Link Fabrication Attack Detection for Software Defined Networks

Adisaya Charoenphol*, Martin J. Reed*
*School of Computer Science and Electronic Engineering, University of Essex, UK
Email: ac22142@essex.ac.uk, mjreed@essex.ac.uk

***Abstract*—Accurate topology knowledge is crucial for software defined networking (SDN) as it uses this to provide effective decisions for SDN applications that provide the network control plane. The SDN controller takes responsibility for creating and maintaining the topology map by gathering information from the data layer through the OpenFlow Southbound API between the controller and the switches. The southbound interface injects link layer discovery protocol (LLDP) packets to obtain the required topology information from switches. However, the topology discovery procedure is open to attack from malicious actors even if secure integrity checking is added to LLDP messages when transmitting packets between switches if valid messages are simply relayed. This vulnerability can result in link fabrication attacks (LFA) and topology poisoning attacks, thereby allowing a malicious actor to reduce performance, perform denial of service, or manipulate routing to maliciously inspect traffic. To detect a LFA in real-time, this research proposes an unsupervised machine learning-based detection based on an isolation forest. The approach was evaluated in SDN switches widely used in cloud environments and with different topology sizes. The evaluation had a high detection performance (F1-score) of 83.92%. It also had a 91.38% attack detection rate (recall) for host-based LLDP relay attacks.**

***Index Terms*—Software-Defined Networking (SDN), Anomaly Detection, Link Fabrication Attack, Unsupervised Learning, Machine Learning**

I. Introduction

Software defined networking (SDN) brings new features to data centre and cloud based networking such as centralised management of quality of services QoS and intent based routing/switching which is difficult with traditional networks [1]. In traditional network devices, decisions such as routing, QoS, and security have to be managed within the devices without an overall network view. SDN is a central control network that separates the control plane from the data plane. The control plane operates as software running on a centralised controller, determining data flows through the network, managing multiple network devices, and making responsible policy decisions. The data plane consists of network devices that forward traffic according to instructions received from the central controller that provides the control plane. This separation allows for centralised network management, dynamic configuration, programmability, and scalability. The benefits of SDN make it attractive to cloud providers and data centres [1] to adopt SDN technology for optimising their network efficiency, automating network management, and scaling network service to their customers. However, security in SDN poses some new challenges [2], and this paper addresses one of these challenges, specifically topology manipulation attacks.

The control plane is delivered by the SDN controller, with key responsibilities including topology discovery for route determination, traffic flow management, and policy enforcement; it serves as the *network's brain*. In terms of topology discovery and route determination, the controller gathers information, including the network devices and links between them. This information is crucial for the controller to function effectively.

The topology discovery process automates the identification of hosts, switches, and links between the devices. The controller and switches communicate via the OpenFlow protocol [3], which uses the OpenFlow discovery protocol (OFDP) to identify links between switches. OFDP uses LLDP to advertise information between active switch ports. The controller builds a centralised topology view after gathering information from LLDP packets. The mechanism of link discovery updates every three seconds to support a dynamic management system.

There are potential threats to SDN's topology discovery using LLDP. A significant vulnerability in this protocol is a link fabrication attack (LFA); attackers can exploit LLDP, an insecure protocol with weak authentication, and no authorization or validation checks, to gain unauthorised access or manipulate network configurations. The LFA aims to poison the controller's central topology view by creating a false link (or links) through compromised devices. Using this false topology an attacker can generate misleading flow traffic, disrupt communication between critical applications, overwhelm network resources, and deny network availability to legitimate devices. The type of LFA discussed in this paper is particularly concerning as it cannot be protected through techniques such as HMAC integrity checking [4] due to the attack simply using a relay of valid messages. Additionally, as switches openly broadcast LLDP messages from all ports, any connected host can relay the LLDP messages to another switch without any opportunity for the management plane to mitigate this attack. Consequently, alternative approaches are needed for mitigation against this attack.

This study proposes an intrusion detection approach to discover host-based LLDP relay detection using unsupervised learning. It can verify bi-directional switch links and detect anomalies in real-time. This approach demonstrates a high recall, which means it effectively identifies a host-based LLDP relay attack. The controller can then use this to mitigate the attack by ignoring these fake LLDP packets.

979-8-3503-5469-0/24/$31.00 ©2024 IEEE

The rest of the paper is organised as follows: Section II introduces the background of SDN topology discovery process and the LFA attack before Section III discusses related work. A model is proposed in Section IV, with the evaluations of this model in Section V. Section VI discusses the proposed deployment, and Section VII concludes the paper.

II. Background

This section provides a detailed description of the SDN topology discovery process, threat model, the LFA, and a brief background on anomaly detection and Isolation Forest.

A. SDN topology discovery process

Topology discovery is an important service for controllers. It collects information from the data plane and creates a central view which many SDN applications use to complete their tasks. OFDP is adopted for SDN topology discovery. It can be described as follows:

1. Host discovery: This process uses a host tracking service (HTS) that depends on the controller platform [5]. For example, the ONOS controller [6]- [7] uses the HostManager to track hosts and identify locations by MAC address and IP address.

