ATLANTIS PRESS
Advances in Computer Science Research (ACRS), volume 54
International Conference on Computer Networks and Communication Technology (CNCT2016)

SDNForensics: A Comprehensive Forensics Framework for Software Defined Network

Shu-hui ZHANG[1,*], Xiang-xu MENG[2] and Lian-hai WANG[3]

[1] School of Computer Science and Technology, Shandong University, Jinan 250101, China

[2] Shandong Computer Science Center (National Supercomputer Center in Jinan), Shandong Provincial Key Laboratory of Computer Networks, Jinan, 250014, China

Keywords: SDN, SDN security, SDN forensics, Control plane.

Abstract. Software-defined networking (SDN) is an emerging network architecture, which decouples the control and data planes of a network. Owing to its openness and standardization, SDN enables researchers to design and implement new innovative network functions and protocols in a much simpler and flexible way. However, the dynamism of programmable networks also brings potential new security challenges relating to various attacks such as scanning, spoofing attacks and denial-of-service attacks. We survey existing research efforts relating to both security challenges and promising solutions for SDN security problems. To the best of our knowledge, there are no published papers on SDN forensics. Before describing our forensics framework, the design goals and technical requirements of SDN forensics are discussed. Then SDNForensics, a comprehensive forensics framework, is proposed for the collection and analysis of digital evidence, built on SDN infrastructure.

Introduction

With the rapid expansion of the Internet and sharp increase in network applications, methods of processing and transmitting network traffic has become an important factor affecting sustainable development of the Internet. In the existing network architecture, a router is primarily responsible for forwarding packets. To adapt to a variety of traffic processing applications, network structures become increasingly complex and contained functions are constantly extended, including packet filtering, differentiated services, multicast, quality of service and traffic engineering. As a result, the forwarding unit of the router has become bloated. At the same time, with the development of cloud computing and big data, higher requirements in terms of reliability, scalability, openness, flexibility and control are imposed on the existing network infrastructure. To alleviate the contradiction between existing network architecture and growing network requirements, many research institutions and scholars have focused on developing new network architectures. Software-defined networking (SDN) has been proposed to meet the needs of the future Internet [1]. SDN has been widely researched in both academia and industry [2].

SDN technology separates the network control plane from the underlying network, replacing the traditional embedded closed control plane with an open software-mode control plane. The centralized controller is used to manage the entire network and allows the network to be programmed. The openness and flexibility of software-defined networking offer great changes to networks, especially to cloud services and data centers. However, problems such as a single point of failure are introduced. The security of the southbound and northbound interface protocol is beyond control. In

addition, the dynamic and open programmable architecture of SDN provides convenient and effective software attack vectors for attackers. With wider development and deployment of SDN, this security issue is becoming a key factor restricting further development [3, 4, 5].

Facing these problems, the contributions of this paper are described as following:

(1) The existing security problems of SDN are described and the state-of-the-art SDN security solution is discussed critically.

(2) Most research on SDN considers security from the OpenFlow perspective [6, 7, 8, 9]. To the best of our knowledge, none so far has focused on SDN forensics. Therefore, a comprehensive SDN forensic framework named "SDNForensics" is proposed. SDNForensics is designed based on SDN layers and security objects, that is, different extraction methods are carried out for different level of SDN.

The remainder of this paper is organized as follows. We introduce a simplified overview of SDN architecture to provide the fundamental background and present the challenges of SDN security in Section 2. The proposed framework based on clustering and convex function evidence theory is described thoroughly in Sections 3 and Section 4. Section 5 summarizes this study and indicates some opportunities for future research in this area.

Challenges of SDN

Structure of SDN

SDN architectures decouple network control and forwarding functions, enabling network control to become directly programmable and the underlying infrastructure to be abstracted from applications and network services [1]. The primary SDN planes have the following functions:

(1) Application plane: contains SDN applications for functions such as network management, policy implementation and security services. SDN applications communicate their network requirements and desired network behavior to the SDN controller via a northbound interface.

(2) Control plane: a logically centralized control framework that maintains a global view of the whole network and provides hardware abstractions to SDN applications. In this layer, the SDN controller translates requests from the SDN application layer down to the SDN datapaths and provides SDN applications with an abstract view of the network. The SDN datapath is a logical network device that exposes visibility and control over its forwarding and data processing capabilities.

(3) Data plane: the combination of forwarding elements used to forward traffic flows based on instructions from the control plane.

Network security techniques can be implemented as applications in the SDN application plane. From control plane, applications can obtain network status or resource information through northbound interface. Similarly, via the control plane, applications can collect samples of packets and redirect the traffic according to higher-level policies using the southbound API.

