ISSN No. 0976-5697

Volume 4, No. 10, September-October 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Model Checking of E-Commerce Protocol using Casper FDR

Dantuluri Sravanthi
Computer Science Engineering
Shri Vishnu Engineering College for Women
Bhimavaram, India
sravanthi.kvkr@gmail.com

K V Krishnam Raju
Computer Science Engineering
SRKR Engineering College
Bhimavaram,India
kvkraju.srkr@gmail.com

Abstract: In present days the popularity of electronic commerce applications are motivated the development of new e-commerce protocols. By using these new protocols the secrecy and agreement properties are achieved. This paper mainly focuses on how to model the e-commerce protocol in CSP using SPL and verified using CasperFDR whether the protocol satisfies the properties specified. Attacks are identified in this version. The specifications through which these attacks are found are presented.

Keywords: Model Checking, E-Commerce Protocol, CSP, SPL, CasperFDR

I. INTRODUCTION

The rising popularity of the WWW (world wide web) has resulted in an increased interest in e-commerce. Therefore a number of e-commerce protocols have been proposed. Most of these protocols ensure that the information that is exchanged between the parties involved in the e-commerce is protected from unauthorized disclosure and modification. In this paper we address the problems of e -commerce protocol verification using CasperFDR. In particular, we use model checking [1, 2, 3] to secrecy and agreement properties of the secure e-commerce protocol proposed in [4].

Modeling and analysis of security protocols with Communicating Sequential Process (CSP) and Failure Divergence Refinement (FDR) have been proven to be effective and have helped the research community find attacks in several protocols. Lowe thus designed Casper [5], which takes more abstract descriptions of protocols as input and translates them into CSP. CSP was first described by Hoare in [6] [7], and has been applied in many fields.

First, we formally model the protocol in SPL (Protocol Specification Language) and analyze the protocol with CasperFDR. Next, we use CasperFDR to show that there are no other known attacks on E-Commerce protocol.

The rest of the paper is organized as follows. Section II deals with related work. In Section III, e-commerce protocol is modeled with CasperFDR and is analyzed and finally we conclude in Section IV.

II. RELATED WORK

Lowe [8, 9, 10] have used the FDR model checker to find attacks on cryptographic protocols. Roscoe et al. [11] have used the FDR model checker together with data independence techniques to prove that some security protocols are free from attacks.

Heintze et al. [12] focus on the non-security aspects of e-commerce protocols and use the FDR model checker to verify the money and goods atomicity properties of two e-commerce protocols NetBill [13, 14] and Digicash [15]. Heintze et al assume that neither the NetBill server nor the communication links to the NetBill server ever fail. The authors substantiate this assumption by arguing that banks (the NetBill server provides the services of a financial institution in this work) provide fail-safe service to customers and, in the worst case, communication with the bank can be made possible using hand-delivery.

III. PROTOCOL SPECIFICATION LANGUAGE

Before discussing the proposed model, it is helpful to know how to specify a protocol in specification language. In protocol engineering, a protocol is specified by the services to be provided, assumptions about the environment, vocabulary of the messages used, encoding (format) of each message, and the procedure rules for consistency of message exchanges. We have taken Needham-Schroeder Public Key protocol as an example for this section and discussed the protocol in the following sub sections.

a. A → B: A,B,{na,A}PK(B)
b. B → A: B,A,{na, nb}PK(A)
c. A → B: A,B,{nb}PK(B)

A. Protocol Description:

The protocol is specified as sequence of messages exchanged between the communication parties. The notation used is similar to the standard method of describing protocols. In order to represent the protocol, we use the notation {m}{k} for message m encrypted with key k. Thus the three messages in Needham-Schroeder public key Protocol can be represented by:

a. A -> B: {na, A}{PK(B)}
b. B -> A: {na, nb}{PK(A)}
c. A -> B: {nb}{PK(B)}

We also need some way of representing starting point of the protocol. We assume that the run is initiated by A receiving some message from a user, or the environment, including B's identity. We represent this by an extra message in the protocol description.

