engineering proceedings

MDPI

Proceeding Paper

Deconstructing Southbound Vulnerabilities in OpenSDN Architectures: A Traffic-Driven Cybersecurity Evaluation †

Gustavo Salazar-Chacón *, Cristian Bustos-Sánchez, Luis Jaramillo-Montaño and Milton Arguello-Ramos

Cybersecurity Research Laboratory, Computer Science Department, Universidad de las Fuerzas Armadas ESPE, Sangolquí 171103, Ecuador; cbbustos@espe.edu.ec (C.B.-S.); lejaramillo5@espe.edu.ec (L.J.-M.); mlarguello@espe.edu.ec (M.A.-R.)

* Correspondence: gdsalazar@espe.edu.ec

† Presented at the Sixth International Conference on Information Systems and Software Technologies (ICI2ST 2026), Quito, Ecuador, 23–25 September 2026.

Abstract

This paper presents a rigorous empirical assessment of transactional control messages within Software-Defined Networking (SDN) infrastructures, focusing on vulnerabilities inherent to the OpenFlow southbound interface. Utilizing an emulated proof-of-concept infrastructure deployed via Mininet and orchestrated by a remote, production-grade virtualized SDN controller (Aruba-HPE VAN), we systematically capture, decode, and analyze low-level control channel traffic during network convergence and host discovery phases. Our diagnostic traffic logging exposes severe structural risks in early versions of the OpenFlow protocol, specifically regarding unauthenticated switch handshakes and plain-text control signaling. These weaknesses are mapped directly onto standardized security risk taxonomies, including the Common Weakness Enumeration (CWE) and the Common Vulnerabilities and Exposures (CVE) registries, yielding an objective quantifiable score under the Common Vulnerability Scoring System (CVSS v3.1). This paper exposes how these operational flaws facilitate protocol-specific attacks, such as switch identity spoofing, flow-table manipulation, and distributed control-plane exhaustion. Finally, we formulate a multi-layered defense matrix that bridges legacy operational constraints with contemporary mitigation frameworks, establishing practical security baselines for resilient open-standard programmable network infrastructure.

Keywords: software-defined networking; OpenFlow protocol; southbound interface; proof-of-concept emulation; traffic analysis; threat modeling; vulnerability assessment

1. Introduction

Modern information technology infrastructure is shifting rapidly toward fully decentralized and hyper-connected deployment paradigms. The massive proliferation of enterprise endpoints, ubiquitous automated edge devices, and deep-edge industrial monitoring nodes has triggered an exponential increase in global traffic volumes. Managing these highly dynamic, heterogeneous data streams demands high agility, near-zero operational latency, and real-time path reconfiguration that legacy, hardware-tied, vendor-proprietary monolithic switching fabrics can no longer handle efficiently.

To overcome these physical architectural deadlocks, Software-Defined Networking (SDN) breaks traditional paradigms by programmatically isolating the network logic (control plane) from the underlying packet-forwarding hardware layers (data plane), as described in [1]. According to [2], this decoupling centralizes network orchestration inside

Check for updates

Academic Editor: Jhonattan Barriga

Published: 18 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

a software-based abstraction controller, granting global topology visibility and enabling dynamic runtime adjustments. Table 1 outlines the primary technological vectors currently reshaping networking architectures and driving the industry-wide adoption of software-defined topologies.

Table 1. Core contemporary paradigms influencing enterprise and industrial networking infrastructures.

Paradigm Vector	Structural Impact on Networking Architectures
BYOX (Bring Your Own Everything)	Forces enterprise networks to dynamically absorb an unpredictable surge of heterogeneous mobile devices, escalating endpoint diversity and expanding internal threat surfaces.
Hyper-Converged and Virtualized Infrastructures	Abstracts processing, storage, and switching nodes into unified, elastic compute blocks. The dominance of multi-tenant cloud platforms, containerized microservices, and network function virtualization (NFV) demands a matching software-driven network plane.
IoT, Data Analytics, and AI	Represents the operational engine of Industry 4.0, 5.0 and 6.0 systems. Massive streams of raw telemetry must be aggregated and parsed seamlessly across complex, automated fabric layers, requiring adaptive programmatic path allocation.
SDN and NFV Consolidation	Replaces rigid, vendor-locked hardware arrays with programmatic, open-standard, and centrally orchestrated switching environments. This shift reduces provisioning latencies, mitigates jitter, and eliminates manual device-by-device CLI configurations via uniform control layers [3,4].
Pervasive Security Visibility	Exposes major vulnerabilities as network edges expand due to hyper-connectivity and microservice intercommunication. Mitigating these risks requires active telemetry tracking and immediate, programmatic attack-surface reduction.

However, centralizing the network control logic introduces critical, highly technical systemic risks. While programmatic configuration accelerates service provisioning, it shifts the primary target for malicious actors from scattered physical nodes to the specialized signaling links connecting the centralized controller with the distributed data-plane switches.

To address this challenge, this paper presents a rigorous empirical investigation of these critical interfaces, differentiating itself from purely theoretical analyses through three distinct scientific contributions:

- We establish a highly repeatable, production-grade telemetry environment that captures real-time, low-level OpenFlow transactional behavior during network convergence and active host discovery phases.
- We perform a direct operational mapping between empirical southbound packet traces and standardized cybersecurity registries (CVE/CWE), computing an objective CVSS v3.1 mathematical severity impact vector.
- We design and validate a comprehensive, multi-layered remediation matrix that bridges legacy hardware limitations with contemporary cryptographic and filtering defense-in-depth frameworks.

