2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems

Study on authentication protocol of SDN trusted domain

Ruikang Zhou
College of Computer Science
Beijing University of Technology
Beijing, China
jasonkidd5586@126.com

Yingxu Lai
College of Computer Science
Beijing University of Technology
Beijing, China
laiyingxu@bjut.edu.cn

Zenghui Liu
Automation Engineering Institute
Beijing Polytechnic
Beijing, China
zenghuiliu@yeah.net

Jing Liu
College of Computer Science
Beijing University of Technology
Beijing, China
kitty_helios@bjut.edu.cn

***Abstract*—Currently Software Define Network (SDN) architecture has become a hot topic. Aiming at the authentication security issues of SDN network architecture, we introduce an authentication protocol based on SDN network architecture without any trusted third party between trusted domains. By applying AVISPA security analysis system of network interaction protocol, we can guarantee protocol security and provide complete safety tests. Our work fill the gap of mutual trust between different trusted domains and provide security foundation for interaction between different trusted domains.**

Keywords—SDN ; trusted technology ; AVISPA security analysis system

I. INTRODUCTION

In the next generation network architecture design, many experts and scholars have completed a series of remarkable research work. They proposed some next generation network architectures, for example, 4D, GENI, FIRE, JGN2plus, SOFIA[1-4]. These architectures decoupled network control layer and network data layer, for improving network transmission efficiency[5]. Meanwhile, according to the demands of cloud computing network, these architectures could modify network decision-making algorithms from a high-level network configuration, optimize network operations, and bring benefits of flexible, robust and credible. Specifically, OpenFlow[6], introduced by Nick McKeown, gives researchers a way to run experimental protocols in the networks. Based on Software Define Network (SDN) and OpenFlow protocol, researchers implemented network management and security functions mainly in the aspects of control, traffic forwarding and load balancing[7]. But the security issue in OpenFlow design should be thoroughly concerned, especially in the protection between the different controllers' nodes. The problem of security certification issues between the different single-controllers limit the application range of a SDN network architecture, which is only used in a single domain with a single controller.

In order to solve above issues, Casado et al.[8] proposed SANE network architecture in which logic controllers could provided secure authentication for device access and loading strategy. Moreover, Ethane[9] further extended SANE network architecture with data forwarding strategy by using core controller method. This new architecture implemented with data forwarding function, in a certain extent, could solve the security threats between controller layer and data forwarding layer. However, the security threats of controller and network equipment were not solved and still exist in the new network architecture.

Li[10] implement a deep analysis of new network architecture GENI and recognized that a large number of malicious attacks and flood attacks to network nodes can be easily spread through logic control layers and data forwarding layers and lead to corresponding secure issues, for instance, DDoS attack. Benton[11] evaluated the weakness of OpenFlow protocol. He thought OpenFlow had an accurate secure certification method between controllers and switches, but this method was not mandatory and did not provide security mechanisms under transport layer. So, an attacker could easily bypass authentication for malicious attacks. Apparently, above researches discovered a serious threat of new network architecture, but did not introduce a specific solution.

Based on SDN network architecture, Shalimov[12] did a comprehensive analysis for evaluating NOX, POX, Ryu, Beacon controllers in compatibility, reliability, security and processing capacity. Focusing on the aspect of security, the paper pointed out that the reason to most controllers' security problems were forged stream message length, protocol version or wrong type of message flow. Besides, Kreutz et al[13] analyzed the entire SDN security issues and pointed out that the new network architecture with a rapid response and anti-threaten security mechanism was needed to re-establish, especially in the differences of security threats between

978-1-4799-8261-5/15 $31.00 © 2015 IEEE
DOI 10.1109/ISADS.2015.29

controllers and traditional network.

Based on the above issues, the contributions of this paper are as follows:

- This paper introduces a new architecture to increase the probability of guaranteeing the credibility for future network architecture by applying the ideas of trusted network to SDN.
- We propose a trusted domain authentication protocol that protects controllers' credibility among entire network architecture when communicates with a non-trusted third party. The protocol gets trust certifications among different trusted domains and provides secure base in domain session.
- We demonstrate security of our trusted domain authentication protocol by AVISPA security analysis system.

II. THE SECURITY AUTHENTICATION PROTOCOL OF SDN TRUSTED DOMAINS

As shown in Fig.1, we propose a SDN trusted domain network architecture which combines OpenFlow and trusted computing. For solving the trust certification problem among trusted domains, we design some modules in SDN controller. Trusted Measurement Module (TMM), based on TCG Software Stack (TSS), is used to measure the sensitive information. Controller Flow Rule (CFR) is trusted policy which was measured by TMM, and Controller Communication Module (CCM) ensures controller authentication process between individual trusted domains.

