Int J Theor Phys (2015) 54:3311–3330
DOI 10.1007/s10773-015-2571-9

Study on Cloud Security Based on Trust Spanning Tree Protocol

Yingxu Lai · Zenghui Liu · Qiuyue Pan · Jing Liu

Received: 16 October 2014 / Accepted: 9 February 2015 / Published online: 25 February 2015

Abstract Attacks executed on Spanning Tree Protocol (STP) expose the weakness of link layer protocols and put the higher layers in jeopardy. Although the problems have been studied for many years and various solutions have been proposed, many security issues remain. To enhance the security and credibility of layer-2 network, we propose a trust-based spanning tree protocol aiming at achieving a higher credibility of LAN switch with a simple and lightweight authentication mechanism. If correctly implemented in each trusted switch, the authentication of trust-based STP can guarantee the credibility of topology information that is announced to other switch in the LAN. To verify the enforcement of the trusted protocol, we present a new trust evaluation method of the STP using a specification-based state model. We implement a prototype of trust-based STP to investigate its practicality. Experiment shows that the trusted protocol can achieve security goals and effectively avoid STP attacks with a lower computation overhead and good convergence performance.

Keywords STP · Cloud security · Trust evaluation · Trusted network

1 Introduction

Cloud computing is one of the current hot topic in the field of information technology. The advantages of convenience, economy and high extensibility attract the attention of more and more enterprises. However, cloud computing is just like a double-edged sword, it brings us great convenience, at the same time, it also carries additional security vulnerabilities. With the popularity of cloud computing, the significance of security issues is gradually increasing and it has become an important factor to restrict cloud applications. Cloud security is complex, not only the traditional firewall, IPS and other security devises are needed to build a

Y. Lai (✉) · Q. Pan · J. Liu
College of Computer Science, Beijing University of Technology, Beijing 100124, China
e-mail: laiyingxu@bjut.edu.cn

Z. Liu
Automation Engineering Institute, Beijing Polytechnic, Beijing 100176, China

security defense system, also switch, as the core network equipment of Ethernet, should be able to prevent attacks or virus. Switch shares an important role with other security devices to build up a full mechanism for security managing.

Switches typically act as the 2nd layer forwarding devices, and their essential function is to optimize the network for packet forwarding. Obviously, all kinds of packets are easy in and out of these open, distributed systems, this increases the probability that the switches and switch protocols suffer from different kinds of security attacks, such as MAC flooding attacks, ARP attacks, and STP attacks.

Spanning Tree Protocol (STP) [1, 2] is one of the most important protocols running on switches. The purpose of STP is to address the problem brought by the redundancy in physical topology. STP dynamically builds a spanning tree when an active looped topology exists in a 2nd layer network, thus avoiding the broadcast storm caused by repeating frame forwarding and unstable MAC address table [3]. Switches collaboratively compute a spanning tree by exchange Bridge Protocol Data Units(BPDU) each other. Each switch is identified by a unique Bridge ID (BID), and the one with minimum BID will be selected as the root of the spanning tree. Due to this feature, an attacker can disguise itself as the root by sending BPDU with BID less than the current root's BID, and then take advantages of the root role for further attacks. This type of attack is called root take-over attack [4]. The spanning tree algorithm recalculates all the switches' metric as the topology is changed, which would be easy for an attacker to launch some other attacks to the network, such as ID changing attack [5], BPDU flooding attacks and silent attacks. Marro [6] showed the pitfalls of STP in three aspects: (1) lack of authentication in BPDU message; (2) slow convergence of STP; and (3) root role is not fully monitored. Thus, many attacks can be exploited to a network running STP due to these pitfalls, especially when an attacker can easily access the network.

Integrating the trusted computing technology into the cloud computing environment, to provide cloud services in a reliable way has become a hot spot in cloud security research field [7, 8]. In this paper, we propose a trust-based spanning tree protocol to address the pitfalls (1) and (3). First, we use Trusted Computing technology to guarantee the credibility of the switch. Then, we guarantee the credibility of the trust-based protocol by a trust-based evaluation model. As we know, virtualization technology is one of the core technologies of cloud computing, it brings great scalability and manageability features for the data center. So, finally, we implement a prototype of trust-based STP on a virtual switch to re-appear the protocol application in the cloud environment to the maximum extent and demonstrate its application.

The remainder of this paper is organized as follows. Section 2 introduces the existing solutions to prevent STP attacks. Section 3 shows the proposed trust-based STP in detail. Section 4 gives a general implementation of the trust-based STP prototype on a virtual switch. Section 5 presents performance analysis on trust-based STP. Finally, we conclude the paper in Section 6.

2 Related Work

Many solutions have been put forward for preventing STP attacks, including Cisco's BPDU Guard [9] and Root Guard [4, 10]. BPDU Guard prevented the port from accepting any BPDU message to avoid the STP related attacks, while Root Guard went against root take-over attack by turning the port that had received the superior BPDUs (conf_BPDU with BID less than the current root BID) into blocking. The former method did not allow new switches to connect the port that had BPDU guard enabled. The latter could not stop other network

infrastructure attacks, and it required manual configuration, which increased the burden of administrators. Both of them went against the STP design concept.

Applying authentication on BPDU messages is another type of studies. Fung [11] proposed a method to achieve simple BPDU authentication mechanism using a Bridge Address Permit List (BAPL). This solution was suitable for a small network. If there were too many bridges in the network, the maintenance of the permit list would become a problem. Whalen and Bishop [12] developed an authentication mechanism, which modified the BPDU by adding three extra fields, including a nonce, a key index, and a SHA-1 digest. The weakness of this proposal was that, the signature of each BPDU spent too much time, affected the normal operation and convergence time of STP.

Yeung et al. [13] proposed a partition-based switched network. It used special switches at the boundary of the STP domain. The domain was divided into a Network Infrastructure (NI) and a Non-Network Infrastructure (NNI). The NI used a normal STP, while the NNI used a modified STP. NNI was used to isolate the attacks launched. However, this proposal required specially designed switches running modified STP, hence it was not adaptable to most networks.

Jieke et al. [5] put forward a specification-based IDS intrusion detection system. The system was modeled as a state machine by the specification of STP, and it could detect the illegal behaviors of the neighbors by keeping tracks of its neighbor's specification. It was not efficient due to the high costs for maintaining state information of all neighbors, and it could not avoid root take-over attack. Therefore, this method was not applicable to a large switching network.

Rai et al. [14] suggested an improved method on the basis of Jieke's approach. It divided the network into different domains, with each domain running an IDS, instead of each switch running an individual IDS. Each IDS in a domain could monitor BPDUs generated from any switch in its local domain. If a BPDU was received from a switch that was not covered by any IDS, it was considered from an attacker. This approach could detect a new switch claimed the root role but could not identify the root take-over attacks launched from the inside network. Another drawback is that the IDS deployment also needed to be changed if the network topology was changed.

Ge et al. [15] designed an abnormality detection system. Each switch had a spanning tree parameters: bridge ID and a PPC (Port Path Cost), which was determined by the network server when the switch joined in the network. Each switch in the network stored a neighbor-tracking table. By comparing the receiving BPDU with the spanning tree parameters stored in the neighbor-tracking table, the system decided whether it was an abnormal BPDU or a normal topology change. This method could prevent the attacks launched by those non-authorized switches, but could not prevent the basic flooding attacks. The main drawback of the design was its overhead of storing the neighbor-tracking table.

