Analyzing Flow Rule Attacks and Policy Enforcement in Software Defined Networking

Vlad Florea
Supervisors: Mauro Conti, Chhagan Lal
EEMCS, Delft University of Technology, The Netherlands

June 19, 2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

Software Defined Networking (SDN) is a new paradigm that allows for greater reliability and more efficient management compared to traditional networks. However, SDN security is a developing field, and research towards fixing significant security vulnerabilities is still ongoing. One major threat to SDN security are attacks that seek to exploit policy and flow rule enforcement. This paper aims to summarise how these attacks are conducted and the weaknesses they target. Then, state-of-the-art solutions to these weaknesses will be presented, along with their use cases, advantages and disadvantages. Finally, an improvement on the state-of-the-art solutions will be proposed, as well as a potential direction for future research..

1 Introduction

The field of networking has become increasingly stagnant as the difficulty of managing traditional networks grows alongside their complexity. The bundling of the control and data planes results in a lack of flexibility when routing traffic under unpredictable load changes. Moreover, network operators have to deal with unexpected changes in topology due to units failing or attacks, lack of proper documentation, and vendor-specific commands. Enforcing the necessary policies in such a volatile environment is a challenging task.[1]

SDN is a new paradigm that seeks to solve these issues by separating the network's control plane from the data plane. This separation results in a system that makes use of a logically centralised controller, enabling more efficient management and better scalability. The controller instates flow rules and policies over the network elements using a southbound API, the most common of which is OpenFlow.[2] However, while the SDN architecture certainly offers security advantages, such as easier transfer of feedback from threat detection, it also poses additional security risks in other areas.[3] Mainly, there is not as much existing research on the topic of security as with traditional networks, and successful attacks on the control plane can result in the entire network being compromised.

The aim of this paper is to find solutions to attacks that affect flow rules and bypass or exploit network policies, analyse the limitations of these solutions, and then attempt to find a new solution to address these threats. This goal will be accomplished by reviewing the existing state-of-the-art literature in the field of SDN security, specifically articles and papers focusing on policy-based solutions.

The second section of the report will provide additional background on the topic of the research and present the most common types of attacks that seek to compromise the control plane by exploiting vulnerabilities in the system's policies. The third section will detail the state-of-the-art solutions that have been collected via literature review on the topic, and provide additional information on the concepts involved. The fourth section will discuss the weaknesses and limitations of current solutions and propose a direction in which future research on the topic could be conducted, in order to advance the state-of-the-art. The fifth section will summarise the ethical implications of this research, along with the steps taken to prevent misconduct. The sixth section will present potential improvements on the state-of-the-art literature and future research directions. The final section will offer a summary of the findings resulting from the research and concludes the paper.

2 Background and Related Work

Research in the domain of SDN security has progressed rapidly over the past few years, and as such, there exist a number of papers that address the topic of threat detection and solutions to security issues in Software Defined Networks. This paper aims to provide a comprehensive overview of the different types of attacks that exploit policies in the SDN control and data planes.

As part of the literature review phase of the project, papers were sought out through the IEEEXplore database, specifically, papers that provide comprehensive overviews of security issues in SDN, as well as others that focus on specific attacks related to policy and flow rule exploitation and solutions that counter them.

The paper [4] addresses policy-related vulnerabilities, different methods for policy conflict resolution and an overview of all implementations and solutions currently in use for countering policy attacks and network flow diversion. Network policies are rules that govern the behaviour of a network. They determine the level of access that different users have within the network, which traffic should be prioritised, and ensure quality of service for users and network security. Flow rules reside within the flow tables of the network switches and have the role of determining the manner in which packets traverse the network. In the pro-active SDN model, flow tables are populated ahead of time, and flow rules in the table are checked against incoming packets. In the reactive SDN model, each packet is extradited to the controller, which instates a flow rule in the switch in response to the packet. This model can adapt to new networking flows but is significantly less scalable than the pro-active model.

