2023-01-21

SECAP Switch—Defeating Topology Poisoning Attacks Using P4 Data Planes

Dylan Smyth
Department of Computer Science, Munster Technological University, Cork, Ireland, dylan.smyth@mtu.ie

Sandra Scott-Hayward
Centre for Secure Information Technologies, Queen's University Belfast, Belfast , Northern Ireland

Victor Cionca
Department of Computer Science, Munster Technological University, Cork, Ireland, victor.cionca@mtu.ie

Sean McSweeney
Department of Computer Science, Munster Technological University, Cork, Ireland, sean.mcsweeney@mtu.ie

Donna O'Shea
Department of Computer Science, Munster Technological University, Cork, Ireland, Donna.OShea@mtu.ie

Recommended Citation

Smyth, D., Scott-Hayward, S., Cionca, V. et al. SECAP Switch—Defeating Topology Poisoning Attacks Using P4 Data Planes. J Netw Syst Manage 31, 28 (2023). https://doi.org/10.1007/s10922-022-09714-z

Journal of Network and Systems Management (2023) 31:28
https://doi.org/10.1007/s10922-022-09714-z

SECAP Switch—Defeating Topology Poisoning Attacks Using P4 Data Planes

Dylan Smyth[1] · Sandra Scott-Hayward[2] · Victor Cionca[1] · Sean McSweeney[1] · Donna O'Shea[1]

Received: 30 August 2022 / Revised: 11 November 2022 / Accepted: 20 December 2022 /
Published online: 21 January 2023

Abstract

Programmable networking is evolving from programmable control plane solutions such as OpenFlow-based software-defined networking (SDN) to programmable data planes such as P4-based SDN. To support the functionality of the SDN, the correct view of the network topology is required. However, multiple attacks aimed at topology poisoning have been demonstrated in SDNs. While several controller-centralised security solutions have been proposed to defeat topology poisoning attacks, some attacks e.g., the Data Plane ARP Cache Poisoning Attack and the relay-type Link Fabrication Attack are difficult to detect using a fully centralised security solution. In this paper, we present the Security-Aware Programmable (SECAP) Switch—a lightweight, in-network, P4-based security solution that is designed to prevent attacks that might otherwise evade control plane solutions. The SECAP switch verifies source address details contained within the headers of protocols commonly used to perform topology poisoning attacks. This function is supported by a novel variance-based anomaly detection solution to provide a layered defence. We demonstrate the ability of the SECAP switch to defeat topology poisoning attacks with minimal memory and processing overhead.

Keywords Software-defined networking · Programmable data planes · p4 · Security · Topology poisoning

1 Introduction

Software-defined network (SDN) security solutions have largely followed the SDN principle of detection through logically centralized control and protection through control plane applications and extensions. These centralised security solutions have

✉ Dylan Smyth
dylan.smyth@mtu.ie

Extended author information available on the last page of the article

been proven to be effective at defeating various threats [1, 2]. However, in addition to traditional network attacks, SDN specific attacks exist which are either capable of avoiding centralised security controls, or are simply difficult to detect from the control plane. The Data Plane ARP Cache Poisoning Attack (DACP) [3], and the relay-type Link Fabrication Attack (LFA) [1] are examples of such attacks. Both of these topology poisoning attacks take advantage of expected network behaviour, and do not present obvious indicators that an attack is taking place. Currently, detecting and defeating these attacks can only be done by implementing techniques that impact the performance of the network. As such, an alternative defence mechanism is required.

Programmable switches are a recent advancement in SDN technology. These devices provide network developers with a level of forwarding pipeline control previously unavailable in SDN switches. Programming Protocol-independent Packet Processors, or P4, is modern programmable switch technology that has presented the opportunity for SDN data planes to be actively involved in forwarding decisions [4]. In-network processing and decision-making offers interesting opportunities for the security of SDNs, particularly with regard to attacks which are difficult to detect with a solely centralised security solution. While attacks, such as the DACP, can potentially avoid interacting with the control plane, they cannot avoid interaction with the network's forwarding devices. P4 provides network developers with the ability to define the structure of rule tables, specify the packet processing pipeline, and include limited logic within the forwarding devices. Of particular importance for security functions, stateful behaviour is supported through the use of registers, which can be read and written to as traffic is forwarded. Recent work has exploited this to enable switches to perform stateful security operations such as flooding attack detection [5] and traffic encryption [6]. However, P4 has not previously been used to develop in-network security solutions targeting the DACP or the relay-type LFA.

In this paper, we present the Security-Aware Programmable (SECAP) Switch—a lightweight, in-network, P4-based security solution that is designed to prevent attacks that might otherwise evade control plane solutions. Numerous topology poisoning attacks involve address spoofing, including the ARP Cache Poisoning Attack, the Host Location Hijacking Attack [1], and the Persona Hijacking Attack [7]. This solution addresses this through two main features; source address verification and in-network anomaly detection. This work focuses on two specific attacks for an in-depth analysis of this approach, the DACP and relay-type LFA. Hence, the contributions of the paper are as follows:

- We present SECAP, a novel, lightweight, in-network security solution capable of defeating topology poisoning attacks with low overhead and small memory footprint.
- We demonstrate the implementation of SECAP in P4 and provide an in-depth analysis of the solution through two stealthy topology poisoning attacks.
- We investigate the ability for P4 switches to independently detect the relay-type LFA using traffic characteristics, and analyse the effectiveness of this solution.
- We provide an evaluation of the cost of the in-network security solution in terms of bandwidth, latency, and memory and processing footprint.

The rest of the paper is structured as follows: Sect. 2 presents a threat model and description of the DACP and relay-type LFA, along with an overview of existing defences and their limitations. Section 3 presents the SECAP Switch - a P4-based defence against topology poisoning attacks. Section 4 describes the testbed and test methodology used to evaluate the SECAP switch with the results of the evaluation presented in Sect. 5. The results are discussed in Sect. 6 and related work is presented in Sect. 7. Finally, Sect. 8 concludes the paper.

2 Threat Model and Attacks

As previously described, topology poisoning attacks pose a particular threat to the correct functioning of networks, from terrestrial networks [1] to space-air-ground-integrated networks, as discussed in [8]. In this section, we present our threat model and introduce two specific topology poisoning attacks, DACP [3], and the relay-type LFA [1] that are explored in this work.

2.1 Threat Model

This work considers an attacker who has direct access to the SDN data plane, either through their own device or a compromised machine. The goal of the attacker is to poison hosts and/or the controller's view of the network topology in order to intercept network traffic. In the case of the relay-type LFA, the attacker is considered to have access to two machines in the network and has created a General Routing Encapsulation (GRE) tunnel between them for the purpose of the attack. This scenario was shown to be possible in work presented in [9].

2.2 The Data Plane ARP Cache Poisoning Attack

2.2.1 The Attack

The DACP attack allows an attacker to poison the ARP cache of a host through address spoofing and protocol header manipulation. The steps of this attack are outlined in Fig. 1. In summary, the attacker first sends a genuine ARP reply to the victim in order to establish a flow in the network. This is then followed by a malicious ARP reply with modified header values, which piggybacks on the previously established flow. The result is that the victim's ARP cache is poisoned. What differentiates this attack from a conventional ARP cache poisoning attack is that it piggybacks the malicious ARP reply over an established flow. By using an existing flow to forward the malicious traffic, the attacker is able to avoid interaction with the controller during the attack, preventing the controller from observing the malicious ARP messages and having its own data stores poisoned. This attack was previously demonstrated against the Floodlight [10] controller in [3] and extended in the same work to develop a stealthy port scanning technique and Denial-of-Service (DoS) attack.

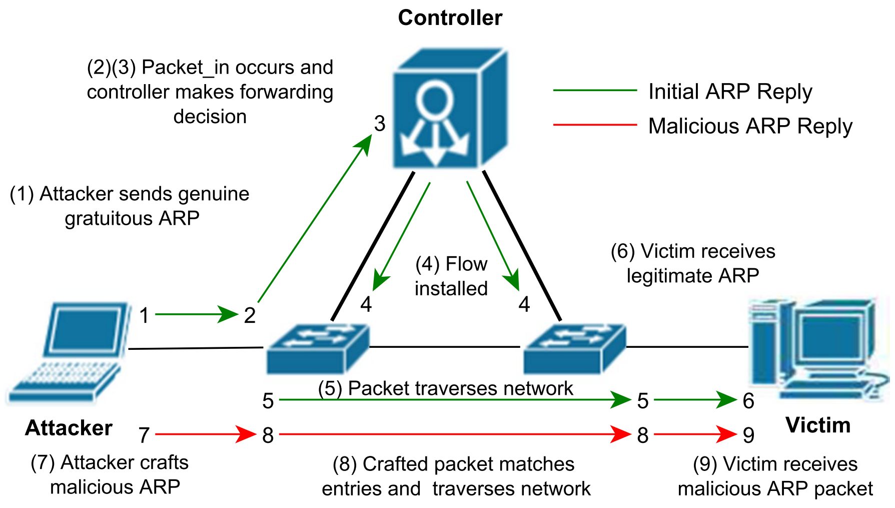

Controller
(2)(3) Packet_in occurs and controller makes forwarding decision
Initial ARP Reply
Malicious ARP Reply
(1) Attacker sends genuine gratuitous ARP
(4) Flow installed
(6) Victim receives legitimate ARP
(5) Packet traverses network
Attacker
Victim
(7) Attacker crafts malicious ARP
(8) Crafted packet matches entries and traverses network
(9) Victim receives malicious ARP packet
1
2
3
4
5
6
7
8
9