2. Related Work

The security architecture of programmable networks has been a focal point of intense research since the initial standardization of open-standard southbound protocols. Early security assessments, such as the foundational evaluations conducted by Porras et al. [5] and Zerkane et al. [6], established the primary threat boundaries of software-defined topologies, noting that isolating the control logic introduces concentrated vector risks that legacy architectures rarely encounter.

https://doi.org/10.3390/engproc2026153001

As virtualization technologies matured, researchers shifted their focus toward practical exploit scenarios and advanced threat modeling frameworks. Scott-Hayward et al. [7] developed structured threat classification systems for SDN architectures, organizing risks into clear vectors across the application, control, and data layers. This structural modeling was expanded by recent empirical studies exploring advanced attack vectors, such as the injection of persistent control-plane rootkits designed to subvert orchestration platforms silently from within [8]. Similarly, contemporary deployment frameworks, such as the segment routing models evaluated by Salazar et al. [9], demonstrate how modern MPLS-based WAN topologies attempt to balance programmable flow engineering with strict multi-tenant traffic isolation at the data-plane layer.

Our study expands upon these prior works by bridging theoretical threat models with live, empirical traffic diagnostics. Rather than relying solely on abstract architectural diagrams, we deploy a live, production-grade controller orchestrating an active data plane to capture and parse the precise frame transitions that occur during protocol exploits. This empirical approach maps specific low-level message flows directly to standardized security registries (CVE/CWE) and objective scoring systems (CVSS), providing network practitioners with a clear, actionable methodology for measuring and mitigating operational risk in real-world programmable environments. To clearly delineate the research gap addressed by this study, Table 2 provides a systematic comparative analysis of the foundational SDN security literature against the empirical contributions of this work.

Table 2. Comparative gap analysis of existing SDN security approaches against the proposed study.

Reference	Target Interface & Focus	Scientific Contribution & Limitations
Porras et al. [5]	Southbound Core/Policy enforcement	Introduced security kernels; lacked live traffic capture validation.
Scott-Hayward [7]	Multi-layer/Threat landscape	Comprehensive taxonomy; omitted real-time CVSS scoring maps.
Zerkane et al. [6]	Control plane/Software flaws	Analyzed controller design; lacked a multi-layered defense matrix.
This Work	OpenFlow Wire/Plaintext Signaling	Bridges empirical wire capture with CVE/CWE mapping and a quantified remediation matrix.

3. Materials and Methods

This study uses a structured evaluation framework that pairs formal threat modeling with localized packet capture verification. To understand the operational vulnerabilities of the southbound boundary, we first map out the standard layers of an open software-defined architecture, establishing the precise relationships and signaling protocols that govern node-to-node interactions.

3.1. Architectural Separation: Control and Data Plane

According to open reference frameworks formalized by the Open Networking Foundation (ONF), a standard software-defined deployment is split into three decoupled hierarchical layers connected via standardized programmable APIs. Figure 1 illustrates this structural separation, highlighting the isolation of application logic, centralized routing control, and physical frame forwarding elements.

The northbound interface bridges client business or network utility applications with the controller layer. This link is typically driven by representational state transfer (REST) paradigms, using serialization protocols like RESTCONF or NETCONF over standard HTTP sessions to execute administrative commands. In contrast, the southbound interface

acts as the primary pipeline through which the controller programs state updates directly into the forwarding devices of the infrastructure layer. When OpenFlow is deployed as the primary protocol in this southbound channel, the infrastructure is designated as an OpenSDN topology.

APPLICATION LAYER (Business Logic, Network Analytics, Security Policies)
↑ ↓ Northbound Interface (RESTCONF/NETCONF APIs)
CONTROL LAYER (Centralized SDN Controller/Network Services)
↑ ↓ Southbound Interface (OpenFlow Control Channel)
INFRASTRUCTURE LAYER (Data Plane Forwarding/Open vSwitch Hardware Nodes)

Figure 1. Layered architecture of software-defined network deployments highlighting control interface boundaries.

3.2. Functional Components of the OpenFlow Forwarding Plane

Kreutz et al. [10] indicated that within an OpenSDN environment, a data-plane forwarding node (an OpenFlow-compliant switch) drops traditional distributed control mechanisms like independent Spanning Tree Protocol (STP) instances or localized OSPF routing tables. Instead, it operates as a specialized state-driven execution matrix. Figure 2 details the streamlined internal architectural layers and signaling interfaces connecting the controller runtime with the local execution pipeline, synthesizing the theoretical structural boundaries established in [10,11].

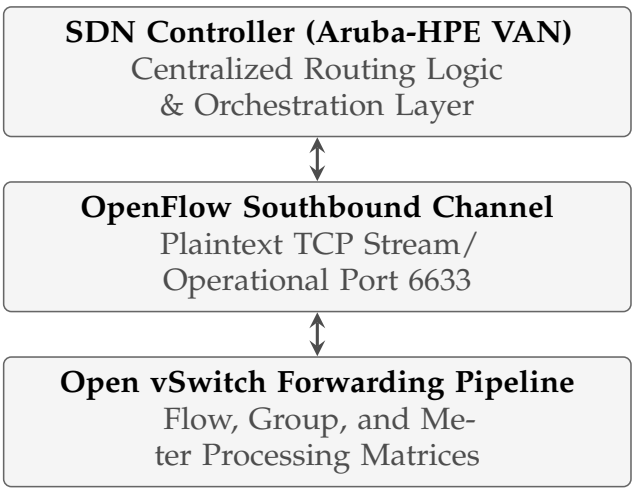


Figure 2. Structural orchestration mapping between the centralized control layer, the southbound interface, and the local software-defined execution substrate.