2. Switch discovery: This is the initial handshaking process [8]. Switches register with the controller using OpenFlow (with preconfigured IP address and TCP port number) using the OFPT_Hello message. An OpenFlow switch notifies the controller about its active switch ports, port IDs, and related MAC addresses as soon as the switch and controller establish a connection, in response to an OpenFlow OFPT_FEATURES_REQUEST and OFPMP_PORT_DESC message. Therefore, the controller has information about each switch to add/modify/delete flow rules and manage each switch.

3. Link discovery: OFDP leverages the Layer 2 LLDP packets to send switch information to controllers and switches. Network devices use LLDP as a standard protocol to announce their presence and capabilities. The LLDP packet consists of a header and a payload. The header includes the destination MAC address (LLDP multicast: 01:80:C2:00:00:0E), source MAC address, and Ethernet type. The Ethernet type field, which is set to 0x88cc, is used to identify LLDP frames from other frames. The payload comprises a variety of type, length, and value (TLV) fields. The mandatory TLVs are Chassis ID TLV, Port ID TLV, TTL TLV, and End of LLDPDU TLV. The optional TLVS are different for each controller platform. For instance, the ONOS controller's optional TLV includes the ONOS chassis, ONOS port, unknown subtype (0x4), and unknown subtype (0x5). The format of the LLDP packet is shown in Figure 1.

When the controller receives all of the information from the switch, it periodically creates an LLDP packet to be encapsulated and sent by a OFPT_PACKET_OUT packet to the switch's OpenFlow port from the southbound interface. The LLDP packet is then sent by the switches from the allocated port. When a switch receives an LLDP packet from a neighbouring switch (or an attacker mimicking a switch) the switch encapsulates the LLDP frame in OFPT_PACKET_IN packet and passes the message to the controller as illustrated in Figure 2. By executing these actions on every switch, the controller constructs the complete network topology. The OFPT_PACKET_OUT packets, with an enclosed LLDP packet, is sent periodically, for example, every three seconds in ONOS. In this way the controller obtains a central topology view which is dynamically updated. We will see that this process is vulnerable to malicious attacks.

Destination MAC Address	Source MAC Address	Ethernet Type	Chassis ID TLV	Port ID TLV	TTL TLV	Optional TLV	End of LLPDDU TLV
01:80:C2:00:00:0E	Source MAC Address	0x88cc	Chassis ID	Port ID	120	e.g., Port Description	End of LLDP

Fig. 1. LLDP packet format.

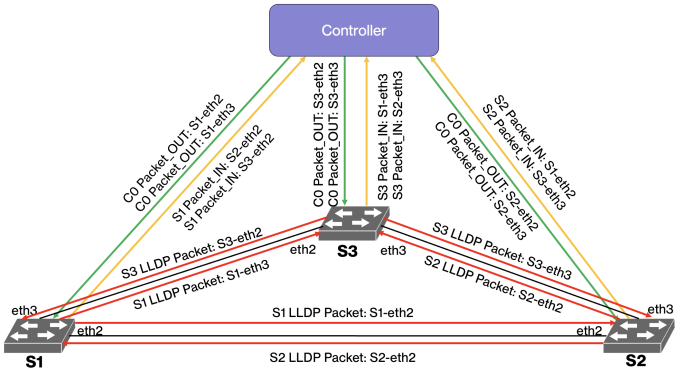

Fig. 2. Link Discovery Process

B. Threat model: link fabrication

As described in the introduction there are a number of potential threats to SDN [2]. This paper addresses a particular type of attack termed a link fabrication. This is of interest to a potential malicious actor as they can use this to introduce a fake link to the SDN network to achieve a potential number of aims including: diverting traffic into the fake link so that the traffic can be monitored; reducing the performance of the network by introducing a slower link; or creating a denial of service attack in the network. This type of attack is more fully described below in II-C, but the high-level threat is first described here. As described above in II-A, LLDP messages are used to automate the network topology discovery process for the controller so that it can detect all the connections between switches (and potentially other network entities such as servers/hosts). To enable this the controller broadcasts the LLDP messages, but to enable the automated discovery the controller has to assume that the LLDP messages are only distributed between valid switches. This implicit trust is at the heart of the threat as there is no obvious mechanism to allow automated discovery without this element of trust. Consequently, this paper shows how to detect when this trust has been breached rather than disallow automated topology

discovery which has uses in dynamic networks or networks that require reduced management overhead.

