# Evolving Network Security in the Era of Network Programmability

Mingming Chen
mzc796@psu.edu
The Pennsylvania State University
University Park, PA, USA

## Abstract

Software-defined networking (SDN) is a centralized network architecture enabling dynamic, programmable, and flexible network management, which advances technologies like network security. However, it also introduces new vulnerabilities due to the segregation of data, control, and application planes, creating additional attack surfaces and security gaps from the increased complexity of programmability, flexibility, and scalability.

To empower network security with SDN, we develop a coordinated sampling strategy using P4 programming for adaptive network monitoring. Additionally, we uncover a flow entry-induced topology poisoning attack to highlight security gaps from unplanned module integration. Finally, we propose to fortify the SDN control plane by generalizing SDN security policies and fuzzing it to uncover unknown vulnerabilities.

## CCS Concepts

• **Security and privacy** → **Distributed systems security**; **Security protocols**; **Intrusion detection systems**; **Software security engineering**; • **Networks** → *Link-layer protocols*; **Network protocol design**.

## Keywords

SDN; P4; Coordinated Sampling; Budgeted Maximum Multi-Coverage; Flow Entry-induced Topology Poisoning; Reinforcement Learning; Misuse Testing; Fuzzing

**ACM Reference Format:**
Mingming Chen. 2024. Evolving Network Security in the Era of Network Programmability. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3658644.3690859

## 1 Introduction

Software-defined networking (SDN) is an architectural approach to network management, that centralizes control plane functions and decouples them from the data plane. This separation allows for flexible, dynamic, and programmable network management, and provides the following benefits:

- SDN stimulates innovation: The centralized control plane enables network virtualization (NV). Network function virtualization (NFV) is also supported on SDN due to its flexible network management which supports orchestration and automation, service function chains, and fast deployment.
- SDN accelerates technologies' development: Cloud computing is well supported because SDN reduces network complexity allowing cloud providers to configure network resources quickly and dynamically even across multiple clouds. SDN also provides fast and flexible data transmission to support emerging distributed and federated learning.
- SDN enhances the performance of various scenarios: SDN empowers traffic engineering to quickly adapt to changing traffic patterns and service demands on data centers, wide area networks, IoT, 5G, and vehicular networks.
- SDN facilitates machine learning (ML) on network applications: With the centralized control provided by SDN, ML has been adopted to solve various network problems including routing, load balancing, and intrusion detection.

In addition to the advantages SDN provides, network security is generally enhanced by SDN because it provides fine-grained control over traffic flows, enforcing security policies at the network level. SDN not only assists in fast blocking/rerouting of known dangerous traffic but also helps monitor traffic to find potentially harmful flows. By collecting and analyzing traffic data, SDN supports various attack detection methods. For example, network traffic monitoring can be integrated into a single device (e.g. controller, switch, or a dedicated network monitor) or distributed [5]. Distributed traffic monitoring is practical because big data processing techniques are available to process and analyze monitored traffic data in distributed systems [5]. SDN stimulates many security applications such as network monitoring and intrusion detection [13].

However, SDN also introduces additional attack surfaces due to its decoupled, centralized architecture. Attack surfaces on the southbound interface (between the control plane and data plane) have been extensively studied, with examples including race condition attacks on insecure controller implementations [12] and topology poisoning attacks initiated from malicious hosts or switches misleading the controller with fabricated links around them [8]. Work has also been done to enhance SDN northbound (between control plane and application plane) security, focusing on preventing malicious SDN applications from attacking northbound APIs [9], and system-level vulnerabilities [2, 11].

Although much effort has been devoted to both the network security enabled by SDN and securing SDN, research gaps remain. On one hand, the programmable data plane enabled by the P4 language releases extensive power augmenting the programmable control


*CCS '24, October 14–18, 2024, Salt Lake City, UT, USA*

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690859

plane enabled by SDN architecture. On the other hand, stealthy attacks can still evade existing detections, with no systematic approach to uncover these vulnerabilities. Besides, the east-westbound interface (between controllers) has been largely overlooked even though the controller cluster is essential in practice.