[3] provides an overview of how policies are enforced in SDN, compared to traditional networks, as well as a list of the entities involved. Network policies are selected in the controller based on the status of the network and whether incoming connections match their conditions. They are then sent to the network switches via the OpenFlow API, where they determine the application of flow rules on the incoming packets. The paper also explains the impact of policy conflicts on the security of the network and details different types of attacks that can exploit the SDN architecture. Moreover, it lists a number of state-of-the-art solutions for policy conflict resolution and real-time policy checking.

[5] offers a high-level view of numerous types of SDN attacks, as well as an explanation of how the OpenFlow API functions. The article outlines six main types of SDN attacks, targeting either the controller, applications or network elements.

1. Spoofing attacks involve an attacker pretending to be another host or using a fake address in order to fool the system into enforcing the wrong policies.

2. Tampering attacks target existing packets entering the network and involve the attacker modifying the header or payload of a packet in order to perform malicious actions.
3. Repudiation attacks are executed by performing an action on the network and, in the absence of appropriate logging functionality, pretending that said action never took place.
4. Denial of Service attacks are performed by overwhelming the system with a large number of packets or requests or by disrupting the network flow such that packets are directed through channels that lack the appropriate capacity to handle them. The resulting congestion prevents legitimate packets from being handled by the network.
5. Information Disclosure attacks involve the attacker discovering crucial information about the structure of the network, the inner workings of the network components or the contents of packets travelling through the network. This goal is usually accomplished via side channels, through the use of certain scanning tools or by eavesdropping on communication channels making use of weak encryption or none at all.
6. Elevation of Privileges can be performed either by impersonating another host after a successful spoofing attack, by increasing privileges in a weakly secured controller or by exploiting improper policy enforcement protocols to create policy conflicts and bypass security policies.

This paper differs from existing papers by providing an overview of a number of different attacks that target policies in SDN, as well as multiple solutions that specifically address these attacks through policy enforcement, policy conflict resolution and policy verification between the control and data planes.

3 Attacks and Solutions

This section will address several attacks that aim to exploit or bypass policies in the SDN control plane, the way they are performed and how they affect the functioning of the network. Solutions for these attacks will also be listed, and their mechanism of action will be explained. A comparison between the solutions can be found in Table 1.

3.1 Attacks

1. **Priority-passing Attack**

   The priority-passing attack is a type of spoofing attack. [6] It is performed by changing a host's IP or MAC address and impersonating another host to bypass the high priority flow rules and cause the controller to install malicious lowest-priority flow rules in the forwarding device. This leads to the targeted switch forwarding packets from the host to a destination originally intended for packets from another host. The attack can allow the malicious actor to take advantage of the system to accomplish different objectives - redirection to a different network or packet drop between two hosts. To accomplish the first objective, through IP/MAC passing, the system assigns flow rules meant for another sender's packets to packets sent by the malicious actor. In case the addresses are in different virtual local area networks (VLAN), VLAN-crossing is performed and the packets can enter a different, potentially sensitive VLAN. For the second objective, priority-passing is used to create Denial of Service (DoS) between two hosts, by changing the flow rule that is executed during packet egress (packet processing before they leave the network). The low-priority rule that is set instead will forward all packets to a different host, thus dropping all packets in the connection with the intended receiving host.

2. **Covert Channel Attack**

   A covert channel attack involves creating a means of transferring data between two entities that are normally not meant to communicate with each other. There are two types of covert channels: storage and timing channels. [7] In storage channel attacks, one process writes information directly or indirectly to a location. This information is then directly or indirectly read from said location by another process. In timing channel attacks, transmitted information is retrieved by interpreting the arrival time of packets at the receiver. Messages can be sent by the attacker by varying the time interval between packets. While jitter can reduce the effectiveness of the attack, simply increasing the time interval between attack packets can render this effect moot.