C. Link fabrication attack (LFA)

The controller's centralised management and control over the whole network is attractive for an attacker to manipulate for malicious purposes. The LFA attack aims to mislead the controller by maliciously using LLDP and creating a fake link (or links) in the central topology view. It can make communication across the network vulnerable to denial of service, monitoring packets and malicious control. An *LLDP packet relay* is a method to mount an LFA as a malicious node in the network captures, copies, relays the LLDP packet and then injects it into a switch port to trick the controller into believing in a fake link. An LLDP packet relay attack, also known as a host-based LLDP relay packet attack, can use compromised end hosts to create LFA, although compromise switches could be used for this attack. This attack requires at least two compromised hosts. They create a channel with traffic data fields that are not visible to the controller and communicate with each other. This approach to building a channel is called an in-band or logical channel [9].

This type of LFA can be understood by a simple example shown in Figure 3 with two hosts H1 and H3 that are under the control of a malicious actor. The controller initiates a new LLDP packet by encapsulating it in an OpenFlow OFPT_PACKET_OUT packet, sends it to S1, instructs it to decapsulate the LLDP packet and send out of eth1 assuming eth1 could be a potential link to a previously unknown switch-to-switch link; likewise, the controller will send other LLDP packets out of all the other interfaces. H1 captures this LLDP packet, encapsulates it in the aforementioned channel and sends it to H3. H3 then injects this LLDP packet into eth1 of S3. S3 then sends this LLDP packet to the controller by encapsulating it in OFPT_PACKET_IN OpenFlow packet. The controller matches the LLDP sent in the OFPT_PACKET_OUT packet and received OFPT_PACKET_IN packet assuming that there is a link between S1 and S3 and records it in the OpenFlow topology table, assuming it is like any other actual link and the LFA attack is complete. H1 and H3 can now capture packets that are sent over this malicious link or rate limit or block data packets allowing other attacks to be performed.

While mitigation strategies exist for LLDP modification attacks [4], the described LFA attack does not require any modification to the LLDP packet. Thus, HMAC based integrity protection, as proposed by Alharbi et al. [4] cannot help against the LFA described in this paper. Instead we show how anomaly detection is possible and this can be used to mitigate the attack by blocking these detected packets.

D. Anomaly detection and Isolation Forest

Identifying data points or events that significantly depart from normal trends is termed *anomaly detection*. Anomaly detection can be accomplished through a variety of strategies, such as ML, rule-based systems, and statistical approaches.

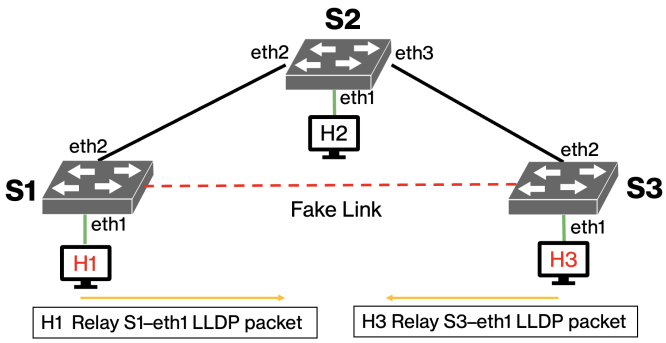

Fig. 3. Host-based LLDP relay packet attack

Accuracy is a major problem in anomaly detection since some techniques might generate a large number of false positives, mistakenly labelling good data as bad. Furthermore, the quality and relevance of the data have a major impact on anomaly detection's efficacy. Depending on the features of the data, the form of anomaly, and the requirements of the application, these approaches can be applied individually or in combination.

ML algorithms for anomaly detection aim to find patterns and anomalies in typical data or events. The most common ML technique for anomaly detection is the supervised learning technique. This model requires labelled data for training. This can be effective for identifying known threats. However, labelling data can be a time-consuming and resource-intensive process, especially for rare anomalies and complex datasets. The unsupervised learning algorithm works with unlabeled data, making it suitable for finding unexpected anomalies without needing pre-defined categories. This paper uses an unsupervised approach.

An *isolation forest* (iForest) [10] is an effective and efficient unsupervised ML for anomaly detection. It detects anomalies by isolating data with a binary tree called an *isolation tree* (iTree), which is effective for isolated data. Anomalies are highly likely to be close to the root or have short average path length on an iTree, while normal data are likely to be isolated at the deeper end of an iTree. The iForest constructs an ensemble of several iTrees, similar to a random forest. The iForest approach isolates abnormal data from normal data, making fast detection performance converge for a small number of trees. An iForest requires a small sampling size for effective detection, which can reduce computational cost and memory usage.