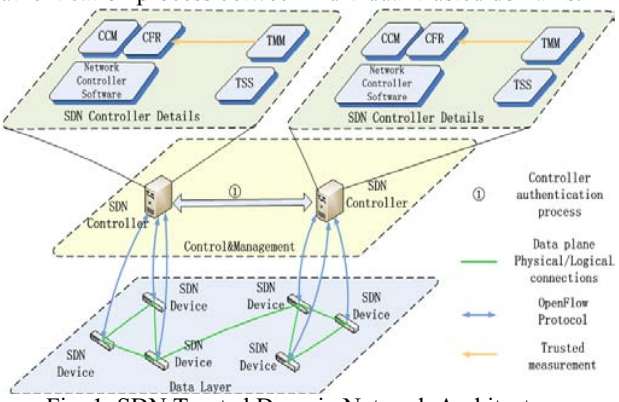


Fig. 1. SDN Trusted Domain Network Architecture

Related symbols explanation:

R: Authentication requester.
N: Authentication receiver.
N_{PUB}: Authentication requester public key.
R_{PUB}: Authentication receiver public key.
N^{-1}: Authentication requester private key.
R^{-1}: Authentication receiver private key.
$Plat\_ID_R$: Authentication requester platform ID.
$Plat\_ID_N$: Authentication receiver platform ID.
$Nonce_R$: Random number of authentication requester, used to measure hash and prevent replay attack.
$Nonce_N$: Random number of authentication receiver, used to measure hash and prevent replay attack.
AK_R: Random number of authentication requester, used to make session key by authentication receiver.
AK: Session key.
ACK: Authentication successful symbol.
CS_ID_R: Controller software ID of authentication requester.
CS_ID_N: Controller software ID of authentication receiver.
R_PCR_1: Controller platform hardware measurement of authentication requester.
R_PCR_2: Controller platform OS measurement of authentication requester.
R_PCR_3: Controller software measurement of authentication requester.
R_PCR_4: Controller application module measurement of authentication requester.
N_PCR_1: Controller platform hardware measurement of authentication receiver.
N_PCR_2: Controller platform OS measurement of authentication receiver.
N_PCR_3: Controller software measurement of authentication receiver.
N_PCR_4: Controller application module measurement of authentication receiver.
Hash(): Hash function based on HMAC_SHA-1 of TPCM

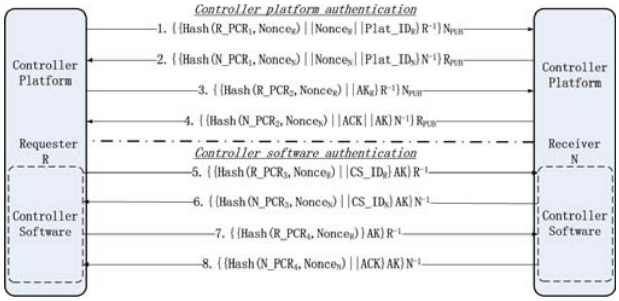


Fig. 2. Trusted Domain Authentication Process

As shown in Fig. 2, the specific certification process will be introduced in details in two parts: the controller platform certification and the controller software certification.

Step 1: R (the requested controller) sends an authentication request to N (the received controller). Primarily, R creates a digital signature for its own controller platform identity information $Plat\_ID_R$, random numbers $Nonce_R$ and hardware sensitive information R_PCR_1 by its own private key. Then, the public-key of N is expected to encrypt the hash value, and finally R sends the encrypted message to N.

$$R \rightarrow N: \{\{Hash(R\_PCR_1, Nonce_R) || Nonce_R || Plat\_ID_R\} R^{-1}\} N_{PUB} \quad (1)$$

Step 2: N receives the authorized information from R. Firstly, N receives the encrypted message including R's ID and N_{PUB} from step 1. Then, N will implement a negotiated registration by using R's platform identity information. And then, aiming the hardware sensitive information in the R's platform, N implements a credible verification which uses PCR values of N's hardware sensitive information to make hash with $Nonce_R$. If the hash result is consistent with N received information in step 1, R's hardware is trusted. Finally, if the verification is completed, N will create a digital signature for signing controller platform identify information $Plat\_ID_N$, random numbers $Nonce_N$ and hardware sensitive

information N_PCR_1, and correspondingly use the public key of R to encrypt them and send it to R. On the contrary, the negotiation is failed.

$$N \rightarrow R:\{\{Hash(N\_PCR_1,Nonce_N)||Nonce_N||Plat\_ID_N\}N^{-1}\}R_{PUB} \quad (2)$$