When the existing mechanism can no longer meet the current need of information security, especially when the threats come from inside the network, we need new methods to address the security problems. In this paper, we implement a prototype of trust-based STP based on the work in [15]. First, for the credibility of the network device: we use Trusted Computing technology to guarantee credibility of the switch. A switch cannot join a trusted network without the TPM (Trusted Platform Moudle) chip, which can prevent the switch BIOS (Basic Input/Output System) from being tampered. Second, for the credibility of STP protocol: we propose a method to ensure the credibility of the protocol, which is trust-based STP. In this design, a switch with a BID less than the current root BID that participated in STP needs to be verified by using its certificate, which prevents the illegal switch participating in STP and launching root take-over attack. Third, for the trust evaluation:

we guarantee the credibility of the trust-based protocol by a trust-based evaluation model. Finally, we present trust-based evaluation of STP using a specification-based state model, which estimates whether the protocol is in accordance with the expected behavior or not.

3 Trust-Based STP

In this section, we describe the improved STP protocol on trusted switch. And we describe a trust evaluation model based on the expected behaviors of the trusted protocol for monitoring the unexpected behaviors.

3.1 Trusted Switch

In our tested network, each switch is equipped with a TPM chip. Each TPM has a unique EK (Endorsement Key), which can randomly generate an arbitrary number of AIK (Attestation Identity Key) key pairs for identity attestation [16]. When a switch initially joins in the Internet, in order to obtain permission to announce topology information, it is required to obtain its own certificate. For this purpose, its platform needs to be attested by the authority. First, the TPM on the switch signs the platform information to the authority to request platform attestation. Upon successful verification, the authority sends a platform identity confirm message to the switch. Then the switch binds its user identity information and platform identity information together for a second round of attestation. After all the information has been confirmed, the authority issues a certificate with both platform and user identity to the switch, and then sends a Certificate Revocation List (CRL) to each switch in the network. This guarantees that a certificate is only released to a normal switch. The process is shown in Fig. 1.

The key information in the certificate to be attested is {user ID, Hash(Bridge ID), PCR, SML}. The user ID contains user information of the switch. We assume that the behaviors of a legitimate user are trusted, including changing priority of the switch. The Bridge ID consists of a unique MAC address and a bridge priority. This part ensures the switch participating in STP is legitimate and prevents changing ID attacks. The PCR (Platform Configuration Register) value and the SML (Stored Measurement Log) ensure that the BIOS system is not tampered during booting up and also guarantee that the trust-based STP and trust evaluation model are not tampered, i.e., program code is not modified or skipped.

A switch or the authority can also revoke a certificate through a similar process. The switch just needs to send a message and its signature to request revocation. Then, the

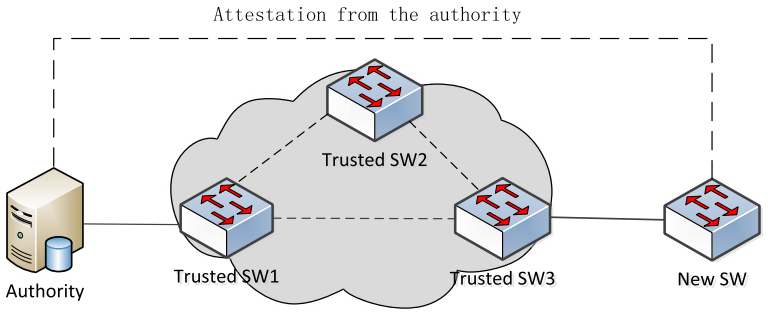

Fig. 1 Attestation from the authority

authority verifies the signature, updates the CRLs and releases a new one to each switch in the network. The certificate is used each time in STP convergence, as the convergence is not frequently happened, so we use CRLs instead of OCSP (Online Certificate Status Protocol) to reduce the overhead.

3.2 Trust-Based STP

We attach a local authentication to the STP that can protect the switch from outside attacks, especially the root take-over attack. Considering the network topology, we deploy the improvement of the trust-based STP in the following situation, and in order to give a better understanding of trusted STP, we take an example to explain the execution of the protocol.

In Fig. 2, switch R is the root bridge. Switch A is a new one, which claims to be a root bridge. It connects directly to switch B, and passes through several switches connected to the root bridge R. Claim process based on trust STP is divided into the following steps.

Step 1: The new switch exchanges BPDU with a non-root switch. Switch A sends a superior BPDU to claim to be a root bridge. When switch B receives the superior BPDU from a target root(switch A) directly connected to it, switch B will send a request BPDU to switch A to ask the certificate. On receiving the requested message, switch A should send a response BPDU back with its own unique certificate, which was acquired as switch A access to the network at the first time. Then switch B checks the certificate: first, it checks the signature and analyzes whether the certificate has been tampered; second, it checks the CRLs and see whether the certificate is revoked. If the authentication succeeds, which means the switch is trusted, the announcement is accepted. Consequently, switch B updates its information of design root and go on to send the superior BPDU with "rootID=A" to its neighbors(e.g., switch C). Otherwise, switch B gives an alarm to prevent switch A from participating in STP.

Step 2: Intermediate switches exchange BPDU. As shown in Fig. 3, switch C receives the superior BPDU from switch B which design root is switch A. Switch C will send a request BPDU to the sender (switch B) directly connected to it, instead of to switch A. When switch C receives the superior BPDU from switch B, it means switch A has passed the authentication by switch B, thus switch C just need to verify switch B. That is because switch B certainly first receives the superior BPDU from switch A, and it does the process mentioned in step 1. Then switch

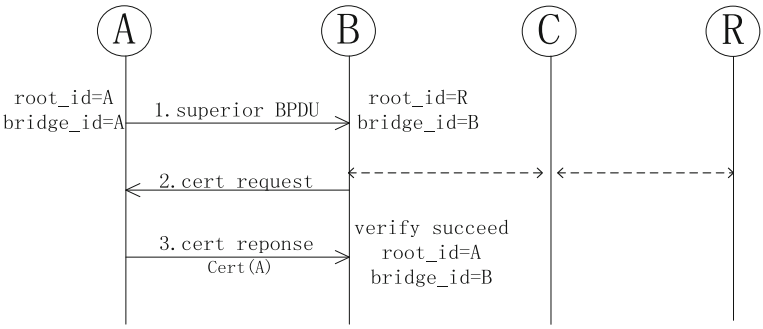

Fig. 2 Step 1: The new switch connects to a non-root switch

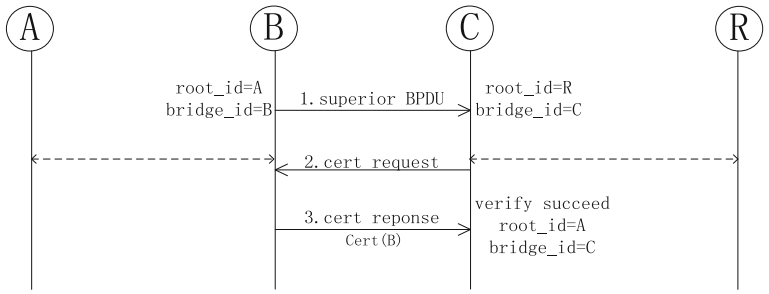

Fig. 3 Step 2: Intermediate switches exchange BPDU

B sends its own certificate back to switch C. If switch B is legitimate, switch C considers the behavior of switch B is trusted too, and thus, the root announcement it sent is trusted. Then switch C updates its information of the design root and takes further steps.