Fig. 1 Data plane ARP cache poisoning

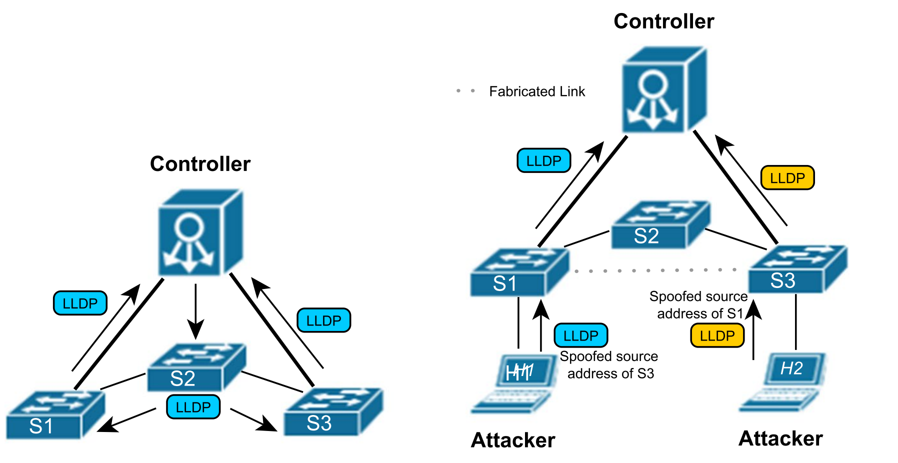
2.2.2 Existing Solutions and Limitations

Security solutions such as Sphinx [2] and Topoguard [1] have been proposed to prevent poisoning attacks such as ARP cache poisoning. However, they rely on observing the traffic related to the attack, which is what DACP aims to avoid. For these solutions to be able to detect the DACP, all ARP traffic must be mirrored and observed by the control plane, introducing more control channel overhead. Moreover, forcing all traffic to be mirrored to the control plane introduces an avenue for a control plane saturation attack [11] or similar DoS attack to be carried out. The DACP relies on piggybacking traffic, an action which can be prevented, or at least greatly restricted, by using distinct flow rules implemented through specific match conditions (i.e. matching on more header values than needed for forwarding). However, these additional match conditions increase the memory requirement per flow rule, and hinder effective aggregation of flow rules in any effort made to minimize memory usage. So, while it is possible to detect and prevent the DACP, it must be done at the cost of control channel overhead, increased controller vulnerability, and increased data plane memory consumption.

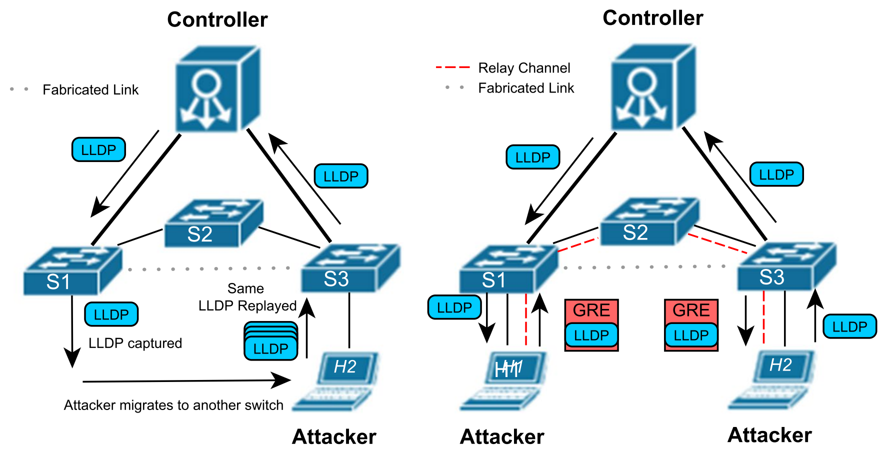
2.3 The Relay-Type Link Fabrication Attack

2.3.1 The Attack

The Link Fabrication Attack (LFA) is a topology poisoning attack that aims to modify the controller's view of the topology by fabricating the existence of network links [1]. This is done through manipulation of the Link Layer Discovery Protocol (LLDP) messages used by the target controller to discover the network topology. The expected LLDP operation is illustrated in Fig. 2a. In this example, an LLDP



(a) Expected LLDP Operation (b) LLDP Generation LFA



(c) LLDP Replay LFA (d) Relay-type LFA

Fig. 2 LLDP link discovery threats

message is sent from the SDN controller to S2, S2 forwards LLDP messages to its connected switches, S1 and S3, and S1 and S3 forward the LLDP message from S2 back to the controller. With this process, the controller detects the link S2-S1 and S2-S3. The LFA can be carried out in three ways; through LLDP generation (shown in Fig. 2b with attackers generating LLDP packets with spoofed source addresses to create the fabricated link), LLDP replaying (shown in Fig. 2c with an attacker replaying a captured LLDP packet to create the fabricated link), and LLDP relaying (shown in Fig. 2d). As effective solutions already exist for the generation and replay attacks, this work will focus on the relay-type attack.

In the relay-type attack, depicted in Fig. 2d, an attacker will relay the LLDP messages from one network attachment point to another without modifying the message.

An example scenario for this attack is where an attacker has physical access to two network attachment points through two separate devices and fabricates a link between these attachment points. The attacker can, through an out-of-band connection or an in-band tunnel, relay LLDP messages and establish a working link between those two points, allowing network traffic to flow and placing themselves in a Machine-in-the-Middle (MitM) position. A study of various attack scenarios involving out-of-band connections is provided in [12].

2.3.2 Existing Solutions and Limitations

Previous work has solved the issue of LLDP generation and replaying by including single-use tokens in LLDP messages, limiting the period for which those messages are valid [13]. However, this approach does not resolve the relay-type attack as the LLDP message is forwarded immediately and without modification. Alternative solutions include detection of the attack by analysing link characteristics and using anomaly detection to verify newly connected links [12]. The analysis of link characteristics approach requires collection of link characteristics over a period of time determined by the number of samples that needs to be collected. To support accurate collection of the link characteristics, no traffic is permitted to use the link during the data collection period. If traffic is prevented from flowing through the new link before verification, then a legitimate link will incur a set-up delay equal to the time taken for the verification process to verify the link. However, if traffic is allowed to flow, then the attacker can take advantage of this period of vulnerability. While this solution prevents the attack impacting legitimate network traffic, it does nothing to proactively prevent the attack from occurring in the first place. Furthermore, the success of a detection solution using anomaly detection is not guaranteed as a resourceful attacker can attempt to reduce the footprint of their attack by utilising a fast in-band or out-of-band relay channel and, in turn, reduce the observable difference between the fabricated link and a legitimate link. A multi-faceted proactive and reactive defence would be a better means of defeating this attack.

3 SECAP Design

As detailed in Sect. 2, while control plane security solutions exist that can potentially detect and prevent the DACP and the relay-type LFA, a trade-off is introduced, where successful detection and prevention can only be achieved at a performance and security cost. Programmable data planes introduce the ability to define logic in the data plane forwarding devices, which can be executed in response to particular events or observed traffic. A security solution implemented using this logic cannot be evaded by an attacker simply due to its position within the network forwarding devices. Existing solutions, such as [14], already use this logic to provide next generation firewalling functionality beyond what can be implemented through flow rules.

The SECAP Switch provides an in-network P4-based defence through two main functions; source address verification and anomaly detection. The source address verification function consists of three components; MAC and IP verification, ARP Verification, and LLDP Address Verification. The anomaly detection function is implemented through a single component. Three operating modes are provided as a means of supporting deployment in different network environments.

An outline of the flow through the SECAP switch pipeline is presented in Fig. 3. Packets entering a switch port are first processed by the parser, a state machine used to extract protocol header details from the received packet. The SECAP security functionality logic is then applied. The application of components in the pipeline shown in Fig. 3 depends on the protocol headers that have been parsed and are controlled through conditional checks. Ethernet frames are always subjected to MAC verification. If an IP header is present, then IP verification will also be performed. Similarly, either ARP or LLDP verification will be applied if the relevant header is present. This design allows for the verification performed by one component to support the operation of another later in the pipeline. If the packet passes the checks imposed through the components, it will continue to the forwarding tables where *match+action* rules will be applied. If the packet fails any of the checks, it will be dropped before reaching the forwarding tables. Much of the data learned by the switch during operation is stored in P4 register arrays. The switch port number serves as an index for each register array, allowing the P4 registers to be used in parallel without needing to maintain an additional mapping of port number to array index.

3.1 Source Address Verification

Source address verification is the first major function of the SECAP Switch. This function performs source address verification for the Ethernet, IP, ARP, and LLDP

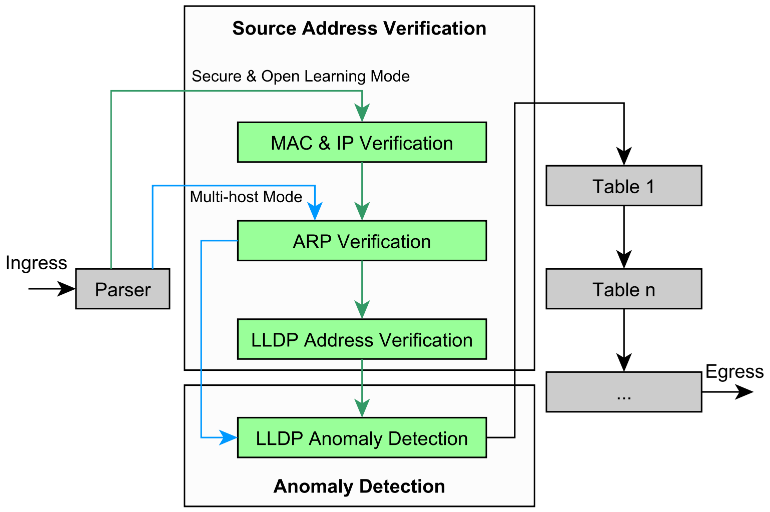


Fig. 3 Overview of flow through SECAP switch P4 pipeline

protocols. The goal of this function is to prevent an attacker from spoofing their source address and performing topology poisoning attacks.

3.1.1 Ethernet and IP (MAC/IP) Verification

The purpose of this component is to capture the MAC and IP address of each host attached to a switch to support address verification. This functions in a similar way to port-based access control in conventional networks. In SECAP, each switch port is associated with a pair of P4 registers. One register stores the attached host's MAC address and the other stores the attached host's IP address. Once an address has been learned and stored, the port will become locked to that address. The addresses in subsequent traffic will be compared against the stored addresses. The relevant traffic will be dropped if the addresses do not match. No action is taken if the addresses do match, and the traffic is allowed to continue through the switch pipeline.

3.1.2 ARP Verification

This component verifies the content of the ARP header. This verification prevents an attacker from sending the malicious ARP reply associated with the DACP. Upon reaching this component, the MAC address of the Ethernet header will have been verified by the MAC/IP address verification check. This MAC address is then compared to the MAC address found within the hardware address field of the ARP header to ensure that there is no discrepancy. Furthermore, if an IP address is available for this port, the stored IP address will be compared with the source IP address field of the ARP header. The ARP message will be dropped if either of these checks fail.

