International Journal of Enhanced Research in Management & Computer Applications, ISSN: 2319-7471
Vol. 3 Issue 3, March-2014, pp: (34-41), Impact Factor: 1.147, Available online at: www.erpublications.com

Specification, Simulation and Validation of Kerberos Protocol against replay attack using SPIN

L. Benarous[1], M. Djoudi[2]
[12]Computer Science & Mathematics Department, University of Amar Telidji, Laghouat, Algeria

Abstract: The general context of this paper is the verification of the Kerberos authentication protocol and its immunity against the replay-attack. A Promela model of the protocoland the intruder executing a replay-attack was build; this model will be simulated and verified using the powerful tool spin. The results prove that the protocol is against the replay attack as it is announced.

Key words: Kerberos, modelling, simulation, verification, SPIN, PROMELA.

1. Introduction

One of the major challenges related to distributed systems and networks is the security. Many cryptographic protocols and algorithms were developed to ensure the security; some of them despite their strength were proven to have errors. Among the powerful protocols existing we choose the Kerberos [6] [7] authentication protocol based on the symmetric cryptography, the protocol is widely used and is announced to be against the replay-attack. Thus, this paper will be aimed to prove this assumption by modelling, simulating and verifying the protocol using SPIN[1].SPIN is powerful model-checker for distributed systems and protocols, the use of this tool to validate security was discussed in a general way in [8] and [9] shows with an example the use of spin to validate the security protocols such as Needham-Schroeder public key protocol.This paper will discuss in an original way the modelling of the Kerberos protocol and the intruder behaviour; as well as the simulation and verification results. The paper is organized as follow:section 2 presents the protocol; the section 3 will be devoted to the presentation of the specification language Promela and its tool Spin. Section 4describes the modeling of the protocol using PROMELA language. Section 5 will contain the simulation results; Section 6 will contain the verification results and the final part draws the conclusion.

