation of possible fields and histories of alternative messages is shown. (Let us remember
that a Messages History is a choice of messages for the specified protocol.

In figure 7 the results obtained according to the predicate sent() that shows the different alternative messages, generated starting from our specification of the protocol are described. Notice that the predicate “sent()” has already been able to generate two possible alternatives of messages according to the contents of the specification. If the Prolog interpreter is now queried about an intruder knowing the session key “ks”, it would provide the Messages History (if it is available) from which the intruder can know the key “ks” (see figure 8).

\[ \vdash \text{sent}(M,H,1). \]
\[ M = [a,c,\text{encrypt}(ksa,b)] \]
\[ H = [] \]
\[ \vdash \text{sent}(M,H,2). \]
\[ M = [c,a,\text{encrypt}(ksa,\text{encrypt}(ksb,ks,a))] \]
\[ H = [[c,a,\text{encrypt}(ksa,ks[a])],[a,c,\text{encrypt}(ksa,b)]] \]
\[ \vdash \text{sent}(M,H,3). \]
\[ M = [a,b,\text{encrypt}(ks,ks[a])] \]
\[ H = [[c,a,\text{encrypt}(ksa,\text{encrypt}(ksb,ks,a))],[a,c,\text{encrypt}(ksa,b)],[a,c,\text{encrypt}(ksa,b)]] \]
\[ \vdash \text{p_knows}(ks,H,3). \]

Figure 7: Simple query on alternative of sent messages

Reconsulting ...
\[ \vdash \text{p_knows}(ks,H). \]
\[ H = [[a,b,\text{encrypt}(ks,ks[a])],[c,a,\text{encrypt}(ksa,\text{encrypt}(ksb,ks,a))],[a,c,\text{encrypt}(ksa,ks[a])],[a,c,\text{encrypt}(ksa,b)],[a,c,\text{encrypt}(ksa,b)]] \]

Figure 8: Complex query about the vulnerability of the session key

The result is that an intruder can end up knowing the key of session “ks” from the message 3 of the encryption pattern “encrypt(ksa,ks[a])” of the Messages History shown in the query. That is to say, in this protocol, an intruder could obtain the session key distributed “ks”, in the message 3 substituting in the encryption pattern “encrypt(ksa,kx)” for the encryption pattern “encrypt(ksa,x)”.

The shown example gives an idea of the possibilities that the Analyzer of Attacks offers in the analysis and verification of security protocols. The consistency of a protocol will have to be evaluated in function of the potential attacks to consider. These attacks will imply the acquisition of information, on the part of an intruder, of some of the exchanged contents and security functions developed by the security protocols. This knowledge will have to be specified through the elements KNOWS and RULES.

Therefore, a complete study of the vulnerability of a protocol requires on the one hand, the clear definition of the potential attacks to consider (security policy) and, on the other, the formal specification of these attacks (expressed in the form of acquisition of knowledge). The Analyzer of Attacks only offers the formal base for the analysis and verification of protocols; it is the final user of the system, who can evaluate the robustness of the protocol through the appropriate specification of attacks.

4. LOGICAL ELEMENT OF IMPLEMENTATION

The Logical Element of Implementation of our model Analyzer of Attacks is responsible for the automatic implementation of the security protocol formally specified. The developed infrastructure is able to incorporate different security protocols dynamically to a communication among TCP/IP applications. It has been achieved that the parent entities in a distributed environment interpret a specification of a security protocol and that, therefore, only have one code independently of the specific security protocol: Traditionally, the implementation of a security protocol (as any communications protocol) implied the implementation of a specific code in each of the participant entities ([20], [21]). In this work, it has been possible to develop an infrastructure in which the pair entities have the necessary functionalities to interpret any protocol.

The software infrastructure developed in the Logical Element of Implementation makes it possible to implement any security protocol with only one code of interpretation of protocols located in each participant entity. Each of the entities will have to know which is its behavior in function of the formal specification of the protocol. In figure 9 this idea is described. The process of automatic implementation of a security protocol begins after the distribution of the formal specification of the security protocol between pair entities [22]. A third entity the Authentication Server will be responsible to distribute the formal specification. This process is an application (a common code) installed in all the participant entities of the protocol. The idea is that the participant entities can scan the formal specification and adopt the role of issuer or receiver according to their role in the protocol.

5. CONCLUSIONS

In this paper a tool for the analysis and automatic implementation of security protocols has been presented. The result of this work is twofold:

- On the one hand it has been proven that the automatic translation of a security protocol to a group of Prolog rules is a valid choice for the exhaustive study of attacks. Either the supplantation of fields of a message, or the complete supplantation of a message, makes it possible to analyze and to study the attacks to a security protocol and its possible vulnerability. Having some information about the contents of the exchanges that an intruder could know and thanks to the Prolog backtracking property one can explore, if there is any possibility, of an intruder accessing sensitive information.
On the other hand it is possible to develop the appropriate functionalities for the automatic interpretation of a formal specification of a security protocol in each of the implied entities. This idea is the starting point for a dynamic conception of implementation of security services: For each application, one can specify the associate security protocol (dynamic design) according to the security policy requirements. The incorporation of our system of automatic implementation in distributed applications like the e-commerce, bank transfers, network management etc, or as integral part of the code of an Agent ([23], [24], [25]) opens a new line for the development of security services in Internet.

6. REFERENCES


