Automating SDN-ACLs with User Groups and Authentication Events

Florian Grießer*, Atsushi Shinoda†, Hirokazu Hasegawa‡, Hajime Shimada†
* *School of Computation, Information and Technology, Technical University Munich, Munich, Germany*
†*Graduate School of Informatics, Nagoya University, Nagoya, Japan*
‡*Center for Strategic Cyber Resilience R&D, National Institute of Informatics, Tokyo, Japan*
§*Information Technology Center, Nagoya University, Nagoya, Japan*
florian.griesser@tum.de, shinoda@net.itc.nagoya-u.ac.jp, hasegawa@nii.ac.jp, shimada@itc.nagoya-u.ac.jp

***Abstract*—Due to emerging cybersecurity threats, traditional networks struggle to adapt to new challenges because of their static nature and need for manual adjustments. In contrast, the inherent flexibility and rapid adaptability of Software-defined Networks (SDN) present an opportunity to overcome these limitations. Leveraging this potential, we propose a novel approach for automatically generating Access Control Lists (ACLs) within SDN environments. The system centralizes Access Control to the User Database and automatically generates derived rules, thus reducing administrators' manual work. By implementing Port Access Control, we can ensure that only authentic clients can access network resources. As a second feature, the system can change ACLs to block traffic or forward traffic to an Intrusion Detection System (IDS) for deeper inspection in case of suspicious activity like failed login attempts. To demonstrate the effectiveness, we evaluated the system in two use cases, initial client connection and dynamic adaption to authentication events, to test and compare the implementation to other systems. The evaluation proved that we can reduce manual processes and enhance the security of a network by dynamically generating ACLs to isolate clients.**

***Index Terms*—Software-defined Networking; Authentication; Access Control Lists**

I. Introduction

Digital transformation has exponentially increased the complexity of network architectures, presenting unprecedented challenges in maintaining robust security frameworks. In this ever-evolving digital landscape, cybersecurity threats have become more sophisticated, leveraging the linkage of modern infrastructures to exploit vulnerabilities at an alarming rate. Traditional network security mechanisms, which mainly rely on static configurations and manual oversight, are increasingly proving inadequate against this backdrop of dynamic and evolving threats [1]. The inherent limitations of these conventional approaches, characterized by their inflexibility and slow response times, underline the urgent need for more adaptable, responsive security measures.

Software-Defined Networking is a revolutionary paradigm that promises to redefine network management and security [2]. At its core, SDN separates the network's control logic from the underlying hardware, facilitating a centralized and programmable framework that transcends traditional hardware limitations [3]. This separation enhances network flexibility and management and introduces dynamism and adaptability, which were unachievable with conventional network architectures until now. According to a report by Global Market Insights, the SDN Market, valued at USD 28.2 billion in 2023, is expected to experience significant growth, with a projected expansion rate exceeding 17% annually from 2024 to 2032 [4]. Through SDN's capabilities, networks gain the flexibility to adapt swiftly to evolving security demands. This flexibility enables the immediate implementation of tailored security measures and configurations to effectively counter new threats, as illustrated in the study by Ali et al. [5].

Furthermore, our contribution is complemented by the work of Yakasai et al. in FlowIdentity, which advances virtualized network access control within SDN through a role-based firewall [6]. We also draw upon the architectural insights of Casado et al. in Ethane, demonstrating the power of centralized policy enforcement [7], and the innovative approach of Mattos et al. in AuthFlow, focusing on authentication and access control mechanisms in SDN environments [8].

Furthermore, this approach was refined by incorporating a sophisticated analysis of authentication logs, drawing upon the work of Xing et al. in SnortFlow, which explores an OpenFlow-based intrusion prevention system in cloud environments [9], and the study by Le et al. on a flexible network-based intrusion detection and prevention system on Software-Defined Networks [10].

The paper progresses from reviewing related SDN security work in Section II to foundational concepts in Section III. Section IV describes our system for automating ACLs, while Section V covers its implementation. Section VI evaluates the system's performance against existing methods, and Section VII encapsulates concluding thoughts and future directions, rounding off our discussion.

II. Related Work

Network security and access control advancements are crucial in the evolving landscape of SDN. The following studies demonstrate that emerging technologies and frameworks are pivotal in addressing these challenges.

A. Intrusion Detection with Authentication Events

In the study by Chu et al. [11], "ALERT-ID," an intrusion detection system for large-scale network infrastructures, is presented. The system distinguishes between normal operations and potential security threats through real-time analysis of authentication, authorization, and accounting (AAA) system logs. It employs behavioral models built on historical access patterns and user profiles, efficiently identifying potential intrusions and misuse. Notably, ALERT-ID balances the need

for thorough security monitoring with a manageable false alarm rate, demonstrating the importance of dynamic security measures in complex network environments.