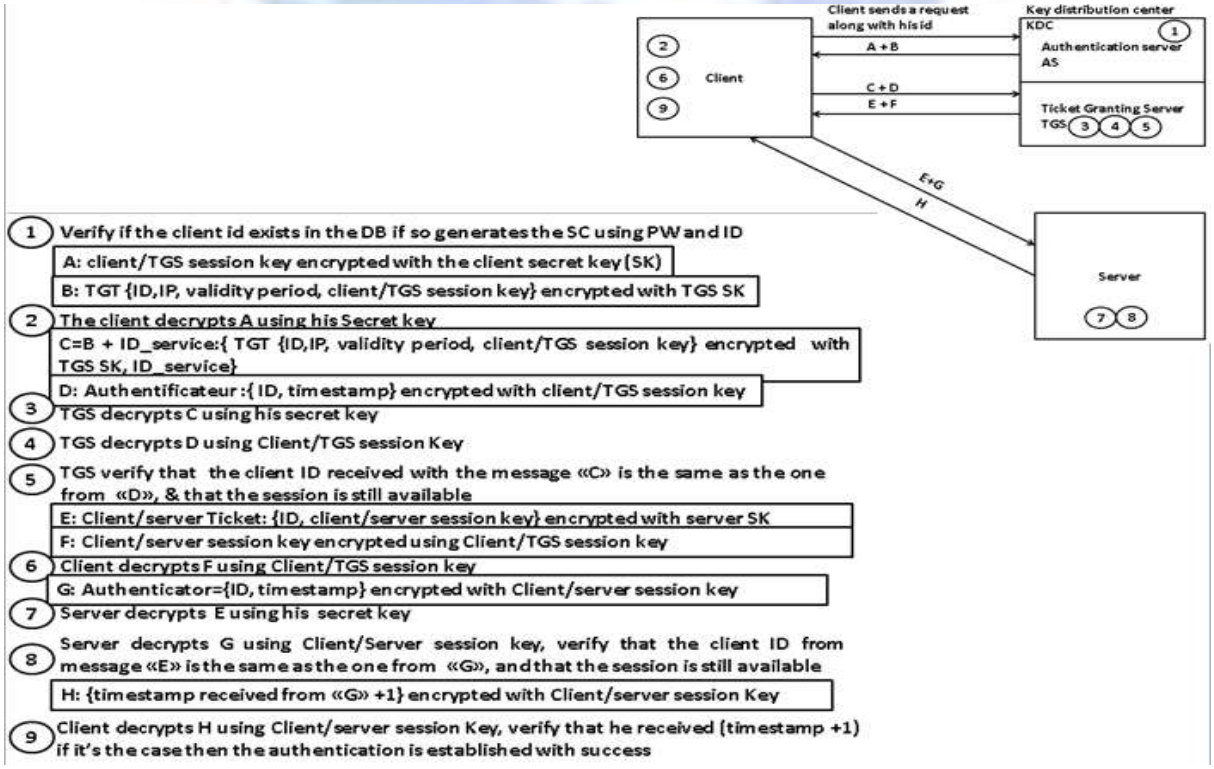

Figure1: Kerberos protocol

2. The Studied Protocol

2.1. The Protocol Definition

Kerberos is a network authentication protocol, it aims to avoid the transmission of the password in unsecure network, it's one of the results of the Athena project (MIT) developed by Miller & Neumann.

The name Kerberos comes from the Greek mythology and it refers to the three headed dog protecting the gates of Hades. Just like the name refers, Kerberos protocol is structured of 3 sets for the authentication process: the server, the client and the Key Distribution Center KDC. The protocol uses the symmetric cryptography based on the use of secret key, for this, Kerberos shares a secret key with each of its clients on the network. [5]

2.2. Protocol description

The protocol is based on the use of tickets; it provides a mutual authentication system between the client and the server to identify each other. The authentication has a limited period of time aiming to prevent the replay-Attack.To obtain a service, a client sends its ID to the authentication server (AS), this latter will verify if the ID exists in its database and sends back a ticket granting ticket (B) and a session key (A). The client then sends an encrypted request(C) and an authenticator (D). The Ticket Granting Server (TGS) checks the validity period of the session and the identity of the client (using C and D), then sends to the client a ticket granting service (E) and a session key. The client sends an authenticator (G) and message E to the server, the server checks the identity of the client (E, G) and the validity period (G) and sends H. the client checks H and ends the authentication and starts the service. The detailed steps of the process are shown in figure 1.For more details visit [6] [7]

3. The Promela language and its associated tool Spin

3.1. The Promela language

PROMELA (PROcessMEta Language) is a specification and verification language for concurrent and distributed systems.[1] [2]

The specifications contains mainly:

Process

Describe the behavior of a system, can be declared using the keyword proctype:

```
ProctypeA(/*typeP parameter*/)
{// body}
```

The process may be instantiated by using the keyword Active, or the run command, as follow:

```
Activeproctype A()
{
//body
}
Or
Init{
RunA();
}
```

Channels

The channels are used for the message transmission. The send operator is !and the receive operator is ?.

A channel can be declared as follow:

Chan channel_name = [c] of {mtype}
Where c is the capacity of the channel and mtype is the message type.

Example: chan p = [3] of {int ,byte};The channel p can transmit 3 messages of type {int, byte};
A message can contain constant which can be declared using mtype.

Example:

- q!a,b: /*send the message containing a & b through the channel q*/
- q?a,b: /*receive the message from channel q and save its content in variables a, b*/

The variables

The variables in Promela can be declared globally or locally, the declaration syntax is the same as in c language. Thus, a variable can be of a predefined type {bit, Boolean, byte, short, integer, unsigned} [1] [4], it can be of a complex type like arrays or can be defined by the user using the typedef (structures).

The constants can be defined as in c by using #define
The Promela language can be used in the verification, because it allows the expression of LTL (Linear Temporal Logic) properties inside the Promela model[3].A Promela specification can contain C code fragments, has the extension .pml and can be interpreted by the simulator and the verifier SPIN.