   Two examples of covert storage channel attacks that affect policies are the host-based attack and the switch-based attack. [8]

   In the host-based attack, the attacker takes advantage of the policies of a network by crafting a packet with a header that does not match existing rules, triggering the controller to apply new rules in the switches. The resulting rule conflict at the level of the switches allows the attacker to then create packets that can bypass filter rules and travel between isolated networks. This is performed via ICMP header modification, which exploits ICMP rules inherent to SDN controllers that are used for troubleshooting. Since these rules can not be excluded, the attack can not be easily detected at the switches.

   In the switch-based attack, a malicious rule that alters the MAC address of packet headers is installed in one of the switches. Then, once attack packets are sent to the switch, the rule is applied and the packet headers are modified. The original flow rule that would deny the attack packets access to their destination is thus circumvented, since the MAC address specified in its conditions no longer matches. This results in the attack packets bypassing filter rules and reaching their destination.

3.2 Solutions

1. **Preacher**

   Preacher [9] is a probabilistic policy checker that detects attacks stemming from compromised switches in an adversarial data plane. Header or payload hashing is utilised to sample a subset of packets travelling through

Table 1: Comparison between policy-related solutions

Solution	Goal	Method
Preacher [9]	Detect adversarial switches and routers	Probabilistic policy checking
Covert Channel Defender [8]	Rule conflict resolution and prevention of covert channel attacks	Classify rules into Equivalence Classes using VeriFlow
Policy Management and Enforcement System [10]	Automatic attack mitigation in ISP networks	Monitor switches and paths to determine network status and apply appropriate policies
Logical Security Architecture [2]	Real-time policy enforcement and attack mitigation	Priority-based policy conflict resolution and detection of attacks at the level of the switches using a modular approach
Switch-based Rule Verification [6]	Efficiently counter priority-passing attacks	Flow rule verification using HashMap

the network, and the policies associated with these samples are then identified and stored. Afterwards, all other sampled packets are checked using the policies of previously stored packets in order to determine their legitimacy.

Preacher works to detect numerous types of attacks since it can identify both injection attacks and Denial-of-Service attacks (DoS). Sampled packets have their information, as well as a timestamp, added to the History. An injection attack is detected when preceding packets stored in the History do not match the policy of the current sample, which specifies that certain packets must have already been received. A packet drop/DoS attack is detected when subsequent packets that are expected according to the policy in the current sample are never received.

Experiments have been performed in order to determine mean detection time for different sampling ratios, detection throughput, sampling overhead and the tradeoff between resources used and detection time.

Detection time was measured for sampling ratios of 0.9% and 1.3%. It was found that doubling the sampling ratio roughly halves the number of packets needed to detect an attack and that having a higher number of compromised switches working together increases the number of packets needed to detect an attack. Moreover, detection time decreases linearly with the increase in packet rate of the compromised switch.

Detection throughput is the number of samples the system can analyse simultaneously, and it was tested by scaling the number of threads and detector cores. The increase in throughput is slower than linear; however, there is a substantial increase in throughput with hyper-threading and higher detection thread counts.

Overhead is not substantial when using Preacher, with only a 1.5 MB increase in memory and CPU usage remaining within acceptable ranges at the level of the controller. At the level of the switches, there is no impact on throughput.

The size of a network affects detection time, and so does the usage of independent hash assignment. For small networks, only a few cores are needed to achieve detection times under 10 minutes, while for larger networks, tens of cores are needed for a detection time of under an hour. Independent hash assignment significantly slows down detection, requiring hundreds of cores for similar detection times, even in small networks.

2. **Covert Channel Defender**

   The Covert Channel Defender [8] aims to detect and resolve rule conflicts that can facilitate covert channel attacks in real time. It takes the form of an additional layer that bridges the controller and network.

   Conflict resolution is accomplished by categorising types of packets handled under the same rules in Equivalence Classes and storing these ECs in an efficient tree-based data structure called an extended Trie. The trie design is based on the one used by VeriFlow.