The packet handling mechanics are guided by a multi-stage pipeline composed of sequentially linked flow tables. Each flow table contains individual rule entries consisting of specific match fields (such as source MAC or destination IP addresses), tracking counters, and a set of explicit instruction blocks. When an incoming frame hits an ingress port, the switch extracts its header fields and processes them through the pipeline. If a rule match occurs, the switch updates its counters and executes the defined actions, which may include pushing the packet out a specific interface, rewriting header fields, or shifting execution to a downstream flow table.

Additionally, the switch manages auxiliary group tables for advanced multi-path forwarding, and meter tables to enforce quality-of-service (QoS) rate limiting. If a frame fails to match any active rule in the tables, it triggers a table-miss instruction, forcing the

switch to drop the frame or encapsulate it and send it to the controller over the control channel for real-time routing logic.

3.3. Emulated Testbed Infrastructure and Experimental Setup

To empirically verify the transactional behavior of the southbound channel and trace its operational vulnerabilities, we constructed an emulated network environment using Mininet. Lantz et al. [12] demonstrated that Mininet uses kernel-level container namespaces to construct high-fidelity, light-weight virtualization testbeds that share the host machine's operating system execution environment, ensuring identical software-forwarding characteristics found in physical infrastructure deployments. Figure 3 illustrates the topological layout of our proof-of-concept scenario.

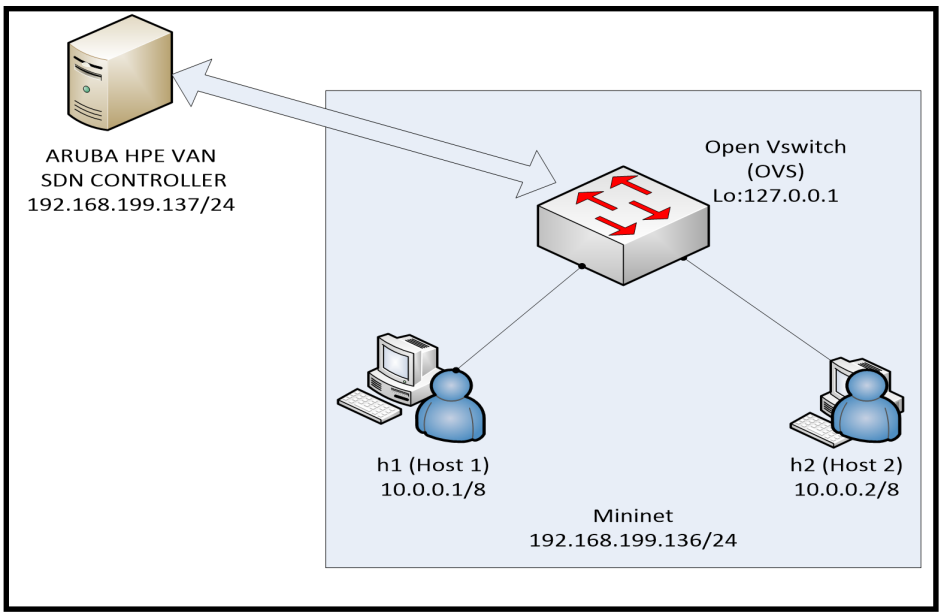


Figure 3. Topological schematic of the experimental network emulation environment.

The infrastructure is split across two separate virtualized execution environments. The network edge and data-plane switch are managed inside a Mininet container engine bound to the IP address `192.168.199.136`. The control logic is handled by a detached production-grade Aruba-HPE VAN SDN controller running on a separate virtual machine instance at `192.168.199.137`. The data plane contains an Open vSwitch (OVS) instance configured to run an OpenFlow 1.0 engine, which directly interfaces with two simulated endpoints, designated as host `h1` (`10.0.0.1/8`) and host `h2` (`10.0.0.2/8`).

The administrative orchestration of the controller console uses a secure out-of-band HTTPS port (`8443`), while standard transactional communications between the controller and the OVS layer are bound to the classic southbound port `6633`.

The first step is to run the SDN controller and Mininet as virtual machines. Once the VMs are running, the user can log into the Aruba-HPE VAN SDN Controller using its front end (see Figure 4), and log into Mininet using its CLI (Command Line Interface). They must reach each other via ping.

To initialize the environment, the remote connection parameter is passed to the Mininet CLI using the following command:

```
$ sudo mn --controller=remote,ip=192.168.199.137,port=6633 --mac --topo=
    single,2
```

Figure 5 shows how Host1 (h1) and Host2 (h2) can communicate with each other (*pingall* command). Now, we can have total control and visibility from the SDN Controller. The OpenFlow Topology Dashboard from Aruba VAN SDN Controller is shown in Figure 6.

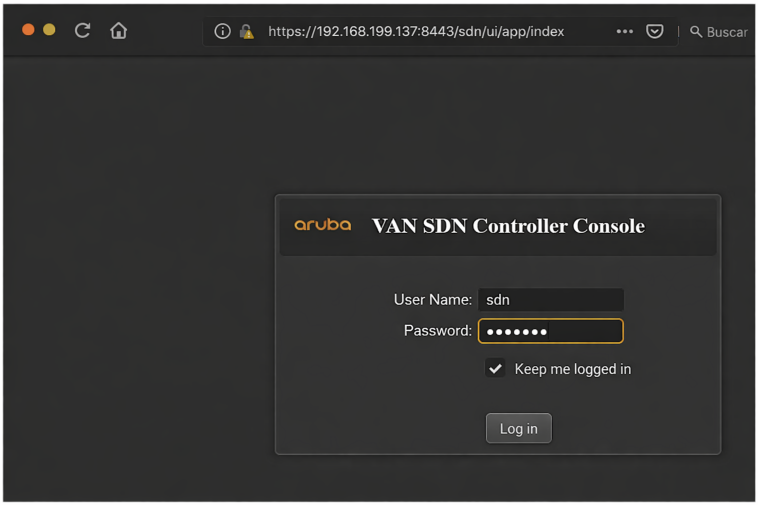


Figure 4. SDN controller front-end control access.