3.2. The Spin tool

Spin (Simple Promela INterpretor) is open-source software verification tool developed in Bell Labs by Gerard Holzmann. The software has been available freely since 1991, and continues to evolve to keep pace with new developments. In April 2002 the tool was awarded the ACM System Software Award.
The tool can be used in four main modes:[1]

- As a simulator: offering three types of simulation; the random, the interactive and the guided simulations
- As an exhaustive verifier
- As proof approximation system
- As a driver for swarm verification.

The spin tool has 3 graphical user interfaces:

- The ispin
- The jspin[3]
- The xspin

We used in this article the ispin graphical user interface.

4. The modeling process

To simplify the model we imposed some assertions:

- The messages A, C and F along with the treatment related to them were not considered in our model.
- The encryption and decryption weren't modeled.
- The cryptographic system used in the encryption and decryption was out of our study.
- The structure of the messages B, D, E, G and H didn't contain an explicit field for the secret keys transported
- within it.
- In the model, we used P (from protected) + the key to refer that the message was protected with this key.
- Example:PCTGSSessK referring to protected by c/TGS session key.
- The simplified protocol we modeled is presented in Figure 2.

The model contains the 3 main processes: the Client, the KDC and the Server as well as an intruder who executes the replay-Attack (by delaying the response and by multiple send to confusing the system).
We started the model by defining the constants we need, by using mtype: mtype
{CLIENT, KDC, SERVER, PCTGSSessK, PServerSK, PCLIENTServerSessK, PTGSSK, trusted}

Next, we defined the channels that will transport the messages:
chan q1= [3] of {mtype, byte}; transporting the first message.
chan q2= [17] of {mtype,byte,int, mtype}; for transporting the messages B and D.
chan q3= [3] of {mtype,mtype}; the trust messages.

chan q4= [3] of {int ,mtype}; transporting the message H.
The client starts his session by sending a request including his ID, the message will be transported in the channel q1: q1! CLIENT, identifier; the KDC will send message B as soon as the 1st message is received:q1? CLIENT, id -> q2! CLIENT, id, validity, PTGSSK;the client then will send the message D:

q2! KDC, identifier, timestamp, PCTGSSessK; the KDC will verify the identity of the client and the validity of the session by comparing the current time (t) with the received time plus the validity period, if the conditions are satisfied the message E will be sent to the client:
if
:: (id1==idD) && (t<time+ validity) -> q2! CLIENT, id, validity, PServerSK; /*E sent */
fi;
Note: The timestamp was calculated from the time and transmitted as an integer value.
The Client has finished the 1st authentication stage by obtaining the ticket granting service (E), and starts the mutual identification with the server to ensure a mutual trust. The client will send the messages E and G to the server:
If
:: q2? CLIENT, eval(identifier),v, PServerSK ->
q2! SERVER, identifier, v, PServerSK;
q2! SERVER, identifier, timestp, PCLIENTServerSessK;
Fi;

The server will check the client identity and the validity of the session and sends back the message H: q4! Timestamp, PCLIENTServerSessK; the timestamp= timestamp from G +1. The client will check that the timestamp received is equal to the one from E + 1, with that the trust is established between the server and the client; the services can be requested and guaranteed. The client sendsq3! SERVER, trusted; and the server as soon as receiving this message will send a trust message as wellq3? SERVER, trusted -> q3! CLIENT, trusted; The protocol modeled specifies the authentication process, to prove that it is against the replay –attack, we tried to specify the behaviour of the attacker and its interaction with the protocol. The intruder will aim to attack the last phase of authentication; it's the part where the client authenticates its self to the server, after obtaining the ticket granting service from the KDC. The intruder tries to intercept the messages E and G from the client and try to:

- Send them many times to confuse the systems. Or,
- Send them later.