Security Threats to SDN

Potential attacks and vulnerabilities in SDN include unauthorized access, data leakage, data modification, malicious/compromised applications, denial of service (DoS),

configuration issue, and system level SDN security. According to its logical architecture, primary security threats to SDN are summarized as follows:

(1) Controller vulnerability: the security of the controller is vital for an SDN to operate safely. The controller has extensive control over the network. This has the advantages of fine granularity, real-time push and traffic monitoring. However, this has led to the controller becoming a focus for attack. Moreover, it is increasingly difficult to discover the origin of a specific attack using traditional intrusion detection system.

(2) Lack of a trust mechanism between the controller and applications: in SDN, there is no effective trust evaluation and trust management mechanism between the controller and the application. Security verification technologies for network devices and applications are different. Malicious applications can easily exploit this. Authorized legitimate applications can also be tampered with and applied to the controller.

(3) Switch vulnerability: after a switch is attacked, packets are abnormal. If an attacker sends false requests to a controller or other switch using this compromised switch, the threat can spread quickly to the whole network.

(4) Lack of trusted resources during evidence collection and repair procedures: For software-defined networking, fault repair requires a secure and reliable forensics mechanism to rapidly recover network functions.

Current Approaches and Limitations

Existing solutions to security problems in SDN can be divided into the following categories:

(1) Controller security

The availability of the SDN controller is of serious concern for the whole network. As a software platform, the SDN controller essentially supports potential hackers in reconfiguring the entire network. By spoofing the address of an SDN controller, an attacker can take over or bring down the entire network by means of a fake controller. Thus, securing the SDN controller is paramount. Research on controller security can be summarized as being either evolutionary or revolutionary. For example, SE-Floodlight was designed as a Security Enhanced version of the widely used OpenFlow Floodlight Controller [9]. In contrast, revolutionary controller is designed to develop entirely new SDN controller with security mechanism. For example, ROSEMARY controller is proposed to implement network application containment and resilience strategy based around the notion of spawning applications independently within a micro-NOS [10].

(2) Application security

SDN application security refers to the application vulnerabilities in the core device. The flow rules in SDN are fundamental for packet processing and are dynamically generated by a variety of security applications. The security of some important applications is a key factor in ensuring SDN security. To improve application security, Ball et al. proposed VeriCon, a tool to verify that an SDN program was correct on admissible topologies and for all possible sequences of network events [11]. Similarly, Canini et al. proposed a detection module for vulnerability testing and verification [12].

(3) DoS/distributed (D)DoS attack defense

There are currently three main methods for mitigating DoS/DDoS attacks. First, DoS/DDoS behavior is detected based on traffic characteristics. For example Braga proposed a lightweight method for DDoS attack detection based on traffic flow features [13]. Second, DoS/DDoS attacks are prevented based on connection migration

mechanism. For instance, Shin et al. proposed "AVANT-GUARD", an effective security framework to detect DDoS attacks [14]. Third, based on the STRIDE and Unified Modeling Language (UML), potential DoS/DDoS attack behaviors in SDN are evaluated and predicted. STRIDE is a classification scheme for characterizing known threats according to the kinds of exploit that are used.

The solutions described above have advanced the theory and development of SDN security. However, some key security issues remain:

(1) Most SDN controllers are primarily designed to schedule and control network resources, including link discovery, topology management and policy formulation. Security features are given less attention.
(2) SDN application and switch security measures are still in their infancy. Functional verification of applications is emphasized and application security validation is frequently overlooked. Similarly, although research on acquisition and analysis of network packets has been conducted, switch security tends to be ignored.
(3) There is still no research on SDN forensics. In 2016, the Digital Forensic Research Workshop organized a challenge to promote the development of SDN security and forensics [15]. The main traditional computer forensics techniques discussed included evidence association analysis, data confusion and automated analysis. Based on data fusion, Ma G et al. proposed a digital forensic model, which carried out time constraint and legal restriction in linear course control. Furthermore, the model supervised the whole forensic process [16]. Wang L et al. proposed a model of live computer forensics based on physical memory analysis [17].

The present paper aims to address the third issue, and to propose a solution for SDN forensics.

Design Goals and Requirements

Design Goals

Based on previous analysis in Section 2, the SDNForensics framework is designed based on the following principles:

(1) Resilience: SDNForensics should tolerate faults, account for failures and adapt dynamically to the configuration changes of SDN network. Overall task execution will be expedited by splitting digital evidence into atomic entities.
(2) Modularity: SDNForensics will be a hosting environment for pluggable modules. The framework will not only launch component modules, but also offer some baseline features such as security, intermodule communication and logging. A module is meant as an opaque container of functions with a predefined common structure. The implementation method of modules should depend on the type of data being processed.
(3) Openness: SDNForensics will be open, as this architecture is more easily portable, interoperable, inspectable and subject to contributions. This is important, particularly in digital forensics, that all stakeholders should be able to reproduce all operations in the most forensically sound way.