   CCD conflict resolution is performed in three phases: Constructing the extended trie, locating conflicting ECs and merging found ECs. First, the extended trie is constructed based on new rules generated by the controller. Then, when a new rule comes from an application, a lookup is performed in the trie by analysing each individual field of the rule. Finally, the leaves of the trie are updated with newly generated ECs and correlated ECs are linked using leaf pointers.

   New rules generated by the controller can cause rule conflicts, which are resolved by constructing bi-directional forwarding graphs between their correlated ECs. If the ECs overlap, the CCD generates new filter rules associated with the conflicting ECs. Conflict-free ECs are also merged in order to remove redundant rules and reduce the complexity of the trie.

   Several experiments were performed to determine rule processing delays in the case of different numbers of ECs, different field substitution rules and EC merging. CCD has been tested using an OpenFlow controller called Floodlight, where it was found to perform similarly to VeriFlow, CCD having a higher overhead with a difference of around 19%. It was found that CCD increases the packet forwarding delay by an average of 8.72% and boasts only a 0.16% reduction in throughput

compared to VeriFlow. Throughput is also reduced by only around 15% when compared to the base Floodlight controller.

3. **Policy Management and Enforcement System**

   The Policy Management and Enforcement System (PMES) [10] is a framework that can automatically configure and enforce policies within a network, responding to threats in real time. It is meant for use in Internet Service Provider (ISP) networks. The system consists of a monitoring plane, which contains the Monitoring Component (MC), and a policy plane, which contains the Policy Database (PD), Policy Decision Point (PDP) and Policy Orchestrator and Implementer (POI). The MC resides at the level of the switches in the network but is implemented as a separate component. The POI is a component that interfaces with the ISP controller, ensuring consistent traffic flow between the customer controllers and switches.

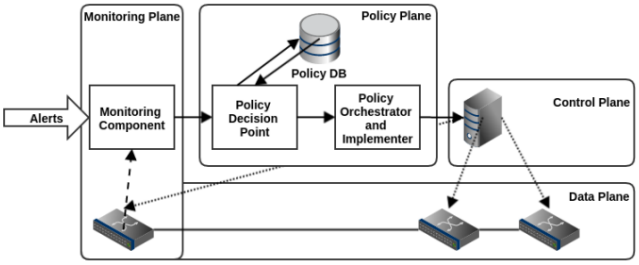

Figure 1: Structure of the PMES [10]

It aims to provide dynamic policy enforcement that adapts to the requests of particular customers and to mitigate Distributed Denial-of-Service (DDoS) attacks. This is achieved by enforcing new policies based on security notifications and alerts received by its Monitoring Component, which assesses the status of the network and determines whether it is under attack or not. When the Monitoring Component detects a congestion, the flow information, impact severity, security class and type of the attack are evaluated. This information is then sent to the Policy Decision Point, where the context is activated based on the event conditions. Afterwards, appropriate high-level policies are selected by the PDP and retrieved from the Policy Database. Then, these policies are sent to the POI, which performs path computation using source and destination IP addresses, policy action and necessary bandwidth from the PDP as input. Finally, the POI inserts Network Service Headers in the packets containing the rules in order to streamline packet header checking and distributes the flow rules to the appropriate switches. This allows for real-time enforcement of high-level policies in the controller and adjustment of network flow in the switches to prevent network congestion.

The authors performed an experiment to demonstrate the effectiveness of the Policy Management and Enforcement System in an ISP network by simulating a DDoS attack. PMES was implemented in Python and run as an OpenFlow application on the Ryu SDN controller, using Mininet. In order to determine the Quality of Service (QoS) of the system, throughput and network jitter of legitimate traffic were measured. Throughput drops sharply as soon as the DDoS attack begins but returns to normal levels in 10-20 seconds for all customers once new flows are established in the network. However, throughput dropped to zero for one of the customers again due to a late alert redirecting traffic through a high QoS path that was already in use. This experiment was run a second time, with the traffic being redirected through a lower bandwidth path after the late alert due to QoS provisioning. In this scenario, throughput remained at normal levels after the distribution of new flow rules. Network Jitter follows the same pattern, increasing once the attack takes place and taking a sharp decrease back to the previous level once new flow rules are set, taking up to 40 seconds in total and 20 seconds from the moment the redirection request is sent.

