KERBEROS AUTHENTICATION PROTOCOL MODELING USING NUSMV MODEL

Brajesh Patel*

Priyanka Dubey**

Abstract:

Authentication is one of the biggest problem related to computer information security in the area of distributed environments. Various protocols are used for the authentication purpose such as Needham-Schroeder protocol Kerberos protocol etc. The aim of this paper is to verify and formalize the Kerberos protocol using NuSMV model checker. The protocol version used in this paper is Kerberos Authentication Protocol. The paper suggests CTL specifications for authentication, secrecy and integrity. We have also proposed an approach to identify presence of intruder in the system.

Keywords: Kerberos Protocol, NuSMV Model Checker, CTL

* Deptt of Computer Science. HoD (Com. Sci.) SRIT, Jabalpur(m.p) India.

** Deptt of Computer Science. M.E. (4th SEM), SRIT, Jabalpur(m.p) India.

I. INTRODUCTION:

With the tremendous development in the field of Computer Networks, it has become easier to avail the services located on various different networks. As these services sometimes uses the personal data of users like on-line account passwords while doing on-line transactions, thus providing security to the clients has become of prime importance. So, in order to provide users faster and secured communication, various authentication pro- tocols which provide both, authorization and authentication and integrity and secrecy of messages are utilized. Authentication protocol are security mechanism whereby each party is assured its identity to one another. One of the authentication protocol which is commonly used is Kerberos. Kerberos was developed in the Athena Project at the Massachusetts Institute of Technology (MIT)[1]. It is a computer network authentication protocol, which allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. It is designed as a client/server model and it provides mutual authentication i.e. both the client and server verify each other's identity.

In this paper we have discuss about the formalization of Kerberos Authentication Protocol through model checking. The aim of paper is to analyze cryptographic protocols by means of NuSMV [2].

Model Checking, one of many formal verification methods[3], is an attractive and increasingly appealing alternative to simulation and testing to validate and verify systems. Given a system model and desired system properties, Thus, when the model checker verifies a given system property, it implies that all behaviors have been explored, and the question of adequate coverage or a missed behavior become irrelevant. There are two main advantages of using model checking compared to other formal verification methods :

1) It is fully automatic, and
2) It provides a counter example whenever the system fails to satisfy a given property.

NuSMV is a model checking tool which can be used to verify finite state machines. In NuSMV [4] we try to map our Finite State Machine (FSM) described in terms of state variables and input variables, which may assume different values in different states, of a transition relation describing how inputs leads from one state to possibly many different states. Earlier some authors are tried to model the behavior of Kerberos through NuSMV. In the M. Panti and L. Spalazzi and S. Tacconi [6], author try to express the basic version of Kerberos.In this paper author will model the basic version of Kerberos and prove the authentication property. Author will also shown the correspondence property for security requirement and show the counter example.

Shreya et. al [7] also has used NuSMV for the basic version of Kerberos for authentication. They have presented the concept of freshness and which helps to find the reply attack. They have demonstrated a possible means by which the weakness of the Kerberos protocol causing the replay attack can be overcome hence making the protocol more stronger.In this paper, we have demonstrated how to write a model for the Kerberos protocol version 4, introducing a concept of Ticket Granting Server (TGS) in NuSMV and the replay attack. It has also been shown that incorporating the concept of timers associated with individual messages in the protocol can be used to overcome the replay attack.The remainder of the paper is organized as follows:- II discusses about the theoretical background for this paper. Section III describes how to model the Kerberos protocol in NuSMV, gives the CTL specifications and describes the results obtained in this work. Finally conclusion has been drawn in Section IV.

II. THE KERBEROS PROTOCOL:

Kerberos is a network authentication protocol developed as part of Project Athena at MIT. Its main aim is to provide strong authentication for client/server applications by using secret-key cryptography. It allows nodes communicating over a non-secure network to prove their identity to one another in a secure manner. It uses symmetric-key cryptography and requires a trusted third party. In this paper our main focus will be on Kerberos version .

This version contains mainly three entities:

1) Client (C): An entity which wants to make use of any service hosted on a server.

2)Server (V): An entity which hosts different services which clients request for.