In this research statement, we exploit SDN programmability to enhance network security and strive to protect SDN architectures with multi-controllers against sophisticated attacks originating from the control plane. First, we introduce the concept of coordinated sampling in Section 2.1. This innovative approach involves distributing the flow sampling workload across multiple switches and coordinating the sampling of the same flows. In Section 2.2, we present a flow entry-induced topology poisoning attack that can be initiated from a malicious controller within a controller cluster or a malicious SDN application. The flow entry-induced topology poisoning causes all controllers to discover the deceptive topology indefinitely by inserting normal flow entries. The deceptive topology is computed to induce benign controllers to assist with malicious activities, such as redirecting flows to an eavesdropping point or bypassing monitoring points.

Ultimately, we strive to strengthen the security design of SDN multi-controller architectures in Section 3. As many SDN security research works have been conducted on the OpenFlow protocol, we propose evaluating various SDN southbound protocols, particularly the P4 runtime interface, which significantly impacts network security with its programmable data plane. As those protocols share the characteristics of connecting SDN forwarding devices to the SDN controllers, the vulnerabilities may also be similar, especially for those initiated from the control plane. We aim to develop a series of SDN control plane security policies by which to securely implement SDN controllers to avoid security flaws by design. Furthermore, a fuzzing framework on the SDN controller may uncover unknown stealthy vulnerabilities.

## 2 SDN: The Double-Edged Sword

SDN offers both opportunities and risks to security. Controllers' topology view enables coordinated sampling, but a deceptive topology can mislead controllers to divert flows from sampling points.

### 2.1 P4-enabled Coordinated Sampling

To address the challenge of supporting high-rate, flow-based sampling within the resource constraints on each switch, a coordinated sampling framework is developed [3]. We deploy multiple P4-programmable switches along flows and coordinate packet sampling among them. This framework enables dynamic sampling point activation, deactivation, and runtime configuration, leveraging P4's programmability. Our P4 coordinated sampling algorithm incurs negligible overhead on throughput and delay with substantially low activation and deactivation time (around 0.05 and 0.01 seconds, respectively) as demonstrated on Arista 7170CD switches.

Given the substantial cost disparity between P4-programmable switches and ordinary SDN switches, we formulated the P4 programmable switch placement problem as a budgeted maximum multi-coverage problem to determine the optimal placement of a budgeted number of P4-programmable switches to achieve maximum flow sampling coverage. This NP-complete integer linear programming is pseudo-polynomial solvable on realistic topologies as confirmed by theoretical proofs and experiments.

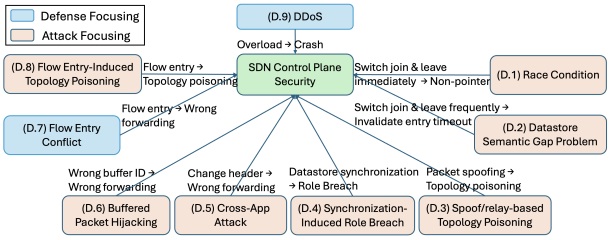


**Figure 1: Attacks on SDN Control Plane**

### 2.2 Flow Entry-induced Topology Poisoning

By identifying an overlooked vulnerability (CVE-2024-37018) in which flow entries designed for traffic routing can influence topology discovery results, we uncovered a new attack vector such that a malicious application or a malicious controller in a multi-controller architecture may craft poisonous flow entries, leading a legitimate controller to independently discover a deceptive topology conducive to malicious activities [4]. Leveraging this attack capability, we designed a reinforcement learning (RL) model to compute a deceptive topology with the same degree sequence and high graph similarity to the real topology, tailored to specific goals. The attack has been successfully tested against five trending open-source SDN controllers and nine SDN topology discovery protocols including OFDP.