3.1.3 LLDP Address Verification

This component prevents an attacker from modifying the source address of relayed LLDP frames by comparing the source MAC address in the Ethernet header with the Chassis ID field in the LLDP header. If they are the same, then the LLDP message will continue to the next stage. If they do not match, the LLDP message will be dropped. This stage of verification works alongside the MAC/IP verification component and existing LLDP message integrity verification solutions such as [13]. The MAC/IP verification component will lock the switch port to which the attacker is connected to their MAC and IP address. As such, the attacker will be unable to directly relay an LLDP message, as the source MAC of that LLDP message will differ from their own source MAC address. The attacker will also be unable to modify the source MAC address of the LLDP message as this source MAC will be compared to the Chassis ID in the LLDP header. Finally, the attacker will be unable to modify both the source MAC address and the Chassis ID as this would then cause the LLDP message to be rejected by the controller due to existing token-based solutions.

To provide an example of the operation of this component, consider the sample LLDP message presented in Fig. 4. This LLDP message contains the

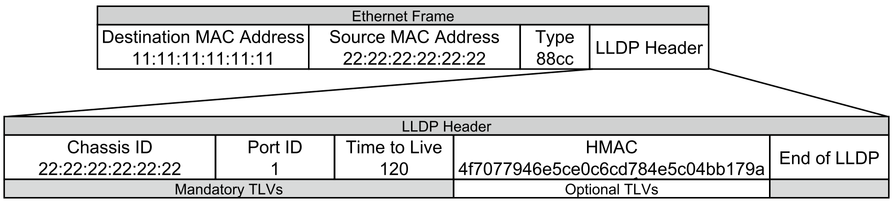


Fig. 4 Overview of LLDP header and relevant type-length-value (TLV) fields

mandatory type-length-value (TLV) fields, as well as an optional TLV containing a hash-based message authentication code (HMAC), typical of what would be found in a token-based LFA prevention solution. The HMAC is an MD5 hash of the Chassis ID, Port ID, and a secret key. This hash is calculated by the controller when the LLDP message is constructed. The controller can verify the contents of the message by recalculating the hash for a given message and comparing the calculated hash with the hash contained within the message. In order for an attacker to pass the source address verification stage of the SECAP solution, they would need to modify the source MAC address of the Ethernet header to match their own MAC address. However, in order to pass the LLDP address verification stage, the attacker would also need to modify the Chassis ID in the LLDP header to also match their own MAC address. Upon doing this, the LLDP message will successfully pass the SECAP checks but will subsequently be rejected by the controller as the HMAC will not calculate correctly due to the modification of the LLDP header.

There is, however, a scenario where an attacker may be able to bypass this component by spoofing their MAC address to match that of the target network switch upon initial connection to the network. The SECAP anomaly detection component is designed to defeat this scenario (see Sect. 3.2).

3.1.4 Address Verification Implementation

As previously described, learned addresses used for the source address verification function are stored in P4 registers. Each switch port maintains a register for a learned MAC address and IP address. When the Ethernet or IP header is parsed, the address mapped to the relevant switch port is read from the register into a temporary variable and compared against the address in the parsed protocol header. ARP header verification does not need any additional storage as it compares addresses within the Ethernet header and ARP header. Similarly, the LLDP address verification component compares addresses found in the parsed protocol header and, as such, does not need any additional stateful storage. A register is used to store a *violation* flag that is used to mark when an address verification check has failed. This flag is a temporary variable created when a packet enters the pipeline, and should a check fail, the flag will be set, marking the packet to be dropped.

3.2 Anomaly Detection

While the source address verification function is enough to defeat topology poisoning attacks in the majority of scenarios, the relay-type LFA could still be performed by a determined and resourceful attacker. An attacker could, before establishing themselves in the network, spoof the address of the switch from which they wish to relay LLDP traffic. By doing this, the attacker can forward LLDP messages to the switch they are connected to, without needing to modify the contents of the message. In this scenario, source address verification checks will be satisfied and will not stop the attack. As such, the SECAP Switch solution has another function, which can detect the relay-type LFA through anomalies in link characteristics. There are two items required for this component:

- A measurable link characteristic.
- A method of identifying a fabricated link using the chosen link characteristic.

In the following sub-sections, we describe the process used to identify a measurable link characteristic, the detection of a fabricated link using the LLDP interval variance as that characteristic, and implementation of the anomaly detection solution.

3.2.1 Identifying a Link Characteristic

Based on the findings of previous work in which it was shown that the relay channel and LLDP forwarding method used during the relay-type LFA can affect the characteristics of the resulting fabricated link [12], we hypothesize that characteristics for a fabricated link will be different to the characteristics of a legitimate network link. The link characteristic used in [12] is latency, measured by comparing the time an LLDP message left the controller to the time that LLDP message returned. In the programmable data plane, this global network view is not available to the individual P4 switch. As such, while a P4 switch can observe LLDP messages during the forwarding process, it cannot determine when an LLDP message left its origin and, therefore, the centralised, controller-based link latency calculation cannot be directly applied in the programmable data plane.

To this end, an investigation was carried out to establish which characteristics could be measured accurately and consistently by an individual P4 switch. An attempt was made to estimate link latency by recording the time at which an LLDP message arrived from the controller (start of an LLDP round), recording the time an LLDP message arrived from a neighbouring switch, and calculating the difference between these times. The calculated value is an estimated latency, where the time at which an LLDP message was received from a neighbouring switch is subtracted from the time that neighbouring switch *may* have received the LLDP message from the controller. In practice, this was found to be unreliable as the controller does not distribute LLDP messages to switches at the exact same time. This leads to race conditions where a switch may have already received an LLDP message from a neighbouring switch before it receives an LLDP message from the controller and records the received time.

For example, take a topology with two switches; S1 and S2. Consider L_c to be the LLDP message sent from the controller, L_s to be the LLDP message sent from a neighbouring switch, T_c to be the timestamp at which L_c arrives, and T_s to be the timestamp at which L_s arrives. S1 could estimate the latency of the link S1–S2 by calculating $S1(T_s) - S1(T_c)$. S2 can estimate the link latency by performing the same calculation. However, this will only produce the same estimated latency if both switches have the same value of T_c, as then both switches will receive L_s at a similar time, resulting in a similar value for T_s. In practice, this is not the case. Both switches receive L_c at slightly different times, and so the value of T_c on both switches is different. In the case where S1 receives L_c after S2 (S1(T_c) > S2(T_c)), then the latency calculated by S1 will be lower than the latency calculated by S2, as there will be a smaller difference between S1(T_s) and S1(T_c). In the next LLDP round, S1 could receive L_c first, resulting in S1 detecting a higher latency and S2 detecting a lower latency, resulting in the aforementioned race conditions.

As illustrated in this example, a switch does not have the ability to accurately calculate the latency over a given link by itself, hence it was necessary to investigate alternative characteristics, which could be used to detect a fabricated link. For this, the LLDP interval was considered.

Analysing LLDP interval as a measurable link characteristic: The inter-arrival time, or interval, of LLDP messages is the rate at which they arrive at a switch port after being emitted by the controller at a configured frequency. For the open network operating system (ONOS) SDN controller [15], the controller used in the evaluation of this work, the LLDP frequency is three seconds by default. The rate at which a switch receives LLDP messages from a neighbouring switch will be slightly lower than the rate at which the messages are sent from the controller, as it will take time for the message to be processed by the switch, travel over the link, and reach the neighbouring switch port (i.e. for a network with ONOS, the LLDP interval will be greater than three seconds). The characteristics of the network link (e.g. latency) is one of the factors which influence the exact interval of the messages. Given that previous work in this area in [12] identified that the latency of a fabricated link can be different to that of a legitimate link, it should also be possible to identify a fabricated link using LLDP interval.

An analysis was performed to investigate whether a difference in LLDP interval could be observed between a legitimate link and fabricated link. This analysis used the topology previously shown in Fig. 2d. A link was fabricated between Switch 1:Port 1 (S1:P1) and Switch 3:Port 3 (S3:P3) by relaying messages through a GRE tunnel within the network. This scenario represents a worst case for defence as the in-band relay channel matches the speed and reliability of the target network, giving the attacker the advantage of minimal LLDP interval difference. Timestamps, in the form of microseconds since the switch was launched, were logged at the switch each time an LLDP message arrived at a switch port and these timestamps were captured and collected by an external script. The LLDP intervals were obtained by calculating the difference between consecutive timestamps. LLDP intervals were logged for three runs, with each run capturing one thousand intervals. A mean LLDP interval was calculated for each timestamp across the three runs, creating a set of one thousand LLDP intervals to be used during analysis.

LLDP Interval Distribution: The plots in Fig. 2 show the distribution of LLDP intervals observed at the switch ports. No significant difference could be observed between the intervals of the messages for a legitimate link and a fabricated one when comparing the mean interval. However, the distribution of LLDP intervals for each port on each switch show a slight but distinct difference in the distribution for the ports associated with the fabricated link (S1:P1 to S3:P3). This can be seen in Fig. 5a and c. While the intervals for genuine network links mostly lie in the centre of the distribution, the distribution for the fabricated link has a greater number of values sitting to the left and right tails. Although this difference is not visible when observing the mean interval, it is visible when the variation of a population of captured samples is calculated.

LLDP Interval Variance: In the absence of a mathematical library and loop operations in P4, a variance calculation is implemented in SECAP based on the standard deviation. The variation calculated for each switch port at each switch after a number of samples are captured is presented in Fig. 6. It can be observed that as the number of samples collected increases, a distinct difference in the variance becomes apparent. For Switch 1 and Switch 3, shown in Fig. 6a and c, there is a clear difference in variance for the genuine network link and the fabricated link. For Switch 2, shown in Fig. 6b, while there is a visual difference in the variation of the legitimate links, the actual difference in variance never falls below $1e9\ \mu s^2$, while the difference in variance for links at the other switches is consistently greater than $1e9\ \mu s^2$.