3)Authentication Server (AS): A trusted Third Party which knows the passwords of all users and stores these in a centralized database. Also, the AS shares a unique secret key with each server. It shares a key K_{CS} with C, a key K_{VS} with V and generates new session keys K_{CV} .

4) Ticket Granting Server (TGS): TGS issues tickets to clients who have been authenticated to AS. Thus, the client first requests a ticket-granting ticket ($Ticket_{tgs}$) from the AS. The client module saves this ticket. Each time the user requires access to a new service, the client applies to the TGS, using the ticket to authenticate itself. The TGS then grants a ticket for the particular service. The client saves each service-granting ticket and uses it to authenticate its user to a server each time a particular service is requested.The protocol makes use of timestamps T_C and T_V and the lifetime L.

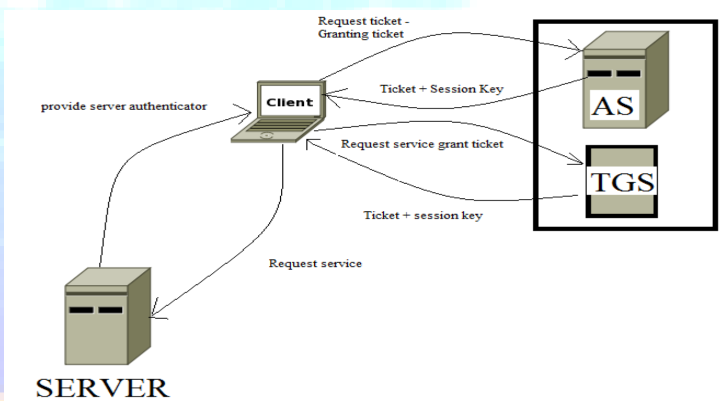

Fig. 1 Kerberos Authentication Protocol

The basic message exchanges [8] between these different entities involved in the authentication process according to Kerberos Version 4 are shown in Fig 2 and are given as follows:

1) $C \rightarrow AS : C \parallel TGS_{ID} \parallel TS_1$

In this message, the client requests a ticket-granting ticket by sending its identity and password to the AS, together with the identity of TGS, indicating a request to use the TGS service and a time stamp, so that the AS knows that the message is timely.

2) $AS \rightarrow C : E(K_C , [K_{C,tgs}\ kTGS_{ID}\ kTS_2\ kLifetime_2\ kTicket_{tgs}])$ $Ticket_{tgs} = E(K_{tgs}, [K_{C,tgs}\ kC\ kAD_C\ kTGS_{ID}\ kTS_2\ kLifetime_2\ ])$

The AS responds with a message, encrypted with a key derived from the user's password that contains the ticket requested by client. This message also contains session key $K_{C,tgs}$ to be shared between TGS and Client. The client retrieves this key.

The ticket is encrypted with TGS's key K_{tgs} which is already shared with AS. The ticket contains session key $K_{C,tgs}$ which is retrieved by TGS. Like this, the session key has been securely delivered to both C and the TGS. The ticket also contains AD_C (Network Address of client) which prevents the use of ticket from workstation other than one that initially requested the ticket.

3) $C \rightarrow TGS : V\ kTicket_{tgs}\ \ kAuthenticator_C$

$Authenticator_C = E(K_{C,tgs} [C\ kAD_C\ kTS_3\ ])$

The client sends message to TGS stating the identification:-

Requested service V, $Ticket_{tgs}$, an authenticator which includes the identity of client C, address of Client AD_C and a time stamp. The authenticator is intended for use only once and has a very short lifetime. The TGS can decrypt the ticket with the key that it shares with the AS. The TGS uses the session key $K_{C,tgs}$ to decrypt the authenticator. The TGS can then check the identity and address from the authenticator with that of the ticket and with the network address of the incoming message. If all match, then the TGS is assured that the sender of the ticket is indeed the ticket's real owner.