An iTree (one of many in the iForest) is constructed in the training phase by taking an arbitrary feature in the data, for example, it could be mean transmission time. The iTree then creates a binary decision tree by choosing a random split point in this feature, for example, one child less than a certain mean transmission time, the other child greater or equal to the split point. Each new item from the data is then used to grow this iTree by proceeding down the iTree and growing with a new random split point if it is unique, or allocating to an existing leaf if it is the same value as a previous item. In practice, with

continuous data with high precision values that do not repeat, the itree will grow deeper with each new item. Consequently, a maximum depth is commonly set as a parameter. During the running phase, new incoming data is placed in each iTree of the iForest according to the random split points allocated in the training phase. The anomaly score, s, is the output of the iForest analysis on this data, it is defined over the data x as [10]:

$$s(x, n) = 2^{\frac{-E(h(x))}{c(n)}} \tag{1}$$

where $E(h(x)$ is the average of the depth, $h(x)$, from the collection of iTrees and $c(n)$ is a normalising factor derived from the the average depth when searching through all the points in the trees n and can be obtained as [10]:

$$c(n) = 2H(n-1) - 2(n-1)/n \tag{2}$$

where $H(i)$ is the Harmonic number, the sum of the reciprocals of the first n natural numbers. $H(i)$ can be approximated (a good approximation for n being high as in our case) as:

$$H(i) \approx \log(i) + \gamma \tag{3}$$

where $\gamma \approx 0.557$ is the Euler–Mascheroni constant. The anomaly score $0 < s \leq 1$, is interpreted as follows:

- if data return s very close to 1, they are anomalies,
- if data have s much smaller than 0.5, they are safe to regard as normal,
- if data return $s \approx 0.5$, they do not display strong anomaly.

III. Related Works

While SDN offers significant advantages in network management and automation, its reliance on accurate topology discovery makes it vulnerable to various attacks. Countermeasures have been proposed by many studies to mitigate host-based LLDP relay attacks in SDN networks. The solutions used to identify anomalies or prevent this attack are briefly discussed in this section.

To mitigate the risk of host-based LLDP relay attacks, some studies introduced a security framework for detecting SDN link discovery attacks. Chou et al. [11] have presented a correlation-based topology anomaly detection (CTAD) framework. CTAD can monitor and record topology status as well as use correlation analysis for defence LFA attacks. The new LLDP packet format was proposed for detecting LFA along with time difference analysis using a Box-and-Whisker plot. However, the CTAD framework required many resources and bandwidth to generate and send probe packets in the SDN network to calculate the correlation coefficient.

Salti et al. [9] proposed a LINK-GUARD framework that can detect LFA using probe packets and calculate link latency. The link latency measurement module uses boxplot like computations to identify outliers. This framework, however, necessitates sending flow rules and probe packets over the network to calculate link latency for fake link identification. It also has restrictions on detecting low-latency networks, which are less than 5 ms.

In order to identify this particular form of attack, it is important to thoroughly analyse the network's behaviour and status. According to Gao et al. [12], LFA relay attacks result in an increased frequency of the victim host's IP address in the mainstream network. This study employs entropy calculations to examine the distribution of destination IP addresses and the expanded frame format of LLDP frames, which are crucial for integrity verification and safeguarding against forgery and replay. However, the framework required many resources to create a host mapping table and encrypt all the LLDP packets.

Soltani et al. [13] have presented a ML-based defence system: a ML-based link guard (MLLG) for the rarely update network and real-time link verification (RLV) for the frequent update network. MLLG is an offline detection approach that is not able to handle big networks effectively, whereas RLV requires a retraining model to cope with different latencies. The RLV's recall or true positive rate (TPR) is moderate, possibly leading to the omission of certain forged links.

A security solution that is based on P4 switch and calls a security-aware programmable (SECAP) was presented by Smyth et al. [14] to detect LFA. SECAP offers two services: source address verification and in-network anomaly detection. SECAP provides these services using a new variance-based anomaly detection method. The limitations of SECAP, including its lack of support for looping and recursive functions, limit the possibilities for the real-world implementation of the P4-based prototype.

Several different approaches were utilised to detect abnormalities for LFA attacks. However, the challenge of unsupervised ML in these attacks remains an unaddressed research area of study and is thus addressed in this paper.

IV. Proposed Model

To mitigate the LFA, we propose two modules for ML-based LFA detection. The link verification module is responsible for capturing and extracting LLDP information in real-time. After obtaining the LLDP information, feature engineering is used to create new features for a stronger signal from the dataset. The second module is the anomaly detection module, which uses ML to detect host-base LLDP relay packet attacks.

A. Link verification module

The link verification module aims to capture LLDP packets and examine bi-directional links between switches by extracting the necessary features. This study focuses on link latency. Once we have extracted the features, we calculate the link latency and its statistical characteristics.