4. **Logical Security Architecture**

   The Logical Security Architecture [2] seeks to counter flooding and injection attacks from end hosts by carrying out dynamic management of security policies across the entire network. This is accomplished using a separate Security Management Application (SMA), which runs as an application in the SDN controller, and Switch Security Components (SSC), which are found in each of the switches.

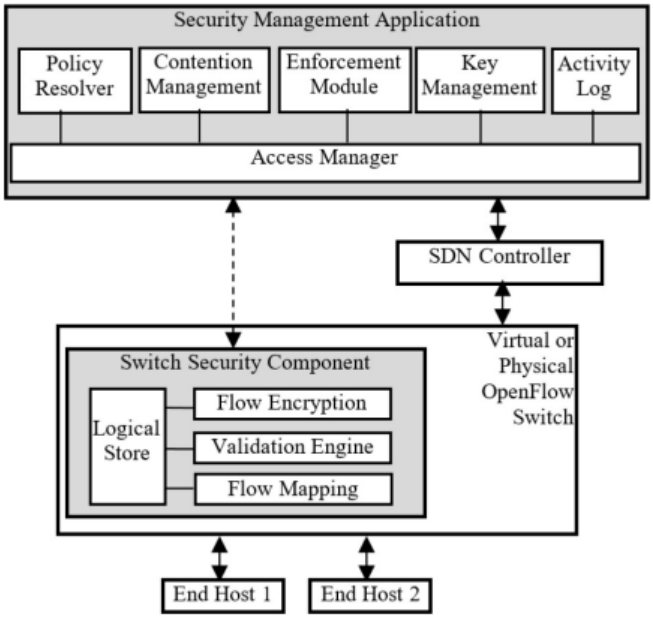

Figure 2: Structure of the Logical Security Architecture [2]

SMA allows network administrators to specify policies based on a number of parameters and conditions, such as time, location, event, type of traffic, source address and destination address. It also allows the triggering of poli-

cies during certain events, like increases or decreases in traffic load. The SMA is comprised of the Policy Resolver, Contention Management, Access Manager, Key Management, Enforcement Module and Activity Logs. The Access Manager serves as the central control unit for the SMA, Policy Resolver is used to determine the appropriate policies, Contention Manager detects potential conflicts with policies already in the switches, Enforcement Module determines the best mechanism for enforcing new policies, Key Management is used for generation of keys for secure communication, and Activity Log keeps track of all messages exchanged with the controller. Conflict resolution is performed in the SMA by checking whether a new policy conflicts with others being enforced in the switches and then applying the policy with a higher priority in the case of a conflict.

The SSCs constantly monitor network flow from hosts and drop malicious packets before they have to be processed by switches. The main components of the SSCs are the Flow Mapper (FM), Logical Store (LS), Validation Engine (VE) and Flow Encryption (FE). The FM creates reports mapping the relation between flows and the applications that generate them at the end hosts. The LS stores these reports, which are then validated in the VE component to identify malicious hosts. FE is used to secure communication between end hosts using symmetric key encryption. The SMA can communicate directly with the SSCs through XML. It holds the policies for all of the network devices within a given SDN domain and makes security decisions based on events in the network, as well as information regarding the hosts received from the SSCs.