4) $TGS \rightarrow C : E(K_{C,tgs} [K_{C,V}\ kV\ kTS_4\ kTicket_V\ ])$

$Ticket_V = E(K_V [K_{C,V}\ kC\ kAD_C\ kV\ kTS_4\ kLifetime_4\ ])$

The TGS then sends C above message as a response to request:-

This message is encrypted by session key shared between TGS and client and includes a session key to be shared between Client and the server V, the identification of V, and the time stamp of the ticket. The ticket also includes session key to be shared between client and server V,and it is encrypted with V's key K_V .

5) $C \rightarrow V : Ticket_V \, \| \, Authenticator_C$

$Authenticator_C = E(K_{C,V} [C$

$\| AD_C \| TS_5]$

In this message the client sends the ticket received by TGS to server V. Client also sends an authenticator. The server can decrypt the ticket, recover the session key, and decrypt the authenticator.

6) $V \rightarrow C : E(K_{C,V} [TS_5 +1])$

The server V sends the value of the time stamp fromthe authenticator, incremented by 1, encrypted in the session key K c, v. C can decrypt this message to recover the incremented time stamp. Because the message was encrypted by the session key, C is assured that it could have been created only by V.

III. **MODEL OF PROTOCOL:**

Modeling the Kerberos Protocol through NuSMV Model checker will ease to find any attack on the protocol by an intruder. To model the Kerberos Protocol in NuSMV we have considered four entities, AS - the Authentication Sever, TGS - the Ticket Granting Server, V - Server, C - Client. These entities will communicate with each other and send message to each other as described in Section II. Each entity exchanges a set of messages with other entities in the model. Apart from these mentioned entities there is one more entity I - the Intruder for the purpose of checking the protocol against any attack.

TABLE I

TABLE OF COUNTERS

Counter Name	Purpose
c as count s	counts the number of messages sent from C to AS
c as count r	counts the number of messages C received from AS
c tgs count s	counts the number of messages sent from C to TGS
c tgs count r	counts the number of messages C received from TGS
c v count s	counts the number of messages sent from C to V
c v count r	counts the number of messages C received from V
as nt s	counts the number of messages sent from AS to C
as count r	counts the number of messages received by AS from C
tgs count s	counts the number of messages sent from TGS to C
tgs count r	counts the number of messages received by TGS from C
v count s	counts the number of messages sent from V to C
v count r	counts the number of messages received by V from C

Each entity viz. AS, TGS, C and V is associated with set of processes. Each process instantiates these entities and all processes work asynchronously and concurrently. To represent this behavior, the entities are mapped to the various states of the finite state automata given in Fig 1. Each entity performs some operation and after performing that operation it moves to a specific state. These operations are mapped to transitions of the finite automata correspondingly. In our model, the states of each entity are represented by a variable named state which takes following values for all the entities:

1) Client(C) : idle, send1, receive1, send2, receive2, send3, receive3
2) Authentication Server (AS) : idle, send1, receive1
3) Ticket Granting Server (TGS) : idle, send1, receive1
4) Server (V) : idle, send1, receive1

5) Intruder (I) : idle, send2, receive2, send3, receive3, eaves- drop, remove, generate, store

For each entity a set of counters is maintained which counts the number of messages sent and received. These counters given in Table I, help to check for the authentication property of the protocol. Depending on the transmission and reception of messages the various entities go from one state to another. Based on these transitions the normal operation of the protocol, in the absence of intruder, can be stated as follows:

1) Initially the state of each entity will be idle.

2) When the client entity C wants some service from the server V, client first authenticates itself to the authen- tication server AS. So, the client will send a message to authentication server AS. Now, the client's state is changed to 'SEND1' and increment value of the counter c_as_count_s.

3) When AS will receive the message from client, its state is changed from idle to 'RECEIVE1' and its counter as_count_r is incremented.

4) AS will reply to client in the form of ticket granting ticket and changes its state from 'RECEIVE1' to 'SEND1' and increments it's another counter as_count_s. AS will also maintain a timer to this ticket.

5) After getting response from AS, client will changed it state form send to 'RECEIVE1' and increments its counter c_as_count_r.

6) Now client will send the ticket received in the reply from AS to the Ticket Granting Server (TGS) and changes its state from 'RECEIVE1' to 'SEND2' and increments its counter c_tgs_counts.