Step 3: the root switch receives the superior BPDU. As shown in Fig. 4, switch R receives the superior BPDU with "rootID=A" from switch C. Where switch R is the root bridge, not only the above step should be executed, but also an additional step should be executed. If the switch has passed the authentication procedure, the root (switch R) will take a self-assessment to decide whether the target root (switch A) should be the root or not. The self-assessment information includes hello time, forward delay, the max age, and the number of alarms aroused by the trust evaluation model. If no more than two items exceed the threshold values, the self-assessment succeeds, and switch R does not update the root information. On the contrary, switch R will give a root take-over attack warning. Otherwise, switch R updates its CRL and accepts switch A as a new root.

The procedure is shown as follows.

BPDU: the normal format of BPDU in STP
RID: Root ID
BID: Bridge ID
X: the switch which has a smaller RID
$C \rightarrow R$: $BPDU\{..., Type(0x00), RID(X), BID(C)\}$
$R \rightarrow C$: $BPDU\{BPDU\{..., Type(0x02)\}\}$
$C \rightarrow R$: $Response\{BPDU\{..., Type(0x04), RID(X), BID(C))\}, ...\}, Cert(C)\}$

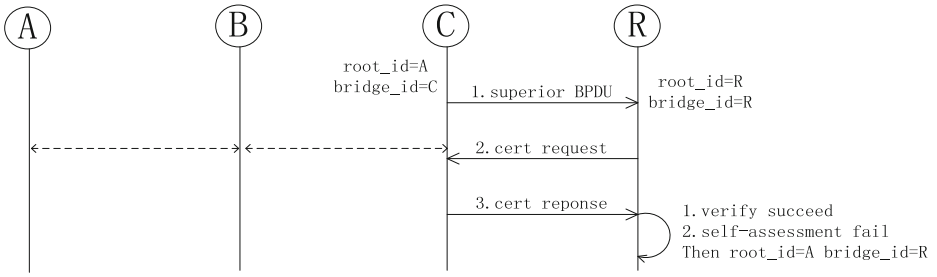

Fig. 4 Step 3: the root switch receives the superior BPDU

3.3 Trust Evaluation

Trusted Computing Group (TCG) uses expected behavior of entities to define credibility: an entity can be trusted if it always acts as expected, towards the desired objectives [17]. The trust of a STP depends on the expected behavior of each switch during their announcing topology information stage. In order to ensure that the trust-based STP is not misconfigured or maliciously modified, we describe a trust evaluation model based on the expected behaviors of the trusted protocol. We can also monitor the unexpected behaviors triggered by the switch itself, in case the switch is taken over by hackers.

The trust evaluation model estimates the trust-based STP using specification-based state model [5], which specifies normal behaviors and estimates any other behaviors as abnormal behaviors. The model divides the protocol into different states according to the type of BPDU that the switch is waiting for. Then it creates a state machine according to the STP operating principle. There are six states in the state machine, and each represents a state of the STP protocol running on a switch. Each arrow represents a transition between two states and the state transfers from one to another by triggering conditions above the line and generating the corresponding event under the line, as shown in Fig. 5. Each switch only maintains a state machine and records its own states, without knowing about its neighbors.

We denote the six states as S1-S6. Where, S1 is Initial state; S2 is Wait for CONF_BPDU state, at this state, switch is waiting for a configuration BPDU; S3 is Wait for TCN_BPDU

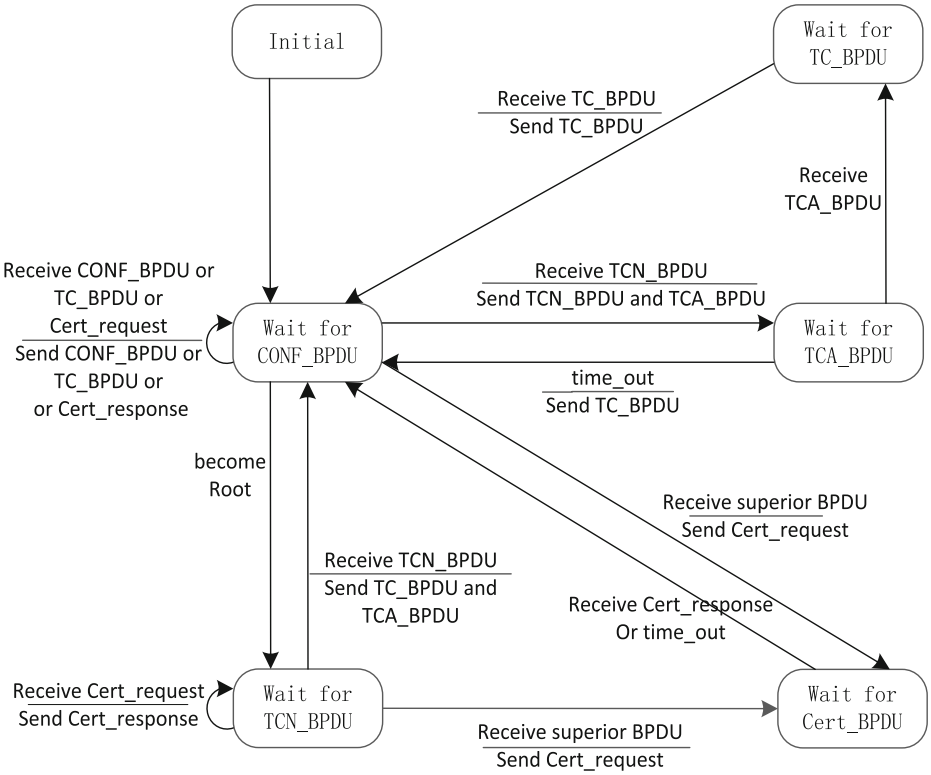

Fig. 5 Trust evaluation model

state, switch is waiting for a topology change notify BPDU at this state; S4 is Wait for TC_BPDU state, switch is waiting for a topology change BPDU at this state; S5 is Wait for TCA_BPDU state, switch is waiting for topology change acknowledgement BPDU at this state; S6 is Wait for Cert_BPDU state, switch is waiting for certification BPDU at this state. We simplify the Figs. 5 to 6.

Then, we extract 11 rules to define the trust evaluation model based on Fig. 6. These rules represent the key process of the trust-based STP, where each transition (each rule) stands for the expected behaviors of STP. We monitor the switch is trusted or not by these rules.

Rule 1: S2 ∩ cert_request received => send cert_response ∩ (S2→S2)

If the state is *Wait for CONF_BPDU*, switch will receive CONF_BPDU every 2 seconds from its neighbors. The switch update its information and forward it to other neighbors. When the topology changes, the switch will receive the TC_BPDU from one neighbor, and then forward the BPDU to other neighbors. At that time, the switch will also receive a certification request message, the switch will send a response BPDU back with its own unique certificate.

Rule 2: S2 ∩ is_root => S2 → S3

When the switch is *Wait for CONF_BPDU* state, if the switch is root bridge, the switch state will be changed to state S3 (*Wait for TCN_BPDU*).

Rule 3: S3 ∩ cert_request received => send cert_response ∩ (S3→S3)

The root bridge receives a certification request message, the switch will send a response BPDU back with its own unique certificate.

Rule 4: S3 ∩ TCN_BPDU received => send TC_BPDU ∩ TCA_BPDU ∩ (S3→S2)

The root bridge receives a TCN_BPDU message, the root bridge will send a confirmation BPDU back. And then it sends a TC_BPDU to other neighbors to announce the topology change.