## 3 Fortifying the SDN Control Plane

A systematic way to detect stealthy vulnerabilities on SDN is desirable because existing defenses do not address vulnerabilities that exist across modules. In Figure 1, we studied recent SDN attacks on the control plane with an observation that they share the character of exploring cross-module vulnerabilities to remain stealthy. We plan to address this issue in two steps. First, we study the misuse vulnerabilities on various SDN southbound interfaces, and develop a series of protocol-independent security policies to safeguard SDN controller implementations. Then, we aim to discover further unknown cross-module vulnerabilities by fuzzing with learning techniques.

### 3.1 Misuse Testing on SDN Southbound

As the de-facto SDN southbound protocol, OpenFlow has been well-studied by the research community. However, OpenFlow is not equivalent to SDN. Both of the open source SDN controllers OpenDaylight and ONOS support multiple southbound interfaces including OpenFlow, P4, OVSDB, BGP, NetConf, etc [6, 7]. In practice, many companies use and customize various southbound interfaces based on their SDN architecture [1]. Consequently, we plan to expand the security analysis on SDN southbound interfaces focusing on P4 runtime, which is an interface used to control P4-programmable switches [10].

We believe that misuse testing on the SDN southbound interfaces can generate a series of security policies for controller implementations due to two observations: (1) The fundamental cause of many existing attacks studied on OpenFlow-based SDN is the SDN architecture. OpenFlow is simply the carrier of the attacks. For example,

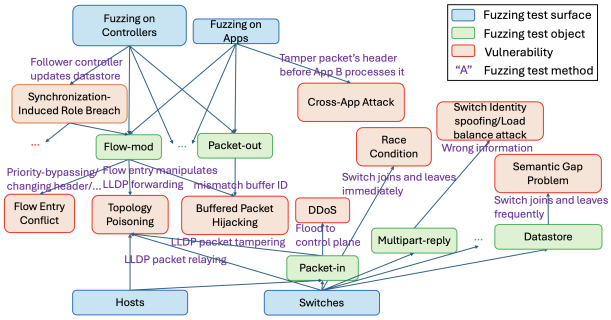


**Figure 2: Fuzzing Prototype on SDN**

the (D.1) race condition attack is essentially a classic TOCTTOU (Time of Check to Time of Use) attack on file systems, exploring the insecure controller implementation which is orthogonal to the southbound protocol. Other cases include the (D.9) DDoS attack, (D.2) Datastore Semantic Gap problem, and (D.4) Synchronization-induced role breach. (2) SDN Southbound interfaces other than OpenFlow tend to adopt OpenFlow methods to achieve SDN functionalities. Because OpenFlow is initiated to enable the SDN architecture, its features fit the SDN architecture naturally. For example, the first physical network running P4 Runtime at the SDN NFV World Congress uses the de-facto OpenFlow discovery protocol (OFDP) to discover links [10]. Consequently, the (D.8) Flow entry-induced topology poisoning attacks can also attack such an SDN architecture composed of P4-programmable switches. Other cases may include (D.3) spoof/relay-based topology poisoning, (D.7) Flow entry conflict, (D.5) Cross-App Attacks, and (D.6) Buffered Packet Hijacking.

## 3.2 Exploring Unknown SDN Vulnerabilities

Ultimately, we plan to use fuzzing techniques to discover unknown vulnerabilities that enable sophisticated attacks originating from the SDN control plane. As most of the stealthy attacks shown in Figure 1 originate from the design flaws across modules, we target the cross-module vulnerabilities and fuzz the related configuration messages to explore potential vulnerabilities. In Figure 2, we depict a fuzzing framework for an SDN architecture with OpenFlow protocol, inspired by Figure 1. The fuzzing framework will extend to the protocol-independent P4 runtime API and be generalized to any southbound interfaces based on the result of Section 3.1.

The architecture operates in dynamic SDN scenarios with application, control, and data planes. We focus on fuzzing the control plane with randomized inputs to configure the network. The invariant is that the network behavior must follow legitimate instructions. The fuzzing oracle observes runtime network behaviors and compares them with the invariant to discover vulnerabilities.

There are several challenges of fuzzing on the SDN control plane. First, a protocol-independent fuzzing framework needs a comprehensive analysis of various SDN southbound protocols to provide fuzzing guidance. Second, the large number of southbound messages makes it challenging to fuzz efficiently. Third, the dynamic analysis depends on various scenarios which are unknown.

