Specifying and Checking Network Protocol Based on TLA

Liang Wan\textsuperscript{1,2,3}  
\textsuperscript{1}School of Information  
Renmin University of China  
Beijing, China  
\textsuperscript{2}School of Computer Science and Information  
Guizhou University  
Guiyang, China  
wanliang@ruc.edu.cn  

Wenchang Shi\textsuperscript{1,3}  
\textsuperscript{3}Key Lab of Data Engineering and Knowledge Engineering of Ministry of Education  
Renmin University of China  
Beijing, China  
wengchang@ruc.edu.cn

Abstract—Network protocol vulnerability detection is paramount to network security. Formalization provides an important way for vulnerability detection. In this paper, we apply TLA, which is a powerful tool for formal analysis, to analyze network protocols. An approach is proposed that aims to detect vulnerabilities of a protocol effectively with the TLA, with the Kerberos protocol being taken as an example. Firstly, roles for the protocol, especially those related to intruders, are created. Then actions of the roles are specified. Sessions among the roles are built. And environment parameters are set. A prototype program is developed to implement the approach, which covers the model and the detection properties of the protocol. Experiments show that our approach is effective and powerful in specifying and checking a protocol, and it is better than SPIN and SMV.

Keywords—TLA; Action; Protocol; Model Checking; Security

I. INTRODUCTION

Formal methods use mathematic and logic method to describe and validate the system. Its description includes two aspects. One, which is also called modeling, is the description of systematic action. It describes the system and action mode through the model with systematic structure. The other one, which is also called specification, is systematic property. It shows the property which the system has, such as safety property, liveness property [5] and so on.

Model checking was first proposed by Clarke and Emerson. It is one of the formal methods. It can be applied to a number of very important systems, and find the defects in the hardware or the software. Given a model of a system, using the tools of model checking, it will test automatically whether this model meets a given specification. And the specification contains safety requirements such as the absence of deadlocks and similar critical states that can cause the system to crash. Model checking is a technique for automatically verifying properties of systems.

TLA (Temporal logic of actions) [1] is logic for specifying and checking concurrent systems, it is brought forward by Lamport.L. It is based on temporal logic and action logic. And it can specify a system’s modeling and property together, this is different from others famous model checking tools, such as SMV, SPIN. It is very useful of system model and their properties are represented in the same logic. It is extremely powerful, both in principle and in practice.

Detecting the loopholes of a protocol is difficult thing. We try to find a method to detect the defects of a protocol. This need the steps of creating the roles for Kerberos protocol, specifying the actions of the roles, building the sessions among the roles, writing the program for it, and checking the program. The key is to specify the security properties of the protocol. We add the security detecting properties in the model program and repeatedly checking and modifying the programs. Through this method we find the security defects of the protocol. It shows the method is effective and powerful.

II. THE SYNTAX AND SEMANTICS OF TLA

Definition 1. A labeled transition system [3] $T=(Q,I,A,L)$, in which,  
$Q$ is a (finite or infinite) set of states,  
$I$ is a set ($I \subseteq Q$) of initial states,  
$A$ is a set of actions, and  
$L$ is a transition relation $L \subseteq Q \times A \times Q$.

Definition 2. Primed value [1,2]: The Primed value is another value in the next state of the value.

Definition 3. Action [1,2]: An action is a Boolean expression which include primed value, variable, constant, for example $hr'=hr+1$,while the state transition caused by several actions.

Definition 4. Stutter action [1,2]: Action $A$ is called stutter action, if and only if $A'=A$.

Definition 5. Behavior [1,2]: Behavior consist of infinite states sequence, denoted as $\sigma$, a state is denoted as $\sigma_i$:

\[
\begin{align*}
\sigma & \triangleq \sigma_0 \rightarrow \sigma_1 \rightarrow \sigma_2 \rightarrow \cdots \\
\sigma^n & \triangleq \sigma_n \rightarrow \sigma_{n+1} \rightarrow \sigma_{n+2} \rightarrow \cdots
\end{align*}
\]
Definition 6. For any action A, we define Enabled A to be the predicate that is true for a state iff it is possible to take an A step starting in that state [3].