The feature extraction process involves collecting the OFPT_PACKET_OUT packet and OFPT_PACKET_IN packet on the controller using packet capture tools. As mentioned in the previous section, the SDN's controller broadcasts OFPT_PACKET_OUT packet to every active port on every switch and receives OFPT_PACKET_IN packet from switches. Once the controller matches OFPT_PACKET_OUT packet and OFPT_PACKET_IN packet, it can create the network's central view. This detection mechanism can verify a bi-directional

TABLE I
ML FEATURES, Δ IS THE LLDP LATENCY, QTXX REPRESENTS THE Δ QUANTILE UP TO XX%

No.	Feature Name	No.	Feature Name
1	Flow	18	QT55
2	SourceSwitch_No	19	QT60
3	DestSwitch_No	20	QT65
4	Number_Packets	21	QT70
5	Mean Δ	22	QT75
6	Std Deviation Δ	23	QT80
7	Minimum Δ	24	QT85
8	QT05	25	QT90
9	QT10	26	QT95
10	QT15	27	Maximum Δ
11	QT20	28	Number_Link
12	QT25	29	Number_Switch
13	QT30	30	Number_Node
14	QT35	31	Number_PackOut
15	QT40	32	Number_PackIn
16	QT45	33	Number_DropPack
17	QT50		

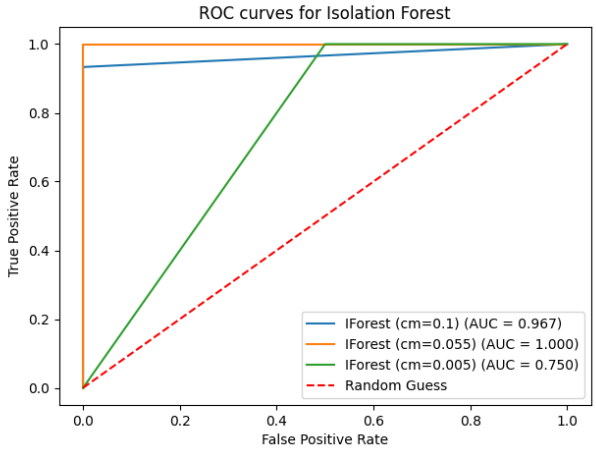

Fig. 4. ROC curve for comparing contamination's Isolation forest

switch link in the same way. We extract features from the packet capture and relate them to bi-directional switch links. This module also calculates the difference between OFPT_PACKET_OUT packet and OFPT_PACKET_IN packet timestamps to determine link latency which is the primary feature used for the anomaly detection. We thus choose the features as shown in Table I.

As the ML needs to operate over a finite number of features and the link latency measurements are arbitrarily spread, leading to a potentially infinitesimal set of values, we need to find a method to categorise these values as unique features. A simple approach is to sample the experimentally determined cumulative distribution function (CDF) of link latency measurements as quantiles. This is more effective than simply binning as a histogram of the probability distribution function as quantiles can easily be summarised more coarsely by simply removing quantile values from the quantile vector without requiring modification. We choose 20th quantiles (20 equally spaced intervals) as the initial choice to give a representation of the CDF in a discrete form. Future work (see VI) is needed to determine how many quantiles are actually needed, we choose 20th quantiles to give a reasonable spread over the CDF without needing an excessive number of features.

B. Anomaly detection module

There are two steps in the anomaly detection module: the training step and the test step. In the formal phase, a large dataframe is required to train the isolation forest model and tune hyperparameter. The latter employs the optimised model in the real network and evaluates the performance shown in Section V.

Feature importance selection is a technique to reflect ML computing's performance and reduce unimportant features that affect resource usage and time consumption. This study uses feature selection for ML dataset preparation. The ML-based LFA solution uses a random forest feature importance selection, which is an ensemble algorithm like isolation forest, to determine the relative importance of input features.

Optimising the hyperparameters of a ML algorithm is also essential for gaining performance. The hyperparameters of the isolation forest model include *max_samples*, *max_features*, *n_estimators*, and *contamination*. The *max_samples* parameter determines the number of examples used to train each individual iTree. The *max_features* parameter determines the number of samples used to train every iTree. The number of iTrees is determined by the *n_estimators* parameter. *Contamination* is defined as the proportion of anomalies in relation to the total amount of data.

Contamination is the crucial and sensitive hyperparameter of the isolation forest algorithm. The setting is based on experience, as well as the size and complexity of the datasets. For this study, the observation contamination values were determined through receiver operating curve (ROC) and precision-recall (PR) curves for an optimal result. Figure 4 and Figure 5 display the ROC and PR curves that compare performance in different contaminations. The optimal contamination for this study is 0.055.

V. EVALUATION

This section presents the experimental setup, the attack model, and its implementation. Then, the performance analysis is presented.

A. Experiment setup

The emulated testbed is built in Mininet [15], which is deployed on server running the Ubuntu operating system and used to emulate the OpenFlow network. Mininet uses OpenvSwitch as the switch fabric as widely used in many cloud networks [16]. The ONOS [6] [7] controller facilitates communication between OpenFlow switches. Note that ONOS has LLDP fabrication mitigation through HMAC protection similar to that described by Alharbi et al. [4]. We ensured this was enabled and noted that the attack was still possible even