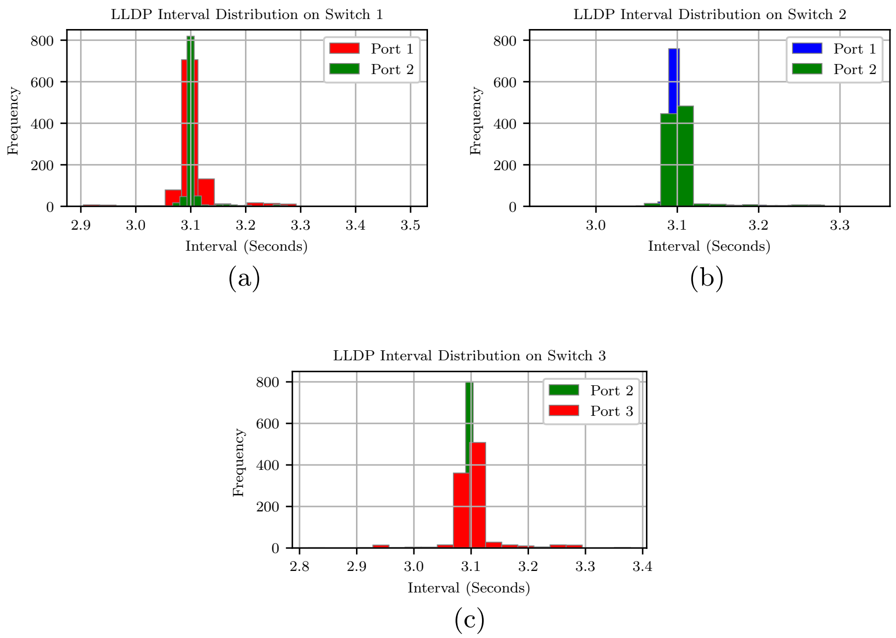


Fig. 5 LLDP interval distributions (seconds)

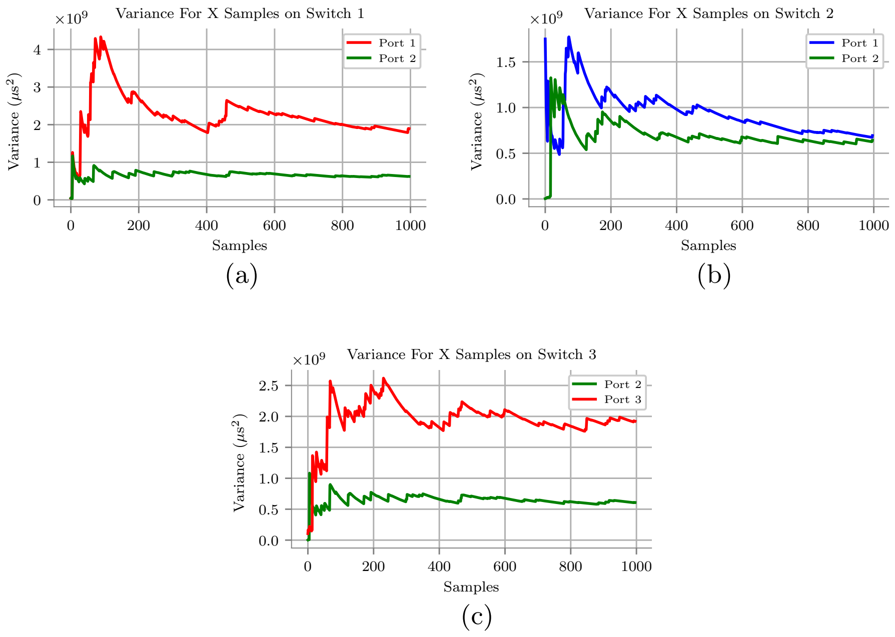


Fig. 6 LLDP interval variance comparison for each switch

3.2.2 Detecting a Fabricated Link

We base our proposed fabricated link detection method on this observed difference in variance. When a new link appears in the network, LLDP interval samples are collected. During this time, no other traffic is forwarded over the new link by the switch, preventing network traffic conditions from influencing the collected samples. The time during which samples are collected for testing is known as the *vetting period*. There are several variables to be configured for the vetting period; the number of samples to collect, S, the baseline LLDP interval variance of legitimate links, B, and the tolerance value, TV. The configuration of these variables are explored in Sect. 5. Once S LLDP interval samples have been collected, their variance, V, is calculated. The percentage difference, D, between V and B is then calculated. If D is within the specified tolerance, TV, then the link is determined to be legitimate and is permitted for use. Otherwise, the link will not be allowed and the vetting period for that candidate link will start again.

3.2.3 Anomaly Detection Implementation

Our prototype implementation of the anomaly detection solution is subject to the constraints of P4. At the time of writing, P4 does not support loop operations, and as such a step of the variance calculation must be repeated for each captured sample. We use a register array to store the observed LLDP intervals. The size of the array is equal to the number of samples to be captured, and each switch port is associated with its own array. The LLDP interval is calculated by reading the timestamp when an LLDP message enters the switch pipeline, *TS*, and comparing this against the previous timestamp obtained when an LLDP message entered the pipeline for this port, TS_{-1}. This does not occur for the very first LLDP message that enters the pipeline as there is no previous timestamp against which to compare. A register maintains the value of TS_{-1}, and this register is updated after the value is used to calculate an interval. TS_{-1} is read into a temporary variable when used for the interval calculation. Each switch port has a flag associated with it, *Pvf*, signalling whether or not the vetting period has been completed for that port. This flag is read into a temporary variable when a packet enters the pipeline. When *Pvf* is active, only LLDP traffic is forwarded over the link. Each port also maintains an LLDP counter, *Plc*, and a running total for the intervals, *Pit*, used to calculate the mean interval for that port. This mean is then used during the calculation of the variance. *Plc* and *Pit* are stored using P4 registers, and use temporary variables to hold their values (Table 1).

Further discussion regarding the implementation of the anomaly detection component is provided in Sect. 6.

3.3 Operating Modes

A range of operating modes enables SECAP compatibility with different network environments. The operating modes are used to define which components are active and how a switch learns host addresses. Moreover, the operating modes define how to reset or unlock the addresses to which a switch port has been locked. The SECAP Switch supports three different operating modes; *Secure Learning Mode*, *Open Learning Mode*, and *Multi-host Mode*.

- Open Learning Mode: All components are active and the switch learns host addresses by observing traffic. This mode is most suited to network environments where static IP addressing is used, each switch port is occupied by a single host, and hosts do not frequently change their attachment point. Switch ports are locked to a host's address upon receiving traffic from an attached host and can

Table 1 Anomaly Detection Variables

Variable	Description
TS	Latest LLDP timestamp
TS_{-1}	Previous LLDP timestamp
B	Baseline variance
S	Number of samples to collect
V	New link variance
D	Percentage difference
TV	Tolerance value
Pvf	Port vetting flag
Plc	Port LLDP count
Pit	Port interval total

be unlocked by a control plane device resetting the P4 registers used to store the host's address.

- Secure Learning Mode: All components are active and host addresses are learned through Dynamic Host Configuration Protocol (DHCP) offers received from the control channel. This mode is suited to dynamic network environments where each switch port is occupied by a single host. DHCP messages are allowed to pass without address verification taking place, and the addresses associated with a switch port can be reset or changed through DHCP offers. Switch ports can also be unlocked by the control plane, as with Open Learning Mode. In this mode, static addresses can be supported through static DHCP entries or manual P4 register updates.
- Multi-host Mode: In this mode, the source address verification component is not active. Switch ports are not locked to an address. The ARP header verification and in-switch anomaly detection components are active. This mode is suited to environments where several end hosts share a switch port, such as virtualised environments. Multi-host mode offers more flexibility at the cost of security, as the ARP header verification component will be unable to verify the source IP address in the ARP header.

The operating modes and their implementation is further discussed in Sect. 6.

4 Test Methodology

This section describes the testbed and the methodology for evaluation of the SECAP switch.

4.1 Testbed

The topology of the testbed (3 switches, 3 hosts) is shown in Fig. 7. The testbed was deployed in an Ubuntu Virtual Machine (VM) with access to 4 processor cores and 16 GB of RAM. The host machine had an Intel i7-7700 CPU running at 3.60 GHz and with 32 GB of DDR4 RAM. Table 2 provides a description of the inter-switch connections. This testbed consisted of three Stratum [16] software P4 switches with each using an instance of BMv2 [17] to perform packet forwarding. Each switch was deployed in a separate Docker container [18] and an overlay network was created to link each connected switch port, and ensure each inter-switch connection was isolated from the underlying Docker network. Additional Docker containers were added to the network to act as hosts. For the single switch tests, an additional host was connected to S1. The ONOS [19] controller version 2.3 was used as the network controller which ran on the host machine. The SECAP Switch solution was added to ONOS and configured as an application to be distributed to the testbed switches upon their connection.

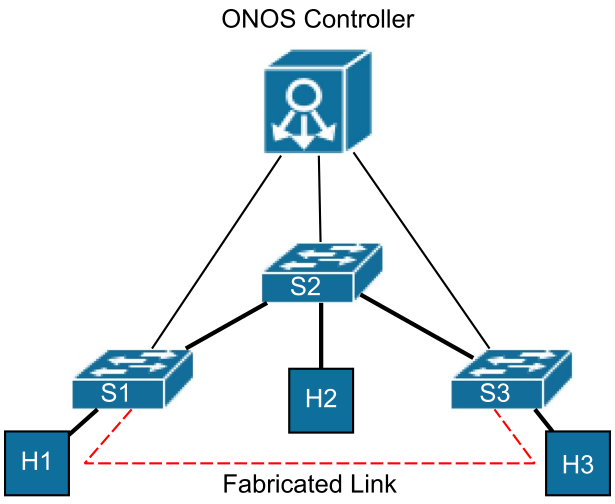


Fig. 7 Testbed topology

Table 2 Testbed network port connections

	Port 1	Port 2	Port 3
Switch 1	Host 1	Switch 2	–
Switch 2	Switch 1	Host 2	Switch 3
Switch 3	–	Switch 2	Host 3

4.1.1 Source Address Verification

Tests were carried out to ensure that the MAC/IP and ARP verification functioned as expected and to understand the bandwidth and latency costs associated with these components. The switch was configured to operate in *Open Learning Mode* for these tests.