\[ \text{Enabled } A \equiv \exists \gamma_1, \ldots, \gamma_n : A(c_1/v_1', \ldots, c_n/v_n') \]

Definition 7. WF_r(\langle A \rangle) [1,2]:

\[ \text{WF}_r(\langle A \rangle) \equiv \Box \Box \neg \text{Enabled } A_r \lor \Box \Box \langle A \rangle_r \]

Definition 8. SF_r(\langle A \rangle) [1,2]:

\[ \text{SF}_r(\langle A \rangle) \equiv \Box \Box \neg \text{Enabled } A_r \lor \Box \Box \langle A \rangle_r \]

Definition 9. P \rightarrow Q [1]: P \rightarrow Q \equiv (P \rightarrow Q) \equiv \forall n \in \mathbb{N} : (\sigma^n = P) \rightarrow \exists m \in \mathbb{N} : (\sigma^{n+m} = Q)

Definition 10. A run of T is a finite or infinite sequence \( \rho = q_1 A q_2 A \cdots A q_i \), where \( q_i \in I \) and \( (q_i, A, q_{i+1}) \in \mathcal{L} \) holds for all i [3].

Definition 11. \( \sigma[n] \) [1,2]: Given a sequence \( \sigma = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots \), we write \( \sigma[n] \) to denote the prefix \( s_0 \rightarrow s_1 \rightarrow \cdots 
\)

Definition 12. For a property \( \Phi \) and a finite sequence \( \rho = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n \), we write \( \rho \models \Phi \) iff \( \rho \models \sigma \in \Phi \) for some finite sequence \( \sigma \) [3].

Definition 13. \( \sigma \models \Phi \) [1,2]: Let Q and A be sets of states and actions. A \( (Q, A) \)-property \( \Phi \) is a set of \( \omega \)-sequences \( \sigma = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_i \rightarrow \cdots \), where \( s_i \in \Phi, A \in A \). We inter-changeably write \( \sigma \models \Phi \) and \( \Phi \models \sigma \).

Definition 14. Liveness property[1,2]: \( \Phi \) is a liveness property iff \( \sigma[n] \models \Phi \) for all \( \sigma \) and \( n \in \mathbb{N} \).

Definition 15. \( \Phi \) is a safety property iff for any infinite sequence \( \sigma : \models \Phi \) if \( \sigma[n] \models \Phi \) for all \( \sigma \) and \( n \in \mathbb{N} \) [3].

TLA mainly support LTL (Linear time Temporal Logic), the main temporal operators are \( \Box \) (always), and \( \Diamond \) (eventually), F is a formula, and P is a predicate. The following is the relation between behavior and action.

III. SPECIFYING THE SYSTEM MODEL ON TLA

TLA is the logic based on temporal logic and action logic. In the late 1980's, Lamport invented TLA, the Temporal Logic of Actions: a simple variant of Pnueli's original logic. TLA makes it practical to describe a system by a single formula. Most of a TLA specification consists of ordinary, nontemporal mathematics. Temporal logic plays a significant role only in describing those properties that it's good at describing. TLA also provides a nice way to formalize the style of reasoning about systems that has proved to be most effective in practice: a style known as assertion reasoning.

TLA+ is quite good for specifying a wide class of systems: from program interfaces (APIs) to distributed systems. It can be used to write a precise, formal description of almost any sort of discrete system. It's especially well-suited to describing asynchronous systems: that is, systems with components that do not operate in strict lock-step.

We specify a system using TLA+. In TLA+, system specifications are of the form:

\[ \text{Init} \land \Box \langle \text{Next} \rangle, \land \text{Liveness} [1-3] \]

\text{Init} : State formula describing the initial state(s)
\text{Next} : Action formula formalizing the transition relation usually a disjunction \( A_1 \lor A_2 \lor \cdots \lor A_n \) of possible actions (events) \( A \).

\text{Liveness} : Temporal formula asserting liveness conditions usually a conjunction \( \text{WF}_r(\langle A \rangle) \land \cdots \land \text{SF}_r(\langle A \rangle) \) of fairness conditions.