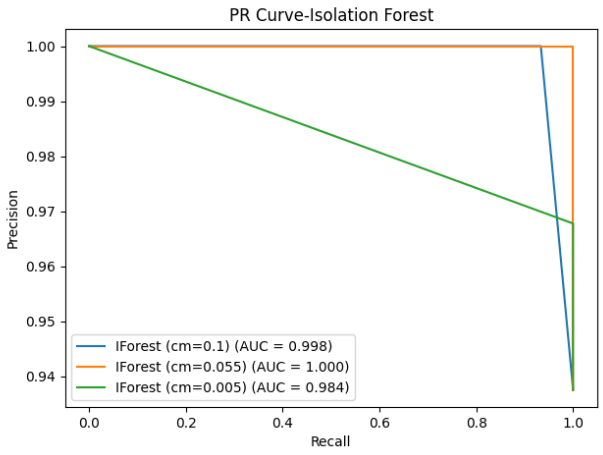

Fig. 5. PR curve for comparing contamination's Isolation forest

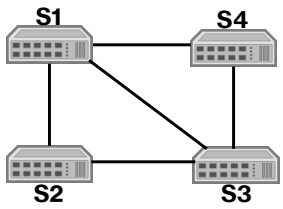

Fig. 6. Scenario 1: Ring topology without attack

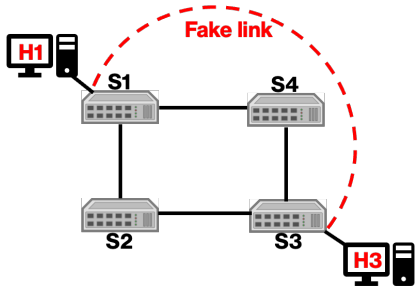

Fig. 7. Scenario 2: Ring topology with Host-based LLDP packet relay attack

with this mitigation as the LFA we are using is a relay based attack rather than a fabrication based attack. We create and craft our LFA using Python.

B. Attack model and implementation

In this study, two compromised hosts set up the host-based LLDP relay packet attack. Python scripts on both compromised hosts establish the attack by listening for and relaying LLDP packets. The first compromised host in the experiment sniffs incoming LLDP packets from the interface connected to the OpenFlow switch. After receiving the LLDP packet, the first compromised host inserts it as a payload inside the User Datagram Protocol (UDP) packet. After that, it establishes a UDP client-server connection and sends the UDP packet to the second compromised. Once the second compromised host receives the UDP packet, it extracts the LLDP packet and forwards it to an incoming port. The second compromised host establishes the bi-directional switch link by sniffing and relaying LLDP packets to the first compromised host. Then, a fabricated link can forward LLDP packets between the two malicious hosts.

The experiment emulates ring topology both without attack and with a host-based LLDP packet relay attack. We randomly create the number of switches in the topology, ranging from 4 to 15, and also assign two compromised hosts at random. There are two scenarios of testing: Scenario 1 is a ring topology without attack, as shown in Figure 6, and Scenario 2 is a ring topology with attack, as shown in Figure 7. In both scenarios, the position and number of links are closely similar. In Scenario 1, the position of two ports between S1 and S3 is unconnected to hosts (normal link), whereas in Scenario 2, the position of two ports between S1 and S3 is connected to hosts (fake link).

C. Dataset creation

The dataset was generated from a python script which automated the operation of the Mininet SDN system. The process of creating a dataset involves two steps. First, we capture the OFPT_PACKET_OUT packet and OFPT_PACKET_IN packets to extract essential information for normal and attack scenarios. In this step, the number of packets depends on the size of the topology which varied between 4 and 15 switches which generated between 820–2720 packets per topology. Second, we verify the bi-directional switch links and summarise the packet information. We analyse the statistical information of bi-directional switch links and convert it into a CSV dataset. The dataset includes network identification features, packet-based features, and statistic-based features, as shown in Table I. The training dataset comprises 31200 records, of which 1200 represent fabricated link states and the rest are normal states. Consequently, 4% of packets are anomalous and the imbalance ratio (IR) is 24:1.

D. Result and performance analysis

As mentioned in the previous section, the link verification module collects the link latency for this study. It is displayed in the cumulative distribution function (CDF) of link latency (Figure 8). The latency range varies dramatically between the normal link, which is approximately 0-0.001 ms, and the fake link, which is approximately 0.007-0.065 ms.

It can measure the effectiveness of ML-based LFA detection using F1 score, precision, and recall scores or true positive rate (TPR) or detection rate (DR). This study takes F1-score, recall, ROC and PR into account. These metrics are as commonly used in assessing ML systems:

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = TPR = DR = \frac{TP}{TP + FN} \tag{5}$$