Once the test network was established, each host first sent an Internet Control Message Protocol (ICMP) request to each other host in the network. This was done to ensure connectivity between the hosts and to lock each switch port to the MAC and IP address of the hosts connected to them. The Python Scapy library [20] was then used to craft packets with spoofed source addresses and modified ARP headers for these tests. A series of tests was then carried out where packets were generated with spoofed source MAC and IP address details and sent into the network. ARP Replies with a modified ARP header, matching those that would be used for the DACP attack, were generated and forwarded into the network.

To examine the impact of the switch's source address verification, Ping, IPerf, and ARPing were used to gather information on ICMP and ARP Round-Trip Time (RTT), as well as the available bandwidth of the link through the network. This data was gathered for traffic travelling through a single switch, two switches, and three switches. Three-hundred samples for the ICMP and ARP RTT were gathered over three runs to calculate a mean RTT value. IPerf was used to measure the TCP bandwidth between hosts across three runs. The tests were carried out with forwarding rules already installed in the switches. To provide more insight into the cost of the source address verification component, the impact of verifying Layer 2 (MAC) addresses only and both L2 and Layer 3 (IP) addresses were analysed. The ARP traffic verification tests also also include the required L2 and L3 address verification.

4.1.2 Link Fabrication Attack—Address Verification

The test method for LLDP address verification is similar to the MAC/IP and ARP verification. Genuine LLDP messages propagated by switches were tracked to ensure that they were accepted and passed to the controller. A relay-type LFA was performed where the attacker modified the LLDP source MAC address in order to satisfy the MAC address check in the MAC and IP address verification component. These modified messages were tracked to ensure that they were dropped by the switch.

4.1.3 Link Fabrication Attack—Anomaly Detection

The speed of the python-based simulator allows for a practical evaluation of the LLDP anomaly detection over a range of sample sizes and tolerances. Specifically, this test measures the accuracy of detection for a given number of LLDP interval samples. The simulator is designed to take a configurable number of LLDP interval values from a given pool of values. The LLDP intervals captured for the interval analysis in Sect. 3.2.1 were used as a pool of values for each of the switch ports. The pool of LLDP intervals captured from a legitimate network link were used to simulate live readings from a legitimate link, and the pool of intervals from a fabricated link were used to simulate live readings from a fabricated link. Each experiment therefore used a simulated link with characteristics in line with what would be expected of that link type.

For the simulations, each switch port in the network was treated as a candidate link and the variance of the LLDP intervals for these ports was tested against the baseline variance, *B*, as described in Sect. 3.2. For the evaluation, *B* was calculated by taking the mean variance across all of the legitimate links in the testbed network. The tolerance levels tested ranged between 10% and 100%, increasing in increments of 10. Sets of interval values, ranging from 3 to 500 samples in size, were used to calculate the variance at each port. 500 was used as an upper limit as the time required to capture any more LLDP samples is not practical (e.g. it would take 25 minutes to capture 500 samples for ONOS using the default LLDP round time of 3 seconds). Each simulation was run 100 times to determine a percentage False Positive Rate (FPR), which is the number of legitimate links detected as fabricated, and a False Negative Rate (FNR), which is the number of fabricated links detected as legitimate. The objective of these tests is to determine how the number of samples in the LLDP interval variance calculation influences the FNR and FPR, and to identify how the selected tolerance value influences the trade-off between FNR and FPR.

4.1.4 Static Analysis: Memory and Operations

As the resources on networking devices can be limited, a static analysis was carried out to determine the resources required for the security components of the SECAP switch. This analysis considers the number of register reads and writes (operations) performed, and the memory requirements. The memory requirements were calculated based on the number of Bytes needed to store data used by each component.

4.1.5 Analysis of Control Channel and Memory Overheads

The effectiveness of the SECAP solution in reducing control channel and memory overheads was analysed through measurement of control channel traffic and forwarding rule memory requirements.

Overhead, in the form of control channel traffic, was analysed by measuring how much data was sent between the controller and S1 in the testbed shown in Fig. 7. For this analysis, the SECAP switch was operating in *Secure Learning Mode* and traffic was measured for three scenarios. First, a baseline was captured for which only traffic maintaining the controller-to-switch connection was present. Secondly, a scenario was considered where the SECAP solution is not deployed and an attacker takes advantage of a centralised ARP monitoring solution by performing a Denial-of-Service (DoS) attack with the goal of consuming control channel and controller resources. This DoS attack is performed by sending ARP replies at a rate of 1/ms. These ARP replies use randomised MAC addresses, forcing each ARP reply to be sent to the controller for inspection. The final scenario considered has the SECAP solution deployed and the same DoS attack taking place. The total traffic was measured for each scenario (3 experiments/scenario) and the control channel overhead calculated as the mean number of Bytes.

Overhead related to forwarding rule memory was analysed by comparing the memory requirements for generalised forwarding rules with the requirement for rules with more specific match conditions. This is analysed in the context of the DACP attack, where specific forwarding rule match conditions would be required to defeat the attack in the absence of the SECAP solution. The required P4 match conditions and associated data type sizes (in Bits) for a specific forwarding rule (no SECAP switch) and a generalised forwarding rule (SECAP switch) were compared.

5 Evaluation Results

This section presents the results of the tests described in Sect. 4.

5.1 Source Address Verification

When testing the MAC/IP, and ARP verification components, the switches successfully drop the ICMP requests in which the MAC or IP address had been spoofed.

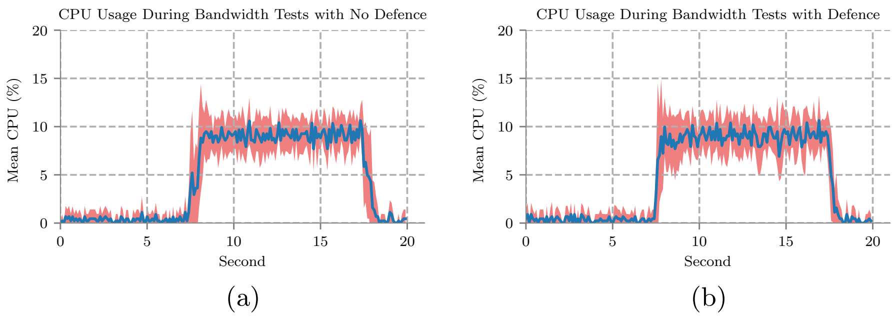


Fig. 8 % CPU usage observed during traffic processing with and without components active

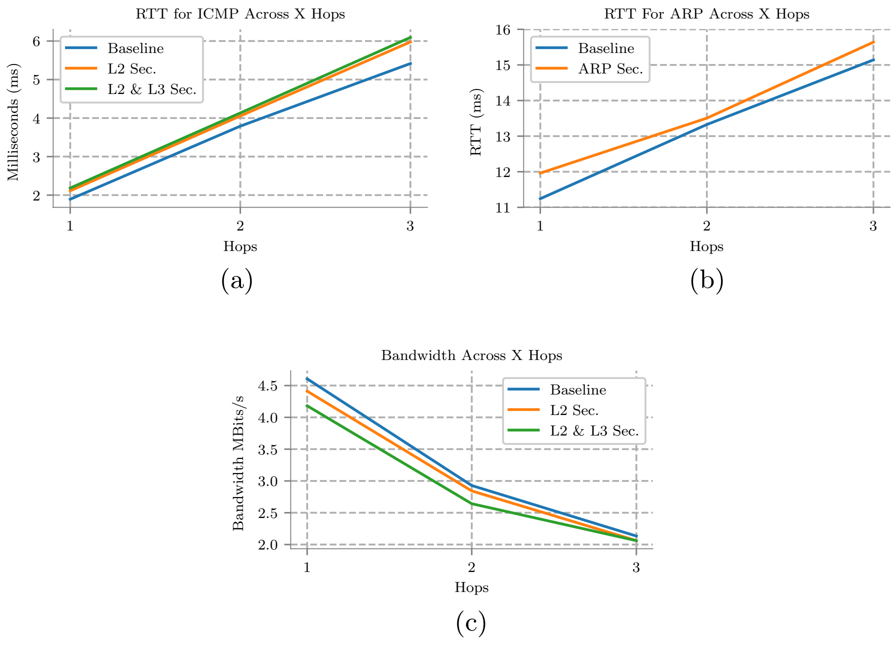


Fig. 9 Impact of solution on ICMP RTT, ARP RTT, and bandwidth

ARP messages were crafted in which the contents of the ARP header were modified in order to carry out an ARP Cache Poisoning attack. These malicious ARP responses were generated at each host targeting another network host. In all tests, the switch detected the discrepancy between the Ethernet header source MAC address and the ARP header source hardware address and dropped the packet.

The processing overhead of MAC/IP verification is shown in Fig. 8. There is no observable impact on the CPU usage. This is expected as the solution does not require complex processing.

The impact of the source address verification components in terms of the RTT and Bandwidth between two nodes in the network over several hops is illustrated in

Fig. 9. The RTT for ICMP traffic, shown in Fig. 9a, increases as the hops between nodes increase. This is expected as each hop adds latency. The RTT across 1 hop and 2 hops shows a minor increase of 0.2 ms when address verification checks are applied to traffic. The largest difference occurs when traffic traverses 3 hops, where a difference of 0.6 ms can be observed between the baseline and when the checks are implemented. This is explained by the additional (though minimal) operations applied to process the message through the switch. The RTT for ARP requests and replies, shown in Fig. 9b, remains within 1ms of the baseline. The bandwidth, measured in MBits/s and shown in Fig. 9c, shows a slight reduction in available bandwidth when the ingress traffic checks are used. However, there is no significant reduction in bandwidth when compared with the baseline. These results are also expected, as the additional latency and processing time will reduce the speed at which data can be sent through the network.

To further explain the increase in RTT and decrease in bandwidth, the exact operations carried out during packet processing are analysed. These operations, specifically the register reads and writes associated with stateful storage, are shown in Table 3. Ultimately, the number of operations required per packet is not significant, and this is reflected in the negligible impact on traffic latency and available bandwidth.

5.2 Link Fabrication Attack—Address Verification

The address checks on LLDP messages were found to drop any relayed LLDP messages in the case where an attacker had previously communicated in the network. Moreover, the address checks did not impede the normal operation of the topology discovery mechanism as the solution does not impede the LLDP messages distributed through the network.