Experiments have been performed using the ONOS SDN controller and Mininet. A simulated attack using malware in a virtual machine allowed attackers to alter the process list in the compromised VM. However, the attack was successfully detected, and the attacker could not modify the report generated by the Flow Mapper. Flow requests were subsequently dropped from the affected VM, and an alert was raised. A simulated topology poisoning attack was also detected in the Validation Engine at the level of the switches. A performance analysis found that the primary source of overhead is the policy enforcement at the level of the switches using the SSCs. The process-level state validation delay scales roughly in a linear fashion, with a delay of 0.052s for one VM and up to 0.596s for 10 VMs. The traffic validation latency scales from 0.08s at 100 rules to 0.813s at 2000 rules. A Mininet simulation running on an Oracle VM Box showed a 2-5% reduction in throughput when running SMA over the ONOS controller.

5. **Switch-Based Rule Verification** Switch-Based Rule Verification (SRV) [6] is a proposed flow rule verification mechanism aiming to mitigate the priority-passing attack's effects by identifying malicious packets. The system makes use of a HashMap table to store all flow rules defined by the controller for incoming packets. The HashMap uses a binary search tree structure for storage in order to enable $O(log(n))$ time retrieval of data, and flow rules are stored in buckets inside this structure.

   SRV determines if a packet is malicious by checking if the IP and MAC addresses of the packet's source and destination match the flow rule defined by the controller. IP and MAC addresses of the packet are retrieved via topology discovery and then concatenated. Afterwards, they are hashed using the SHA1 hash function, which enables fast hashing with a low risk of collision, and checked against flow rules in the bucket corresponding to the resulting hash key. In case of a mismatch on both addresses, the packet is blocked. Otherwise, if the packet matches, the defined rule is enforced. If only one of the addresses matches, the packet's priority is checked against the defined rule's priority, and if the defined rule's priority is higher, the packet is blocked. Otherwise, the rule is enforced.

4 Benefits and Limitations of Existing Solutions

4.1 Preacher

Benefits

- Preacher's probabilistic policy checking allows it to counter a wide range of attacks, including denial of service, injections, reroutes, man-in-the-middle attacks, and covert channel attacks. [9]
- It is lightweight, adding almost no overhead to networks where it has been implemented since it is highly parallelizable and uses an efficient data structure.
- Additionally, it takes the form of a separate component that communicates with the controller via OpenFlow and can therefore be implemented on a wide range of different network types, such as wide-area networks, ISP networks, and Clos networks or datacenters.

Drawbacks

- The probabilistic nature of Preacher is also one of its limitations, as the time required to detect attacks in networks with low packet rates and limited processing power can rise to over an hour, time during which a malicious actor could take advantage of vulnerabilities. This also means that low-volume injection attacks are more unlikely to be detected, especially if the sample rate is low.
- Preacher can deal with a large fraction of untrusted switches in a network. However, detection could be avoided if malicious switches were to modify traffic before it can be checked by trusted switches or if there are no trusted switches at all.
- Covert timing channel attacks can slip undetected in the scenario where there is no precise timestamping at the level of the controller.

4.2 Covert Channel Defender

Benefits

- The main advantage of the Covert Channel Defender is its real-time protection against covert channel attacks, an attack type that is difficult to detect and one that other state-of-the-art solutions based on VeriFlow can not consistently defend against. [8]
- Additionally, CCD can perform policy conflict resolution, a phase which is very lightweight and does not affect performance much relative to VeriFlow.

Drawbacks

- The computational complexity of the trie extension phase is high, at $O((mn)^x)$, where m is the depth of each dimension in the trie, and n is the number of matched trie leaves in each dimension. This is partly counteracted by the fact that only a limited number of fields from the flow rules need to traverse the trie.
- The process of locating Equivalence Classes in the trie is also slow relative to the actual conflict resolution and causes most of the overhead.

4.3 Policy Management and Enforcement System

Benefits

- The Policy Management and Enforcement System allows for dynamic policy instantiation of high-level security policies in order to combat DDoS attacks. [10]
- Policies can be instantiated based on changes in the network, as well as security alerts sent by hosts.
- PMES also heavily reduces jitter in legitimate traffic once the controller starts redirecting traffic, denoting its ability to maintain Quality of Service for legitimate traffic during a DDoS attack.
- Another advantage is the reduced need for network management since policies only need to be specified at a high level by the network administrator.