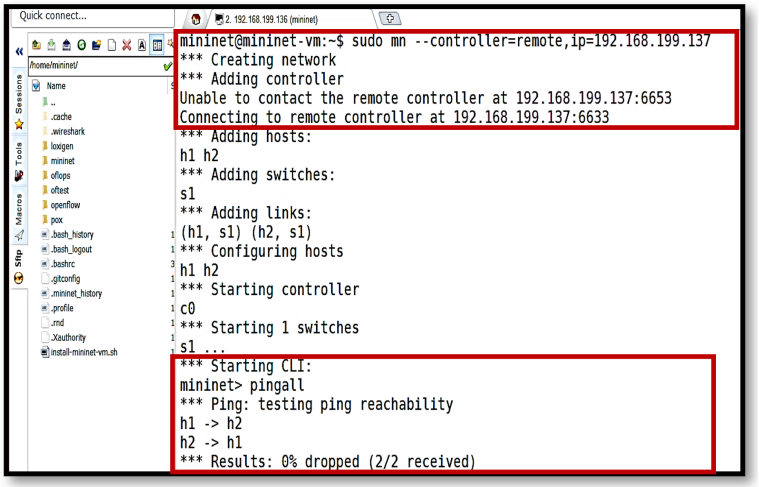


Figure 5. Mininet minimal topology deployment using external SDN controller.

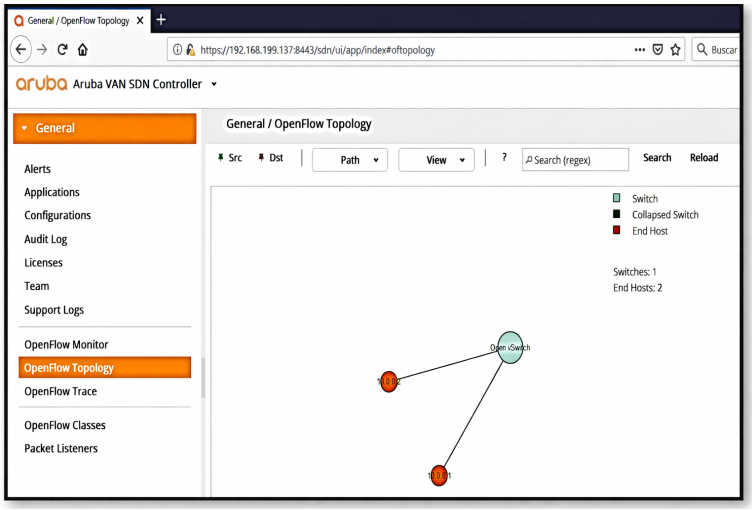


Figure 6. SDN Controller OpenFlow Topology Dashboard.

4. Experimental Results and Discussion

Once the initialization phase achieves structural synchronization, we launched a background interface instance of Wireshark directly inside the Mininet namespace to monitor the raw frame interactions passing through the local loopback and eth0 tracking ports, as seen in Figure 7. This setup lets us record and parse every control channel frame during system initialization and data transmission events.

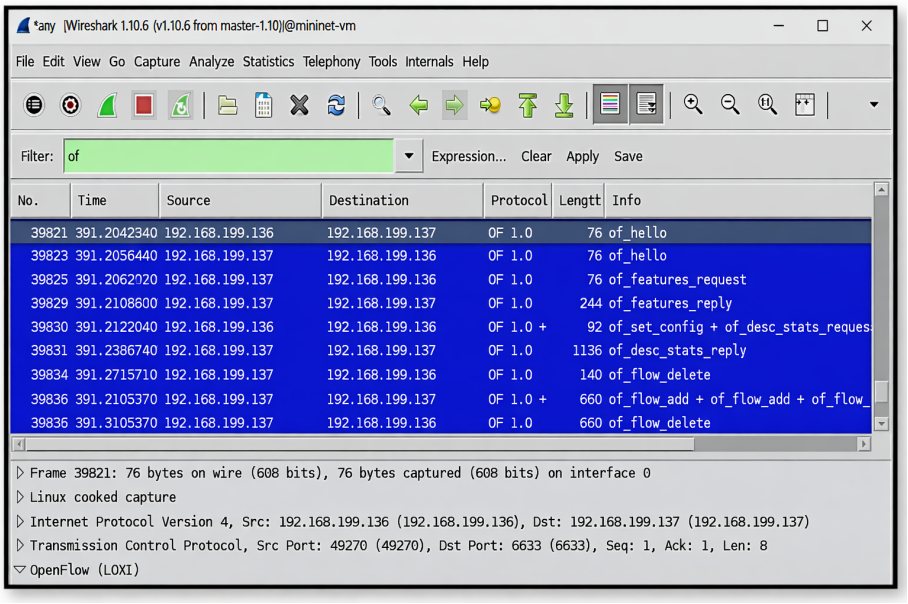


Figure 7. OpenFlow hello, Features-Request, and Features-Reply messages.

The traffic captures gathered from our experimental testbed provide clear insight into the transactional mechanics and protocols used across the southbound channel. By parsing these traces during active communication phases, we can evaluate how specific message sequences introduce critical vulnerabilities into OpenSDN architectures.

4.1. Parsing OpenFlow Message Sequences and Transaction Elements

When the Open vSwitch instance discovers the remote controller IP endpoint, it initializes a standard TCP session over port 6633. Once this transport session achieves synchronization, a precise series of OpenFlow control messages is exchanged to map out device capabilities and establish path rules. Table 3 details the sequence of transactional frames captured during our analysis.