B. Dynamic Access Control in SDN

Transitioning to dynamic access control, the work by Nayak et al. introduces "Resonance: Dynamic Access Control for Enterprise Networks" [12]. Resonance implements dynamic security policies with a registration phase, complemented by real-time monitoring and inference mechanisms specified by administrator rules.

Further extending the concept of network security, the study by Martins et al. [13] introduces an innovative access control architecture for SDN, leveraging the ITU X.812 standard. This framework incorporates Role-Based Access Control (RBAC) with traffic prioritization rules, significantly advancing towards more granular and role-specific access control mechanisms in network environments. A notable feature of this architecture is its reliance on predefined rules, which are carefully established and mapped to user roles by administrators. This approach requires administrators to proactively define comprehensive security and access parameters, ensuring a tailored access control environment but also necessitating substantial initial setup and configuration efforts.

These studies highlight a significant progression in network security approaches, evolving from intrusion detection systems to dynamic and sophisticated role-based access control models. Our forthcoming work aims to advance the field by introducing a new system that significantly enhances these foundational methodologies and requires less administrative work.

III. Preliminary Concepts

This section introduces foundational concepts relevant to network management and security, including OpenFlow switches in SDN, 802.1X Port-Based Network Access Control, access management with Active Directory and LDAP Authentication, and the Extensible Authentication Protocol over LAN (EAPOL).

A. The Role of OpenFlow Switches in SDN

SDN represents a paradigm shift in network management, with OpenFlow switches being a cornerstone of this architecture. These programmable switches enable dynamic network control, efficient routing, and robust access control mechanisms [14].

Despite their advantages, OpenFlow switches introduce security challenges, such as controller security and flow table vulnerabilities [15].

B. Strengthening Network Defenses with 802.1X Port-Based Network Access Control

IEEE 802.1X Port-Based Network Access Control significantly strengthens network security by implementing stringent access control measures. It restricts network entry to verified devices and users, ensuring high network integrity and protection [16].

The interaction follows a precise sequence: initiation, identity presentation, and authentication verification, utilizing Extensible Authentication Protocol (EAP) over LAN [17]. This structured process confirms the identity of the devices and users and mitigates potential replay and impersonation threats.

In summary, 802.1X Port-Based Network Access Control is a foundational network security pillar, effectively managing access through rigorous authentication and encryption practices [17].

C. Access Management with Active Directory and LDAP Authentication

Active Directory (AD) [18] and Lightweight Directory Access Protocol (LDAP) [19] Authentication play vital roles in access management within network environments. AD streamlines user group and role management in Windows networks, while LDAP provides a unified authentication framework across multiple services.

A centralized user database with Access Rights and user Groups and Roles is employed in a typical company network, often based on Active Directory. This centralized management offers scalability, ease of administration, and seamless integration, making it indispensable for effective access control and user management.

IV. Proposed System

This section presents a comprehensive overview of our proposed system, designed to significantly enhance network security and efficiency through advanced ACL management and authentication mechanisms.

A. Previous works Problem and Approach

Building on our established framework for automating ACL generation through statistical analysis of communication patterns, this work seeks to leverage further and enhance the existing infrastructure. Our initial efforts in [20] laid the foundational groundwork for this approach, which saw significant development and refinement in subsequent studies, such as [21]. A primary challenge identified in our exploration was addressing the need for authentication proof for IP addresses.

We have refined our approach to take advantage of a typical user database, like Active Directory [18], a standard part of a company network. This centralized database offers a significant advantage because user and group management and their corresponding resource access permissions are already handled there. We aim to leverage this existing infrastructure to streamline the process and eliminate redundant tasks for administrators.

B. Architecture of the proposed system

In the proposed system, we enhance network security through Port Access Control, which limits network port access exclusively to authenticated users. This approach is grounded in a security model where each port is individually secured and requires authentication before granting access. As a result, each user undergoes an authentication process, enhancing