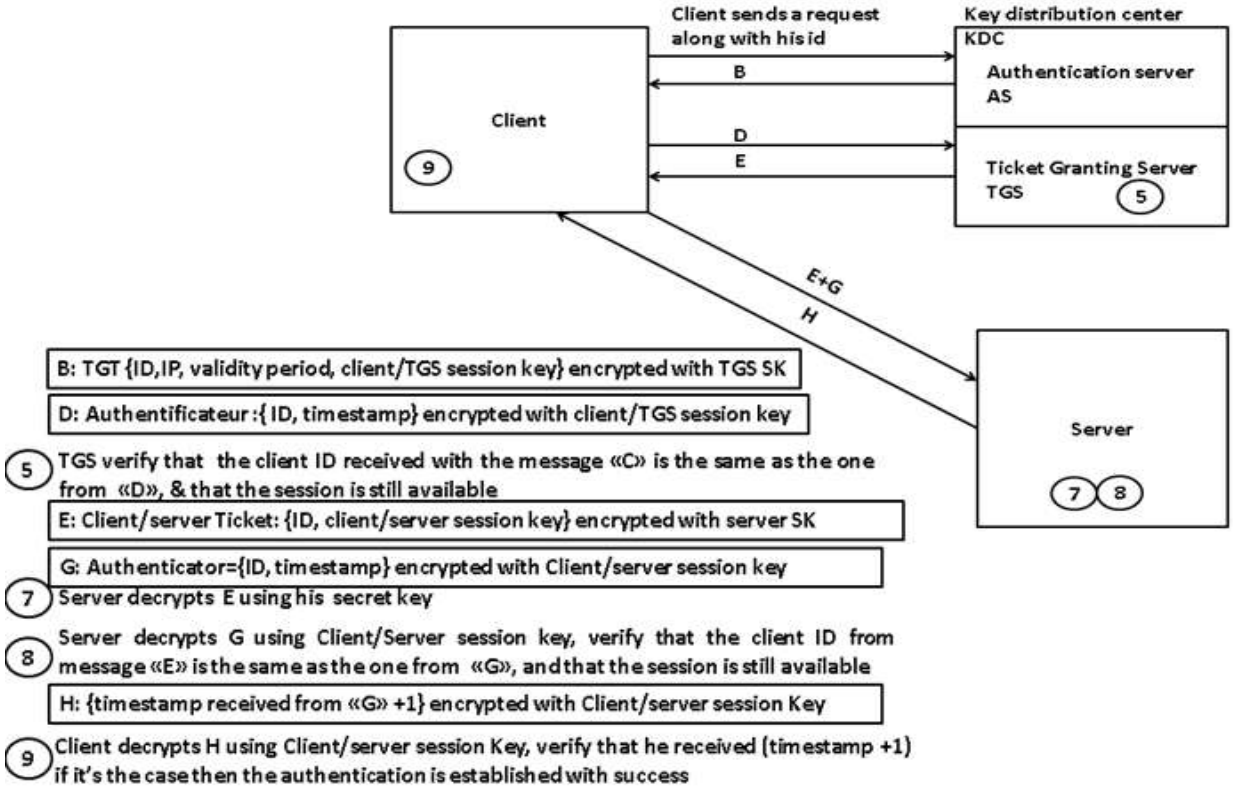

Figure 2: Kerberos simplified model

The protocol focuses mainly on the use of timestamp to check the validity of the session, and a cache to ensure that the message will be received once only to avoid the system inundation. If an intruder succeed to intercept the messages E & G the client will be blocked and the intruder will be trusted instead, so the part of sending E & G will be the same as in the client model. The intruder will eventually wait if it is a delay attack so the time will be greater than the timestamp + validity period and will eventually be blocked by the server. If it's a multiple send it will be blocked as well after the verification of the cache value, in our model it's the received value. For this, we added in the Server code the test of (received< 1) before receiving the messages E & G. and the test of the session availability before sending the message H.
The attacker will send many messages trying to confuse the system, here E & G will be sent ten times to the server, and for this we used a loop:

```
do
:: j <10 ->printf("trying to confuse the server");
q2! SERVER ,identifier,v,PServerSK;
q2! SERVER ,identifier,timestp,PCLIENTServerSessK; j=j+1
:: j>=10 -> break;
od;
```

For the delay attack, we imposed a waiting time before the intruder can send E & G.

5. The Simulation Phase

Three scenarios were obtained from the random simulation; the 1st one shows the expected behaviour of the protocol, the client aiming to obtain a service is authenticated without intrusion; figure 3 will show the simulation result given by spin.

