A Hybrid Control Framework for Security Management Automation in Software-Defined Networks

Jiwon Suh*, Juwon Hong*, and Jaehoon (Paul) Jeong*
*Department of Computer Science and Engineering, Sungkyunkwan University, Suwon, Republic of Korea
Email: {sjw6136, hongju2024, pauljeong}@skku.edu

***Abstract*—This paper proposes a novel hybrid security management framework, which combines OpenFlow and NETCONF/YANG technologies, in a Software-Defined Networking (SDN) environment. The increasing complexity of SDN infrastructure demands a sophisticated security solution that can adapt to emerging threats in real time. Our proposed framework integrates OpenFlow's granular traffic control capabilities with NETCONF/YANG's robust policy management features. This hybrid approach creates a comprehensive security solution in the SDN environment. The system enables automated threat detection and rapid response through a dual-technology approach: OpenFlow handles real-time traffic monitoring and control, while NETCONF/YANG manages security policies and network configurations. Testing demonstrates that this hybrid approach significantly shortens threat mitigation time and accuracy compared to the legal approach based on OpenFlow, while maintaining scalability across diverse SDN deployments. Through experiments, it is shown that our hybrid approach outperforms the the legacy approach for security services in the SDN environment.**

***Index Terms*—Software-Defined Networking, Security Management Framework, Hybrid Control, OpenFlow, NETCONF, YANG.**

I. Introduction

The rapid evolution of network infrastructure and increasing cybersecurity threats have made network security management more complex and challenging. Software-Defined Networking (SDN) [1] has emerged as a promising solution for modern network management by separating the control plane from the data plane, offering centralized control and programmability. However, current SDN security implementations often rely on single-technology approaches that may not fully address the complex security challenges of today's networks, particularly in terms of automated threat response and policy management.

In SDN environments, two prominent protocols - OpenFlow and NETCONF/YANG - serve different but complementary purposes. OpenFlow excels in real-time traffic control and forwarding decisions [2], [3], while NETCONF/YANG provides robust configuration management and policy enforcement capabilities with network forwarding elements (e.g., switch and router) [4], [5]. As demonstrated by Hyun et al. [6], SDN-based security functions can effectively mitigate threats like DDoS attacks, but their approach primarily focused on using single protocol solutions. While these protocols have been used separately for different networking tasks, their potential for creating an integrated security management framework remains largely unexplored.

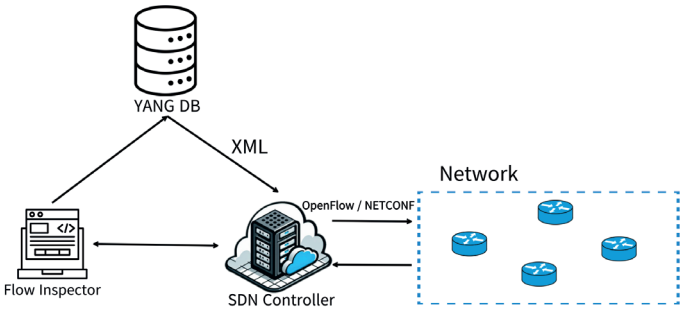

Fig. 1. A Framework of OpenFlow and NETCONF/YANG Hybrid System.

II. Related Work

Previous research in SDN security management has established distinct advantages of different protocol approaches. Kunz and Muthukumar [7] showed NETCONF's efficiency for large-scale configurations and OpenFlow's strengths in flow processing. Hyun et al. [6] demonstrated effective Distributed Denial-of-Service(DDoS) attack mitigation using SDN and Network Functions Virtualization(NFV) with NETCONF/YANG. However, the potential of combining these complementary capabilities for automated security management remains largely unexplored, creating an opportunity for novel hybrid approaches that could leverage the advantages of both protocols.

III. Design

In the proposed framework, as shown in Fig. 1, the SDN controller gathers packet-related data and flow table information from the SDN switches.

This data is then passed to the Flow Inspector, which is responsible for analyzing the traffic and detecting security attacks. Upon detecting a security attack, the Flow Inspector immediately executes a primary defense mechanism through the OpenFlow protocol, issuing drop commands to mitigate malicious traffic. OpenFlow is particularly well-suited for individual flow processing, enabling quick and precise actions

against malicious traffic. Additionally, the Flow Inspector generates an appropriate XML file based on the YANG database, which is sent back to the SDN controller. The SDN controller then applies the XML-based configuration to SDN switches under its control via the NETCONF protocol, which is efficient for performing large-scale and sophisticate configuration changes. By leveraging both protocols, the framework achieves a balance between real-time response and flexible, comprehensive network management.