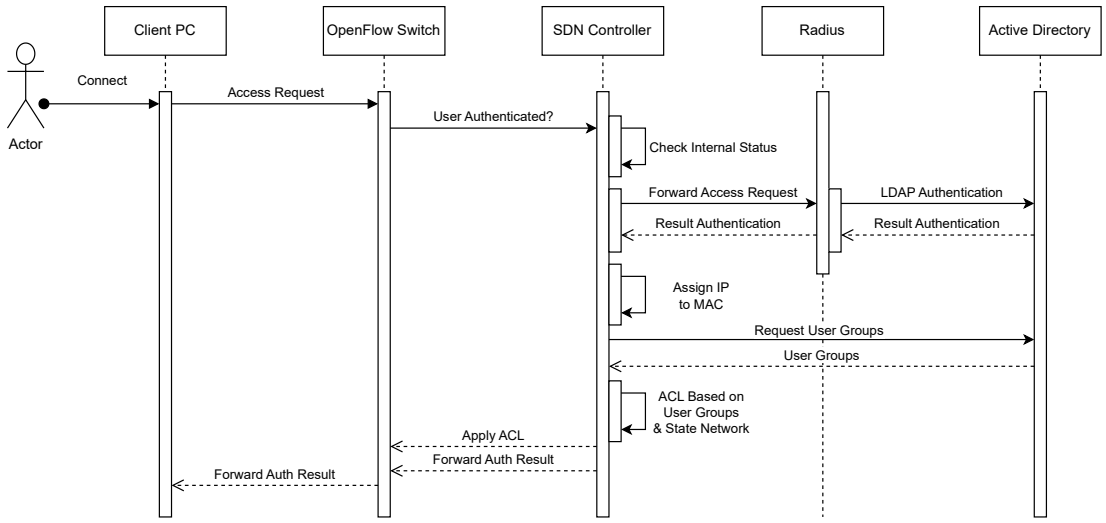

Figure 1. Dynamic ACL Adaption based on Authentication Events

the network's overall security posture. The process for user connection is designed with precision to ensure a secure and efficient authentication mechanism and can be found in Figure 1.

Initially, the system is configured only to allow EAPOL messages, which are then directed to an authenticator component. This step ensures that there is no communication before the client authenticates.

The SDN Controller checks its internal state for preconfigured users based on MAC and IP addresses. This information is crucial for comparing against new data received during authentication. Authentication messages for EAPOL are forwarded to a Radius server, which validates the credentials against a common user database, typically an active directory.

Upon successful authentication, the system assigns an IP address to the specific MAC address by inserting a record in a DHCP server. This procedure ensures that the assigned IP address corresponds to a specific MAC address and is associated with a specific port. The system then generates User-Specific Access Control Lists tied to a particular port by requesting user groups for the specific username from the Active Directory via LDAP. The fundamental idea is that users in the same group as a specific server should also have access to that server. For instance, if the user 'Ben' is a member of the 'Mail' group, to which our Mailserver also belongs, we understand that we need to create ACLs that permit this specific traffic. Consequently, we establish a whitelist to allow this connection and block all other traffic.

The system can identify users labeled as servers, which differ from standard clients, through a unique identifier group assigned explicitly to servers. The same concept would also be possible with a specific role depending on the existing usage in a company network, but in our case, we focused on groups.

Thus, any user belonging to this shared group is required to be able to establish a connection with the designated server. A port scan is conducted for servers to identify open ports and protocols. This information is linked to the user group of the server. For clients, the system constructs ACLs based on user groups and existing database information about servers, ensuring only communication between the user's MAC & IP address and the server's IP address and port. Since, for the system, only the port is necessary, it is also possible to apply this to multiple SDN switches since the ACL is only bound to a port on a switch.

Administrators can create templates for specific scenarios, such as restricting SSH access to only administrators. This template can be created by adding a template for port 22 associated with, for example, the user group "Administrator". These templates are scanned before actual ACL generation, with a higher priority than user roles and ports discovered during the generation process. Additionally, templates are essential for managing Internet traffic, with administrators defining traffic routing rules that cannot be determined using available information.

Since the update of active users and the checking for new ports on the servers occur periodically, once a day, and can be adjusted by an administrator, the system maintains minimal applied ACLs and removes unused ones if a host is no longer connected, thereby leading to higher efficiency [22].

The system also accounts for dynamic authentication events, using an LDAP proxy to monitor authentication activities. This monitoring detects suspicious activities based on failed login events for a specific service. The key idea is that if there are a certain number of failed login attempts, we aim to modify the network layout and ACLs for the particular user being observed, as their actions may be considered suspicious. Therefore, we can also redirect traffic to an Intrusion Detection System to look further since this traffic is probably suspicious or block all traffic or certain parts by adapting the ACLs dynamically. Alert-ID inspires this approach [11] with the extension that we do direct changes in the network when we encounter malicious behavior.

This comprehensive approach to Port Access Control, supported by a robust authentication framework, ensures a secure and efficient network access management system, safeguarding the integrity and security of the network infrastructure. To further enhance the effectiveness of our proposed system, we incorporate a quantitative analysis of Access Control Lists within the network when these settings are applied.

C. Quantitative Analysis of Access Control Lists

We rely on a quantitative understanding of ACLs within the network to understand the system's complexity. The ACL count is determined based on user, group, and port configurations, providing insights into the scale and complexity of the access control mechanisms.