0. -> A: A, B

The absence of a sender field in the above line represents that this message is sent by the environment. The complete protocol description then takes the form:

#Protocol description

© 2010, IJARCS All Rights Reserved

0. -> A: B
   a. A -> B: {na, A}{PK(B)}
   b. B -> A: {na, nb}{PK(A)}
   c. A -> B: {nb}{PK(B)}

B. Free Variables :

The variables and their types and the functions that are used in the protocol definition are defined under #Free variables section. For Needham-Schroeder authentication protocol the free variables are defined as follows.

#Free variables
A, B : Agent
na, nb : Nonce
PK : Agent -> PublicKey
SK : Agent -> SecretKey

In above example the variables na and nb should be taken to be of type Nonce. The functions PK and SK return an agent's public key and secret key, respectively. We term these "free variables" because they will be instantiated with actual values when an actual system running the protocol. Under the free variables section only we also define which keys are inverses to other keys.

InverseKeys = (PK, SK)

The above line means that the functions PK and SK, when applied to the same identity, return keys that are inverses of each other; so for every agent A, PK(A) (A's public key) and SK(A) (A's secret key) are inverses of one another.

C. Processes:

Each agent running in the system will be represented by a CSP process under processes section. For Needham-Schroeder authentication protocol the processes section is defined below.

#Processes
INITIATOR(A,na) knows PK, SK(A)
RESPONDER(B,nb) knows PK, SK(B)

These lines give names to the roles played by the different agents (here INITIATOR and RESPONDER). The parameters and the variables following the keyword "knows" define the knowledge that the agent in question is expected to have at the beginning of the protocol run. For example, the initiator A is expected to know his own identity A, the nonce na, the public key function PK (i.e. he can look up public keys in some table), and his own secret key SK(A).

D. Specifications:

The requirements of the protocol are specified under #Specification section For Needham-Schroeder authentication protocol the specification section is defined as below.

#Specification
Secret(A, na, [B])
Secret(B, nb, [A])
Agreement(A,B,[na,nb])
Agreement(B,A,[na,nb])

In above example the lines starting with Secret specify that certain data items should be secret. The first secret specification Secret(A,na,[B]) specifies that A thinks that na is a secret that can be known to only himself and B. However, this line will cause a CSP specification to be generated with the meaning: if A runs the protocol with B, and B is not the intruder, then the intruder will never learn the value of na.

The lines starting with Agreement are authentication specifications. The first authentication specification Agreement(A,B,[na,nb]) specifies that A is correctly authenticated to B, and the two agents agree on the data values na and nb.

E. The System Definition:

The system definition contains following sections.

a. Type Definitions:

The types of variables to be used in the actual system to be checked are defined in a similar way to the types of the free variables under Actual variables section. For Needham-Schroeder authentication protocol the Actual variables section is defined as below.

#Actual variables
Alice, Bob, Mallory : Agent
Na, Nb, Nm : Nonce

According to above example the system dealing with three agents (Mallory will be the intruder), and three nonce's. The public and secret keys of these agents are defined in the #Functions section, below.

We use the convention that free variables representing agents are denoted by a single capital letter (A, B, etc.) while actual variables representing agents are denoted by real names (Alice, Bob, etc.). Similarly, free variables representing other data items are denoted by small letters (e.g. na) while the corresponding actual values are denoted by identifiers starting capital letter (e.g. Na).

b. Functions:

The functions that are used by the agents in the protocol description have to be defined under the #Functions section. For Needham-Schroeder authentication protocol the Functions section is defined as below.

#Functions
symbolic PK, SK

The above lines represent that the functions PK (which returns an agent's public key) and SK (which returns an agent's secret key) to be symbolic: this means that Casper produces its own values to represent the results of function applications.

c. System definition:

The system definition section represents which agents should be present in the system to be checked. For Needham-Schroeder authentication protocol the System section is defined as below.

#System
INITIATOR(Alice, Na)
RESPONDER(Bob, Nb)

From the above lines we consider a system with a single initiator, Alice (taking the role of A in the protocol description), and a single responder, Bob, they use nonce's Na and Nb. The types of the parameters of the processes should match the types of the parameters of the corresponding processes defined under the #Processes section.

d. The intruder:

Finally, the intruder section defines the operation of the intruder is specified by giving his identity, and the set of

© 2010, IJARCS All Rights Reserved

data values that he knows initially. For Needham-Schroeder authentication protocol the Intruder Information section is defined as below.