5.3 Link Fabrication Attack–Anomaly Detection

In Fig. 10, the FPR for legitimate links and the FNR for fabricated links are shown for varying sample size and tolerance value. The FPR and FNR are presented for each separate switch port. The results for the percentage tolerance of 10%, 20%, 80%, 90%, and 100% are omitted from the plots as they result in opposing extremes in terms of the FPR and the FNR. For example, while a tolerance of 100% results in a low FPR, it in turn results in a high FNR. Tolerances between 30% and 70%, inclusive, provide both an FPR and FNR closer to an acceptable level. A tolerance value of 50% provides the most balanced FPR and FNR, as a result of the similarity between the LLDP interval values for the legitimate and fabricated link. Using a tolerance value of 50%, an FNR and FPR of 20% or less can be achieved after collecting 200 samples. Approximately 500 samples would be required to reduce the FNR and FPR to 5%.

There are two main reasons for these results. First, the variance between the legitimate link and the fabricated link is quite small, and as such a large number of samples is required to establish a detectable difference. The small difference

Table 3 No. of operations (register reads and writes)

Functionality	Operations	Occurs
Vetting over flag check	1 register read	Each incoming frame/packet
DHCP update	2 register writes	Each DHCP offer from controller
MAC address verification		
Get MAC for port	1 register read	Each incoming ethernet frame at this port
Set MAC for port	1 register write	If no MAC set for this port
IP address verification		
Get IP for port	1 register read	Each incoming IP packet at this port
Set IP for port	1 register write	If no IP set for this port
ARP header verification		
Get IP for port	1 register read	Each incoming IP packet at this port
Set IP for port	1 register write	If no IP set for this port
LLDP anomaly detection		
Check if vetting complete	1 register read	Each incoming LLDP message
Retrieve interval total	1 register read	Each incoming LLDP message if vetting not complete
Update interval total	1 register write	Each interval calculation if vetting not complete
Add new interval	1 register write	Each interval calculation if vetting not complete
Increment LLDP Count	1 register read and write	Each incoming LLDP message if vetting not complete
Total		
3 register reads (+2 writes for new host)		Each IP packet
3 register reads (+2 writes for new host)		Each ARP reply
2 register reads (+1 write for new host)		Each frame carrying anything other than IP or ARP
2 register read + 3 writes (+1 read per captured sample at end of vetting period)		Each LLDP frame during link verification

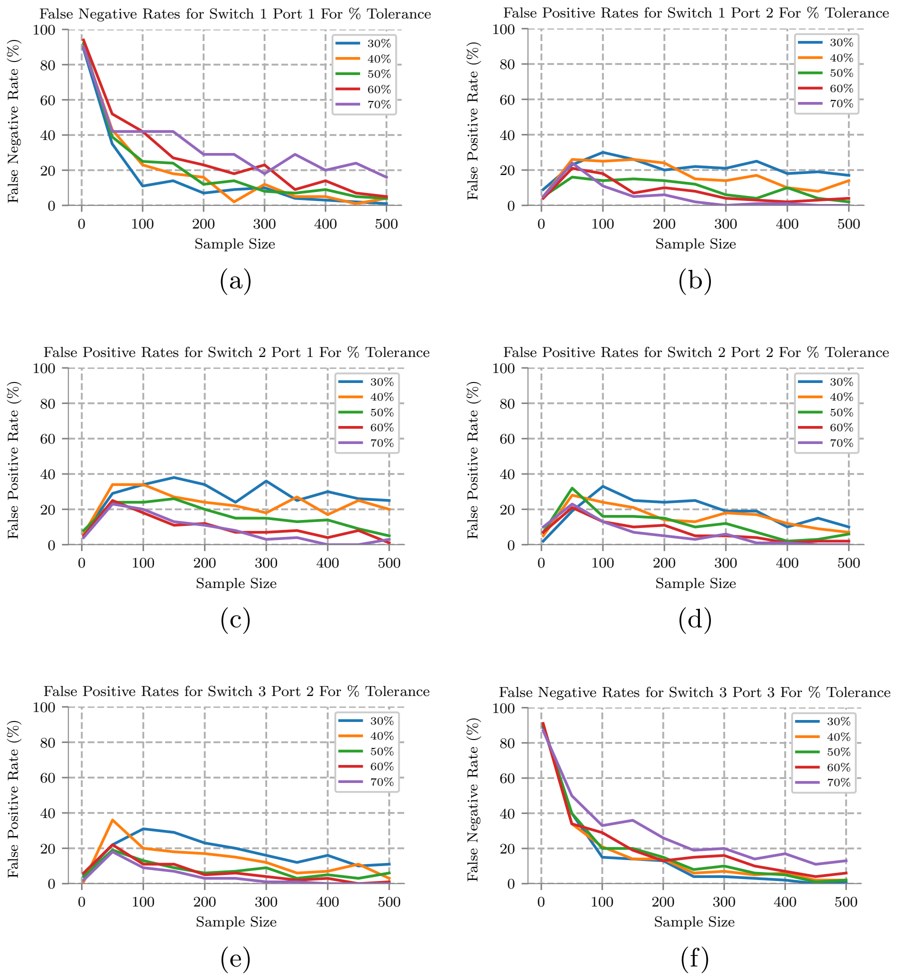


Fig. 10 False positive and false negative rates For LFA detection using LLDP interval variance

causes the results for each tolerance to be similar in their FNR and FPR. The second reason that these results are expected, and the reason the difference in variance is so small is due to the threat model and attack method used here. In this evaluation, the attacker is relaying messages through the network itself, using a relay channel with the speed and reliability of the legitimate links. The attacker's forwarding method causes the fabricated link to be slightly different when compared to a legitimate link. In summary, this is the best possible scenario for the attacker, and while adequate detection takes a large number of samples, it is promising that it can be achieved.

Table 4 Memory requirement for MAC/IP and ARP verification: temporary (temp) & persistent variables

Item	Memory	Storage type
Ethernet header	14 Bytes	Stateless
IP header	20 Bytes	Stateless
ARP header	28 Bytes	Stateless
MAC temp store	6 Bytes	Stateless
IP temp store	4 Bytes	Stateless
Violation flag	1 Bit	Stateless
MAC address register array	6 Bytes (per port)	Stateful
IP address register array	4 Bytes (per port)	Stateful
Memory overhead		
Total for 24-port Switch	312 Bytes + 1 Bit	
Total for 48-port Switch	552 Bytes + 1 Bit	

5.4 Static Memory Requirement Analysis

The number of Bytes required for the MAC/IP and ARP verification components are shown in Table 4. 72 Bytes are required to store relevant data during packet processing and address verification, plus an additional bit used to indicate a security check failure. An additional 10 Bytes is required per switch port. The MAC and IP address associated with a switch port uses stateful storage (P4 register) to allow the data to persist across packets. A typical 24-port switch would therefore need a total of 312 Bytes + 1 Bit to host the components required to defeat the DACP.

Table 5 presents the memory analysis for the LLDP address verification and LLDP anomaly detection. This component requires 146 Bytes of stateless non-persistent storage during packet processing. This stateless storage refers to temporary local variables created to hold the result of an operation or data read from registers. Most of this is required for anomaly detection, as the address check can be performed once the LLDP header has been read and parsed. An additional bit is also needed for use as a flag, which is stateful and uses persistent storage. A further 12 Bytes of stateful persistent storage is required to store data needed for interval, mean, and variance calculations. In total, 437 Bytes and 1 bit is therefore required for a 24-port switch to host this component. A 24 port switch will also require an additional 96 Bytes for each interval sample that will be collected.

Outside of the security features, the solution also requires a further 2 bits used to indicate the operating mode.

5.4.1 Analysis of Control Channel and Memory Overheads

The results of the analysis of control channel traffic overhead is shown in Fig. 11. The baseline value shows the mean number of Bytes required to maintain a connection to a network switch. During the DoS attack, the number of Bytes observed is over three times greater than the baseline. This increase in control channel traffic

Table 5 Memory requirement for LFA defence: temporary & persistent variables

Item	Memory	Storage type
LLDP header	110 Bytes	Stateless
Vetting over temp	1 Bit	Stateless
Prev LLDP Msg time temp	4 Bytes	Stateless
Current time temp	4 Bytes	Stateless
Calculated tnterval temp	4 Bytes	Stateless
LLDP counter temp	4 Bytes	Stateless
Interval running total temp	4 Bytes	Stateless
Interval mean	4 Bytes	Stateless
New link variance (*V*)	4 Bytes	Stateless
Percentage difference (*D*)	4 Bytes	Stateless
Baseline variance (*B*)	4 Bytes	Stateful
Port LLDP counter (*Plc*)	4 Bytes (per port)	Stateful
Port interval running total (*Pit*)	4 Bytes (per port)	Stateful
Previous LLDP message time (TS_{-1})	4 Bytes (per port)	Stateful
Vetting over flag (*Pvf*)	1 Bit (per port)	Stateful
Interval store	4 Bytes (per sample, per port)	Stateful
Memory overhead		
Total for 24-port switch	437 Bytes and 1 Bit, plus 96 Bytes per interval sample	
Total for 48-port switch	728 Bytes and 1 Bit, plus 192 Bytes per interval sample	

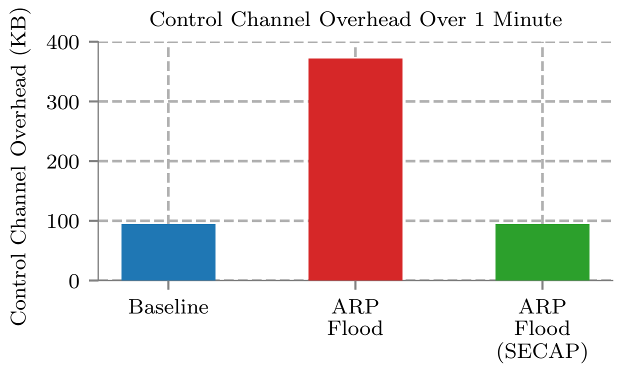


Fig. 11 Control channel traffic comparison with and without the SECAP solution during a DoS attack

is expected, as in this scenario all ARP traffic is mirrored to the controller. With the SECAP Switch solution in place, malicious ARP traffic is filtered at the switch, and as a result, there is no increase in control channel traffic.