- **Number of ACLs per User** (N_u): This metric quantifies the number of ACL entries associated with each user. It is calculated by summing the ports across all groups a user belongs to, given by

$$N_{u_i} = \sum_{g=1}^{G_i} P_{g_i} \tag{1}$$

where P_{g_i} is the number of ports for group g for user i and G_i is the total number of groups for user i.

- **Global Number of ACLs** (N_{global}): The total number of ACLs across the network reflects the overall complexity of access control. It is computed as

$$N_{global} = \sum_{i=1}^{U} N_{u_i} \tag{2}$$

where U represents the total number of users, and N_{u_i} is the number of ACLs for user i.

- **Number of ACLs per Switch** (N_s): Understanding the ACL count for each switch helps in optimizing access control at the local level. This metric is determined by

$$N_s = \sum_{i \in S} N_{u_i} \tag{3}$$

where S is the set of users connected to the switch, and N_{u_i} represents the number of ACLs for user i in the set S.

The subsequent Sections will examine the implementation details, demonstrating its capabilities and effectiveness.

V. Implementation

Following the conceptual framework outlined in Section IV, the practical implementation of the Port Access Control system integrated various components. The design in Figure 2 presents how different parts work together.

We chose the Faucet SDN Controller [23] for our prototype, which uses Ryu [24] in the backend and Gauge to view events on the switch. It has the considerable advantage that the rules are defined in YAML (YAML Ain't Markup Language). One significant advantage of this architecture is that these files are human-readable and easy to understand. The initial setup of the Openflow Switch contains just the port information

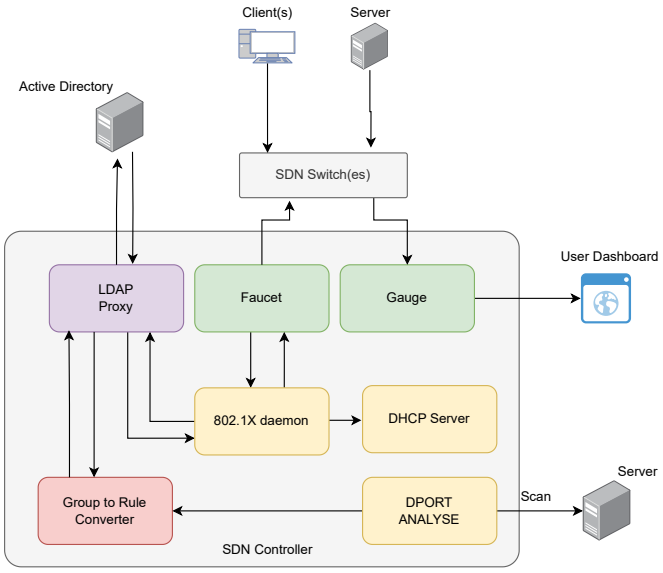

Figure 2. Architecture SDN Controller

and requires authentication before connecting to the Network. Furthermore, specific default rules, such as special treatment for the SDN controller and the Active Directory, were specified beforehand, as these settings are essential when configuring a new network. We used the 802.1X daemon Chewie [23] as a starting point, and then it was heavily adapted to get the user groups via a simple LDAP proxy. A second Service called "Group to Rule" will apply the ACLs as discussed in Section IV. An example rule can be found in Figure 3. It shows the resulting rule with a defined protocol, port, source MAC & IP address, and destination IP address. Since this is directly applied to the port, no other traffic can pass the OpenFlow switch port.

```
acls:
  mac_whitelist_user_ben:
    - rule:
        dl_type: 0x800       # ipv4
        nw_proto: 6          # tcp
        tcp_dst: 80          # port
        eth_src:  32:90:43:57:f2:01
        ipv4_src: 192.168.0.1
        ipv4_dst: 192.168.0.9
        actions:
            allow: 1
    - rule:
        actions:
          allow: False
```

Figure 3. FAUCET ACL CONFIGURATION

A Python script that searches for UDP and TCP ports on the Server provides the open ports needed to craft the ACLs. It then saves this information into a database with the corresponding MAC address and IP address. One problem is that the Server does not directly have an IP address when we try to scan it. We must wait until the IP address is handed

over via the DHCP server to start scanning. Therefore, for a server, the ACLs can only be applied later on and not directly, which is not a problem since the default rules still block all access to the Network, and only DHCP is then allowed to obtain an IP address. A simple folder structure was defined for the templates where an administrator can place templates for Groups and specific Ports and the initial Network operations like DHCP and DNS.

