Summary for Papers on SDN Security

Mohammed Latif Siddiq

August 31, 2020

Abstract

In this report, we will summarize the findings and possible future work of some papers on Software-Defined Networking (SDN) security.

1 Introduction

Software-Defined Networking (SDN) refers to anapproach for network programmability, that is, the capacity to initialize, control, change, and manage network behavior dynamically via open interfaces. SDN emphasizes the role of software in running networks through the introduction of an abstraction for the data forwarding plane and, by doing so, separates it from the control plane. This separation allows faster innovation cycles at both planes as experience has already shown. It has basically three layers: Application, Infrastructure and Control layer. SDN is mainly associated with OpenFlow protocol for infrastructure layer and there are some open source controllers, such as, Floodlight, OpenDaylight, OpenContrail, POX and RYU controller.

2 Findings on SDN Security

Table 1 summarizes the finding from papers on SDN security. In the next sections, we are going to discuss the findings in details.

2.1 Unexpected Data Dependency Creation and Chaining

In this paper, they introduce D2C2(data dependency creation and chaining) attack, which leverages some widely-used SDN protocol features (e.g., custom fields) to create and chain unexpected data dependencies in order to achieve greater reachability. They have developed a novel tool, SVHunter, which can effectively identify D2C2 vulnerabilities.

To perform the attack, there are two insights. One is custom field, can be abused to help attackers send malicious data into the control plane and another is the data dependencies among the many seemingly-separated SDN services/applications can actually be created in unexpected ways(i.e., data item poisoning) to result in the exposure of previously unreachable sensitive methods/APIs to attackers.

Serial No.	Findings in short	Defence Tool
[1]	Creating malicious data dependencies to abuse previously inaccessible sensitive methods/APIs in controllers.	**SVHunter**, pinpoint a wide range of sensitive methods in SDN controllers and create dependencies to attack these methods.
[2]	Buffered Packet Hijacking, a malicious application to hijack buffered packets to launch a number of attacks bypassing existing defense systems which is due to the lack of consistency check between buffer IDs and match fields when installing flow rules	**ConCheck**, blocks API calls that an application uses to generate flow rules when there is inconsistency.
[3]	CrossPath attack that disrupts the SDN control channel by exploiting the shared links in paths of control traffic and data traffic.	
[4]	A novel attack against SDN networks that can cause serious security and reliability risks by exploiting harmful race conditions in the SDN controllers, similar in spirit to classic TOCTTOU (Time of Check to Time of Use) attacks against file systems.	**CONGUARD**, that can effectively detect and exploit harmful race conditions.
[5]	In SDN, control plane decisions are often based on the data plane, it is possible for carefully crafted malicious data plane inputs to direct the control plane towards unwanted states that bypass network security restrictions(i.e., cross-plane attacks)	**EVENTSCOPE**, a vulnerability detection tool that automatically analyzes SDN control plane event usage, discovers candidate vulnerabilities based on missing event-handling routines, and validates vulnerabilities based on data plane effects.
[6]	The data/control plane decoupling of the SDN framework makes the traditional network troubleshooting tools unsuitable for pinpointing the root cause in the control plane.	**ForenGuard**, which provides flow-level forensics and diagnosis functions in SDN networks.

Table 1: Findings from Papers on SDN Security

Using the insights, there tool SVHunter uses a tracer which identifies the usages of sensitive methods and analysis techniques to backward trace data flows from each parameter of each identified sensitive method to its data source. It will then used as context information for the next steps. In the next step, the tool represent the poisoning events in a unified model got from the tracer which describes the causality of each poisoning event. Then the Reasoning Engine reasons the generated causality representations to decide whether and how two or more poisoning events can be chained together. After chaining the events, it crafts the payload by custom field data injector.

The paper works only on Java based controller which can be extended to other languages i.e., python based controller such as RYU, POX controller.

2.2 Buffered Packet Hijacking in SDN

In this paper, they discussed about a new attack called buffered packet hijacking. There are two types of buffered packet hijacking: intra-chain and inter-chain. In both way, whenever there is a process to create a new flow rule, there are several applications involved in this process. If there is a malicious application before the application which will write the legitimate flow rule, it can pretend to add or update flow rules for which it is responsible. This can cause Cross-app poisoning, Network Security Policy Bypass, TCP Three-Way Handshake Disruption, Control Traffic Amplification.

For the defence, they developed a tool called *ConCheck*, which has two modules: API Calls Extractor and Consistency Checker.

2.3 The CrossPath Attack

In this paper, they presented the CrossPath attack to significantly disrupt the SDN control channel by exploiting the shared links between paths of control traffic and data traffic. They developed a probing technique called *adversarial path reconnaissance* that can find a target path containing the shared links. They proved the conditions of successful probing, analyzed the expected number of explored paths to find a target path.