The reduction in data plane memory usage is explained through the following example. The DACP attack takes advantage of generalised forwarding rules. Forwarding rules with specific match conditions are required in order to defeat it. An example of a forwarding table with specific match conditions for ARP can be seen in Listing 1. When matching on fields of the ARP header, the total memory requirement for that rule is 304 Bits, or 38 Bytes. A generalised forwarding rule, one which removes match conditions related to the ARP header (shown in Listing 2), requires a total of 128 Bits, or 16 Bytes in total. The SECAP switch enables safe deployment of these generalised forwarding rules, providing an overall reduction in data plane memory used.

Listing 1: ARP - Specific Match Conditions

```
table arp_example {
    key = {
        standard_metadata.ingress_port   : ternary;   // 16 bits
        hdr.ethernet.src_addr            : ternary;   // 48 bits
        hdr.ethernet.dst_addr            : ternary;   // 48 bits
        hdr.ethernet.ether_type          : ternary;   // 16 bits
        hdr.arp.type                     : ternary;   // 16 bits
        hdr.arp.hwsrc                    : ternary;   // 48 bits
        hdr.arp.ipsrc                    : ternary;   // 32 bits
        hdr.arp.hwdst                    : ternary;   // 48 bits
        hdr.arp.ipdst                    : ternary;   // 32 bits
    }
}
```

Listing 2: ARP - Generalised Match Conditions

```
table arp_example {
    key = {
        standard_metadata.ingress_port   : ternary;   // 16 bits
        hdr.ethernet.src_addr            : ternary;   // 48 bits
        hdr.ethernet.dst_addr            : ternary;   // 48 bits
        hdr.ethernet.ether_type          : ternary;   // 16 bits
}
```

6 Discussion

In this section various aspects of the work presented in this paper are discussed.

6.1 Scope of the SECAP Solution

This work focuses on defeating the DACP attack and relay-type LFA, both of which are challenging to accurately detect. In Sect. 5, we demonstrated that the SECAP switch can defeat both of these attacks. However, the SECAP switch can also defend against a range of topology poisoning attacks such as the Host Location Hijacking Attack [1], and the Persona Hijacking Attack [7]. This is achieved by locking a host to a single address to prevent an attacker from spoofing their address. Furthermore, preventing the addresses in the ARP header from deviating from the Ethernet header prevents ARP cache poisoning attacks. The SECAP solution is effective but lightweight in defending against topology poisoning attacks.

6.2 LFA Defence and Results

The address-based LFA defense presented in this paper would be enough to prevent the relay-type LFA in many scenarios. A notable benefit here is that the attack can be prevented without the need for the link to go through a vetting period, as was the case in the previously developed centralised solution [12]. A further defence, such as the variation-based detection method is needed to cover edge cases and other scenarios where an attacker is in a position whereby they can bypass the address-based checks. While the variance-based solution requires a high number of samples to accurately detect the attack, it should be noted that in the experiments the attacker was in an advantageous position. An attacker using the SDN infrastructure to host a relay channel is the worst case scenario for a defence based on anomaly detection, as the attacker can match the speed and reliability of the SDN closely. In the evaluation, the attack was conducted using a GRE tunnel through the SDN and the attacker was able to create a fabricated link that matches closely to the characteristics of the genuine network links. The fact that the attack *can* be detected, even in this difficult scenario, is a positive result. Furthermore, the analysis of the various attack scenarios provided in [12] strongly indicate the possibility of detecting a fabricated link with a lower number of samples in scenarios where out-of-band relay channels or user-space forwarding is used.

As previously noted, some P4 constraints such as lack of support for looping and recursive functions, limit the potential for real-world deployment of the P4-based prototype. The variance-based solution becomes cumbersome, especially when a large number of interval samples are required for accurate detection. However, in [21], Gao et. al present their implementation of statistical calculations using P4 code. These calculations are similar to those required for the SECAP switch anomaly detection component. They also identify the difficulty in implementing these calculations in P4. While their method would not directly apply to the SECAP switch, it does provide the necessary tools to explore alternative statistical functions to use for anomaly detection. For example, it would be worth exploring if the approximate square root function described in [21] would be sufficiently accurate to use for detecting the relay-type LFA scenario. We leave this and the study of alternative statistical functions to future work.

6.3 Deployment

The SECAP switch is not restricted for use in a specific network type. The operating modes presented in Sect. 3.3 are designed to provide flexibility for various deployments. Regardless of the position of the SECAP switch in a network, it is capable of performing valuable security operations. In the context of this work, a SECAP switch deployed in multi-host mode on a network backbone or wireless network can equally improve resilience to topology poisoning attacks.

6.4 Scalability—Topology and Traffic

A key feature of the SECAP switch is its ability to provide security functions independently from the controller and other network switches. As the solution operates independently (i.e. on individual switches), the size of the network topology does not impact upon its effectiveness. However, a larger topology implies more active ports and more network traffic. Tables 4, 5 in Sect. 5.4 provide insight into the amount of memory required for switches of various sizes. The tables indicate the number of Bytes required for storage for 24- or 48-port switches. This storage requirement assumes that the ports are active, and as such the memory requirement will never exceed that detailed in the tables.

In regards to networks with high traffic volumes and high rates of traffic, any impact on network performance caused by the SECAP switch will be linked to the number of operations the solution must perform. The operations performed by the SECAP switch are dictated by the operating mode of the switch and the type of port from which the traffic is arriving. For example, an inter-switch link requires a single register read, while a host port requires two register reads in order to validate source addresses. As such, the number of operations carried out by the switch is finite, and these operations will be carried out at a rate equal to the rate at which the switch can process packets. Therefore, the SECAP solution can be expected to have a limited and consistent impact on network traffic, regardless of the speed of that traffic. The highest impact would be expected in environments where connected hosts change frequently, as the details for these hosts would need to be updated as they connect, leading to further operations being carried out. In future work, we will consider further optimizations to improve performance in all scenarios and evaluate the SECAP solution on hardware switches with commensurate processing capabilities.

7 Related Work

The security of next-generation networks is a critical research area, both from a defensive and offensive perspective. The evolution of programmable networking has led to the investigation of the capability to implement security functionality in programmable data planes such as P4. Voros *et al.* [5] uses P4 to program security middleware (i.e. Firewalls). The middleware provides filtering based on IP, port, protocol, as well as detection of flooding attacks. It can also make decisions based on protocol header fields. Flooding attacks are detected using the meters offered by the P4 language. The work largely functions as a guide showing how firewall features can be built into a P4 program.

Datta *et al.* [22] present P4Guard, a P4-based firewall for SDN. The solution provided in the paper provides a firewall design equipped with parsers to allow flow rules to drop or allow traffic based on criteria in packet headers. Moreover, the work uses counters to gather statistics for traffic in the network, allowing the software to detect flooding attacks. The work also introduces software which acts as a controller specifically for P4Guard, allowing firewalls to be deployed and removed from the network, as required. Apart from the use of counters to deter flooding attacks, this work does not take full advantage of the logic offered by P4 to defeat attacks, and instead focuses on using P4 to improve the match criteria for flow rules and allow fine-grained firewall rules to be installed.

Similar work has been carried out with a focus on p4-based intrusion detection and prevention. Lewis et al. [14] present P4ID, a P4-based Intrusion Detection System (IDS) that uses P4 features as well as flow rules for attack detection. Ndonda et al. [23] develop an IDS with a focus on industrial control systems, specifically ModBus systems. Musumeci et al. [24] focus on defeating TCP flooding attacks using machine learning, showing an improvement in detection capability when in-network P4 components are used. Sanghi et al. [25] develop a similar solution, where in-network P4 components are used to collect data which is then passed to an external data collector, where anomaly detection is used to detect DoS attacks.

P4-based in-network security solutions beyond firewalling have also been explored. The authors of [6] introduce P4-MACsec, a MACSec implementation in P4 that offers protection for network traffic by encrypting traffic using AES-GCM. They also propose a new secure link discovery mechanism, which uses encrypted LLDP messages, an authentication token in the form of a nonce, and a sequence number. These additions to the link discovery protocol aim to defeat generation-type and replay-type LFAs. The defences presented in [6] do not, however, protect against the relay-type LFA.

Extensions to the relay-type LFA have been explored. For example, in [26], Shrivastava et al. study the relay-type LFA in the context of hybrid networks, where broadcast domain discovery protocol (BDDP) messages are relayed rather than LLDP to fabricate multi-hop links. They propose Hybrid-Shield, which verifies links by analysing host traffic and probing the functionality of the conventional switches in the link. They identify a limitation, whereby an attacker can potentially bypass the defence by generating host traffic with spoofed source details. This limitation would be directly solved by the source address verification feature of the SECAP Switch. Host ports set to the appropriate operating mode would not allow host traffic with spoofed source address details to be transmitted in the network. *Multi-host mode* set on other ports would facilitate connections to conventional networking devices. Deploying both the Hybrid-Shield solution and the SECAP Switch together could provide a superior LFA defence.

Other work develops a controller-based, layered approach to defeating the Host Location Hijacking attack, and the various types of LFA [27]. This work builds on previous works to create a modular solution to these attacks. In regards to the LFA, the solution adds a verification field to LLDP messages, preventing them from being forged or replayed. The relay-type LFA defence uses a two-step process in defeating the attack. First, the source port of the LLDP message is checked against a map of host attachment points. If the LLDP message originates from a host port, it will be dropped. Secondly, the transmission delay of LLDP messages is tested against a static threshold value, and links with an LLDP delay greater than this threshold are deemed to be malicious. Furthermore, the entropy of network traffic is considered when determining the legitimacy of a link. While this work delivers a layered solution to the relay-type LFA, it is unclear whether or not it would be capable of detecting the attack in scenarios that are advantageous to the attacker, such as the scenario considered in this paper.

Most recently, Wang et al. present an in-depth analysis of the Host Location Hijacking attack in the context of space–air–ground-integrated vehicular networks [8]. Their work focuses on developing a recovery scheme for this hijacking attack, which was originally presented by Hong et al. [1]. As highlighted in Sect. 6, the Host Location Hijacking attack can be prevented through the source address verification function of the SECAP Switch. In contrast to the SECAP switch security functions, which proactively prevent the attack from taking place, the solution presented by Wang et al. in [8] is reactive, only taking action after an attack has begun. The SECAP Switch solution prevents the attack in a proactive manner, preventing the attack from taking place.