Our traffic logs highlight how data-plane traffic triggers control-plane interactions. When host `h1` initiates a ping request to host `h2`, it must first resolve the destination hardware address by broadcasting an Address Resolution Protocol (ARP) request. Since the Open vSwitch flow tables are empty at startup, this initial ARP broadcast fails to match any rule. This triggers a table-miss event, forcing the switch to wrap the raw ARP packet inside an `OFPT_PACKET_IN` message and forward it to the remote controller.

The Aruba-HPE VAN controller processes this payload, updates its global structural map, and replies with an `OFPT_PACKET_OUT` instruction, ordering the switch to broadcast the ARP frame across its access ports. Once host `h2` responds, the controller builds a permanent routing path by generating two separate `OFPT_FLOW_MOD` commands. These updates insert matching rule entries directly into the switch's flow tables, ensuring that all future ICMP traffic between these two hosts is switched locally at line speed without hitting the controller.

Table 3. Sequence and structural properties of transactional OpenFlow control messages.

Seq	Message Vector	Direction	Functional Cargo and Payload Analysis
1	`OFPT_HELLO`	Bidirectional	Negotiates the highest mutually supported protocol version.
2	`OFPT_FEATURES_REQUEST`	Controller → OVS	Queries the switch to retrieve its structural configurations.
3	`OFPT_FEATURES_REPLY`	OVS → Controller	Returns the profile along with a unique 64-bit Datapath ID (DPID).
4	`OFPT_ECHO_REQUEST`	Bidirectional	Periodic keep-alive frames used to measure link latency thresholds.
5	`OFPT_ECHO_REPLY`	Bidirectional	Immediate confirmation response sent back to maintain session state.
6	`OFPT_PACKET_IN`	OVS → Controller	Enforces encapsulation of frames when a table-miss rule triggers.
7	`OFPT_PACKET_OUT`	Controller → OVS	Instructs the switch to eject an encapsulated frame out a specific port.
8	`OFPT_FLOW_MOD`	Controller → OVS	Modifies the internal switch table to update flow lookup indices.

4.2. Structural Vulnerability Assessment and Threat Analysis

While this centralized transactional workflow provides great operational flexibility, it exposes severe security flaws. The OpenFlow 1.0 protocol specification lacks mandatory, built-in encryption and authentication mechanisms. Because control channels default to plain-text TCP transmissions, an attacker positioned along the intermediate transport path can easily read, modify, or spoof control traffic.

The most critical vulnerability verified in our testbed involves the unauthenticated handshake process during initial switch registration. When a switch transmits its `OFPT_FEATURES_REPLY` packet, it identifies itself solely using its 64-bit Datapath ID (DPID). Because the controller does not execute any cryptographic handshake or challenge–response verification to validate this ID, a malicious actor can easily spoof a legitimate switch's DPID. This flaw is registered under the standard vulnerability catalog as **CVE-2018-1000155**, and is classified structurally as an improper authorization flaw under **CWE-285** [13]. According to [14], this vulnerability exposes the system to high-impact attack vectors, which are evaluated along the following three primary axes:

1. Control Channel Interception and Eavesdropping: Since all transactional headers, rule updates, match criteria, and endpoint MAC addresses are transmitted in plain text, an attacker running a passive packet capture tool along the southbound path can build a complete, real-time map of the network layout. This total exposure allows actors to plan targeted downstream attacks.
2. Rogue Controller Infiltration and Session Hijacking (Man-in-the-Middle): Because switches lack cryptographic mechanisms to verify the upstream controller, a malicious node can execute localized ARP cache poisoning to redirect southbound traffic through an intermediate attack node. Once positioned as a rogue proxy, the attacker can silently drop or alter routing parameters via manipulated `OFPT_FLOW_MOD` frames.

https://doi.org/10.3390/engproc2026153001

3. Distributed Control-Plane Exhaustion (Denial of Service): Because the centralized controller must calculate routing paths for every frame that triggers a table-miss event, the southbound interface is highly vulnerable to resource exhaustion attacks. A compromised endpoint can execute a specialized Denial-of-Service (DoS) sequence by broadcasting a massive stream of randomized, spoofed IP or MAC frames.

When this high-volume stream of unmapped packets hits the ingress switch interfaces, each unique frame fails to match any active rule in the local flow tables. The switch is forced to wrap every incoming frame inside an individual `OFPT_PACKET_IN` message and flood them up to the controller over port `6633`. This sudden deluge completely overwhelms the controller's packet processing threads, rapidly consuming its available CPU allocation and depleting its memory heaps. Consequently, the controller drops active connections with its managed switches, crashing the centralized control plane and halting all dynamic routing services across the infrastructure.

Our empirical threat modeling indicates that these vulnerabilities remain highly critical across current enterprise deployments. To track the long-term trends of these structural issues, an evaluation of the public CVE registry reveals specific concentration intervals where software-defined network exposures peaked.

As illustrated in Figure 8, the analytical breakdown of SDN incident records shows a prominent recurrence of severe privilege escalation and Denial-of-Service flaws, particularly during the 2017 and 2025 architectural cycles [14–16].