IV. Emulation Results

This section describes the emulation environment and experiment results of our proposed framework.

TABLE I
EMULATION CONFIGURATION

Parameter	Description
Operating System	Ubuntu 20.04
SDN Controller	RYU Controller v4.34.
Emulation Environment	Mininet v2.2.2.
Topology	Single controller with 6 switches and 8 hosts.
Protocols	OpenFlow 1.3 and NETCONF/YANG.
Network Monitoring Information	Packet-related data and switches' flow table information.
Traffic Generation	ICMP, TCP, UDP, and HTTP traffic with randomized intervals.
Network Attack Emulation	DoS, brute force attacks, and network port scans.

A. Emulation Setup

We conducted an emulation of an SDN network using the Mininet framework [8], with the RYU controller [9] managing the network operations. Mininet provided a realistic emulation of the SDN environment, allowing us to evaluate network behaviors under different threat scenarios. The RYU controller was selected due to its flexibility and support for OpenFlow, making it appropriate for testing our proposed security framework. Table I summarizes the emulation setup and configuration.

B. Results

Experiment results are provided here to test and verify the feasibility of the proposed framework. In the emulation environment, normal traffic comprising ICMP, TCP, UDP, and HTTP flows with randomized intervals was generated to emulate real-world network conditions. Based on these conditions, we generated DoS, brute-force, and network port scan attacks to evaluate the system response against those attacks.

Fig. 2 shows the detection and mitigation of a Denial-of-Service(DoS) attack. To test our system, we generated SYN flood DoS attack from host h1 through host h5. The system detects abnormal traffic by analyzing packet-related data and flow table entries from the SDN switches. Upon detecting the attack, the SDN controller immediately issues drop commands for the attacker's IP address to all the switches via the OpenFlow protocol. Simultaneously, it generates an

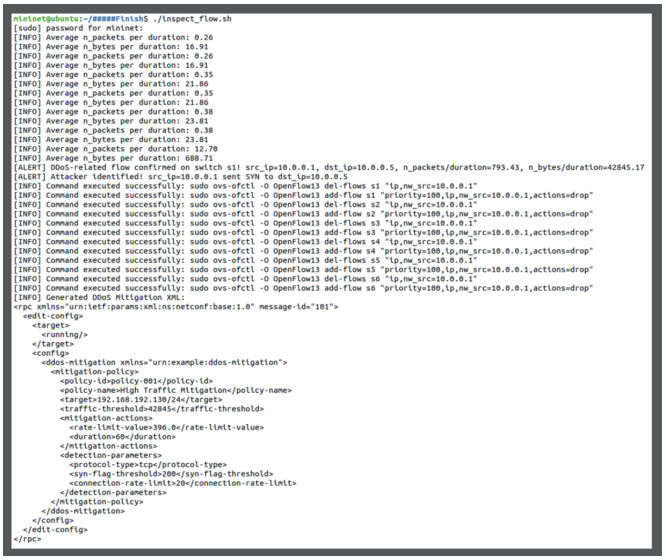

Fig. 2. DoS Attack Scenario.

XML file tailored to the type of attack and the current network environment, based on a predefined YANG data model for such a DoS attack stored in the YANG database, and delivers it to the switches. In this scenario, the SDN controller selects the "High Traffic Mitigation" policy and sets mitigation actions, including traffic thresholds and rate-limit values, according to the current network conditions. Since the detected attack is a SYN flood DoS attack, detection parameters are configured in the switches to block similar attacks automatically in the future.