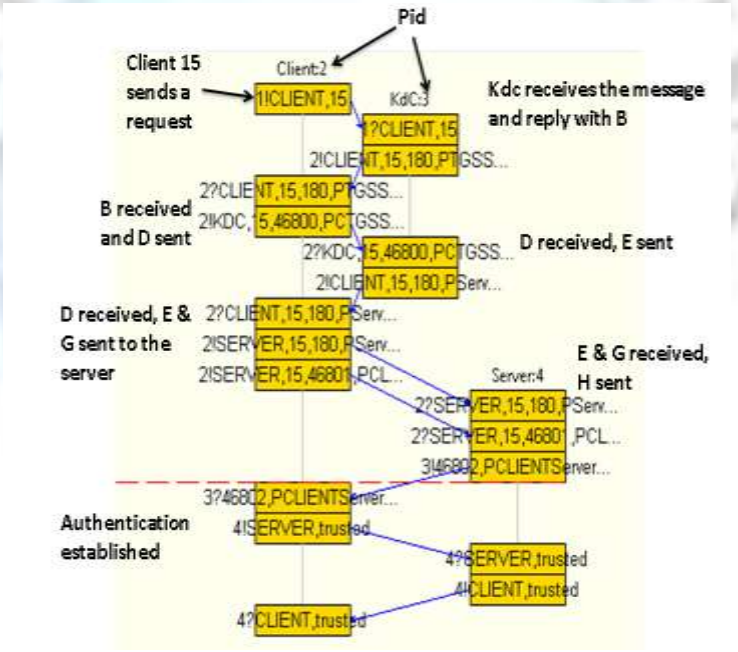

Figure 3: Kerberos protocol simulation S1

The 2nd scenario concerns the intruder executing the 1st type of replay attack trying to confuse the system by multiple send but it gets blocked by the server. As shown in figure 4. The 3rd scenario is when the intruder executes the replay attack by late reply, blocking the client and trying to authenticate later to the system, it gets blocked by the server that verifies the session validity.

```
60:    proc  5 (Intruder) KAt.pml:114 (state 53) [((j<10))]
61:    proc  4 (Server) KAt.pml:69 (state 9) [assert((received==1))]
62:    proc  2 (Client) KAt.pml:44 (state 19) [t = timestp]
63:    proc  4 (Server) KAt.pml:70 (state 18) [else]
64:    proc  2 (Client) KAt.pml:44 (state 20) [t = (t+1)]
end of session client hacked 65: proc  4 (Server) KAt.pml:73 (state 17) [printf('end of session client hacked')]
```