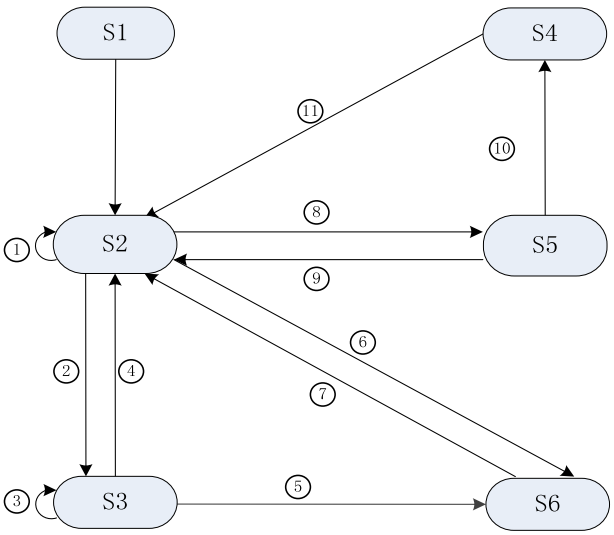

Fig. 6 Simplified trust evaluation model

Rule 5: S3 ∩ superior BPDU received => send cert_request ∩ (S3→S6)
If the state is *Wait for TCN_BPDU*, the root bridge receives a superior BPDU with BID less than the current root's BID, it will send a certification request BPDU, and then the state changes from *Wait for TCN_BPDU* to *Wait for CERT_BPDU*.

Rule 6: S2 ∩ superior BPDU received => send cert_request ∩ (S2→S6)
If the state is*Wait for CONF_BPDU*, the root bridge receives a superior BPDU with BID less than the current root's BID, it will send a certification request BPDU, and then the state changes from *Wait for TCN_BPDU* to *Wait for CERT_BPDU*.

Rule 7: S6 ∩ (cert_response received ∪ time_out) => S6→S2
If the state is *Wait for CERT_BPDU*, the bridge receives target switch certification response or time out, the state will be changed to *Wait for CONF_BPDU*.

Rule 8: S2 ∩ TCN_BPDU received => send TCN_BPDU ∩ TCA_BPDU ∩ (S2→S5)
If the state is *Wait for CONF_BPDU,* the switch does not receive any message until time out, it will send TCN_BPDU message to root bridge and be changed to *Wait for TCA_BPDU*. If the switch received the TCN_BPDU message, it will send TCA_BPDU back and send TCN_BPDU from root port. And then the state of the switch will be changed to *Wait for TCA_BPDU*.

Rule 9: S5 ∩ time_out => send TC_BPDU ∩ (S5→S2)
If the state is *Wait for CERT_BPDU*, the bridge does not receive any message until time out, it will send TC setting BPDU to notify root bridge to recalculate spanning tree, and the state will be changed to *Wait for CONF_BPDU*.

Rule 10: S5 ∩ TCA_BPDU received => S5→S4
If the state is *Wait for TCA_BPDU*, the switch receive TCA_BPDU response message. The state will be changed to *Wait for TC_BPDU*.

Rule 11: S4 ∩ TC_BPDU received => send TC_BPDU ∩ (S4→S2)
If the state is *Wait for TCA_BPDU*, the switch receive TC_BPDU response message, The state will be changed to *Wait for CONF_BPDU*.

Case Study: Execution Analysis Figure 7 shows a simple network running trust-based STP. SW1 is the root. The port of SW4 that connects to SW3 is blocked. When STP is running, the trust evaluation model is also beginning to work at the same time. Now a new switch

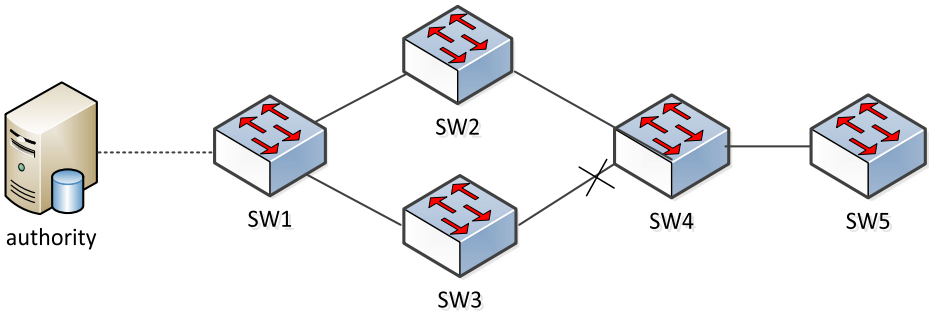

Fig. 7 Test network topology

Table 1 Fields of cert_request_bpdu

Byte	Field
2	Protocol ID
1	Version
1	BPDU type (0x02)

SW5 is connected to the network and claims to be a root. SW4 first receives the superior BPDU, and it sends a probe to SW5. The state of SW4 changes from *Wait for CONF_BPDU* to *Wait for CERT_BPDU*. If SW5 is compromised by an illegal user, or a tampered platform, or SW4 doesn't receive a certification response for a certain amount of time, it will come back to *Wait for CONF_BPDU* state and if this lasts for 3 times, SW5 will be detected as malicious. Otherwise, SW5 will pass the authentication and SW4 modifies the BPDU and sends it to SW2. For SW2, it receives the superior BPDU from SW4 but the sender is SW5. Thus, SW2 sends a probe to SW4 and goes on further steps. Finally, SW1 receives the superior BPDU from SW2 and SW2 passes the verification, SW1 begins self-assessment. If SW1 modifies the hello time, forward delay, or max age frequently, or it triggers alarm many times, the self-assessment will not succeed, because the root SW1 has shown a poor credit. Then SW5 is selected as the root by STP protocol. Otherwise, SW5 will be deemed as launching a root take-over attack.

4 Prototype Implementation

We implement a prototype of trust-based STP on a virtual switch called Open vSwitch [18]. In this section, we present three important components used in trust-based STP: 1) basic data structure; 2) verifying STP announcements; 3) trust evaluation model. We give the

Table 2 Fields of cert_response_bpdu

Byte	Field
2	Protocol ID
1	Version
1	BPDU type
1	Flag
8	Root Identifier
4	Root Path Cost
8	Bridge Identifier
2	Port Identifier
2	Message Age
2	Max Age
2	Hello Time
2	Forward Delay
1028	X509_CERT

key structure of the BPDU messages and describe the trust-based spanning tree protocol in detail. Then show the rules of the trust evaluation model.

4.1 Basic Data Structure

We add two new BPDU type in trusted STP: *cert_request_bpdu* and *cert_response_bpdu*. The format of the messages is shown as follows (Tables 1 and 2).

4.2 Verifying STP Announcements

We simulate our improved protocol on a virtual switch called Open vSwitch. We add two new BPDU types on the basis of the original message: *cert_request_bpdu* and *cert_response_bpdu*.