In defence for the attack, they proposed to ensure forwarding control traffic with high priority, which thus can protect control traffic from being congested by malicious data traffic and proactively reserve bandwidth for control traffic.

2.4 Races in the SDN Control Plane

The network states maintained in the SDN control plane is subject to harmful race conditions. Such as, *Non-adversarial causality*, asynchronous network events and non-determinist schedules and *Adversarial causality*, an attacker can intentionally inject right network events to exploit vulnerabilities i.e. State Manipulation Attacks. This can causes System Crash Connection Disruption, Service Disruption, Service Chain Interference, Privacy Leakage etc.

CONGUARD is proposed in this paper which contains two main phases: (i) locating harmful race conditions in the controller source code by utilizing dynamic analysis and adversarial state racing, (ii) triggering harmful race conditions in the running SDN controller by remotely injecting right external network events with the proper timing.

There are some limitation for CONGUARD for finding harmful excution path and controller dependencies.

2.5 Cross-Plane Event-Based Vulnerabilities

In this paper, they built an automated approach to analyze event use by applications that identifies likely missing event handling and checks whether this lack of event handling can cause data-plane effects in combination with other apps. They also designed the event flow graph data structure, which allows for succinct identification of (a) event dispatching, event listening, and API use among SDN components, as well as (b) the context to realize vulnerabilities.

They designed EVENTSCOPE to identify cross-plane event-based vulnerabilities in three phases. The first phase, the candidate *vulnerability generator*, takes the set of SDN apps as input and produces a list of unhandled event types for each app. The second phase, the *event flow graph generator*, takes the apps' code, the controller's code, and a definition of controller API calls as inputs and constructs an event flow graph that records how events propagate and influence the system. Finally, the event flow graph and the unhandled event types from the first two phases are combined in the third phase, the *vulnerability validator*, to identify the data plane impacts of unhandled event types. The output of this phase results in a list of vulnerabilities that can influence the data plane as a result of unhandled event types.

2.6 Network Security Forensics and Diagnosis in the SDN Era

In this paper, they proposed a novel forensics scheme which dynamically logs the activities of both the SDN control plane and data plane, and builds event-oriented execution traces and state transition graphs for diagnosing network forwarding problems. They proposed a diagnosis tool which provides an inference-based approach to query the logged elements that have dependency relationships with the queried ones.

They developed ForenGuard, which helps network operators trace back past activities of both the control plane and data plane and pinpoint the root causes of network security problems. Our evaluation shows that ForenGuard is useful for diagnosing common SDN net-working security problems with minor run time overhead.

3 Conclusions

In paper [1], [5] and [6], they try to find out the dependencies between data and control plane to pinpoint the possible attack on sensitive functions and data dependencies in SDN network. In paper [2], they use Buffer Packet in SDN switch to demonstrate attacks to bypass existing defence systems. They imply on checking consistency in the buffer packets. In paper [3], they try to find out shared link which can later be used on disrupts SDN control channel. In paper [4], they state about race condition among SDN applications which can be exploited for creating security attack. In paper [5] and [6], they try to find out the dependencies between data and control plane to pinpoint the possible attack on sensitive functions in SDN network.

References

[1] Feng Xiao, Jinquan Zhang, Jianwei Huang, Guofei Gu, Dinghao Wu, and Peng Liu, *Unexpected Data Dependency Creation and Chaining: A New Attack to SDN*, (2020 IEEE Symposium on Security and Privacy (SP)).

[2] Jiahao Cao, Renjie Xie, Kun Sun, Qi Li, Guofei Gu, and Mingwei Xu, *When Match Fields Do Not Need to Match: Buffered Packet Hijacking in SDN*, (Network and Distributed Systems Security (NDSS) Symposium 2020)).

[3] Jiahao Cao, Qi Li, Renjie Xie, Kun Sun, Guofei Gu, Mingwei Xu, and Yuan Yang, *The CrossPath Attack: Disrupting the SDN Control Channel via Shared Link*, (28th USENIX Security Symposium (USENIX Security 19)).

[4] Lei Xu, Jeff Huang, Sungmin Hong, Jialong Zhang, and Guofei Gu, *Attacking the Brain: Races in the SDN Control Plane*, (26th USENIX Security Symposium (USENIX Security 17)).

[5] Benjamin E. Ujcich, Samuel Jero, Richard Skowyra, Steven R. Gomez, Adam Bates, William H. Sanders, Hamed Okhravi , *Automated Discovery of Cross-Plane Event-Based Vulnerabilities in Software-Defined Networking*, (2020 Network and Distributed System Security Symposium (NDSS)).

[6] Haopei Wang, Guangliang Yang, Phakpoom Chinprutthiwong, Lei Xu, Yangyong Zhang, Guofei Gu, , *Towards Fine-grained Network Security Forensics and Diagnosis in the SDN Era*, (CCS '18: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security).