Step 3: R receives the authorized information from N. Firstly, R receives the verify feedback from N, which is encrypted by the public key of R, indicating that there is a trusted negotiated base between R and N. Secondly, $Plat\_ID_N$ could similarly implement its negotiated registration. Thirdly, as shown in step 2, R begins the credible verification with the hardware sensitive information of N, if the results are consistent, N can be trusted in hardware. Finally, if the verification passes, R will produce trust negotiation session random numbers, meanwhile N creates the negotiated private key and implements the next round trust negotiation. R will sign sensitive information of controller platform operation system hash value and encrypt the sensitive information by session key. The negotiation fails if the verification is not successful.

$$R\rightarrow N:\{\{Hash(R\_PCR_2,Nonce_R)||AK_R\}R^{-1}\}N_{PUB} \quad (3)$$

Step 4: N receives the hash sensitive information of the controller platform operation system R_PCR_2 and key random numbers of R, AK_R. Firstly, N receives R_PCR_2 and implements a credible verification, as shown in step 2, with the operation system measurement result N_PCR_2 of its own controller. Secondly, if the verification passes, N will produce a negotiated private key random number AK_N. Then, using AK_R, AK_N, the pseudo-random numbers and function PRGF, N will create a controller software verification session private key AK, which is bound by the platform information of R. Finally, N's platform sends the sensitive information hash value of N to R. The negotiation is failed if the credible verification is not successful.

$$N\rightarrow R:\{\{Hash(N\_PCR_2,Nonce_N)||ACK||AK\}N^{-1}\}R_{PUB} \quad (4)$$

Step 5: R receives the sensitive information hash value from the received platform operating system. Firstly, R receives the operating system sensitive information from the received controller platform and implements a credible verification, as shown in step 2, with the operating system measurement result R_PCR_2 of its own controller platform. Then, if the credible verification passes, the trust negotiation will enter into the controller software verification process, and, therefore, session private key AK will bind the information of N, in which AK is proceeding encryption on the sensitive information hash value of controller software and the controller software ID, also signing the encrypted information through R's private key. Finally, R returns the signed information to N. The negotiation is failed if the credible verification is not successful.

$$R\rightarrow N:\{\{Hash(R\_PCR_3,Nonce_R)||CS\_ID_R\}AK\}R^{-1} \quad (5)$$

Step 6: N received the controller software verification request from R. Firstly, N decrypts the information by R's public key, proving the existed base of the trust negotiation between R and N. Then, using the decryption of the sensitive information hash value of the controller software, N implements a credible verification as shown in step 2. Finally, if the credible verification passes, N will proceed a negotiated registration using the controller software information CS_ID_R of R, and encrypt the sensitive information hash value of controller software and the controller software ID through session private key AK and sign the encrypted information through R's private key. Finally, N returns the signed information to R. The negotiation is failed if the credible verification is not successful.

$$N\rightarrow R:\{\{Hash(N\_PCR_3,Nonce_N)||CS\_ID_N\}AK\}N^{-1} \quad (6)$$

Step 7: R received the verification information from N. Firstly, as shown in step 6, using the sensitive information hash value of the controller software, the received controller implements a credible verification as shown in step 2. Then, if the credible verification passes, R will implement a negotiated registration using the controller software information of N. Finally, R encrypts the sensitive information hash value of R_PCR_4 and $Nonce_R$ through session private key AK, and sign the encrypted information by R's private key. Finally, N returns the signed information to R. The negotiation is failed if the credible verification is not successful.

$$R\rightarrow N:\{\{Hash(R\_PCR_4,Nonce_R)\}AK\}R^{-1} \quad (7)$$

Step 8: Firstly, after receiving controller software application modules hash value measurement of R, N implements a credible verification as shown in step 2. Then, if the credible verification passes, N will encrypt the sensitive information hash value of N_PCR_4 and $Nonce_N$ through session private key AK and sign the encrypted information through R's private key. Finally, R implements a credible verification on application module sensitive information of N, shown in step 2. If the verification passes, the verification of the controller software will complete and the controller verification will pass. Otherwise, the negotiation is failed.

$$N\rightarrow R:\{\{Hash(N\_PCR_4,Nonce_N)||ACK\}AK\}N^{-1} \quad (8)$$

III. PROTOCOL SECURITY TESTING

This section will use the Dolev-Yao (DY) attack model[25] for actual security testing. The DY attack model could have the following variety of knowledge:

- Attackers are familiar with encryption, decryption, hashing, and other cryptographic operations, and they have the public key and private key of themselves.
- Attackers hold the network identity and public key of each subject.
- Attackers have basic password analysis ability.
- Attackers could perform a variety of attacks, such as replay attack.

A. Security goals

The DY attack model can control the entire network, and catch the data for tampering attack and replay attack. For ensuring the protocol secure, the authentication protocol must be detected by DY attack model. This paper sets up multiple security objectives for the DY attack model.