We simulate the authority as a CA server, and x509 certificate stands for the user certificate released by the authority. Each switch has a CA certificate and a user certification of its own when it accesses to the network. When a new BPDU arrives, *stp_receive_bpdu* function is triggered. The switch will verify that if the certificate is issued by the CA server, that is: 1) if the signature of the certificate is valid or not; 2) if the certificate is within the validity period or not; 3) if the certificate is revoked or not. We use the OpenSSL library function *X509_verify_cert* to achieve this goal. The detail is shown as follows.

```
Input: bpdu packet *bpdu
Result: Notifies the STP entity that bridge protocol
data unit 'bpdu'
1   if bpdu.type=conf_bpdu then
2       if receive superior_bpdu  then
3           call transmit_cert_request_bpdu();
4       else call received_conf_bpdu();
5   else if  bpdu.type=tcn_bpdu  then
6           call stp_received_tcn_bpdu();
7   else if  bpdu.type=cert_request_bpdu then
8           read certificate file;
9          call transmit_cert_response_bpdu();
10  else if bpdu.type=cert_response_bpdu:
11          read CA certificate from local storage;
12          verify the user certificate in the bpdu;
13      Else
14         error←error+1;
15  End
```

4.3 Trust Evaluation Model

The trust evaluation is used to provide the mechanism to verify if the authentication rules of trust-based STP are enforced by the trusted switch correctly. State machine is key

component of the trust evaluation model. We specify normal behaviors of a certain switch running STP on a certain state.

We take the state *Wait for CERT_BPDU* as an example and show detail as follows.

Rule 7: S6 $\bigcap$ (cert_response received $\bigcup$ time_out) => S6 $\rightarrow$ S2

```
Input: unsigned char state, const char *function
Output: return 1 if succeed, otherwise return 0.
1    If state=WAIT_FOR_CERT_BPDU then
2        if receive cert_response_bpdu    then
3            state ←WAIT_FOR_CONF_BPDU;
4            return 1;
5        else if stp_timer expired then
6            state ← WAIT_FOR_CONF_BPDU;
7            return 1;
8        else
9            return 0;
10       End
```

5 Performance Evaluation

5.1 Test Topology

We use experiments to evaluate the performance of trust-based STP. In our experiments, we deploy our trust-based STP prototype in three Linux 3.0.0 PCs with Intel Core Duo 2.5 GHz CPU and 2 GB memory. The formed topology is shown in Fig. 8.

We propose a trust-based spanning tree protocol to protect the network from different STP attacks. We use a STP attacking tool named "yersinia" [19, 20], which is designed to explore certain weakness of different network protocols including STP. In this section, we use attack experiments to demonstrate the main function of the trust-based STP, and present performance analysis.

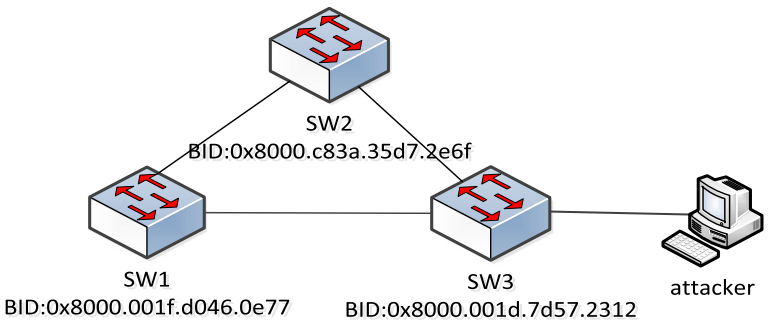

Fig. 8 Test network

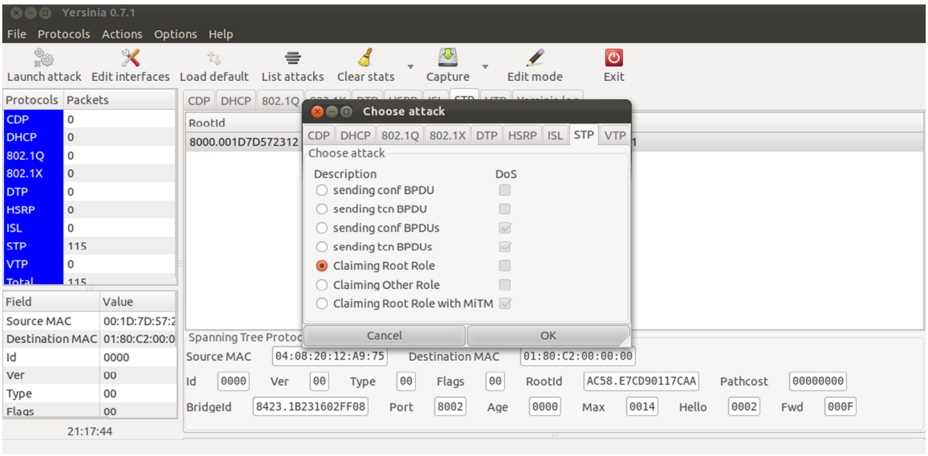

Fig. 9 Launch a root take-over attack

5.2 Experiment Analysis

Experiment A. We first launch a root take-over attack, as shown in Fig. 9, to the original STP protocol on SW3 (here SW3 is a TPM-free switch), which is the root in the network. We use the topology in Fig. 8, and the result is discussed in following figures.

In Fig. 10, we can see that a new root 001d.7d56.2312 is announced, which is smaller than the root (SW3:001d.7d57.2312). We use the command *ovs-vsctl list Bridge br0* to show the STP state information in Fig. 11. We find that the root bridge has been changed.

Experiment B. In this experiment, we launch the same attack on SW3(here, SW3 embedded a TPM chip) to test the trust-based STP, and we use the same topology in Fig. 8.

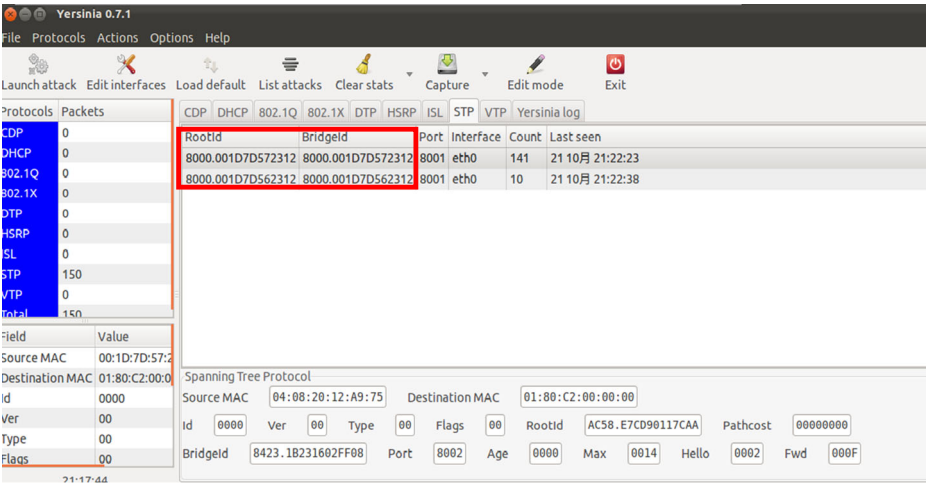

Fig. 10 Yersinia result

```
root@administrator-ubuntu:/home/administrator/openvswitch-1.7.1.bak/openvswitch-1.7.1# ovs-vsctl list Bridge br0
_uuid               : d9ec7115-ace3-4bc7-a257-870a1bc584eb
controller          : []
datapath_id         : "0000001d7d572312"
datapath_type       : ""
external_ids        : {}
fail_mode           : []
flood_vlans         : []
flow_tables         : {}
mirrors             : []
name                : "br0"
netflow             : []
other_config        : {stp-past-cost="10", stp-priority="0x8000"}
ports               : [030f0a76-f372-4c66-a49c-6b1b12f70cb2, c2b7e74e-9c8a-4a27-8ac1-bf13b88f480c]
sflow               : []
status              : {stp_bridge_id="8000.001d7d572312", stp_designated_root="8000.001d7d562312", stp_root_path_cost="19"}
stp_enable          : true
root@administrator-ubuntu:/home/administrator/openvswitch-1.7.1.bak/openvswitch-1.7.1#
```

Fig. 11 The information of STP on SW3 being attacked

We can see that the yersinia also sends a message that claims to be a root (001d.7d56.2312) in Fig. 12. However, the new protocol forces SW3 to send a request message, which is shown as 0000.0000000000 in Fig. 12 and whose type is 0x02, to the attacker. The attacker will fail to the authentication. SW3 triggers the alarms and the system log is shown in Fig. 13. The information of STP on SW3 being attacked is shown in Fig. 14 and we can find that the root does not changed.