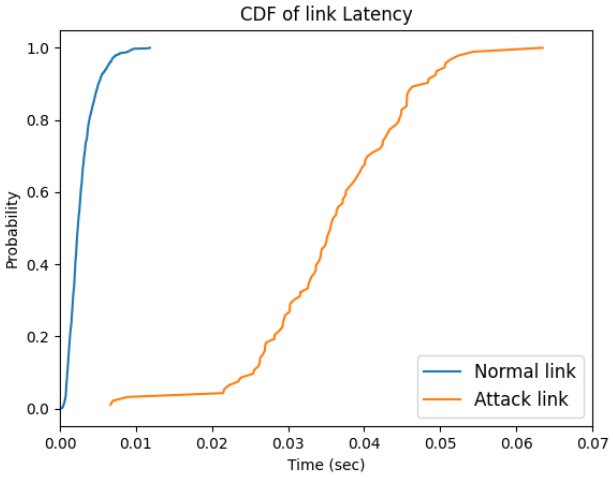

Fig. 8. CDF of link latency

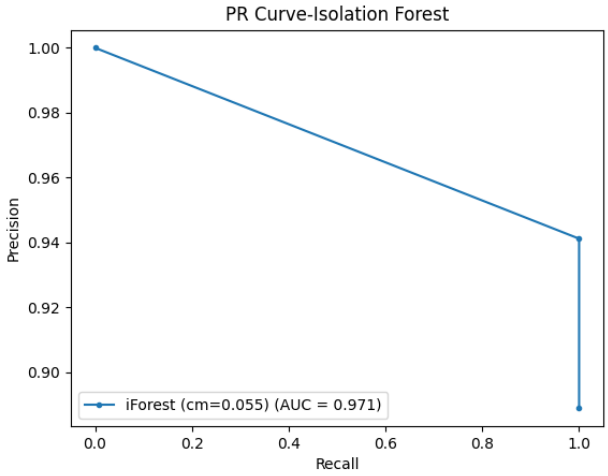

Fig. 10. PR-AUC of ML-based LFA detection

TABLE II
PERFORMANCE OF ML-BASED LFA DETECTION

Algorithm	Score		
	Precision	***Recall***	***F1***
Isolation Forest	0.8448	0.9138	0.8392

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{6}$$

where true positive (TP) is the number of abnormal instances that are correctly classified as fake instances, false positive (FP) is the number of normal instances that are incorrectly classified as fake instances, and false negative (FN) is the number of abnormal instances that are incorrectly classified as normal instances.

The detection performance metrics of the ML-based LFA detection are shown in Table II.

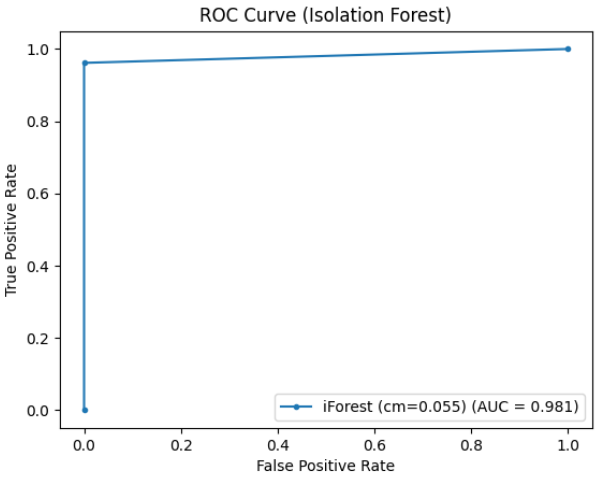

Fig. 9. ROC-AUC of ML-based LFA detection

Another metric that can measure the performance of ML-based LFA detection is the ROC-AUC (receiver operating characteristic area under the curve), which measures the ability of a model to discriminate across different classes accurately. Figure 9 shows an ROC curve that compares True Positive Rate (TPR) to True Positive Rate (FPR). The AUC is found to be 0.981, close to 1, indicating a high level of effectiveness, with the highest TPR and lowest FPR.

Moreover, ML metric known as PR-AUC (precision-recall area under curve) primarily assesses a model's performance in cases of class imbalance. A higher PR-AUC indicates the model's outstanding performance in achieving a balance between precision and recall for the positive class. Figure 10 displays a PR curve that illustrates precision versus recall and an AUC-PR of 0.971. Overall the results show that an unsupervised approach using an iForest can be highly effective in detecting LFA attacks.

VI. DISCUSSION ON DEPLOYMENT OF THE PROPOSAL

This section discusses practical issues of the iForest ML-based LFA detection, along with potential future work.

To verify the link between switches and create a dataset, iForest ML-based LFA detection requires link discovery information. We conduct these processes in the SDN emulation that is close to a real-world cloud environment. This is a different deployment scenario compared to a network consisting of only hardware switches where network challenges such as topology complexity, cost computation, and jitter or noise in the traffic network may have different results. Consequently, future work should consider applying this approach in hardware switches (and hybrid soft/hardware scenarios) to understand the differences.

Another aspect that needs further investigation is the latency between attack start time and detection. This should be investigated by determining the trade-off between precision, recall and detection time. In practice, this would be a decision

made by the developer of a tool or by a network management team. In the experiments here we used 150 s of attack data to determine the results. Consequently, we can confidently state that an attack can be detected within 150 s. As the period between LLDP packets is 3 s and at least two would be required to detect a potential attack using this approach, a detection window between 6 s and 150 s is the most we can determine from this work. Future work should investigate the precision/recall/detection-delay trade-off.