Requirements

The SDNForensics framework requirements are detailed as follows:

(1) Alternative analysis: when accuracy could be deliberately traded for speed, for instance, it is imperative to achieve a very swift overview of the SDN network. SDNForensics is therefore required to give the user the opportunity to select completeness or speed for content and metadata extraction from digital evidence.
(2) Information extraction: traditional data acquisition techniques entail deep domain knowledge, because the investigator is required to know in advance what to look for to feed the search engines. SDNForensics should include a data extraction layer, so that data can be viewed from different or unexpected perspectives.
(3) Simplified interfaces: SDNForensics should avoid overwhelming the investigator attention calls and provide intuitive interfaces.
(4) Case management: SDNForensics will present enhanced case management features. Some evidence details such as acquisition hashes should be calculated automatically, if available, in most widespread formats. SDNForensics should support common disk image file formats and data packages. For example, for network captures, import modules should be able to parse high-level protocols, and extract relevant stream content and metadata.
(5) Security: the threat model considers a private deployment and assumes that no harm can come from insiders. Minimum security requirements include encryption, user multifactor authentication, evidence content tampering control and audit trails.

SDNForensics: A Comprehensive Forensics Framework

Based on the concept of on-demand forensics, the SDNForensics framework is proposed to obtain, purify, store and integrate various data related to cybercrime. A credible chain of evidence can be formed using SDNForensics (described in figure 1). This comprises the data acquisition module, behavior database, data extraction module, data fusion module, abnormal behavior detection module, security alarm module, evidence conservation module and evidence reporting module. The main parts of SDNForensics are described as follows:

(1) Data acquisition: different types of data are obtained from distinct layers. From the application plane, runtime log information is obtained. From the control plane, memory and runtime log information are obtained. From the data plane, memory, network packet and runtime log information are obtained. Runtime logs and memory and disk information are obtained from hosts.
(2) Data extraction: startup time, end time and exception information may be extracted from runtime logs. Abundant real-time information can be extracted from memory images. For instance, by analyzing a memory image from an SDN switch, the type of SDN switch and controller in use, connected hosts and flow rules can be extracted. Running processes, loaded modules, network connections, system configuration information and running internal virtual machines can be extracted from memory images of hosts.
(3) Data fusion: information of different types and from different sources should be integrated, including physical memory information from the controller, physical memory information from the switch, network data, physical memory

information from the host layer and disk information. Cluster analysis is used to preprocess the collected data to form clusters. By using all the distributions belonging to each cluster, the centroid of each cluster is calculated as the representative information of the cluster. The basic probability assignment method is used to realize multi-source information fusion.

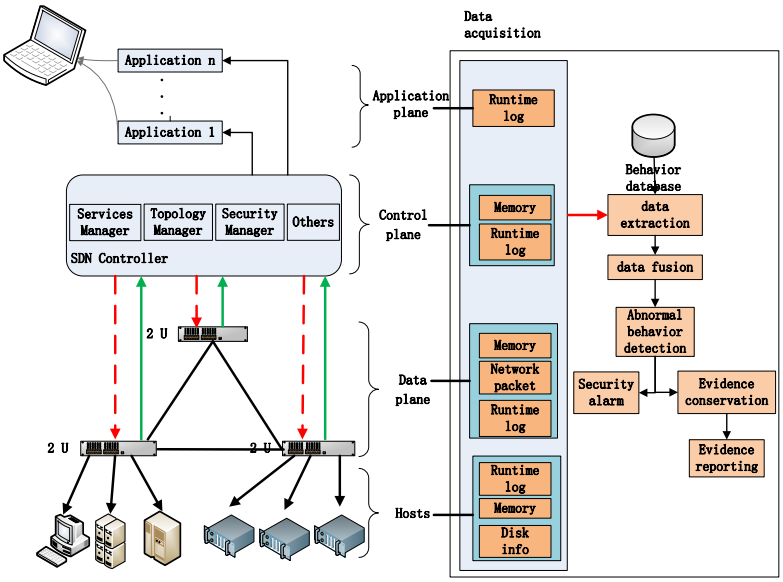

Figure 1 structure of SDNForensics

(4) Anomaly detection: anomalies may be detected using machine learning algorithms, including k-nearest neighbor, support vector machine and fuzzy logic based methods. Anomaly detection can operate in supervised, semi-supervised and unsupervised anomaly detection modes.

(5) Security alarm: according to the analysis above, the security alarm module can be triggered in various ways, including a siren, strobe, and/or notifications to a control room or directly to the administrator via email or phone.

(6) Evidence conservation: the collected data and the analysis results are preserved according to case management rules. Different types of evidence are stored in appropriate file formats or databases.