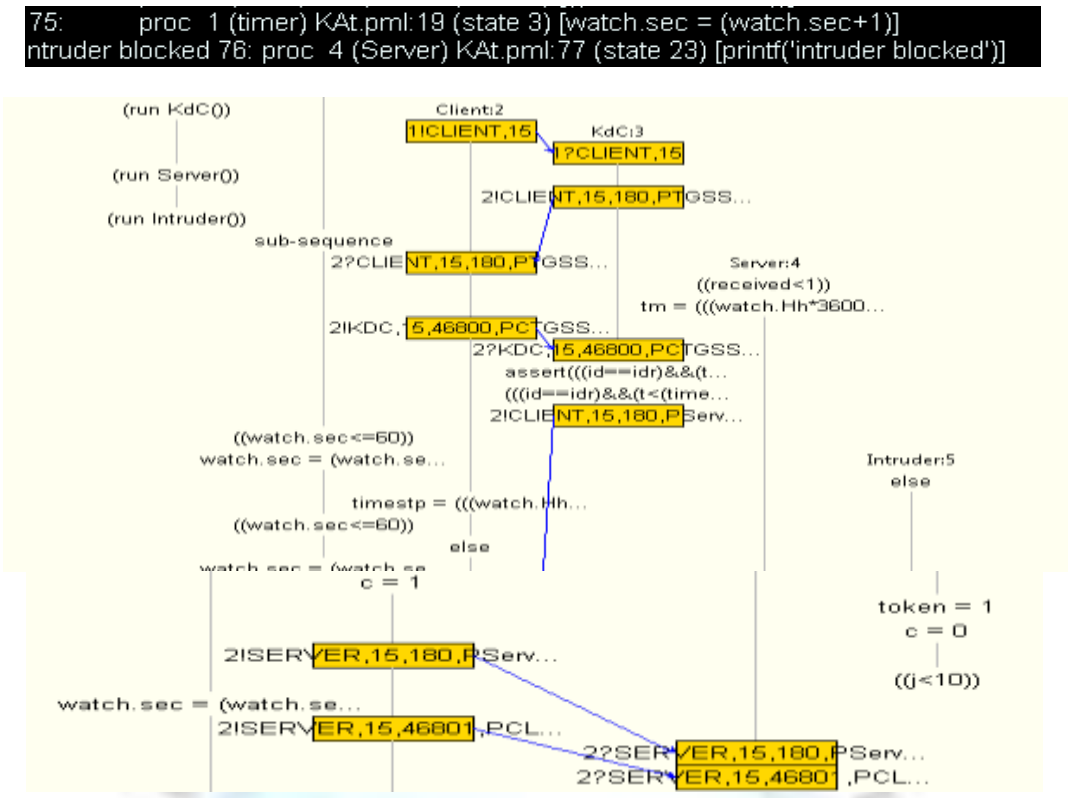

Figure 4: Kerberos protocol simulation S2

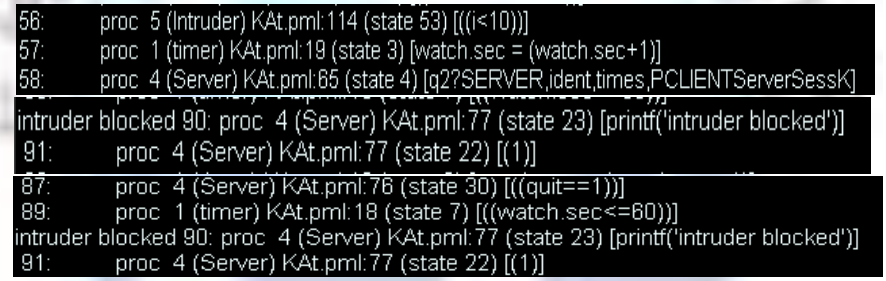

Figure 5: Kerberos protocol simulation S3

6. The Verification Phase

This paragraph is intended to show that our model is exempt from errors and that all the desired properties are satisfied. To achieve this objective, we used the functionality label offered by Promela, the labels used are:[1][3]

- **Trace:** to show all send and receive of the messages, was used in the specification as follow:

```
trace3: {
q2? SERVER ,idt,v,PServerSK; /*E received*/
q2?SERVER,ident,times,PCLIENTServerSessK; /*G received*/}
```

- **Accept:** to show that a part of the model must be executed when reached. As in the client code:

```
accept1:{
if
::  q2? CLIENT,eval(identifier),v,PServerSK ->
    q2! SERVER ,identifier,v,PServerSK;
    q2!SERVER,identifier,timestp, PCLIENTServerSessK;
fi;
…}
```

- **End:** to indicate the abnormal end of process, as in the intruder code:

```
end4: {q2?CLIENT,eval(identifier),v,PServerSK ; /*message E*/
…}
```

Explanation: Due to the fact thatthe intruder may and may not be able to intercept message E and start its attack. The end label was put in the specification so as it allows the analyser to end the intruder execution at this level (if the client authenticates successfully to the server and the intruder fail to intercept the message E)

- **Assert:** to verify the correctness of the statement, such as the validity period value (assert(v==180)), the session availability, the identity of the client (assert(id==idr&&t<time+validity);)…
The protocol was validated using SPIN;we checked the safety and liveness properties. More precisely, the absences of deadlock, assertion violation, non-progress cycles and accept cycle.No errors were found, which denotes that the protocol is strong and respond efficiently against replay attack (delay response, multiple send).
The figure 6 will show a screen recap from the verification.