8 Conclusion

As SDN continues to mature, it will be deployed and used in new and exciting contexts such as space networks. Hostile environments where compute power and memory are constrained require straight-forward, lightweight, robust solutions that proactively prevent attacks. In this paper, we presented the SECurity-Aware Programmable (SECAP) Switch, a lightweight, in-network security solution designed to defeat topology poisoning attacks. This solution specifically addresses attacks that are difficult to detect and defeat using a centralized solution. With a P4-based implementation, we demonstrated the capability of the SECAP switch to defend against the data plane ARP cache poisoning (DACP) attack and the relay-type Link Fabrication Attack (LFA) with minimal memory and processing overhead. We presented an LLDP interval variance-based anomaly detection technique, which supports attack detection even in attacker-advantaged scenarios. In future work, we plan to explore alternative statistical calculations for more efficient and accurate detection.

Author Contributions All authors wrote and reviewed the manuscript.

Funding This research was funded by MTU RISAM Scholarship PhD programme and the HEA HCIP3 (18364682) Cyber Skills initiative.

Data Availability Please contact the corresponding author for access to data sets and code generated as part of this work.

Declarations

Competing interest The direction of the work carried out in the paper was not influenced by any third party, nor do the authors of this paper have any conflict of interest in relation to the technologies used.

Ethical Approval Not Applicable.

References

1. Hong, S., Xu, L., Wang, H., Gu, G.: Poisoning network visibility in software-defined networks: new attacks and countermeasures. Ndss **15**, 8–11 (2015)
2. Dhawan, M., Poddar, R., Mahajan, K., Mann, V.: Sphinx: detecting security attacks in software-defined networks. Ndss **15**, 8–11 (2015)
3. Smyth, D., Cionca, V., McSweeney, S., O'Shea, D.: Exploiting pitfalls in software-defined networking implementation. In: 2016 International conference on cyber security and protection of digital services (cyber security), pp. 1–8. IEEE (2016)
4. Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., et al.: P4: programming protocol-independent packet processors. ACM SIGCOMM Comput Commun Rev **44**(3), 87–95 (2014)
5. Vörös, P., Kiss, A.: Security middleware programming using p4. In: International conference on human aspects of information security, privacy, and trust, pp. 277–287. Springer (2016)
6. Hauser, F., Schmidt, M., Häberle, M., Menth, M.: P4-macsec: dynamic topology monitoring and data layer protection with macsec in p4-based sdn. IEEE Access **8**, 58845–58858 (2020)
7. Jero, S., Koch, W., Skowyra, R., Okhravi, H., Nita-Rotaru, C., Bigelow, D.: (2017) Identifier binding attacks and defenses in software-defined networks. In: 26th USENIX security symposium, pp. 415–432. USENIX Security (2017)
8. Wang, J., Liu, J.: Location hijacking attack in software-defined space-air-ground-integrated vehicular network. IEEE Internet Things J **9**(8), 5971–5981 (2021)
9. Smyth, D., McSweeney, S., O'Shea, D., Cionca, V.: Detecting link fabrication attacks in software-defined networks. http://www.smythtech.net/slides/Detecting_LFA_in_SDN_Softfire_2018.pdf
10. Floodlight, P.: Floodlight. https://github.com/floodlight/floodlight
11. Shin, S., Gu, G.: Attacking software-defined networks: a first feasibility study. In: Proc of the second ACM SIGCOMM workshop on hot topics in software defined networking, pp. 165–166. ACM (2013)
12. Smyth, D., McSweeney, S., O'Shea, D., Cionca, V.: Detecting link fabrication attacks in software-defined networks. In: 2017 26th International conference on computer communication and networks (ICCCN), pp. 1–8. IEEE (2017)
13. Alharbi, T., Portmann, M., Pakzad, F.: The (in) security of topology discovery in software defined networks. In: Proc of the 40th conference on local computer networks (LCN). IEEE (2015)
14. Lewis, B., Broadbent, M., Race, N.: P4id: P4 enhanced intrusion detection. In: 2019 IEEE conference on network function virtualization and software defined networks (NFV-SDN), pp. 1–4. IEEE (2019)
15. Open Networking lab: ONOS. https://github.com/opennetworkinglab/onos
16. O'Connor, B., Tseng, Y., Pudelko, M., Cascone, C., Endurthi, A., Wang, Y., Ghaffarkhah, A., Gopalpur, D., Everman, T., Madejski, T., et al.: Using p4 on fixed-pipeline and programmable stratum switches. In: 2019 ACM/IEEE symposium on architectures for networking and communications systems (ANCS), pp. 1–2. IEEE (2019)
17. P4 Language Consortium: P4 behaviour model reference switch. P4 Language Consortium. https://github.com/p4lang/behavioral-model Accessed Aug 2022
18. Inc., D.: What is docker? https://www.docker.com/what-docker
19. Foundation, O.N.: ONOS overview. https://opennetworking.org/onos/
20. Project, S.: Scapy. https://scapy.net/

21. Gao, S., Handley, M., Vissicchio, S.: Stats 101 in p4: towards in-switch anomaly detection. In: Proceedings of the twentieth ACM workshop on hot topics in networks, pp. 84–90. (2021)
22. Datta, R., Choi, S., Chowdhary, A., Park, Y.: P4guard: designing p4 based firewall. In: MILCOM 2018-2018 IEEE military communications conference (MILCOM), pp. 1–6. IEEE (2018)
23. Ndonda, G.K., Sadre, R.: A two-level intrusion detection system for industrial control system networks using p4. In: 5th International symposium for ICS & SCADA cyber security research 2018, pp. 31–40 (2018)
24. Musumeci, F., Fidanci, A.C., Paolucci, F., Cugini, F., Tornatore, M.: Machine-learning-enabled ddos attacks detection in p4 programmable networks. J Netw Syst Manag **30**(1), 1–27 (2022)
25. Sanghi, A., Kadiyala, K.P., Tammana, P., Joshi, S.: Anomaly detection in data plane systems using packet execution paths. In: Proceedings of the ACM SIGCOMM 2021 workshop on secure programmable network infrastructure. pp. 9–15 (2021)
26. Shrivastava, P., Kataoka, K.: Topology poisoning attacks and prevention in hybrid software-defined networks. IEEE Trans Netw Service Manag **19**(1), 510–523 (2021)
27. Gao, Y., Xu, M.: Defense against software-defined network topology poisoning attacks. Tsinghua Sci Technol **28**(1), 39–46 (2022)

Dylan Smyth is an Assistant Lecturer with the Department of Computer Science, Munster Technological University, Cork, Ireland. He has industry experience in software development and a background in networking and security. Through his research, he has contributed to the security of modern software-defined networks by discovering and analysing new vulnerabilities and by development new defence solutions. He is currently a member of the Modern Networking (MoNet) research group in MTU, where his research focus is network security and the real-world application of programmable networks.

Sandra Scott-Hayward is an Associate Professor with the School of Electronics, Electrical Engineering and Computer Science, and a Member of the Centre for Secure Information Technologies at Queen's University Belfast (QUB). She began her career in industry and became a Chartered Engineer in 2006. Since joining academia, she has contributed security designs and solutions for softwarized networks based on her research on network security architectures and security functions for emerging networks. She received Outstanding Technical Contributor and Outstanding Leadership awards from the Open Networking Foundation in 2015 and 2016, respectively, having been elected and serving as the Vice-Chair of the ONF Security Working Group from 2015 to 2017. She is Director of the QUB Academic Centre of Excellence in Cyber Security Education (ACE-CSE), co-lead of the QUB Leverhulme Interdisciplinary Network on Algorithmic Solutions (LINAS) doctoral training programme, and a Polymath Fellow of the Global Fellowship Initiative at the Geneva Centre for Security Policy (GCSP) from 2021 to 2023. In addition to her research on programmable network security, with LINAS and GCSP, she explores the impact of ML and AI technologies on security and society.

Victor Cionca received his Ph.D. in Electronic and Computer Engineering from the University of Limerick, Ireland. Currently he works as a Lecturer in the Computer Science Dept of the Munster Technological University, Cork, Ireland. He is a Funded Investigator in the SFI CONFIRM and CONNECT research centres, and leads the MoNet (Modern Networking Technologies) project at MTU. His expertise and research is in design, configuration, control, and optimisation of high speed wireless and wired networks, with an emphasis of QoS and application constraints.

Sean McSweeney is the Head of the Department of Computer Science at Munster Technological University. Seán is currently involved in a number of national and international research projects. The projects Seán is currently involved in include CYBER SKILLS, a national €8.2M HCI Pillar 3 project where he is a co-applicant and part of the executive team, BODAH a €1.57M European Interreg project and WARIFA, a €6.7M H2020 project, in which he represents MTU at the project management board. Sean's research interests include the application of offensive techniques to critical infrastructure and enterprise systems, digital forensics for enterprise systems, cyber in the cloud and AI, acoustic sensing and autonomous systems localization and navigation.

Donna O'Shea is a leading national figure in cybersecurity research and innovation. She holds the position of HEA SALI Chair of Cybersecurity at MTU. She has extensive experience in winning funding for and leading large-scale national cybersecurity initiatives including the HEA HCI3 'Cyber-Skills' project on cybersecurity skills, SFI 'Cyber-Futures' EPE project on engaging school students in cybersecurity and 'Cyber-Ireland' Industry Cluster. She is a PI, specialising in I4.0 cybersecurity in the CONFIRM SFI Centre for Smart Manufacturing, an FI, specialising in cybersecurity, in the CONNECT SFI Centre for Future Networks and Group Lead for the MTU RÍOMH - Intelligent Secure Systems group. She has a strong track record in doctoral supervision and in learning. She was Head of Department and Lecturer in MTU Computer Science (2013-2020) and she worked in IBM as a Software Developer 2008-2013.

Authors and Affiliations

Dylan Smyth[1] · Sandra Scott-Hayward[2] · Victor Cionca[1] · Sean McSweeney[1] · Donna O'Shea[1]

Sandra Scott-Hayward
s.scott-hayward@qub.ac.uk

Victor Cionca
victor.cionca@mtu.ie

Sean McSweeney
sean.mcsweeney@mtu.ie

Donna O'Shea
donna.oshea@mtu.ie

[1] Department of Computer Science, Munster Technological University, Cork, Ireland

[2] Centre for Secure Information Technologies, Queen's University Belfast, Belfast, UK