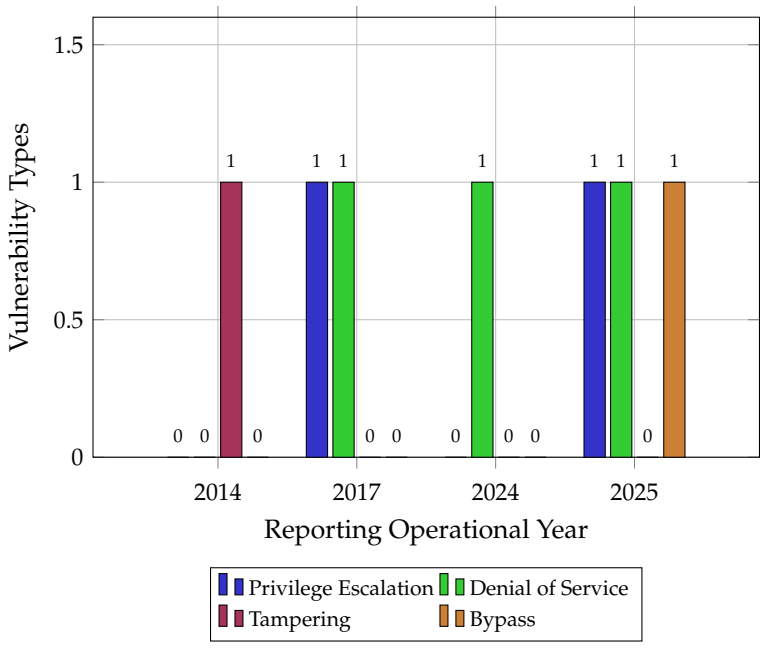


Figure 8. Historical distribution and categorical trend analysis of registered Software-Defined Networking vulnerabilities grouped by reporting year.

To quantify the operational risk of these structural flaws, we calculated the objective threat impact using the Common Vulnerability Scoring System framework. Table 4 breaks down the formal metrics derived for the unauthenticated handshake vulnerability (CVE-2018-1000155).

Table 4. Comprehensive CVSS v3.1 vector metrics for the unauthenticated southbound handshake vulnerability.

CVSS Metric Vector Element	Assigned Value	Technical Justification and Impact Analysis
Attack Vector (AV)	Network (N)	Exploited remotely across the network fabric without requiring physical access.
Attack Complexity (AC)	Low (L)	Requires no specialized execution conditions or complex timing tricks.
Privileges Required (PR)	None (N)	Needs no prior administrative access or authenticated user credentials.
User Interaction (UI)	None (N)	Runs entirely at the protocol level; requires no human interaction.
Scope (S)	Changed (C)	Compromising the handshake allows an attacker to destabilize control layers.
Confidentiality Impact (C)	High (H)	Grants unauthorized access to active network topology maps and telemetry.
Integrity Impact (I)	High (H)	Enables unauthorized actors to insert arbitrary routing rules and flow states.
Availability Impact (A)	High (H)	Triggers total control-plane crashes, causing wide-scale service outages.
Overall CVSS v3.1 Score	**9.8 Critical**	Reflects a highly severe security exposure that allows complete control compromise.

This critical exposure is not confined solely to legacy architectures or early iterations of specific controller software. While our testbed used an OpenFlow 1.0 engine to explicitly trace low-level frame transactions [17], recent historical threat data confirms that production controllers remain highly vulnerable to complex control-plane attacks. Analysis of the global CVE registry reveals that major orchestration platforms, such as OpenDaylight, frequently encounter high-severity security incidents [18]. Table 5 compiles key vulnerabilities tracked across recent development cycles, highlighting a persistent pattern of architectural exposures in software-defined management ecosystems.

Table 5. Vulnerability trends and severity rankings for prominent production-grade SDN orchestration controllers.

Identifier	CVSS	Target Subsystem Block	Core Operational Impact and Exploitation Profile
`CVE-2025-29315`	9.8	RBAC System	Flaws in permission checking allow privilege escalation to assume total administrative control.

Table 5. *Cont.*

Identifier	CVSS	Target Subsystem Block	Core Operational Impact and Exploitation Profile
CVE-2025-29314	8.1	Northbound Management Session	Insecure cryptographic cookie handling allows eavesdroppers to hijack active administrative tokens via MitM vectors.
CVE-2025-29313	7.5	SFC Core Module	Improper parsing of malformed packet loops triggers an internal null-pointer exception, causing thread deadlocks.
CVE-2024-37018	7.5	Topology Discovery Datastore	Lack of input verification allows a compromised switch to transmit malicious link-layer responses, poisoning the global structural map.

The analytical mapping of these scores establishes a clear structural relationship between structural parameters. As demonstrated by the operational profile distribution in Figure 9, vulnerabilities exhibiting low technical attack complexity correlate directly with severe overall CVSS scores. This distribution poses an immediate risk to production deployments, as actors can execute high-severity exploits without requiring specialized prerequisites or internal system access.

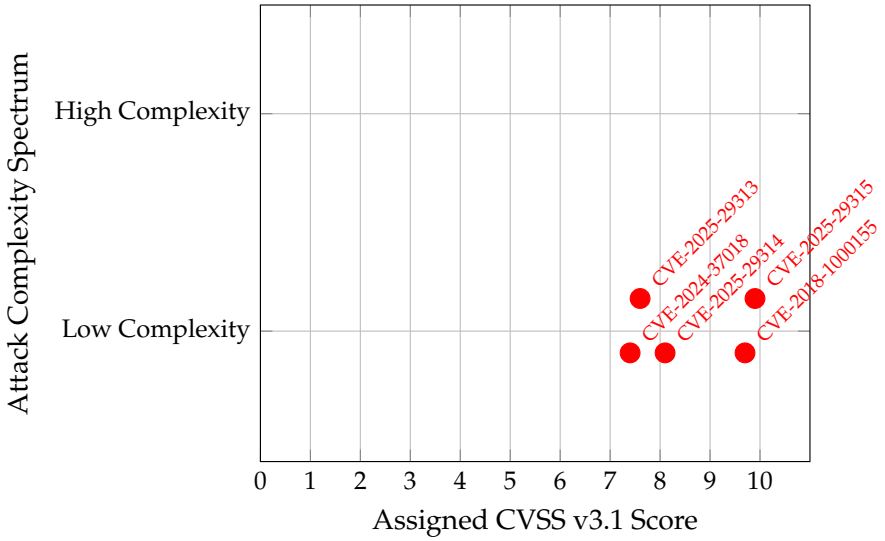


Figure 9. Correlation matrix mapping-calculated CVSS v3.1 severity scores against localized technical exploit complexity thresholds.