Drawbacks

- The Policy Management and Enforcement System is only designed for ISP networks and is therefore not suitable for smaller-scale networks.
- The tested version of PMES can not handle policy conflicts stemming from simultaneous policy enforcement for different customers.
- PMES does not ensure protection against covert channel attacks.

4.4 Logical Security Architecture

Benefits

- LSA is an effective framework for policy management and countering attacks in the SDN domain. [2] Its combined approach to using components for attack detection at the level of both the controller and switches yields markedly improved results compared to other proposed solutions from recent years.
- It continuously monitors flow from the end hosts at the level of the switches and dynamically modifies security policies across the entire SDN domain at the controller level.
- It can also deal with spoofing attacks, which are normally difficult to counter, directly at the source.
- Separating the policy decisions in the SMA and traffic monitoring in the SSC results in lower delays in flow application and minimal congestion.
- An extra layer of security is added by the fact that SSC Monitoring components placed at compromised hosts are not susceptible to attacks since they reside outside the host that they are monitoring.
- The modular SSC design can also be adapted to different switches, with the ability to use preferred modules in the switches for network monitoring, depending on the capabilities of the switch.

Drawbacks

- The current version of LSA can only be used in a single-domain SDN.
- Does not offer protection against all covert channel attacks.

4.5 Switch-Based Rule Verification

Benefits

- The main advantage of Switch-Based Rule Verification is its improved computational complexity when compared to the existing model for countering priority-passing attacks, boasting a $O(log(n))$ time complexity instead of $O(n)$.

Drawbacks

- However, SRV only offers policy conflict detection and blocking of malicious flow rules but no capacity for conflict resolution. This makes it less appealing when compared to other solutions, which have a much wider scope when it comes to attack prevention.

5 Responsible Research

This section deals with the ethical ramifications of the research. Since our project mainly focuses on outlining the advantages and disadvantages of solutions to SDN attacks, no experiments have been conducted as part of the research process. Therefore, ethical considerations such as voluntary participation, informed consent, confidentiality and anonymity do not apply to this research project. The only ethical concerns, in this case, are ensuring that the research has no potential to cause harm, that the work is free of plagiarism, that the results of the project are not biased and that the goals of the project are communicated clearly and in a transparent manner.

The aim of the research is to further understanding of security in Software Defined Networks, a fact that is clearly conveyed in the introduction to this paper. Additionally, the potential for harm and biased results is low, as care has been taken to provide a review of the state-of-the-art solutions that represents their advantages and disadvantages as objectively as possible. Sources have been cited for relevant passages in this paper, and all the relevant information has been taken from relevant and trustworthy literature obtained through IEEEXplore.

6 Discussion and Future Work

- Building on the solutions analysed in this paper, an incremental improvement on the state-of-the-art could be made by creating a hybrid system that incorporates the functionality of the Logical Security Architecture and that of Preacher. Both solutions are written in Java and are developed for use with the ONOS controller, which is among the best performing SDN controllers currently on the market and supports both proactive and reactive flow enforcement. Moreover, both solutions offer relatively low overhead when implemented in single-domain SDN. [9][2] LSA does not explicitly protect against covert channel attacks, and Preacher can normally only guard against covert storage channels. However, as long as there exists accurate timestamping of packets travelling through the network, Preacher can also counter covert timing channel attacks. LSA provides accurate logging functionality via its Activity Log module, meaning that detection of all covert channel attacks should be possible with this setup. Additionally, in LSA all packet headers are validated against the source addresses at the level of the SSC. Therefore, priority-passing attacks can be prevented since spoofing is easily detectable.
- With recent advances in machine learning, another potential way to advance the field of SDN security could lie in the usage of machine learning algorithms for the purpose of attack detection and policy enforcement. The idea of enhancing SDN security via machine learning has been proposed as early as 2018, in [11], in which an intrusion detection framework that relies on deep learning was proposed. This concept has the potential to be extended further into a framework that uses machine learning to adapt to the network structure and create flow rules that guide the traffic through the network as efficiently as possible and maximise QoS for all clients. However, this might only be feasible on small scale networks due to the high processing power requirements involved in training machine learning algorithms.