#Intruder Information
Intruder = Mallory
IntruderKnowledge = {Alice, Bob, Mallory, Nm, PK, SK(Mallory)}

The above example defines that the intruder's identity to be Mallory and initially knows all the agents' identities, a single nonce Nm, the public key function PK, and his own secret key SK. The inclusion of the function PK in the intruder's knowledge means that the intruder knows the public key information of all agents.

IV. MODELING AND ANALYSIS OF E-COMMERCE PROTOCOL

A. E-Commerce Protocol Structure:

a. TP->C : download of product encrypted with key K1
b. C->M : purchase order
c. M->C : product encrypted with a second key K2
d. M->TP : the decrypting key K for the product and the approved purchase order
e. C->TP : the payment token and copy of the purchase order
f. TP->C : the decrypting key
g. TP->M : payment token

The above messages represent different steps in the e-commerce protocol. In this protocol the messages are exchanged between a customer (*C*), a merchant (*M*) and a trusted third party (*TP*). The exchange of value P between X and Y is represented using the notation *X ->Y*: *P*. A merchant has several products to sell.

The merchant places a description of each product on an on-line catalog service with the trusted third party together with an encrypted copy of the product. If the customer is interested in a product, he downloads the encrypted version of the product (step 1) and then sends a purchase order to the merchant (step 2). Note that the customer cannot use the product unless he has decrypted it. Now the merchant does not send the decrypting key unless the merchant receives payment.

The customer does not pay unless he is sure that he is getting the right product.This is handled as follows: the merchant sends the product (step 3) encrypted with a second key, *K*2, such that *K*2 bears a particular mathematical relation with the key, *K*1, where *K*1 is the key the merchant used when uploading the encrypted product on the trusted third party. Additionally, the merchant escrows the decryption key, *K* , corresponding to *K*2, with the trusted third party (step 4).

The mathematical relation between the keys *K*1 and *K*2, is the basis for the theory of cross validation that has been proposed [4]. Thus, by comparing the encrypted product received from the merchant with the encrypted product that the customer downloaded from the trusted third party, the customer can be sure that the product he is about to pay for is indeed the product he wanted. At this stage the customer is yet to obtain the actual product because he does not have the key, *K*, to decrypt the encrypted product. Once the customer is satisfied with his comparison, he sends his payment token to the third party (step 5).

The third party verifies the customer's financial information and forwards the decrypting key to the customer (step 6) and the payment token to the merchant (step 7).

B. Modeling E-Commerce Protocol in CasperFDR:

The modeled E-Commerce protocol in CasperFDR is shown below. In the specification the initiator C and Responder M represents Customer and Merchant. TP represents Third Party Server.

```
#Free variables
c, m : Agent
tp : Server
p, po, pt : Message
kcm : Sessionkey
InverseKeys = (kcm, kcm)
#Processes
INITIATOR(c,tp,p,po,pt,kcm)
RESPONDER(m,p,po,pt,kcm)
SERVER(tp,p,po,pt,kcm)
#Protocol description
0.  -> tp : c
[tp != c]
1. tp -> c : {p}{kcm}
2.  -> c : m
[c != m]
3. c -> m : po
4. m -> c : {p}{kcm}
5.  -> m : tp
[m != tp]
6. m -> tp : kcm
7. c -> tp : po, pt
8. tp -> c : kcm
9. tp -> m : pt
#Specification
Secret(tp, p, [c])
Secret(c, p, [tp])
Agreement(tp, c, [po, pt])
Agreement(c, tp, [po, pt])
#Actual variables
Customer, Merchant, Mallory : Agent
Thirdparty : Server
P, Po, Pt : Message
Kcm : Sessionkey
InverseKeys = (Kcm, Kcm)
#Inline functions
#System
INITIATOR(Customer,Thirdparty,P,Po,Pt,Kcm)
RESPONDER(Merchant,P,Po,Pt,Kcm)
SERVER(Thirdparty,P,Po,Pt,Kcm)
#Intruder Information
Intruder = Mallory
IntruderKnowledge = {Customer, Merchant, Mallory,
Thirdparty, Kcm}
```

C. Analysis of E-Commerce Protocol using CasperFDR:

After compiling and checking the above model in CasperFDR tool, attacks are found for every property declared in specification part. Out of four property1 and property2 are related to secret specifications, property3 and property4 are related to authentication specifications. CasperFDR tool found attacks for secret and authentication properties in the specification part.