IV. CHECKING A PROTOCOL

For checking a protocol, we might take the steps as follow:


AVISPA is a share-cost RTD (FET open) project, funded by the European Commission under the Information Society Technologies Programme operating within the Fifth Framework Programme, started on January 1st, 2003.

The AVISPA Tool provides a suite of applications for building and analyzing formal models of security protocols. Protocol models are written in the High Level Protocol Specification Language (HLPSL) [7] based on TLA for describing security protocols and specifying and checking their security properties.
of the concurrent system. In a protocol specification, finding the roles and their actions is primary, but the states and their transferring are major part in specification.

A. Setting the roles for a protocol

The roles are the major actors of protocol. When the roles of the protocol are defined, the actions are specified for the protocol. So the actors should firstly be made for a protocol, and this is not different from a concurrent system modeling. In specifying a concurrent system, the states and their transferring are the major parts in the specification of a concurrent system.

In Kerberos V5 protocol, the client authenticates itself to the Authentication Server (AS) which forwards the username to a Key Distribution Center (KDC). The KDC issues a Ticket Granting Ticket (TGT), which is time stamped, encrypts it using the user's password and returns the encrypted result to the user's workstation. If successful, this gives the user desktop access.

From the protocol we should define five roles. Where A is the client; B is the server; S is the Authentication Server; T is the Ticket Granting Server, and I is the intruder. Then we could specify the actions of the roles.

B. Building the model of the protocol and specifying the actions among the roles

Kerberos V5 protocol has six steps in the course of certification. We might build the model of the protocol as shown in figure.

![Figure 1. The Roles and certification course of Kerberos protocol](image)

Then we could specify the actions among the roles as follows.

1) The role A and its actions

When the A receives a B’s information, it produces an action to S, as Figure 2. (In the figure the dotted line represents the event; the solid line shows the behavior)

1. State = 0 \& RCV_BA(A,B) =|>
   State'= 2 \& Na':= new() \& SND_SA(A,B,\{Na'\}_Ka)

![Figure 2. The Role A and B,S](image)

When A receives information coming from B, it produces action, changes the state, and sends information to T. As shown in figure 3.

2. State = 2 \& RCV_SA(A,B,\{Kt.Na.Nt\}_Kt) =|>
   State'= 6 \& SND_BA(A,B,\{X'.Kt.Na.Nt\}_Kt')

![Figure 3. The Role A and S,T](image)

When A receives information coming from T, it produces action, changes the state, and sends information to B. As shown in figure 4.

   State'= 8 \& request(A,B,alice_bob_na,Na)

![Figure 4. The Role A and T,B](image)

2) The role B and its actions

When A receives information coming from B, it produces action, and changes the state. As shown in figure 5.

4. State = 6 \& RCV_BA(A,B,\{Na.Ns\}_Kn) =|>
   State'= 8 \& request(A,B,alice_bob_na,Na)

![Figure 5. The Role A and B](image)

The role A’s behavior and states transition. as figure 6.

![Figure 6. State transition of Role A](image)

3) The role S and its actions

When S receives information coming from A, it produces action, and changes the state. As shown in figure 9.

1. State = 1 \& RCV_AS(A,\{Na\}_Ka) =|>
   State'= 3 \& Na':= new() \& SND_SA(A,B,\{Na\}_Ka)

![Figure 7. The Role B and A](image)

The role B’s behavior and states transition. as figure 8.

![Figure 8. State transition of Role B](image)

   State'= 4 \& SND_TA(A,B,\{Kt.Na.Ns\}_Kt)

![Figure 9. The Role S and A](image)
4) The role T and its actions

When T receives information coming from A, it produces action, and changes the state. As shown in figure 10.

\[ \text{State} = 5 \land RCV\_AT(A.B, \{TG.T\} \_\_Kt.\{K\_\_Na\} \_\_K) \]

\[ \Rightarrow \text{State}' = 7 \land Nt' := \text{new}() \land Kn' := \text{new}() \land SND\_AT(A.B\.\{Kn'.Na.Nt\} \_\_K.\{TG.TKn'\} \_\_Kb) \land \text{secret}(Kn', k, \{A, B, T\}) \]

Figure 10. The Role T and A

5) The role I and its actions

The intruder has the ability of monitoring, analysis, falsify information in the channel.

intruder\_knowledge = \{a, b, s, t, ki\}

C. Write the program for the protocol

The TLA+ and HLPSL are the languages based on TLA. The two languages are suitable for the system in the concurrent environment. The HLPSL provides more functions for checking protocols and The TLA+ is more flexible. So we could use the two languages to write Kerberos protocol.