Figure 15 shows the STP content when it is attacked. When SW3(001d.7d57.2312) received a superior BPDU, it sends a cert_request message. The length of packet is 7 bytes (3 bytes logic-link control and 4 bytes cert request), the content of cert request is 0x00000002: 2 bytes protocol ID, 1 byte version and 1 byte BPDU type. The attacker has not the correct certification, so it fails to send a cert_response BPDU.

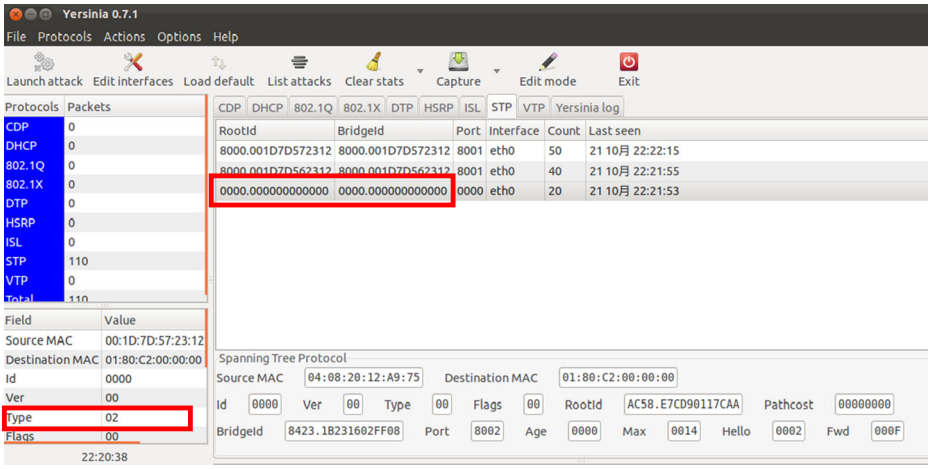

Fig. 12 Yersinia result

```
1071 Oct 21 22:10:57 ubuntu ovs-vswitchd: 00014|stp|INFO|state:#2 WAIT_FOR_CONF_BPDU to WAIT_FOR_TCN_BPDU
1072 Oct 21 22:10:57 ubuntu ovs-vswitchd: 00015|stp|INFO|state:#5 WAIT_FOR_TCN_BPDU to WAIT_FOR_CERT_BPDU
1073 Oct 21 22:10:58 ubuntu ovs-vswitchd: 00016|stp|WARN|alarm=2,state=#005,stp_received_superior_bpdu
1074 Oct 21 22:11:01 ubuntu ovs-vswitchd: 00017|stp|WARN|alarm=3,state=#005,stp_received_superior_bpdu
1075 Oct 21 22:11:04 ubuntu ovs-vswitchd: 00018|stp|WARN|alarm=4,state=#005,stp_received_superior_bpdu
1076 Oct 21 22:11:07 ubuntu ovs-vswitchd: 00019|stp|WARN|alarm=5,state=#005,stp_received_superior_bpdu
1077 Oct 21 22:11:10 ubuntu ovs-vswitchd: 00020|stp|WARN|alarm=6,state=#005,stp_received_superior_bpdu
1078 Oct 21 22:11:13 ubuntu ovs-vswitchd: 00021|stp|WARN|alarm=7,state=#005,stp_received_superior_bpdu
1079 Oct 21 22:11:16 ubuntu ovs-vswitchd: 00022|stp|WARN|alarm=8,state=#005,stp_received_superior_bpdu
```

Fig. 13 system log

```
root@administrator-ubuntu:/home/administrator/openvswitch-1.7.1# ovs-vsctl lis
t Bridge br0
_uuid               : d9ec7115-ace3-4bc7-a257-870a1bc584eb
controller          : []
datapath_id         : "0000001d7d572312"
datapath_type       : ""
external_ids        : {}
fail_mode           : []
flood_vlans         : []
flow_tables         : {}
mirrors             : []
name                : "br0"
netflow             : []
other_config        : {stp-past-cost="10", stp-priority="0x8000"}
ports               : [030f0a76-f372-4c66-a49c-6b1b12f70cb2, c2b7e74e-9c8a-4a2
7-8ac1-bf13b88f480c]
sflow               : []
status              : {stp_bridge_id="8000.001d7d572312", stp_designated_root=
"8000.001d7d572312", stp_root_path_cost="0"}
stp_enable          : true
root@administrator-ubuntu:/home/administrator/openvswitch-1.7.1#
```