Conclusions and Future Work

According to logical architecture of SDN, principal security threats to SDN are summarized, including controller vulnerability, lack of a trust mechanism between the controller and applications, switch vulnerability and lack of trusted resources. Although several solutions to controller security, application security and DoS/distributed (D)DoS attack defense have existed. Aiming at the prolem of SDN forensics, SDNForensics, a comprehensive SDN forensic framework is proposed. Software development of a working prototype is underway. Meanwhile, research on data extraction and data fusion is in progress.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (Grant Nos. 61572297, and 61602281), the Shandong Provincial Natural Science Foundation of China (Grant Nos. ZR2016YL014, ZR2016YL011, ZR2014FM003, and ZY2015YL018), the Shandong Provincial Outstanding Research Award Fund for Young Scientists of China (Grant Nos. BS2015DX006, and BS2014DX007), and the Shandong Academy of Sciences Youth Fund Project, China (Grant Nos. 2015QN003).

References

[1] Zhang CK, Cui Y, Tang HY and Wu JP, State-of-the-Art survey on software-defined networking (SDN), Ruan Jian Xue Bao/Journal of Software, 2015,26(1), pp. 62-81 (in Chinese).

[2] Jin D, Nicol D M, Parallel simulation of software defined networks, In: Proc. of the 2013 ACM SIGSIM Conf. on Principles of Advanced Discrete Simulation. ACM, 2013, pp. 91-102.

[3] Sandra Scott-Hayward, Sriram Natarajan and Sakir Sezer, A Survey of Security in Software Defined Networks, IEEE COMMUNICATION SURVEYS & TUTORIALS, 2016, 18(1), pp. 623-654.

[4] Syed Taha Ali, Vijay Sivaraman and Sanjay Jha, A Survey of Securing Networks Using Software Defined Networking, IEEE TRANSACTIONS ON RELIABILITY, 2015, 64(3), pp. 1086-1097.

[5] Izzat Alsmadi, Dianxiang Xu, Security of Software Defined Networks: A survey, Computers & Security, 2015, 53, pp. 79-108.

[6] Zuo QY, Chen M, Zhao GS, Xing CY, Zhang GM and Jiang PC, Research on OpenFlow-based SDN technologies, Ruan Jian Xue Bao/Journal of Software, 2013,24(5), pp. 1078−1097 (in Chinese).

[7] Porras P, Shin S, Yegneswaran V, Fong M, Tyson M and Gu G, A security enforcement kernel for OpenFlow networks, In: Proc. of the 1st Workshop on Hot topics in Software Defined Networks. Helsinki: ACM, 2012, pp. 121-126.

[8] Son S, Seungwon S, Yegneswaran V, Porras P and Guofei G, Model checking invariant security properties in OpenFlow, In: Proc. of the 2013 IEEE Int'l Conf. on Communications (ICC). Budapest: IEEE, 2013, pp. 1974-1979.

[9] Porras P, Cheung S, Fong M, Skinner K and Yegneswaran V, Securing the software-defined network control layer, In: Proc. of the 2015 Annual Network and Distributed System Security Symp. (NDSS 2015). San Diego: Internet Society, 2015, pp. 1-15.

[10] Seungwon Shin, Yongjoo Song, Taekyung Lee, Sangho Lee, Jaewoong Chung, Phillip Porras, Vinod Yegneswaran, Jiseong Noh, and Brent Byunghoon Kang, Rosemary: A Robust, Secure, and High-performance Network Operating System. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14). ACM, New York, NY, USA, 2014, pp. 78-89.

[11] Ball T, Bj N, Rner, Gember A, Itzhaky S, Karbyshev A, Sagiv M, Schapira M and Valadarsky A, VeriCon: Towards verifying controller programs in software-defined

networks, In: Proc. of the 35th ACM SIGPLAN Conf. on Programming Language Design and Implementation. Edinburgh: ACM, 2014, pp. 282-293.

[12] Canini M, Venzano D, Pere P, Ni, Kosti D and Rexford J, A NICE way to test openflow applications, In: Proc. of the 9th USENIX Conf. on Networked Systems Design and Implementation. San Jose: USENIX Association, 2012, pp. 1-14.

[13] Braga R, Mota E and Passito A, Lightweight DDoS flooding attack detection using NOX/OpenFlow, Local Computer Networks (LCN), 2010 IEEE 35th Conference on. IEEE, 2010, pp. 408-415.

[14] Shin S, Yegneswaran V, Porras P and Guofei Gu, AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks, Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013, pp. 413-424.

[15] DFRWS 2016 Forensics Challenge. https:// www.dfrws.org/ 2016/ challenge/ index.shtml

[16] G. Ma, C. Sun and Z. Wang, Study on digital forensics model based on data fusion, 2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), Jilin, 2011, pp. 898-901.

[17] L. Wang, R. Zhang and S. Zhang, A Model of Computer Live Forensics Based on Physical Memory Analysis, 2009 First International Conference on Information Science and Engineering, Nanjing, 2009, pp. 4647-4649.