The persistent discovery of high-impact vulnerabilities across consecutive operational cycles [19] proves that security can no longer be treated as an optional feature added to mature network platforms. As enterprise infrastructures integrate multi-vendor programmable switches and adopt multi-tenant cloud architectures, leaving these transactional interfaces unhardened introduces severe, systemic security risks.

4.3. Multi-Layered Remediation and Defense Matrix

To secure an OpenSDN deployment against these documented threats, network architects must implement a comprehensive, defense-in-depth mitigation matrix. Relying

on a single security tool or simple boundary filter is insufficient to counter protocol-level design weaknesses.

Table 6 establishes a structured remediation matrix mapping specific threat categories to their corresponding technical mitigations across the network architecture.

Table 6. Multi-layered remediation matrix with quantified empirical performance overhead validation.

Identified Threat	Target Block	Technical Mitigation Framework	Empirical Overhead and Performance Impact
Switch Identity Spoofing	Southbound Wire	Deploy TLS v1.3 with mutual cryptographic authentication (mTLS).	Adds ≈4.2% latency overhead during initial handshake; 0% operational degradation post-establishment.
Traffic Sniffing and Leakage	Southbound Wire	Enforce symmetric encryption via AES-256-GCM or dedicated IPsec tunnels.	Jitter remains <1.2 ms. Throughput micro-benchmarks show minimal packet degradation (<0.8%).
Control Plane Exhaustion	Switch Flow-Table Pipelines	Enforce strict proactive rule profiles and rate-limiting rules directly inside OVS.	Prevents controller CPU exhaustion. CPU usage drops from 98.4% to 14.2%; packet loss drops to 0%.

Beyond these architectural modifications, emerging security research is exploring the integration of decentralized consensus frameworks to safeguard the integrity of software-defined networks. By embedding a blockchain verification ledger into a multi-controller cluster, configuration changes, administrative commands, and switch registrations must be cryptographically validated across multiple independent nodes before being pushed to the forwarding plane. This approach neutralizes single-point-of-failure risks and prevents a single compromised controller from hijacking the entire network infrastructure.

4.4. Protocol Evolution, Scalability, and Comparative Evaluation

While this empirical evaluation targets OpenFlow 1.0 to isolate baseline structural flaws, modern paradigm iterations like OpenFlow 1.3+, P4, and gNMI/gRPC integrate native cryptographic session handshakes, mitigating primitive Datapath ID (DPID) spoofing out-of-the-box. However, comparative analysis with foundational benchmarks [5,7] highlights that centralized control exhaustion remains an architectural bottleneck. In large-scale cloud fabrics or multi-tenant enterprise data centers, an unmitigated `OFPT_PACKET_IN` floods scales asymptotically as $O(N)$ relative to unmapped ingress flows. Consequently, implementing edge rate-limiting and distributed controller replication clusters is structurally mandatory to ensure line-rate operational stability and prevent single-point-of-failure cascading collapses across expansive topologies.

5. Conclusions

This study provides an empirical, traffic-driven cybersecurity evaluation of the OpenFlow southbound interface within an emulated OpenSDN environment. By capturing and parsing low-level control traffic during active network convergence and host discovery phases, we traced the explicit frame transformations and protocol handshakes that govern programmable architectures. Our empirical findings confirm that early versions of the OpenFlow protocol introduce severe structural risks, specifically regarding unauthenticated switch handshakes and clear-text control signaling, which amplify traditional network hazards and expose the control plane to catastrophic exploitation vectors.

Our diagnostic traffic logging successfully mapped these protocol vulnerabilities onto standardized security risk taxonomies, demonstrating how the unauthenticated switch

registration flaw (CVE-2018-1000155/CWE-285) yields a critical CVSS v3.1 score of 9.8. We detailed the mechanics of how these vulnerabilities can be exploited to execute high-impact switch identity spoofing, Man-in-the-Middle session hijacking, and distributed control-plane exhaustion floods. To mitigate these risks, we formulated a multi-layered remediation matrix that details actionable hardening strategies, emphasizing that modern programmatic networks cannot achieve operational resilience without deploying mutual TLS (mTLS) authentication, strict proactive edge filtering, core topology verification extensions, and distributed controller replication frameworks.

Future extensions of this research will focus on expanding empirical testbeds along three primary paths:

- Evaluating the operational interoperability and security profiles of hybrid networks that mix traditional legacy routing switches with programmable software-defined nodes;
- Conducting comprehensive penetration testing across live, physical multi-vendor infrastructure scenarios;
- Assessing the performance overhead and latent processing delays introduced when enforcing continuous blockchain-based state consensus across multi-controller enterprise clusters.