7 Conclusion

This paper has surveyed a number of attacks that exploit vulnerabilities related to policies and flow rules in the SDN domain. A summary of solutions targeting different policy-related vulnerabilities has been provided, along with the benefits and drawbacks of each solution. An incremental improvement on the state-of-the-art and a future research direction that could advance the state-of-the-art in the field of SDN have also been suggested.

Using a logically centralised controller means that the control plane's ability to enforce appropriate policies affects the security and performance of the entire network. Therefore, using attack mitigation tools and policy management frameworks with comprehensive coverage and low overhead is necessary. The solutions analysed in this paper aim to defend against common attack types such as DoS, injection, and man-in-the-middle but also attacks that are hard to detect and specifically target the system's policies, namely the covert channel attack and the priority-passing attack. In conclusion, further research into the field of policy management is essential towards improving SDN security. This can either be done through iterative improvement on state-of-the-art solutions or new research directions such as the use of machine learning in policy enforcement and flow rule management.

References

[1] P. E. Verissimo C. E. Rothenberg S. Azodolmolkys D. Kreutz, F. M. Ramos and S. Uhlig. Software-defined networking: A comprehensive survey. *Proc. IEEE*, vol. 103(no. 1):14–76, Jan. 2015.

[2] V. Varadharajan and U. Tupakula. Counteracting Attacks From Malicious End Hosts in Software Defined Networks. *IEEE Transactions On Network And Service Management*, vol. 17(no. 1):160–175, March 2020.

[3] S. Natarajan S. Scott-Hayward and S. Sezer. A Survey of Security in Software Defined Networks. *IEEE Communication Surveys and Tutorials*, vol. 18(no. 1):623–630, First Quarter 2016.

[4] Y. Xin S. K. Jagatheesaperumal M. Ayyash M. Shaheed M. Rahouti, K. Xiong. SDN Security Review: Threat Taxonomy, Implications, and Open Challenges. *unpublished*, pages 15–27, 2022.

[5] S. Scott-Hayward H. Song M. Winandy A. Danping, M. Pourzandi and D. Zhang. Threat Analysis for the SDN Architecture. pages 6–19, 07 2016.

[6] P. Swain R. Kumar, S. Sahoo. An Improved Flow Rule Verification Against the Priority-passing attack in SDN. *IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security*, vol. 14(no. 4):1–6, 2020.

[7] C. E. Brodley S. Cabuk and C. Shields. IP Covert Timing Channels: Design and Detection. *Proceedings of the 11th ACM Conference on Computer and Communications Security*, page 178–187, 2004.

[8] P. P. C. Lee M. Xu Q. Li, Y. Chen and K. Ren. Security Policy Violations in SDN Data Plane. *IEEE/ACM Transactions on Networking*, vol. 26(no. 4):1715–1725, August 2018.

[9] L. Schiff K. Thimmaraju and S. Schmid. Preacher: Network Policy Checker for Adversarial Environments. *IEEE/ACM Transactions on Networking*, vol. 29(no. 5):2087–2099, October 2021.

[10] Z. Zhang R. S. G. Blanc and K. T. H. Debar. Adaptive Policy-driven Attack Mitigation in SDN. *XDOMO'17: Proceedings of the 1st International Workshop on Security and Dependability of Multi-Domain Infrastructures*, (no. 4):1–6, April 2017.

[11] S. Shahristani A. Dawoud and C. Raun. A Deep Learning Framework to Enhance Software Defined Networks Security. *32nd International Conference on Advanced Information Networking and Applications Workshops*, pages 709–713, 2018.