Fig. 14 The information of STP on SW3 after being attacked

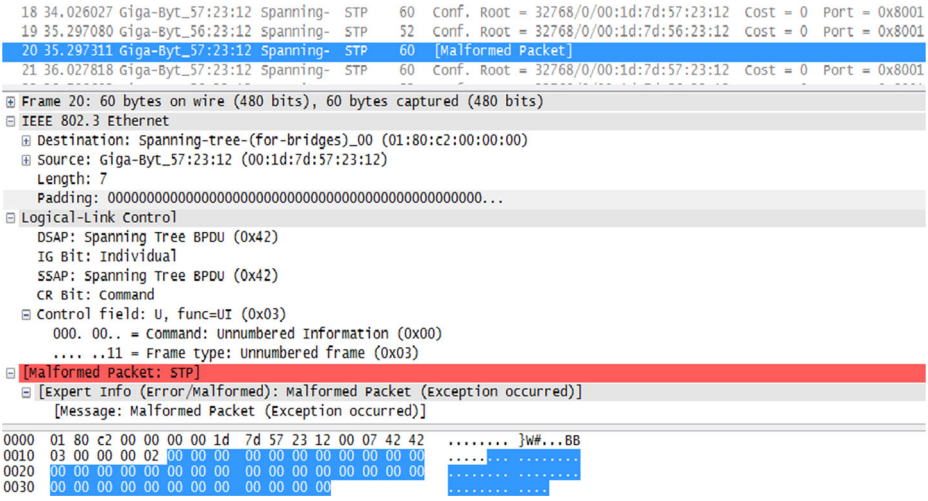

Fig. 15 Content of STP: send request

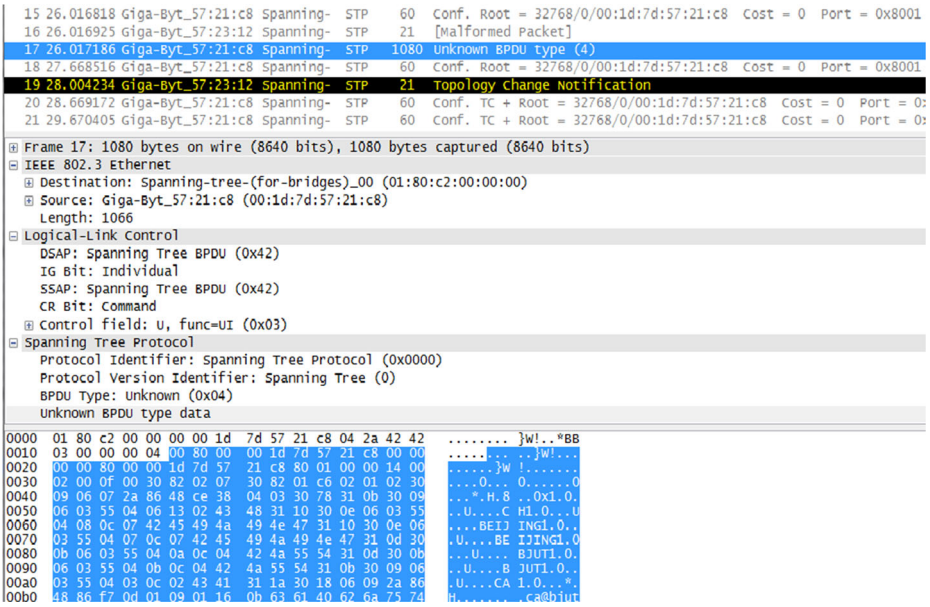

Fig. 16 Content of STP: send response

Under normal circumstances, when receiving the superior cert_request, root bridge will send back its certificate as response, which is shown in Fig. 16. We can see that the type of cert_response is 0x04, length is 1066 bytes (3 bytes logic-link control and 1063 bytes cert_response).

Then we demonstrate several experiments and show contrasting result in Table 3.

- **Root take-over attack.** Repeat the experiment B, attacker without certificate or with forged certificate can be detected easily. The state of trust evaluation model in SW3 stays at *Wait for Cert_BPDU* when the superior BPDU is received. It remains at this state for 2 seconds until Cert_BPDU arrived and other new superior BPDUs will be discarded during this period, thus **superior BPDU flooding** is avoided. Malicious switch with correct certification, however, who wants to replace the root bridge frequently, can be detected by the root self-assessment mechanism also. The mechanism can effectively prevent the root from take-over attack without impeding the root bridge election under normal circumstances.

Table 3 The Contrast Result

Attacks	STP	Root guard	BPDU guard	Trust-based STP
Root take-over attack	×	√	√	√
Superior BPDU flooding	×	×	√	√
Conf BPDU flooding	×	×	√	√
tcn BPDU flooding	×	×	√	√

note: √ avoid × not avoid

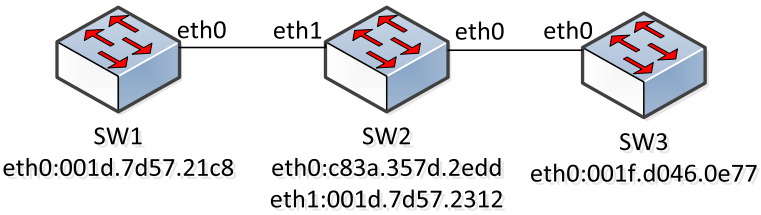

Fig. 17 Test network for performance analysis

- **Conf/tcn BPDU flooding attack.** These attacks are detected by trust evaluation model when sending or receiving BPDUs more than the expected times in a unit time, and other malicious behaviors will be detected also, such as sending BPDUs without receiving corresponding BPDUs on certain states.

Other attacks:

- **Silent attack.** A switch sends configuration BPDUs continuously to indicate it is alive after the spanning tree is built. When a switch stays silent on purpose (receive CONF_BPDU without sending it in state *Wait for CON_BPDU*), the trust evaluation model will alert it as a silent attack. Also other non-logic actions will be detected. Furthermore, silent attack launched from outside also can be detected.
- **Faked failure attack.** These attacks aim at network reconfigurations by sending fake TCN_BPDU frequently. They are detected if it sends TCN_BPDU without trigger event in any other state but *Wait for TCA_BPDU*, or sends normal BPDU more than the expected times in a unit time in the *Wait for TCA_BPDU* state .

5.3 Performance Analysis

Time delay is one of the most important factors that affect the performance of STP. In order to test the time delay of the authentication, we deploy the network structure as shown in Fig. 17. SW2 is connected to SW1 through interface eth1 and connected SW3 through interface eth0. The reason why we connect the switches in a line instead of a cycle is that the topology can avoid the influence which SW1 brings to SW3, so that we can get an exact data of the authentication time. Table 4 recorded the behavior of SW2 at each step of authenti-

Table 4 The Record of SW2

time (s)	sendorreceive	BPDU type and content	port
26.016594	Send	rootID=001d.7d57.2312	eth0,eth1
26.016818	Receive	Superior BPDU with rootID=001d.7d57.21c8	eth1
26.016925	Send	Cert_request	eth1
26.017186	Receive	Cert_response	eth1
26.018905	Send	New BPDU with rootID=001d.7d57.21c8	eth0

Table 5 Experiment result of verification time

No.	1	2	3	4	5	6	7	8	9	10
time(ms)	1.626	1.638	1.629	1.700	1.496	1.641	1.655	1.662	1.656	1.672

cation. In Fig. 17, SW2 first starts up, it sends the message with rootID=001d.7d57.2312. Then we start up SW1 and SW3.

Table 4 shows that the delay is 1.98ms (26.018905-26.016925) from receiving the superior BPDU to the end of authentication. The average is 1.719ms. At the same time, we record the verification time cost of X509 and the result is list in Table 5. The average is 1.638ms.

The result 1.719 ms is only one time authentication overhead but the authentication is much more then one. The switch asks for authentication only when it receives the superior BPDU (the bridge ID is less then the current root ID) without knowing other fields in the BPDU. Thus, we consider two limiting cases. First, in the best case, we suppose that each switch receives the BPDU with minimum BIDat the first time. So the authentication times is $n-1$ (n is the switch times in the network, each switch has to authenticate for one time except the root bridge). For example, in Fig. 18, the ascending order of BID is A,B,C,D ..., switch A is the minimum BID. In the best case, each switch receive the BPDU with "rootID= A" from its neighbors, so the switch just needs to authenticate once to the first BPDU's sender. The example shows that the authentication time has nothing to do with the topology structure.

Then in the worst case, we suppose that each switch receives the superior BPDU in decending order of BID. Every time the switch receives the superior BPDU, it will ask for authentication once. In other words, each switch before receiving the minimum BID, it will authenticate for all the switch in the network which BID is smaller than its. So if the number of switch in the nerwork is n, the switch with maximum BID will authenticate for $n-1$ times, the second biggest is $n-2$, and so on, then the total is $(n-1)+(n-2)+\ldots+2+1$, that is $n\times(n-1)/2$. For example, in Fig. 19 (3), switch E first starts up, then switch D. When switch E receives the superior BPDU from switch D, it authenticate for once. Later,