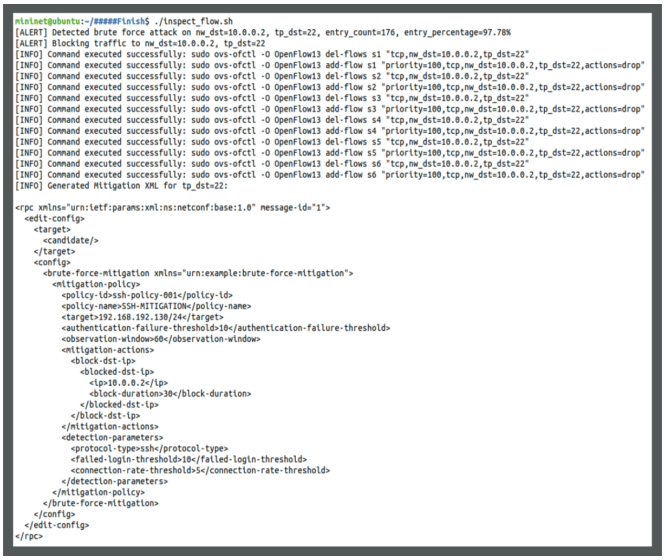

Fig. 3. Brute Force Attack Scenario.

Fig. 3 shows the detection and mitigation of a brute-force attack over the SSH protocol. Similar to the DoS attack scenario, the SDN controller sends drop commands to all SDN switches for the victim's IP address and port number. It also generates an XML file to enforce sophisticated mitigation rules. Since the attack is an SSH brute-force attack, the SDN

controller selects the "SSH-MITIGATION" policy, configures mitigation actions , and adjusts detection parameters based on the current network conditions.

```
mininet@ubuntu:~/#####Finish$ ./inspect_flow.sh
[ALERT] Detected port scan attack from src_ip=10.0.0.5, flow_count=320, total_flows=330
[ALERT] Blocking traffic from src_ip=10.0.0.5
[INFO] Command executed successfully: sudo ovs-ofctl -O OpenFlow13 del-flows s1 "ip,nw_src=10.0.0.5"
[INFO] Command executed successfully: sudo ovs-ofctl -O OpenFlow13 add-flow s1 "priority=100,ip,nw_src=10.0.0.5,actions=drop"
[INFO] Command executed successfully: sudo ovs-ofctl -O OpenFlow13 del-flows s2 "ip,nw_src=10.0.0.5"
[INFO] Command executed successfully: sudo ovs-ofctl -O OpenFlow13 add-flow s2 "priority=100,ip,nw_src=10.0.0.5,actions=drop"
[INFO] Command executed successfully: sudo ovs-ofctl -O OpenFlow13 del-flows s3 "ip,nw_src=10.0.0.5"
[INFO] Command executed successfully: sudo ovs-ofctl -O OpenFlow13 add-flow s3 "priority=100,ip,nw_src=10.0.0.5,actions=drop"
[INFO] Command executed successfully: sudo ovs-ofctl -O OpenFlow13 del-flows s4 "ip,nw_src=10.0.0.5"
[INFO] Command executed successfully: sudo ovs-ofctl -O OpenFlow13 add-flow s4 "priority=100,ip,nw_src=10.0.0.5,actions=drop"
[INFO] Command executed successfully: sudo ovs-ofctl -O OpenFlow13 del-flows s5 "ip,nw_src=10.0.0.5"
[INFO] Command executed successfully: sudo ovs-ofctl -O OpenFlow13 add-flow s5 "priority=100,ip,nw_src=10.0.0.5,actions=drop"
[INFO] Command executed successfully: sudo ovs-ofctl -O OpenFlow13 del-flows s6 "ip,nw_src=10.0.0.5"
[INFO] Command executed successfully: sudo ovs-ofctl -O OpenFlow13 add-flow s6 "priority=100,ip,nw_src=10.0.0.5,actions=drop"
[INFO] Generated Mitigation XML for src_ip=10.0.0.5:
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <port-scan-mitigation xmlns="urn:example:port-scan-mitigation">
        <mitigation-policy>
          <policy-id>tcp-scan-policy-001</policy-id>
          <policy-name>TCP-SCAN-MITIGATION</policy-name>
          <target>192.168.192.130/24</target>
          <scan-detection-threshold>30</scan-detection-threshold>
          <observation-window>60</observation-window>
          <mitigation-actions>
            <block-src-ip>
              <blocked-src-ip>
                <ip>10.0.0.5</ip>
                <block-duration>300</block-duration>
              </blocked-src-ip>
            </block-src-ip>
          </mitigation-actions>
          <detection-parameters>
            <scan-type>tcp-syn-scan</scan-type>
            <short-duration-threshold>5</short-duration-threshold>
            <connection-rate-threshold>100</connection-rate-threshold>
            <alert-threshold>50</alert-threshold>
            <geo-location-block>false</geo-location-block>
          </detection-parameters>
        </mitigation-policy>
      </port-scan-mitigation>
    </config>
  </edit-config>
</rpc>
```

Fig. 4. Port Scan Attack Scenario.