The computational cost of this work is relatively low as the periodicity of data is only every 3 s (i.e. a low data rate) and an isolation forest has relatively low complexity compared to other anomaly detection techniques, such as statistical methods, classification-based methods, and cluster-based methods. Specifically, the isolation forest has a linear computational complexity for evaluation (detection) of $O(nt \log \psi)$ where n is the number of data points evaluated over t trees with ψ elements in the tree [10]. The authors of the Isolation Forest suggest that ψ should be 256 over $t = 100$ trees and we found these defaults to work well [10].

As discussed earlier, additional future work is needed to determine optimum algorithm parameters such as the number of quantiles used for features in the isolation forest and sampling windows used; the latter is closely related to the detection delay.

In this scenario, the switches are presumed to be trustworthy and attacks are mounted via compromised hosts; although, a switch (or switches) may also be compromised. The future work will cover attacks that are launched via compromised switches.

VII. Conclusion

SDN is a dynamic network architecture that uses controller to maintain the central view of the entire network. However, SDN also introduces vulnerabilities that attackers can exploit. While there are a number of mitigation strategies against LLDP based topology attacks, this paper investigates an LLDP relay attack which does not currently have a working mitigation as it cannot be protected by techniques such as HMAC integrity checks. In this study, a host-based LFA is implemented in a random number of switches creating ring topologies. An iForest ML-anomaly detection approach with a simple statistical feature set is used to detect host-based LLDP relay packet attacks and shows a highly effective result. The conclusion is that an iForest-based anomaly detection can detect host-based LLDP relay attacks and provide very good results in terms of recall or TPR.

Acknowledgment

The authors would like to express their gratitude to the Royal Thai Government's Scholarships for providing the necessary funding.

References

[1] L. Poutievski, O. Mashayekhi, J. Ong, A. Singh, M. Tariq, R. Wang, J. Zhang, V. Beauregard, P. Conner, S. Gribble, R. Kapoor, S. Kratzer, N. Li, H. Liu, K. Nagaraj, J. Ornstein, S. Sawhney, R. Urata, L. Vicisano, K. Yasumura, S. Zhang, J. Zhou, and A. Vahdat, "Jupiter evolving: transforming google's datacenter network via optical circuit switches and software-defined networking," in *Proceedings of the ACM SIGCOMM 2022 Conference*, SIGCOMM '22, (New York, NY, USA), p. 66–85, Association for Computing Machinery, 2022.

[2] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 623–654, 2016.

[3] N. Anerousis, P. Chemouil, A. A. Lazar, N. Mihai, and S. B. Weinstein, "The origin and evolution of open programmable networks and SDN," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1956–1971, 2021.

[4] T. Alharbi, M. Portmann, and F. Pakzad, "The (in)security of topology discovery in software defined networks," in *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, pp. 502–505, 2015.

[5] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined networks: New attacks and countermeasures," in *Network and Distributed System Security Symposium*, 2015.

[6] "ONOS." Open Networking Foundation. https://opennetworking.org/onos/.

[7] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: towards an open, distributed SDN OS," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, (New York, NY, USA), p. 1–6, Association for Computing Machinery, 2014.

[8] S. Khan, A. Gani, A. W. Abdul Wahab, M. Guizani, and M. K. Khan, "Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 303–324, 2017.

[9] I. A. Salti and N. Zhang, "LINK-GUARD: An effective and scalable security framework for link discovery in SDN networks," *IEEE Access*, vol. 10, pp. 130233–130252, 2022.

[10] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008.

[11] L.-D. Chou, C.-C. Liu, M.-S. Lai, K.-C. Chiu, H.-H. Tu, S. Su, C.-L. Lai, C.-K. Yen, and W.-H. Tsai, "Behavior anomaly detection in SDN control plane: A case study of topology discovery attacks," in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 357–362, 2019.

[12] Y. Gao and M. Xu, "Defense against software-defined network topology poisoning attacks," *Tsinghua Science and Technology*, vol. 28, no. 1, pp. 39–46, 2023.

[13] S. Soltani, M. Shojafar, H. Mostafaei, and R. Tafazolli, "Real-time link verification in software-defined networks," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 3596–3611, 2023.

[14] D. Smyth, S. Scott-Hayward, V. Cionca, S. McSweeney, and D. O'Shea, "SECAP switch—defeating topology poisoning attacks using P4 data planes," *J. Netw. Syst. Manage.*, vol. 31, jan 2023.

[15] B. Lantz and B. O'Connor, "A mininet-based virtual testbed for distributed SDN development," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, (New York, NY, USA), p. 365–366, Association for Computing Machinery, 2015.

[16] W. Tu, Y.-H. Wei, G. Antichi, and B. Pfaff, "Revisiting the open vSwitch dataplane ten years later," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, (New York, NY, USA), p. 245–257, Association for Computing Machinery, 2021.