An attack generated by CasperFDR on the property1 Secret(tp, p, [c]) is shown below.

© 2010, IJARCS All Rights Reserved

0. -> Thirdparty : Merchant
1. Thirdparty -> I_Merchant : {P}{Kcm}
1. I_Thirdparty -> Customer : {P}{Kcm}
2. -> Customer : Mallory
3. Customer -> I_Mallory : Po
4. I_Mallory -> Customer : {P}{Kcm}
7. Customer -> I_Thirdparty : Po, Pt
8. I_Thirdparty -> Customer : Kcm
The intruder knows P
Goodbye

The above attack can be explained in the following steps.

a) Initially the intruder acts as a merchant (I_merchant) when the Third party server sends Product information (P) value to the Customer as shown in the above messages 0 and first instance of message 1.
b) During the second instance of message 1 the intruder acts as Third party server and sends P value to the Customer.
c) Now the intruder directly communicates with the customer and captures the information about payment order (PO).
d) Now the intruder acts as a legitimate Third party during the 7^{th} and 8^{th} messages and finally captures information about payment token (PT).
e) However the communication between Customer and Third party server is captured by the intruder by using man-in-the-middle attack.

After receiving the entire information the intruder will be in a position such that it can know the P, PO and PT.

An attack generated by CasperFDR on the property2 Secret(c, p, [tp]) is shown below.

0. -> Thirdparty : Merchant
1. Thirdparty -> I_Merchant : {P}{Kcm}
1. I_Thirdparty -> Customer : {P}{Kcm}
2. -> Customer : Mallory
3. Customer -> I_Mallory : Po
4. I_Mallory -> Customer : {P}{Kcm}
7. Customer -> I_Thirdparty : Po, Pt
The intruder knows P
Goodbye

The above attack can be explained in the following steps.

a) Initially the intruder acts as a merchant (I_merchant) when the Third party server sends Product information (P) value to the Customer as shown in the above messages 0 and first instance of message 1.
b) During the second instance of message 1 the intruder acts as Third party server and sends P value to the Customer.
c) Now the intruder directly communicates with the customer and captures the information about payment order (PO).
d) Now the intruder acts as a legitimate Third party during the 7^{th} message and finally captures information about payment token (PT).
e) However the communication between Customer and Third party server is captured by the intruder by using man-in-the-middle attack.

After receiving the entire information the intruder will be in a position such that it can know the P, PO and PT.

An attack generated by CasperFDR on the property3 Agreement(tp, c, [po, pt]) is shown below.

0. -> Thirdparty : Merchant
1. Thirdparty -> I_Merchant : {P}{Kcm}
1. I_Thirdparty -> Customer : {P}{Kcm}
2. -> Customer : Merchant
3. Customer -> I_Merchant : Po
4. I_Merchant -> Customer : {P}{Kcm}
7. Customer -> I_Thirdparty : Po, Pt
Customer believes (s)he has completed a run of the protocol, taking role INITIATOR, with Thirdparty, using data items Po, Pt
Goodbye

The above attack can be explained in the following steps.

a) Initially the intruder acts as a merchant (I_merchant) when the Third party server sends Product information (P) value to the Customer as shown in the above messages 0 and first instance of message 1.
b) During the second instance of message 1 the intruder acts as Third party server and sends P value to the Customer.
c) Now the intruder acts as merchant and communicates with the customer and captures the information about payment order (PO).
d) Now the intruder acts as a legitimate Third party during the 7^{th} message and finally captures information about payment token (PT).
e) However the communication between Customer and Third party server is captured by the intruder by using man-in-the-middle attack.

After receiving the entire information not only the customer and merchant the intruder also in a position such that it can know the P, PO and PT.

An attack generated by CasperFDR on the property3 Agreement(c, tp, [po, pt]) is shown below.