The dynamic adaption of the ACLs depending on authentication data is done via the LDAP Proxy. All servers in the Network try to authenticate their users via LDAP Bind requests to the LDAP Proxy, which then forwards this to the Active Directory. This approach allows us to determine whether authentication was successful. We implemented a counter for each user with a threshold of six, which will reset after some time, as determined in Alert-ID [11]. To identify the client who tries to connect, a small script monitors the log file with authentication events and then sends the event from the Server to the controller via a small script. This procedure serves only for demonstration purposes, and, in the future, an SSO Server can perform this task since every client needs to authenticate there when they access a server. For example, blocking users who attempt to authenticate with a username different from the one they initially used for network access would also be possible. Additionally, we could restrict access by blocking only the specific port or server access for such users. Another option would be to reroute all their traffic through an IDS. This approach could alleviate the load on IDS systems by selectively forwarding traffic from suspicious hosts.

The implementation phase reaffirmed the proposed system's potential to enhance network security through fine-grained access controls. However, it highlighted the complexities of managing an extensive rule set, especially in larger networks.

VI. Evaluation

In this evaluation chapter, we begin with a detailed examination of the technical aspects of our experimental setup, laying the groundwork for a thorough assessment. We then delve into the feasibility of the proposed system, followed by a comparative analysis of its efficiency and complexity against existing systems. This analysis sets the stage for a nuanced discussion synthesizing our findings and their implications.

A. Experimental Conditions

Our experimental setup was designed to mirror a realistic laboratory environment consisting of multiple physical PCs and servers to simulate a conventional corporate network infrastructure. The network configuration included five Windows clients alongside a singular Linux client. We assigned the clients to different user groups in the active directory. The configuration of each Client and its connected port can be found in Table I. In setting up our experiment, we went with a mix that one would typically find in an office: a mail server for emails and a GitLab instance for the devs to collaborate on code. This way, we could see how different roles, like developers needing GitLab and managers relying on emails, would interact with the system. It is a practical approach that helps us understand how our setup performs in a real-world scenario.

At the core of our Network was an Active Directory on a Windows Server 2019, connected to a dedicated port at the OpenFlow switch. This switch was a Linux PC running Ubuntu 22.10, with an Intel(R) Core(TM) i7-8700 CPU supporting OpenFlow protocol version 1.3.

This detailed setup provides a solid foundation for evaluating the system's feasibility, performance, and complexity.

TABLE I. CLIENTS IN THE NETWORK

Client Name	Port	Source MAC	Groups
Client1	3	1C:69:7A:6D:C6:27	mail, gitlab
Client2	4	1C:69:7A:43:7C:12	mail
Client3	5	1C:69:7A:6D:C8:B0	mail, gitLab
Client4	6	1C:69:7A:6D:C7:EE	mail
Client5	7	1C:69:7A:6D:C8:16	mail

B. Feasibility

The project aimed to demonstrate the feasibility of such a system and highlight its advantages. Therefore, we conducted multiple experiments to verify the system's operability to achieve this.

1) Experiment 1: Connection to the Network: In our initial experiment, we aimed to verify the functionality of the system's initial configuration and the practical application of Access Control Lists. We began by attempting to connect a server to the Network. Initially, all packets except EAP packets were blocked, preventing any network connection without proper authentication.

To facilitate authentication, we configured the server's wpa_supplicant with EAP after setting up a dedicated user account in the Active Directory for the server, marked by the "server" group identifier, to distinguish it as such. Additionally, the server was assigned to the "mail" group to define its access rights. The authentication process utilized standard Username and Password credentials defined within the Active Directory.

Upon initiating these configurations, we observed successful authentication, followed by the server obtaining an IP address via DHCP. The IP address assignment was managed by the SDN Controller, ensuring the server's connectivity post-authentication. We then proceeded with a port scan, which was feasible only after the OpenFlow switch recognized the server's IP, confirming that the server was operational. The procedure was repeated for the second GitLab server.

Subsequently, we connected a client machine to the Network. Like the server setup, this Client was denied network access until authentication credentials were provided. After authentication, the SDN Controller dynamically generated ACLs based on the Client's group memberships.

For example, the first Client, identified as a developer, was granted access to both the mail server and GitLab, as reflected in the applied ACLs (refer to Table II, lines 1 and 2). This access control was strictly enforced, with all unauthorized traffic being blocked at the port level based on the authenticated