D. Checking the program safety properties

The main idea is using two dimensional methods of dynamic actions and static properties to check conformity, security and other property for a protocol. The protocol model, in fact the behavior rules of dynamic actions are described; with some logic representation of attribute in the system static properties are described. The two factors to check the dynamic behavior of the system could be satisfied the system static properties, thus found the system behavior and requirements inconsistency. So a specification of protocol should be including the security attributes under certain conditions, and then testing whether these security properties are satisfied.

E. Analyzing the Results of Checking

One of the checking results is as follow, analyzing all the results we could obtain the corresponding conclusion.

ATTACK TRACE

<table>
<thead>
<tr>
<th>Action</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i -&gt; (a,3)</td>
<td>: start</td>
<td></td>
</tr>
<tr>
<td>(a,3) -&gt; i</td>
<td>: a.{na(a,3)} _ka</td>
<td></td>
</tr>
<tr>
<td>i -&gt; (a,8)</td>
<td>: start</td>
<td></td>
</tr>
<tr>
<td>(a,8) -&gt; i</td>
<td>: a.{na(a,8)} _ka</td>
<td></td>
</tr>
<tr>
<td>i -&gt; (s,4)</td>
<td>: a.{na(a,3)} _ka</td>
<td></td>
</tr>
<tr>
<td>(s,4) -&gt; i</td>
<td>: a.{k0(s,4),na(a,3),ns(s,4)} _ka.{dummy_nonce.k0(s,4)}__dummy_sk</td>
<td></td>
</tr>
<tr>
<td>i -&gt; (s,9)</td>
<td>: a.{na(a,8)} _ka</td>
<td></td>
</tr>
<tr>
<td>(s,9) -&gt; i</td>
<td>: a.{k0(s,9),na(a,8),ns(s,9)} _ka.{dummy_nonce.k0(s,9)}__dummy_sk</td>
<td></td>
</tr>
<tr>
<td>i -&gt; (a,3)</td>
<td>: a.{k0(s,4),na(a,3),ns(s,4)} _ka.start</td>
<td></td>
</tr>
<tr>
<td>(a,3) -&gt; i</td>
<td>: a.b.k0(s,4).{na(a,3),ns(s,4)} _k0(s,4).start</td>
<td></td>
</tr>
</tbody>
</table>

V. CONCLUSION

Finding the loopholes of a protocol is hard work. We use a methods based on TLA to detect the security of Kerberos protocol, which creating the roles for it, specifying the actions of the roles, building the sessions among the roles, setting the environment parameters for it, writing the program for it, and repeatedly checking and modifying the program is very effective. Especially the program based on TLA which covers the model and the detection properties of the protocol is different from SPIN and SMV. The description based on TLA is natural and clear; provides a large number of functions for specifying protocol; and can put the model program and the properties of the protocol together in a description.

The Attack Trace before show the attack steps, in which ‘i’ is role intruder, ‘a’ is role A, the number is nonce, and so on. The last step shows that ‘a’ regards ‘i’ as the server and sends the key.

The results show the protocol has limitation, in certain conditions could be broken. Firstly its security level is not high, because users are often willing to set up a simpler password; Secondly in testing, the server using time value as a verification method, which requires clock synchronization in the both server, but in the larger network clock asynchronous is often occurs, so it also affects the correctness in checking the protocol; Thirdly Kerberos protocol use private key and prevent password guessing ability is very weak, the attacker can collect a large number of permits, through the calculation and analysis of key password guessing. When the user selects a password is not strong enough, it could not effectively prevent the password guessing attack.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their insightful comments that helped improve the presentation of this paper. The work of the paper was supported in part by the National Natural Science Foundation of China (61070192, 90108008, 61170240), Natural Science Foundation of Beijing (4122041), and National High-Tech Research Development Program of China (2007AA012Z414).

REFERENCES