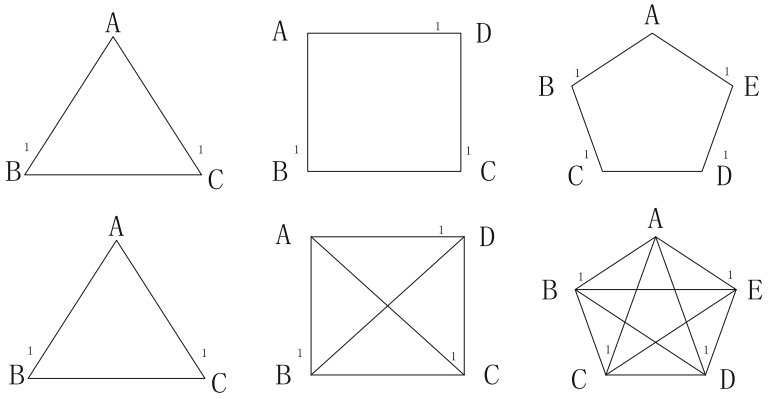

Fig. 18 Authentication times under the best situation

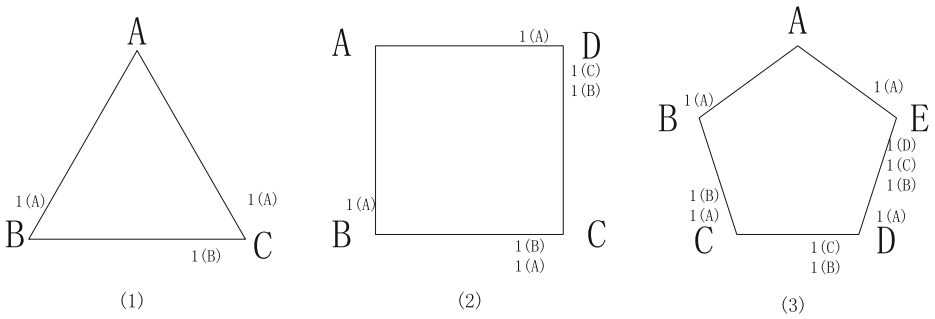

Fig. 19 Authentication times under the worst case

switch C starts up. Switch D asks switch C for authentication once and switch E asks switch D for another time (at this time, switch D sends the superior BPDU with "BID= C" to switch E). Then switch B starts up. And in this way, finally switch E respectively asks for authentication for 4 times aimming at rootID equal to D, C, B and A. Switch D asks for authentication for 3 times aimming at rootID equal to C, B and A and so on. The total times is 10 $(4 + 3 + 2 + 1 = 10)$.

In conclusion, a network with n switches, the authentication times is:

$$Sn = \begin{cases} n - 1, (n > 2, \text{ in the best case}) \\ \frac{n(n-1)}{2}, (n > 2, \text{ in the worst case}) \end{cases} \quad (1)$$

Contrast with traditional STP, the convergency time of trusted STP is $[1.719\times (n-1), 1.719\times n \times (n-1)/2]$ms. When the number of switch is increased, the time cost increase too, the corresponding time curves are shown in Fig. 20. y1 and y2 are corresponding to the authentication time cost in the worst case and in the best case respectively.

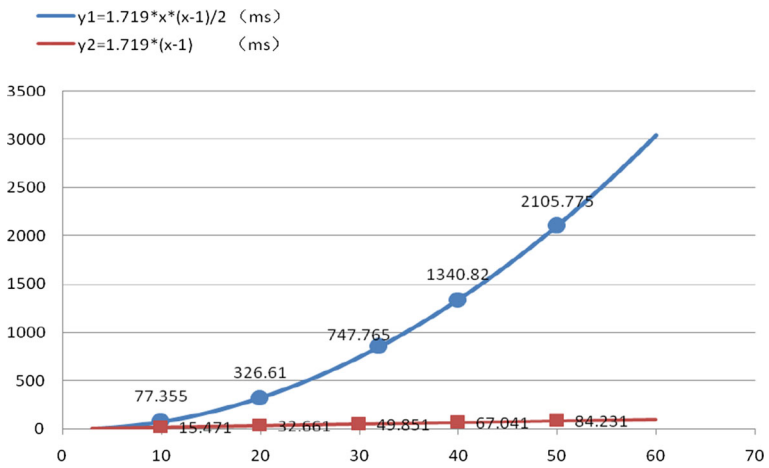

Fig. 20 Trend of authentication time

6 Conclusion

This paper developed a comprehensive security solution for low level network protocols based on trusted network. The trust-based security for STP was applied to enhance the security of spanning tree protocol, which in turn enhanced the network security in layer 2. Combined with the trust evaluation model, the protocol gave comprehensive protection to the threats from both outside and inside the switches. Experiments showed that our improved protocol could effectively avoid the root take-over attack, flooding attacks and other attacks.

Acknowledgments This research was supported by Funding Project for Academic Human Resources Development in Institutions of Higher Learning under the Jurisdiction of Beijing Municipality (PHR201108016).

References

1. ANSI/IEEE Std IEEE 802.1D ed. IEEE Standard for Media Access Control (MAC) Bridges[S], LAN/MAN Standards Committee of the IEEE Computer Society (1998)
2. ANSI/IEEE Std IEEE802.1D ed. IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges[S], LAN/MAN Standards Committee of the IEEE Computer Society (2004)
3. Lewis, W.: Networking Academy Program, CCNA 3:Switching Basics and Intermediate Routing [M],Beijing: People's Posts and Telecommunications Press (2008)
4. Cisco Systems, Spanning Tree Protocol Root Guard Enhancement [OL], 2005, document ID:10588
5. Jieke, P., Redol, J., Correia, M.: Specification-based intrusion detection system for carrier ethernet [C]. In: Proceedings of the International Conference on Web Information Systems and Technologies (WEBIST 2007) (2007)
6. Marro, G.M.: Attacks at the Data Link Layer, MSC thesis[D]. University of California at Davis (2003)
7. Santos, N., Gummadi, K.P., Rodrigues, R.: Towards trusted cloud computing[OL]. In: Sahu S, ed, USENIX Association Proc. of the Workshop on Hot Topics in Cloud Computing 2009. San Diego. (2009). http://www.usenix.org/events/hotcloud09/tech/full_papers/santos.pdf
8. Sadeghi, A.R., Schneider, T., Winandy, M.: Token-Based cloud computing: Secure outsourcing of data and arbitrary computations with lower latency[C]. In: Proceedings of the 3rd Int'l Conference on Trust and Trustworthy Computing, pp. 417–429. Springer-Verlag, Berlin (2010)
9. Cisco Systems: Spanning Tree Port Fast BPDU Guard Enhancement [OL], 2005, document ID:10586
10. Lewis, W.: Cisco Networking Academy Program, CCNP3: Multilayer Switching Lab Companion[M]. People's Posts and Telecommunications Press, Beijing (2006)
11. Fung, H.: Bridge Protocol Data Unit(BPDU) authentication mechanism using Bridge Address Permit List(BAPL), US 2005/0071672 A1[P] (2005)
12. Whalen, S., Bishop, M.: Layer 2 Authentication[D], University of California, Davis (USA), Technical Report XP002387477 (2009)
13. Yeung, K.H., Yan, F., Leung, C.: Improving Network Infrastructure Security by Partitioning Networks Running Spanning Tree Protocol[C], International Conference on Internet Surveillance and Protection (2006)
14. Rai, A., Barbhuiya, F.A., Sur, A., et al.: Exploit Detection Techniques for STP using Distributed IDS[C], Information and Communication Technologies (WICT). World Congress on. (2011)
15. Ge, A., Chiruvolu, G., Ali, M.: Bridge Network Spanning Tree Abnormality Detection, US 7471647B2[P]. (2008)
16. Chen, Y., Chen, Y., Tsai, W.T. Service-Oriented Computing and Web Software Integration, 3rd edn. Kendall Hunt Publishing (2011)
17. Trusted Computing Group (TCG).TCPA Main Specification[OL], Version 1.1b (2002)
18. Production Quality: Multilayer Open Virtual Switch[OL]. http://openvswitch.org/
19. Yersinia Home page[OL]: http://www.yersinia.net/
20. Barroso, D., Andres, A.: Yersinia presentation[OL], Blackhat Europe (2005)