Fig. 4 shows the detection and mitigation of a port scan attack. In this scenario, a TCP SYN port scan targeting all ports of host h2 was launched from host h5. After detecting the attack, the SDN controller issues drop commands for all packets from the attacker to the switches and generates an XML file to enforce mitigation measures. The XML file specifies actions such as blocking the attacker's IP address and setting a block duration. Since the attack is a TCP SYN scan, the controller configures detection parameters, including a short block duration, connection rate thresholds, and alert thresholds, based on the current network environment.

V. Performance Evaluation

In this section, we evaluate the performance of the proposed Hybrid Framework compared to the OpenFlow and NETCONF/YANG protocols. The evaluation metric is average network normalization time, defined as the time required for the network to return to a fully operational state after three types of attacks that we conducted before are detected by the controller. The experiments were conducted with varying the number of switches, ranging from 10 to 50, to emulate realistic network environments.

Fig. 5 shows that while OpenFlow is the fastest solution, the Hybrid Framework makes a strong balance between performance and configurability, achieving efficiency close to OpenFlow's efficiency. This allows networks using the Hybrid Framework to benefit from detailed configuration capabilities akin to NETCONF/YANG with good performance against security attacks. In contrast, NETCONF/YANG offers granular control but incurs a significantly higher convergence time, especially as the number of switches increases.

The source code and the demonstration video clip for the implementation of our Hybrid Control Framework are

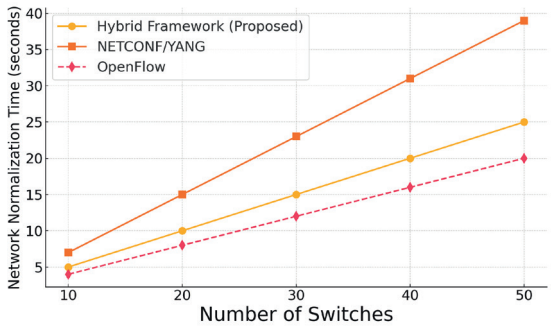

Fig. 5. Network Normalization Time according to the Number of Switches.

available at GitHub of https://github.com/jaehoonpauljeong/Data-Modeling-Group-1-Project and YouTube of https://youtu.be/b8Nyvd7TcRU?si=2uZv5e0GpXPY85Be, respectively.

VI. Conclusion

This paper proposes a Hybrid Framework that integrates OpenFlow and NETCONF/YANG protocols, demonstrating effective detection and mitigation of various security attacks in SDN environments, including DoS, SSH brute force, and port scan attacks, by leveraging the strengths of both protocols. Through experiments, we showed that the Hybrid Framework achieves a strong balance between the performance of OpenFlow and the configurability of NETCONF/YANG, with a good trade-off in terms of network normalization time.

Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Ministry of Science and ICT (MSIT), South Korea (No. RS-2024-00398199 and RS-2022-II221015). Note that Jaehoon (Paul) Jeong is the corresponding author.

References

[1] M. Boucadair and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment," *RFC 7149*, March 2014. [Online]. Available: https://www.rfceditor.org/rfc/rfc7149

[2] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2014.

[3] Open Networking Foundation, "Openflow switch specification (version 1.3.0)," ONF, Tech. Rep., Oct 2013.

[4] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," *RFC 6241*, January 2011. [Online]. Available: https://datatracker.ietf.org/doc/rfc6241/

[5] B. Martin, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," *RFC 6020*, october 2010. [Online]. Available: https://www.rfc-editor.org/info/rfc6020

[6] D. Hyun, J. Kim, D. Hong, and J. P. Jeong, "SDN-based Network Security Functions for Effective DDoS Attack Mitigation," *International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 834–839, 2017.

[7] T. Kunz and K. Muthukumar, "Comparing OpenFlow and NETCONF when Interconnecting Data Centers," *31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 391–396, 2017.

[8] Mininet, "An instant virtual network on your laptop," 2022, [Online]. Available: http://mininet.org/.

[9] RYU project team, "Ryu sdn framework," 2024, [Online]. Available: https://ryu-sdn.org/.