To tackle those challenges, we envision reinforcement learning and feedback on the network state may help with efficiency. The protocol-independent feature depends on the result of Section 3.1 to provide generalized guidance for fuzzing.

# 4 Conclusion

Since SDN technology has been widely implemented in the industry, building a secure and robust SDN control plane is crucial to prevent evolving cyberattacks. A systematic framework for testing SDN controllers' security is essential to release their full potential.

# 5 Acknowledgements

I thank my advisors, Thomas La Porta and Trent Jaeger, for their invaluable advice and persistent support. I also thank my collaborators, Teryl Taylor, Frederico Araujo, and Benjamin E. Ujcich, for their constructive feedback and expert guidance. This research was sponsored by the U.S. Army Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory of the U.S. government. The U.S. government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation here on.

# References

[1] Engineering at Meta. Facebook open switching system ("fboss") and wedge in the open. https://engineering.fb.com/2015/03/10/data-center-engineering/facebook-open-switching-system-fboss-and-wedge-in-the-open/, 2015-03-05. Accessed on 2024-07-09.

[2] Jiahao Cao, Renjie Xie, Kun Sun, Qi Li, Guofei Gu, and Mingwei Xu. When match fields do not need to match: Buffered packets hijacking in sdn. In *Proc. of the Network and Distributed System Security Symposium (NDSS'20)*, 2020.

[3] Mingming Chen, Thomas La Porta, Trent Jaeger, and Srikanth Krishnamurthy. Lightweight coordinated sampling for dynamic flows under budget constraints. In *2024 33rd International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9. IEEE, 2024.

[4] Mingming Chen, Thomas La Porta, Teryl Taylor, Frederico Araujo, and Trent Jaeger. Manipulating openflow link discovery packet forwarding for topology poisoning. https://doi.org/10.48550/arXiv.2408.16940, 2024.

[5] Alessandro D'Alconzo, Idilio Drago, Andrea Morichetta, Marco Mellia, and Pedro Casas. A survey on big data for network traffic monitoring and analysis. *IEEE Transactions on Network and Service Management*, 16(3):800–813, 2019.

[6] Linux Foundation. Opendaylight techniqual overview. http://archive15.opendaylight.org/project/technical-overview, 2015. Accessed on 2024-07-09.

[7] Open Networking Foundation. Onos feature. https://opennetworking.org/wp-content/uploads/2019/12/ONOS-Features_v1.pdf, 2019. Accessed on 2024-07-09.

[8] Sungmin Hong, Lei Xu, Haopei Wang, and Guofei Gu. Poisoning network visibility in software-defined networks: New attacks and countermeasures. In *Ndss*, volume 15, pages 8–11, 2015.

[9] Tao Hu, Zhen Zhang, Peng Yi, Dong Liang, Ziyong Li, Quan Ren, Yuxiang Hu, and Julong Lan. Seapp: A secure application management framework based on rest api access control in sdn-enabled cloud environment. *Journal of Parallel and Distributed Computing*, 147:108–123, 2021.

[10] Nick McKeown. P4 runtime – putting the control plane in charge of the forwarding plane. https://opennetworking.org/news-and-events/blog/p4-runtime-putting-the-control-plane-in-charge-of-the-forwarding-plane/, 2017-12-04. Accessed on 2024-07-09.

[11] Benjamin E Ujcich, Samuel Jero, Richard Skowyra, Adam Bates, William H Sanders, and Hamed Okhravi. Causal analysis for {Software-Defined} networking attacks. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3183–3200, 2021.

[12] Lei Xu, Jeff Huang, Sungmin Hong, Jialong Zhang, and Guofei Gu. Attacking the brain: Races in the {SDN} control plane. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 451–468, 2017.

[13] Huancheng Zhou and Guofei Gu. Cerberus: Enabling efficient and effective in-network monitoring on programmable switches. In *2024 IEEE Symposium on*

*Security and Privacy (SP)*, pages 16–16. IEEE Computer Society, 2023.