- N_PCR/R_PCR value is confidential during transmission.
- Random, $Nonce_R$/$Nonce_N$, is confidential during transmission.
- AK_R is confidential during transmission.
- Controller platform ID/Controller software ID is confidential during transmission.
- Controller software authentication session key is confidential.
- Successful certification logo(ACK) is completed.

B. *Security test*

1) *Test scenarios*

For the security goals, the paper sets up three test scenarios to test whether the protocol satisfies the security goals or not. As shown in table 1, in Scenario 1, we have implemented a single session with all the roles played by legitimate agents. In Scenario 2 and Scenario 3, we have tested the situations in which the intruder would impersonate each of the legitimate agents: the requested authentication controller platform (Scenario 2), the received authentication controller platform (Scenario 3)

TABLE I. SCENARIO DESCRIBE OF SECURITY TEST

Scenario	Scenario Describe
1	session(a,b,kab,h, ap_start,ap_init,ap_Asuccess,ap_Bsuccess)
2	session(a,i,kai,h, ap_start,ap_init,ap_Asuccess,ap_Bsuccess)
3	session(i,b,kib,h, ap_start,ap_init,ap_Asuccess,ap_Bsuccess)

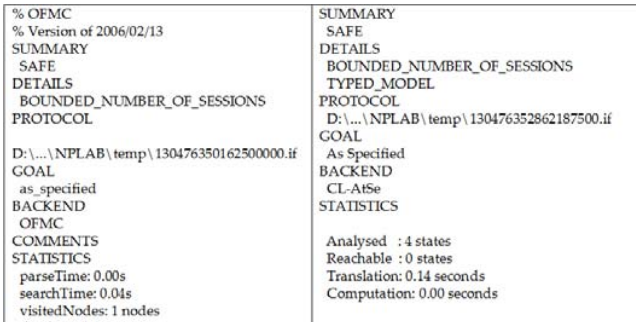
2) *Test results*

As shown in Fig. 3, the authentication protocol passed OFMC security test and ATSE security test, and the authentication protocol has not been attack by DY model, so we can conclude that the authentication protocol is secure.

From the Fig. 3, we could know that our authentication protocol was tested by CL-AtSe model and OFMC system, based on constraint logic, and our protocol was analyzed by 4 states. So, the conclusion of our test is quite convincing.

IV. CONCLUSION

In this paper, we introduced and demonstrated a security authentication protocol of SDN trusted domain, and designed the trusted domain network architecture to solve the credible problem of SDN network architecture. The trust negotiation concept with non-trusted third party is a prerequisite for communication between different SDN trusted domains. In the paper we only considered replay attacks and intermediator attacks. In the future, we plan to consider other attacker behavior models including opportunistic collusion attacks, random attacks and insidious attacks to further demonstrate superiority of our protocol.



a. OFMC System b. ATSE System

Fig. 3. Security Test Resul

References

[1] Greenberg A, Hjalmtysson G, Maltz D A, A clean slate 4D approach to network control and management[J]. ACM SIGCOMM Computer Communication Review, 2005, 35(5): 41-54.

[2] Elliott C. GENI: Opening up new classes of experiments in global networking[J]. IEEE internet computing, 2010, 14(1): 39-42.

[3] Gavras A, Karila A, Fdida S, Future internet research and experimentation: the FIRE initiative[J]. ACM SIGCOMM Computer Communication Review, 2007, 37(3): 89-92.

[4] JGN2plus. 2012. http://www.jgn.nict.go.jp/english/index.html.

[5] SOFIA. 2012. http://fi.ict.ac.cn/research/sofia_overview.html..

[6] Mckeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J. OpenFlow: Enabling innovationin campus networks. ACM SIGCOMM Computer Communication Review, 2008,38(2):69−74.

[7] Egilmez H E, Gorkemli B, Tekalp A M, Scalable video streaming over OpenFlow networks: An optimization framework for QoS routing[C]. Image Processing (ICIP), 2011 18th IEEE International Conference on. IEEE, 2011: 2241-2244.

[8] Casado M, Garfinkel T, Akella A, SANE: A protection architecture for enterprise networks[C]//USENIX Security Symposium. 2006.

[9] Casado M, Freedman M J, Pettit J, Ethane: Taking control of the enterprise[J]. ACM SIGCOMM Computer Communication Review, 2007, 37(4): 1-12.

[10] Li D, Hong X, Bowman J. Evaluation of security vulnerabilities by using ProtoGENI as a launchpad[C]//Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE. IEEE, 2011: 1-6.

[11] Benton K, Camp L J, Small C. Openflow vulnerability assessment[C]//Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM, 2013: 151-152.

[12] Shalimov A, Zuikov D, Zimarina D, Advanced study of SDN/OpenFlow controllers[C]//Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia. ACM, 2013: 1.

[13] Kreutz D, Ramos F, Verissimo P. Towards secure and dependable software-defined networks[C]//Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM, 2013: 55-60.