TABLE II. ACL CONFIGURATION FOR FIVE CONNECTED CLIENTS

OpenFlow Port	Source MAC	Source IP	Group	Destination IP	Destination Port	Description
3	1C:69:7A:6D:C6:27	192.168.11.11	mail	192.168.11.101	25, 993, 995	Mailserver
3	1C:69:7A:6D:C6:27	192.168.11.11	gitlab	192.168.11.102	22, 80, 443	GitLab
4	1C:69:7A:43:7C:12	192.168.11.12	mail	192.168.11.101	25, 993, 995	Mailserver
5	1C:69:7A:6D:C8:B0	192.168.11.13	mail	192.168.11.101	25, 993, 995	Mailserver
5	1C:69:7A:6D:C8:B0	192.168.11.13	gitlab	192.168.11.102	22, 80, 443	GitLab
6	1C:69:7A:6D:C7:EE	192.168.11.14	mail	192.168.11.101	25, 993, 995	Mailserver
7	1C:69:7A:6D:C8:16	192.168.11.15	mail	192.168.11.101	25, 993, 995	Mailserver

source MAC address and specified port. In contrast, a client identified as a manager, and thus only requiring access to the mail server, demonstrated restricted network access in line with their role (refer to Table II, line 3). Attempts to access GitLab by this Client were blocked, illustrating the ACLs' role-based access control. After connecting all clients, each Client and port results can be found in Table II.

Upon issuing a logoff command to the RADIUS server, all associated ACLs were cleared, reverting the system to its default state of blocking all traffic from the disconnected Client. Logging in with a different username on the same PC triggered a reallocation of ACLs, aligning with the new user's access rights. This experiment demonstrated the feasibility of initially creating and effectively applying ACLs within our network environment.

2) Experiment 2: Failed Logins: In the second experiment, we tested failed login attempts to evaluate the systems' response mechanisms. This test simulated incorrect authentication attempts on the GitLab server to observe the system's reaction.

The experiment began with a series of failed login attempts, with each unsuccessful attempt logged by the SDN Controller. After the sixth failed attempt, the SDN Controller adjusted the ACLs, cutting off the Client's access to the server and other network components. An alert was automatically sent to the network administrator, who could either restore the Client's access after a successful re-authentication or suspend the Client for further investigation.

Additionally, we tested the Network's traffic mirroring feature. In this part of the experiment, despite multiple failed login attempts, the Client was not disconnected from the Network. Instead, the Client's traffic was mirrored to a specific port on an OpenFlow switch. This procedure was verified using tcpdump to confirm that the traffic mirroring was functioning as intended, without the integration of an IDS, since this was not in the scope of the experiment.

C. Complexity and Efficiency

To evaluate our system's complexity and OpenFlow rule management capability, we compared it against other SDN security methods by examining the number of OpenFlow rules in different scenarios. Our analysis included a baseline scenario without ACLs, a basic ACL setup, and scenarios involving VLANs. The scenario with Basic ACLs has only rules for direct IP access. That means we only specify that user X can access server Y without further defining which ports or protocols. The VLAN example does not have any specific ACLs. It splits the users into two groups, usually some kind of department in a corporate network. This option has the disadvantage of allowing clients from the same department to communicate, which does not prevent malware from spreading.

Table III summarizes the OpenFlow rule count for each scenario. As observed, the number of OpenFlow rules directly reflects the count of Faucet ACLs. As discussed in Formula 2, the number of ACLs will increase linearly to the number of users. The dynamic ACL configuration, while more complex, demonstrates the system's flexibility and responsiveness to network changes without significantly impacting performance.

TABLE III. RULE COUNT COMPARISON

Scenario	Faucet ACLs	OpenFlow Rules
No ACLs	0	27
Basic ACLs	8	67
VLANs	N/A	67
Dynamic ACLs	32	91

D. Discussion

In discussing the outcomes and implications of our experiments, it is essential to consider both the implemented system's strengths and potential challenges. To offer a comprehensive understanding of our system's enhanced performance and its innovative approach to network security and management, we performed an extensive comparison across several key metrics, including security level, scalability, and manageability. This comparison, detailed in Table IV, is based on empirical data from ACL number analytics, a comparative analysis of system architectures, and their maintenance needs, highlighting our proposed system's superiority in terms of security, scalability, and ease of management.

The introduction of automated, fine-grained whitelist ACLs represents a significant step forward in network security management. The configuration process substantially decreases administrative overhead and mitigates the risk of human error, which is prevalent in manual configurations. A crucial advantage of this approach is the centralization of security decisions, such as access rights, in a singular user database. This consolidation ensures that modifications to access rights are uniformly applied across the Network and all services utilizing this common user database, thereby enhancing consistency and security within the system. As demonstrated in Experiment 1, the automation of ACL configuration significantly reduced administrative overhead since all needed restrictions are applied individually for each Client without the need for additional