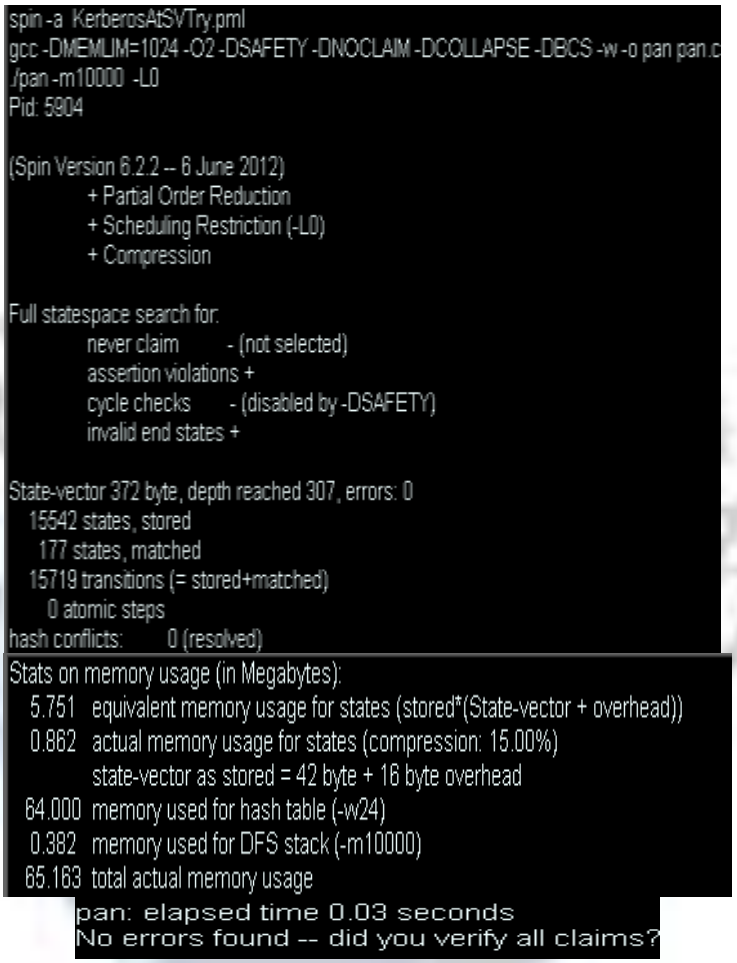

Figure 6: Verification of safety propriety

Conclusion

This paper concludes a part of the work done in a license graduation project.We choose to model the Kerberos authentication protocol, widely known to be strong against the replay attack. This was achieved by verifying the protocol's behaviour and its interactions with the attacker. Thus, we studied and modelledthe intruder's behaviour. The modelling of the kerberos protocol and the intruder were specified from the protocol rules and by understandingthe different steps of the replay-attack, the specification was done using the modelling language PROMELA, the simulations and verification were done using the simulator and analyser SPIN.We concluded after verification and intensive simulation that a client can be comforted and secured from a replay attack when using this protocol.

References

[1]. Holzmann G J. Spin Model Checker, ThePrimer and Reference Manual: Addison-Wesley Professional, 2003.
[2]. Holzmann G j. Design and Validation of Computer Protocols: Prentice Hall, 1991
[3]. Ben-Ari M.Principles of the Spin Model Checker, London: Springer, 2008.
[4]. Petrucci L. Un exemple de langage parallèle asynchrone : Promela, 1999.
[5]. Garman J. Kerberos, The Definitive Guide, O'reilly, 2003
[6]. Miller S P, Neuman B C, Schiller J I et al.Kerberos Authentication and Authorization System:

ftp://athena-dist.mit.edu/pub/kerberos/doc/techplan.txt.
[7]. Kohl J, NeumanC. The Kerberos Network Authentication Service:ftp://ftp.isi.edu/in-notes/rfc1510.txt.
[8]. JosangA. Security protocol verification using SPIN, SPIN'95 Workshop, 1995
[9]. Maggi P, Sisto R. Using SPIN to verify security Properties of cryptographic protocols, 2002.