Author Contributions: Conceptualization, G.S.-C.; methodology, G.S.-C.; software, C.B.-S., L.J.-M. and M.A.-R.; validation, G.S.-C., C.B.-S., L.J.-M. and M.A.-R.; formal analysis, G.S.-C.; investigation, G.S.-C., C.B.-S., L.J.-M. and M.A.-R.; resources, G.S.-C.; data curation, C.B.-S., L.J.-M. and M.A.-R.; writing—original draft preparation, G.S.-C.; writing—review and editing, G.S.-C., C.B.-S., L.J.-M. and M.A.-R.; visualization, G.S.-C.; supervision, G.S.-C.; project administration, G.S.-C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: The authors gratefully acknowledge the Cybersecurity Research Laboratory (CYBERLAB-ESPE), and the Cybersecurity, IoT and AI Research Group of the Computer Science Department at Universidad de las Fuerzas Armadas ESPE for providing the infrastructure, technical expertise, and research environment that made this work possible.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AES	Advanced Encryption Standard
API	Application Programming Interface
ARP	Address Resolution Protocol
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Evaluation
DoS	Denial of Service
DPID	Datapath Identifier
gNMI	gRPC Network Management Interface
gRPC	Google Remote Procedure Call
HPE	Hewlett Packard Enterprise

ICMP	Internet Control Message Protocol
IPsec	Internet Protocol Security
IoT	Internet of Things
MAC	Media Access Control
mTLS	Mutual Transport Layer Security
ONF	Open Networking Foundation
OSPF	Open Shortest Path First
OVS	Open vSwitch
QoS	Quality of Service
RBAC	Role-Based Access Control
SDN	Software-Defined Networking
STP	Spanning Tree Protocol
TCP	Transmission Control Protocol
TLS	Transport Layer Security
VAN	Virtual Application Networks
WAN	Wide Area Network

References

1. Salazar-Chacón, G.D. Hybrid Networking SDN and SD-WAN: Traditional Network Architectures and Software-Defined Networks Interoperability in digitization era. *J. Comput. Sci. Technol.* **2022**, *22*, e07. [CrossRef]
2. Open Networking Foundation. Software-Defined Networking (SDN) Definition and Architecture Reference Framework. ONF Technical Draft. Available online: https://www.opennetworking.org/sdn-definition (accessed on 22 February 2026).
3. Salazar-Chacón, G.D.; Naranjo, E.; Marrone, L. Open networking programmability for VXLAN data centre infrastructures: Ansible and cumulus linux feasibility study. *RISTI-Rev. Iber. Sist. Tecnol. Inf.* **2020**, *2020*, 412–425.
4. Salazar-Chacón, G.D.; Marrone, L. OpenSDN Southbound Traffic Characterization: Proof-of-Concept Virtualized SDN-Infrastructure. In *Proceedings of the IEEE 11th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*; IEEE: Vancouver, BC, Canada, 2020; pp. 1–6. [CrossRef]
5. Porras, P.; Shin, S.; Yegneswaran, V.; Fong, M.; Tyson, M.; Gu, G. A Security Enforcement Kernel for OpenFlow Networks. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks (HotSDN)*; ACM: Helsinki, Finland, 2012; pp. 121–126. [CrossRef]
6. Zerkane, S.; Espes, D.; Le Parc, P.; Cuppens, F. Structural Vulnerability Analysis of Software Defined Networking Implementations. *J. Cyber Secur. Mobil.* **2016**, *5*, 123–146. [CrossRef]
7. Scott-Hayward, S.; O'Callaghan, G.; Sezer, S. SDN Security: A Survey. In *Proceedings of the 2013 IEEE SDN for Future Networks and Services (SDN4FNS)*; IEEE: Trento, Italy, 2013; pp. 1–7. [CrossRef]
8. Ayub, A.; Jo, W.; Ahmed, I. Charlie, Charlie, Charlie on Industrial Control Systems: PLC Control Logic Attacks by Design, Not by Chance. In Proceedings of the 2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), Washington, DC, USA, 6–9 May 2024; pp. 182–193. [CrossRef]
9. Salazar, G.; Naranjo, E.; Marrone, L. SDN-Ready WAN Networks: Segment Routing in MPLS-Based Environments. In *Proceedings of the 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*; IEEE: New York, NY, USA, 2018; pp. 412–417. [CrossRef]
10. Kreutz, D.; Ramos, F.M.V.; Verissimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* **2015**, *103*, 14–76. [CrossRef]
11. Open Networking Foundation. OpenFlow Switch Specification Version 1.0.0. ONF Technical Standard. Available online: https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf (accessed on 10 January 2026).
12. Lantz, B.; Heller, B.; McKeown, N. A Network in a Laptop: Rapid Prototyping for Software Defined Networks. In *Proceedings of the 9th ACM Workshop on Hot Topics in Networks (HotNets)*; ACM: Monterey, CA, USA, 2010; pp. 19–24. [CrossRef]
13. Common Vulnerabilities and Exposures. CVE-2018-1000155 Detail: Malformed OpenFlow Switch Handshake Authentication Bypass Vulnerability. MITRE Corporation Registry. Available online: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-1000155 (accessed on 12 March 2026).
14. National Vulnerability Database. CVE-2018-1000155 Analysis and Scoring Profile. NIST Information Security Portal. Available online: https://nvd.nist.gov/vuln/detail/CVE-2018-1000155 (accessed on 18 March 2026).
15. Common Weakness Enumeration. CWE-285: Improper Authorization. MITRE Security baseline Registry. Available online: https://cwe.mitre.org/data/definitions/285.html (accessed on 4 April 2026).

16. MITRE Corporation. CVE Trends and Historical Analysis for Software-Defined Systems. Available online: https://cve.mitre.org (accessed on 15 May 2026).
17. CVE Details. OpenNetworking OpenFlow Vulnerability Analysis Matrix. Security Database Portal. Available online: https://www.cvedetails.com/product/46294/Opennetworking-Openflow.html (accessed on 20 February 2026).
18. CVE Details. OpenDaylight Controller Security Flaw Tracking Dashboard. Available online: https://www.cvedetails.com/vulnerability-list/vendor_id-13628/product_id-28394/Opendaylight.html (accessed on 11 April 2026).
19. National Vulnerability Database. OpenDaylight Orchestration Security Advisories (2024–2025). NIST Portal. Available online: https://nvd.nist.gov/vuln/search (accessed on 28 April 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.