TABLE IV. COMPARISON OF NETWORK SECURITY AND MANAGEMENT APPROACHES

Metric	**No ACLs**	**Basic ACLs**	**VLANs**	**Resonance**[12]	**ACL Based on X812**[13]	**Proposed System**
Security Level	Low	Medium	Medium	High	Very High	Very High
Port Security	None	None	None	None	Full	Full
Performance Impact	Low	Medium	Medium	Moderate	Moderate	Moderate
Scalability	High	Moderate	Good	Moderate	Moderate	Excellent
Manageability	Easy	Moderate	Moderate	Moderate	Moderate	Easy
Centralization	None	Low	Low	Medium	Medium	High
Flexibility	Low	Moderate	Low	Very High	Very High	High
Cost Efficiency	High	Moderate	Moderate	Low	Moderate	High
Integration Capability	Seamless	Moderate	Challenging	High	High	Low
Resilience	Low	Medium	Medium	High	High	High
Automation & Dynamic Response	None	None	None	Semi-Automated	Semi-Automated	Fully Automated
ACLs based on Authentication Events	None	None	None	None	None	Supported

adaption by the administrator. Contrarily, this centralized, automated approach ensures that only authenticated users and their associated MAC addresses are actively maintained in the system, limiting access to authorized entities and inherently reducing the risk of unauthorized access. Another significant benefit of our approach is its scalability and ease of integration across multiple switches without additional overhead. Since ACLs and clients are bound to specific ports and not to the physical switches themselves, our system can seamlessly scale to accommodate an extensive network infrastructure with multiple SDN switches. ACLs are applied uniquely to each switch, as delineated in Formula 3, ensuring efficient and tailored security measures are in place, irrespective of the size or complexity of the Network.

However, this automation and simplification come at the cost of increased complexity due to the more significant number of ACLs required to maintain fine-grained control over network access. The number of ACLs does not directly impact the system's performance since it only inspects the TCP header to minimize performance impact, compared to, for example, complex rules that inspect the TCP payload. According to Cabrera et al. [25], the time required to check the payload is, on average, 4.5 times longer than that required for header checks. Therefore, even with many ACL rules, the focus on header information ensures minimal impact on network throughput, as even a single ACL with a TCP header rule necessitates the inspection of every packet. One drawback is that we need to prepare the clients and the server to perform an 802.1X authentication.

One of the more critical considerations is the system's approach to handling failed login attempts, as demonstrated in Experiment 2. Completely blocking access after a series of incorrect credentials can safeguard against brute-force attacks but also pose a risk to business continuity. For instance, automated tools using outdated credentials could inadvertently trigger these security measures, leading to unnecessary disruptions. This aspect of the system necessitates a careful balance between maintaining robust security and ensuring uninterrupted business operations.

Integrating traffic mirroring for suspicious hosts presents a nuanced approach to enhancing security monitoring without overloading the Network or the IDS. By selectively mirroring traffic from potentially compromised hosts, the system can focus on analyzing and responding to genuine threats, improving overall security efficiency. This concept aligns with the approach discussed in [26], which proposes a clustering-based flow grouping scheme that assigns Network flows to various IDSs based on routing information and flow data rates, aiming to optimize the load distribution among IDSs and enhance attack detection capabilities.

VII. Conclusion

In conclusion, the proposed system for Port Access Control presents a straightforward implementation framework that significantly enhances network security by enforcing fine-grained access control rules. By leveraging a common user database, such as Active Directory, and binding access controls to specific MAC addresses, the system ensures that only authenticated users can access network ports, providing a robust security posture. Moreover, the approach of mirroring traffic from suspicious hosts, particularly those with repeated failed login attempts, suggests a promising avenue for optimizing IDS performance. Optimizing algorithms for handling multiple concurrent access requests, managing extensive rule sets without compromising performance, and assessing the impact of selective traffic mirroring on IDS efficiency are critical future research areas to enhance scalability and maintain security in complex network architectures.

Acknowledgment

The authors would like to thank Prof. Hiroki Takakura for useful advice. This work was partially supported by JSPS KAKENHI Grant Number JP19K20268 and JP23H03396.

References

[1] H. Zhou, C. Wu, M. Jiang, B. Zhou, W. Gao, T. Pan, and M. Huang, "Evolving defense mechanism for future network security," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 45–51, 2015.

[2] S. T. Ali, V. Sivaraman, A. Radford, and S. Jha, "A survey of securing networks using software defined networking," *IEEE transactions on reliability*, vol. 64, no. 3, pp. 1086–1097, 2015.

[3] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN control: Survey, taxonomy, and challenges," *IEEE*

Communications Surveys & Tutorials, vol. 20, no. 1, pp. 333–354, 2017.

[4] Global Market Insights, "Software Defined Networking (SDN) Market," 2024, report ID: GMI2395, Accessed: 2024-04-09. [Online]. Available: https://www.gminsights.com/industry-analysis/software-defined-networking-sdn-market

[5] F. S. Ali, R. Amin, M. Majeed, and M. M. Iqbal, "Dynamic ACL Policy Implementation in Software Defined Networks," in *2022 International Conference on IT and Industrial Technologies (ICIT)*, Oct 2022, pp. 01–07.