0. -> Thirdparty : Merchant
1. Thirdparty -> I_Merchant : {P}{Kcm}
1. I_Thirdparty -> Customer : {P}{Kcm}
6. I_Merchant -> Thirdparty : Kcm
2. -> Customer : Merchant
3. Customer -> I_Merchant : Po
4. I_Merchant -> Customer : {P}{Kcm}
Customer believes (s)he is running the protocol, taking role INITIATOR, with Thirdparty, using data items Po, Pt
7. I_Merchant -> Thirdparty : Po, Pt
8. Thirdparty -> I_Merchant : Kcm
Thirdparty believes (s)he has completed a run of the protocol, taking role SERVER, with Merchant, using data items Po, Pt
Goodbye

The above attack can be explained in the following steps.

a) Initially the intruder acts as a merchant (I_merchant) when the Third party server sends Product information (P) value to the Customer as shown in the above messages 0 and first instance of message 1.
b) During the second instance of message 1 the intruder acts as Third party server and sends P value to the Customer.
c) Now the intruder acts as merchant and communicates with the customer and captures the information about payment order (PO).

© 2010, IJARCS All Rights Reserved

d) Now the intruder acts as a legitimate Third party during the 7th message and finally captures information about payment token (PT).
e) However the communication between Customer and Third party server is captured by the intruder by using man-in-the-middle attack.

After receiving the entire information not only the customer and merchant the intruder also in a position such that it can know the P, PO and PT.

V. CONCLUSION AND FUTURE WORK

In this paper, the e-commerce protocol is modeled using CasperFDR. The compilation was done with CasperFDR. Attacks are found in this version. The attacks are interpreted by CasperFDR and the message sequence results are reported. In future we will fix the attacks found in the e-commerec protocol.

VI. REFERENCES

[1] J. M. Atlee and J. D. Gannon, "State-based Model Checking of Event Driven Systems Requirements," IEEE Transactions on Software Engineering, 19(1):13–23, January 1993.

[2] W. Marrero, E. Clarke, and S. Jha, "A Model Checker for Authentication Protocols. In Proceedings of the DIMACS Workshop on Design and Formal Verification of Security Protocols," Rutgers University, NJ, September 1997.

[3] J. Mitchell, M. Mitchell, and U. Stern, "Automated Analysis of Cryptographic Protocols Using Murphi," In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, pages 141–151, 1997.

[4] I. Ray, I. Ray, and N. Narasimhamurthy, "A Fair-Exchange Protocol with Automated Dispute Resolution," Technical report, University of Michigan-Dearborn, 1999. Submitted for publication.

[5] G. Lowe, "Casper: A compiler for the analysis of security protocols," Journal of Computer Security, vol. 6, pp. 53–84, (1998)

[6] C. A. R. Hoare, "Communicating sequential processes," Communications of ACM, vol. 21, no. 8, pp. 666–677, (1978)

[7] C. A. R. Hoare, Ed., "Communicating Sequential Processes," Prentice Hall International, (1985).

[8] G. Lowe, "Breaking and Fixing the Needham-Schroeder Public-key Protocol Using FDR," In Tools and Algorithms for the Construction and Analysis of Systems: Second International Workshop, TACAS 96, pages 147–166, 1996.

[9] G. Lowe, "Some New Attacks Upon Security Protocols," In Proceedings of the 1996 IEEE Computer Security Foundations Workshop, IEEE Computer Society Press, 1996.

[10] G. Lowe and A. W. Roscoe. "Using CSP to detect errors in the TMN protocol," IEEE Transactions on Software Engineering, 23:659–669, 1997.

[11] A. W. Roscoe, "Proving Security Protocols with Model Checkers by Data Independence Techniques," In Proceedings of the 1998 IEEE Computer Security Foundations Workshop, IEEE Computer Society Press, 1998.

[12] N. Heintze, J. Tygar, J. Wing, and H. Wong, "Model Checking Electronic Commerce Protocols," In Proceedings of the 2nd USENIX Workshop in Electronic Commerce, pages 146–164, November 1996.

[13] B. Cox, J. D. Tygar, and M. Sirbu, "NetBill Security and Transaction Protocol," In Proceedings of the First USENIX Workshop in Electronic Commerce, pages 77–88, July 1995.

[14] M. Sirbu and J. D. Tygar, "NetBill: An Internet Commerce System Optimized for Network Delivered Services," IEEE Personal Communications, pages 34–39, August 1995.

[15] A. Fiat D. Chaum and M. Naor, "untraceable electronic cash," In Advances in Cryptology – Proceedings of CRYPTO '88, pages 200–212. Springer-Verlag, 1990.

© 2010, IJARCS All Rights Reserved