7) Now TGS will change its state from 'IDLE' to 'RE- CEIVE1' and increment its counter tgs_count_r.

8) TGS replies to the client in form of server granting ticket and changes its state from 'RECEIVE1' to 'SEND1'. TGS also increments the counter tgs_count_s.

9) Now, client has got the ticket from TGS and changes its state from 'SEND2' to 'RECEIVE2' and increases the value of counter c_tgs_count_r.

10) Now, client sends this ticket to server V and changes its state from 'RECEIVE2' to

'SEND3' and increments its counter c_v_count_s.

11) The server V receives the ticket from the client C and changes its state from 'IDLE' to 'RECEIVE1'. It also increments its counter v_count_r.

12) Now, V replies to the client C and changes its state from 'RECEIVE1' to 'SEND1' and increments its counter v_count_s.

13) Finally, the client C on receiving the reply from the server V changes its state from 'SEND3' to 'RECEIVE3' and increments its counter value c_v_count_r.

To model the presence of an intruder following assumptions for finding thebehavior of protocol are required:

1)An intruder will be able to perform every operation with respect to communication.

2) An intruder can eavesdrop each message.

3) An intruder can remove the message sent to other entities.

4) An intruder can impersonate itself as the client or server.

5) An intruder is able to generate new messages.

The various transitions involved in modeling the presence of the intruder are :

1) When client C sends message to server V, and changes its state to 'SEND3', at that time intruder can read the message.

2) Intruder can remove the message and change its state to

'RECEIVE3' and send new message to server V.

3) Server V on receiving the message from intruder changed its state to 'RECEIVE3'. Here the intruder is impersonates as server V.

The CTL Specifications for Authentication are :

Context	Index	Select	Value	Trace	Type	Property
Main						
	0	☑	True		CTL	c.c_as_counter_s = as.as_counter_r
	1	☑	True		CTL	c.c_as_counter_r = as.as_counter_s
	2	☑	True		CTL	c.c_tgs_counter_s = t.tgs_counter_r
	3	☑	True		CTL	c.c_tgs_counter_r = t.tgs_counter_s
	4	☑	True		CTL	c.c_v_counter_s = v.v_counter_r
	5	☑	True		CTL	c.c_v_counter_r = v.v_counter_s

Fig. 3. CTL Specifications

Figure 3 shows the specifications with respect to authentication.

1) AG(C.c_as_count_s = AS.as_count_r)

2) AG(AS.as_count_s = C.c_as_count_r)

The above stated pair of specifications describe that if the client ends n number of protocol executions with the AS as responder, then in past the AS must have started at least n protocol executions with the client as initiator.

3)AG(C.c_tgs_count__s = T GS.tgs__count_r) AG(TGS.tgs_count__s = C.c_tgs count_r)

The above stated pair of specifications describe that if the client ends n protocol executions with the TGS as responder, then in past the TGS must have started at least n protocol executions with the client as initiator.

4) AG(C.c_v_count_s = V.v_count_r) AG(V.v_count_s = C.c_v_count_r)

The above stated pair of specifications describe that if the client ends n protocol executions with the V as responder, then in past the V must have started at least n protocol executions with the client as initiator.

B. Secrecy and Integrity

The CTL Specifications for Secrecy and Integrity are :

1)AG(c.send_$K_{C,V}$ = **I**.receive_$K_{C,V}$)

2)AG(c.send_$K_{C,tgs}$ = **I**.receive_$K_{C,tgs}$

3)AG(as.send_$K_{C,tgs}$ = **I**.receive_$K_{C,tgs}$

4)AG(tgs.send_k_c, v = **I**.receive_k_c, v)

The above stated pair of specifications describe that an Intruder can receive the keys send to it. Thus I cannot get session keys from an entity X more times than these keys have been sent to it. Thus since I is not authorized member of the system, nobody send keys to it. Thus **I**.receive_$K_{C,V}$, **I**.receive_$K_{C,tgs}$, **I**.receive_$K_{C,tgs}$, **I**.receive_k_c, v are all zero.