[6] S. T. Yakasai and C. G. Guy, "FlowIdentity: Software-defined network access control," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. IEEE, 2015, pp. 115–120.

[7] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: taking control of the enterprise," in *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 1–12.

[8] D. M. Ferrazani Mattos and O. C. M. B. Duarte, "AuthFlow: authentication and access control mechanism for software defined networking," *annals of telecommunications*, vol. 71, pp. 607–615, 2016.

[9] T. Xing, D. Huang, L. Xu, C.-J. Chung, and P. Khatkar, "SnortFlow: A OpenFlow-Based Intrusion Prevention System in Cloud Environment," in *2013 Second GENI Research and Educational Experiment Workshop*, March 2013, pp. 89–92.

[10] A. Le, P. Dinh, H. Le, and N. C. Tran, "Flexible Network-Based Intrusion Detection and Prevention System on Software-Defined Networks," in *2015 International Conference on Advanced Computing and Applications (ACOMP)*, Nov 2015, pp. 106–111.

[11] J. Chu, Z. Ge, R. Huber, P. Ji, J. Yates, and Y.-C. Yu, "ALERT-ID: analyze logs of the network element in real time for intrusion detection," in *Research in Attacks, Intrusions, and Defenses: 15th International Symposium, RAID 2012, Amsterdam, The Netherlands, September 12-14, 2012. Proceedings 15*. Springer, 2012, pp. 294–313.

[12] A. K. Nayak, A. Reimers, N. Feamster, and R. Clark, "Resonance: dynamic access control for enterprise networks," in *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, ser. WREN '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 11–18.

[13] B. J. C. de A. Martins, D. M. Mattos, N. C. Fernandes, D. Muchaluat-Saade, A. B. Vieira, and E. F. Silva, "An Extensible Access Control Architecture for Software Defined Networks based on X.812," in *2019 IEEE Latin-American Conference on Communications (LATINCOM)*, 2019, pp. 1–6.

[14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008.

[15] M. S. Farooq, S. Riaz, and A. Alvi, "Security and Privacy Issues in Software-Defined Networking (SDN): A Systematic Literature Review," *Electronics*, vol. 12, no. 14, 2023.

[16] "IEEE Standard for Local and Metropolitan Area Networks–Port-Based Network Access Control," *IEEE Std 802.1X-2020 (Revision of IEEE Std 802.1X-2010 Incorporating IEEE Std 802.1Xbx-2014 and IEEE Std 802.1Xck-2018)*, pp. 1–289, 2020.

[17] J. Vollbrecht, J. D. Carlson, L. Blunk, D. B. D. Aboba, and H. Levkowetz, "Extensible Authentication Protocol (EAP)," RFC 3748, Jun. 2004.

[18] B. Desmond, J. Richards, R. Allen, and A. G. Lowe-Norris, *Active Directory: Designing, Deploying, and Running Active Directory*. " O'Reilly Media, Inc.", 2008.

[19] J. Sermersheim, "Lightweight Directory Access Protocol (LDAP): The Protocol," RFC 4511, Jun. 2006.

[20] H. Hasegawa, Y. Sato, and H. Takakura, "Construction of Secure Internal Network with Communication Classifying System Using Multiple Judgment Methods," *International Journal on Advances in Telecommunications*, vol. 13, no. 3 & 4, 2020.

[21] Y. Sato, H. Hasegawa, and H. Takakura, "Construction of Secure Internal Networks with Communication Classifying System," in *ICISSP*, 2019, pp. 552–557.

[22] M. Ali, N. Shah, and M. A. Khan Khattak, "DAI: Dynamic ACL Policy Implementation for Software-Defined Networking," in *2020 IEEE 17th International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET)*, Dec 2020, pp. 138–142.

[23] FaucetSDN, "Faucet," 2024, accessed: 2024-04-09. [Online]. Available: https://github.com/faucetsdn/faucet

[24] Ryu SDN Framework Community, "Ryu sdn framework," 2024, accessed: 2024-04-09. [Online]. Available: https://ryu-sdn.org

[25] J. B. Cabrera, J. Gosar, W. Lee, and R. K. Mehra, "On the statistical distribution of processing times in network intrusion detection," in *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, vol. 1. IEEE, 2004, pp. 75–80.

[26] T. Ha, S. Yoon, A. C. Risdianto, J. Kim, and H. Lim, "Suspicious flow forwarding for multiple intrusion detection systems on software-defined networks," *IEEE Network*, vol. 30, no. 6, pp. 22–27, 2016.