C. Replay Attack

Figure 4 explains the replay attack being done by the intruder I by capturing the ticket $Ticket_V$ when the client C sends it.

as.state	c.key	c.state	i.exist	i.icounter	i.state	t.key	t.state	v.key	v.state
idle	k_c	idle	0	0	idle	k_tgs	idle	k_v	idle
idle	k_c	send1	1	0	idle	k_tgs	idle	k_v	idle
receive1	k_c	wait	0	0	idle	k_tgs	idle	k_v	idle
send1	k_c	wait	0	0	idle	k_tgs	idle	k_v	idle
idle	k_c	receive1	1	0	idle	k_tgs	idle	k_v	idle
idle	k_c	send2	1	0	idle	k_tgs	idle	k_v	idle
idle	k_c	wait	0	0	idle	k_tgs	receive1	k_v	idle
idle	k_c	wait	0	0	idle	k_tgs	send1	k_v	idle
idle	k_c	receive2	1	0	idle	k_tgs	idle	k_v	idle
idle	k_c	send3	1	0	idle	k_tgs	idle	k_v	idle
idle	k_c	wait	0	0	receive1	k_tgs	idle	k_v	receive1
idle	k_c	wait	0	0	remove	k_tgs	idle	k_v	send1
idle	k_c	receive3	1	0	idle	k_tgs	idle	k_v	idle
idle	k_c	send2	0	0	idle	k_tgs	idle	k_v	idle
idle	k_c	wait	0	0	idle	k_tgs	receive1	k_v	idle
idle	k_c	wait	1	0	idle	k_tgs	send1	k_v	idle

Fig. 4.Attack

to the server V. The intruder I captures this ticket and sends it to the server V after a some period of time impersonating as the client C. When the server receives this packet it thinks that the client C is trying to setup two sessions with it, in fact the client C actually tries to setup only one session with the server with the other session belonging to the intruder I.

IV. CONCLUSION:

In this paper, we have verified and modeled Kerberos Authentication Protocol using NuSMV model checker. We also considered the presence of Intruder in the system and found out the possible replay attack between various entities. Using NuSMV tool we found a flaw with respect to replay attack and moreover through step by step tracing of execution path any other flaws in the model can be detected and solutions can be suggested using model checking. We have used the correspondence property to obtain CTL specifications of all the security requirements of the given protocol with the help of various counters. In future this method of modeling and verifying of the protocol can be extended to Kerberos Version 5 authentication protocol.

REFERENCE:

- S.P.Miller, B.C.Neumann, J.I.Schiller, and J.H.Saltzer. (1988, October) Kerberos authentication and authorization system. [Online]. Available: http://web.mit.edu/Saltzer/www/publications/athenaplan/e.2.1.pdf
- R. Cavada, A. Cimatti, C. A. Jochim, G. Keighren, E. Olivetti, M. Pistore, M. Roveri, and A. Tchaltsev. (2005) Nusmv 2.4 user manual. [Online]. Available: http://nusmv.fbk.eu/NuSMV/userman/v24/nusmv.pdf
- O. G. Edmund M. Clarke and D. A. Peled, Model Checking. Cambridge, USA: MIT-PRESS,, Jan. 1999.
- (2011) Nusmv 2.4. [Online]. Available: http://nusmv.fbk.eu/
- A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "Nusmv 2: An opensource tool for symbolic model checking," in Proceedings of the 14th International Conference on Computer Aided Verification, Copenhagen, Denmark, July 2002, pp. 241–268.

- M. Panti, L. Spalazzi, and S. Tacconi. (2007) Using the nusmv model checker to verify the kerberos protocol. [Online]. Available: http://www.inform.unian.it/personale/spalazzi/repo
- S. Adyanthaya, S. Rukmangada, A. Tiwari, and S. Singh, "Modeling freshness concept to overcome replay attack in kerberos protocol using nusmv," in Computer and Communication Technology (ICCCT), 2010
- International Conference on, Allahabad, India, Sept. 2010, pp. 125–129. [8] W. Stallings, Cryptography and Network Security Principles and Practices, 5th ed. New Delhi, India: Prentice Hall, 2010.
- M. Huth and M. Ryan, Logic In computer Science. Cambridge,U.K.: Cambridge University Press, 2004.