Mitigation of LLDP Topological Poisoning Attack in SDN Environments Using Mininet Emulator

Mattia Giovanni Spina[a], Mauro Tropea[b] and Floriano De Rango[c]
Department of Computer Engineering, Modeling, Electronics and Systems (DIMES), University of Calabria, via P. Bucci 39c, 87036 Rende, Italy

Keywords: LLDP Attack, SDN, Mininet, Elliptic Curve Cryptography, RSA, HMAC, ECDSA, POX.

Abstract: Software-Defined Networking (SDN) paradigm permits to have scalability and flexibility in the network management throughout a centralized control that has the global view of the network topology, but it introduces new security issues. In this paper, the Link Layer Discovery Protocol (LLDP) topological poisoning attack has been studied and analysed in order to provide possible mitigation solutions through the use of Mininet emulator and the POX controller. In particular, it is added to the LLDP protocol the integrity check using three different types of cryptographic algorithms such as Hash-based message authentication code (HMAC), Digital Signature Algorithm (DSA) using RSA and Elliptic Curve DSA (ECDSA). The performance evaluation of the proposal is provided considering a network topology where an attacker hijacks/impersonates an host already connected to the network.

1 INTRODUCTION

Software-Defined Networking (SDN) was introduced to provide greater flexibility and agility to computer networks thanks to the decoupling between control plane and data plane. Before the introduction of SDN, networks were managed by specialized network devices (switches, routers, firewalls) that were preconfigured using specific and proprietary vendor software (protocols, firmware, operative systems) (Liu et al., 2019). With SDN, network management is centralized by means of a central entity, called controller, which represents the brain of the whole network and which leads all network devices dynamically programming their behavior in order to adapt it to specific needs of the network (Rahouti et al., 2022). This new networking paradigm defines a three layers architecture:

- *Application Plane (AP)*: it comprises services and applications that rule over the controller and define the behavior to apply to the network;
- *Control Plane (CP)*: it represents the layer in which resides the controller and in which all the managing, monitoring and configuration operations related to data plane devices take place;
- *Data Plane (DP)*: it is the layer that comprises forwarding devices.

Two interfaces are defined to establish the communication between AP-CP and CP-DP that are Northbound and Southbound interface, respectively. For the first interface there is no standard communication protocol to be used and usually RESTful API are employed. For the latter, one of the standard protocols widely adopted is OpenFlow (Li et al., 2016). Despite all the benefits that SDN paradigm can provide, each of these layers are affected by security vulnerabilities due to its software based nature.

All applications at the AP layer, since they need to communicate with the controller, introduce possible flaws that require authentication mechanisms that most of the time are neglected. Due to this problem, attackers can introduce malicious applications in the network or exploit their vulnerabilities in order to hijack the legacy forwarding rules installed by the controller on the network devices. Security issues concerning CP are inherently related to the centralized nature of the controller. If it is hijacked by malicious users the whole network will be compromised (Melkov and Paulikas, 2021; Fioravanti et al., 2023). For what concern DP more complex attacks can be considered such as malicious flow rules injec-

[a] https://orcid.org/0009-0000-8407-2551
[b] https://orcid.org/0000-0003-0593-5254
[c] https://orcid.org/0000-0003-4901-6233

Spina, M., Tropea, M. and De Rango, F.
Mitigation of LLDP Topological Poisoning Attack in SDN Environments Using Mininet Emulator.
DOI: 10.5220/0012086200003546
In *Proceedings of the 13th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2023)*, pages 318-325
ISBN: 978-989-758-668-2; ISSN: 2184-2841

tion and/or tampering aiming at disrupting the actual topology and the legacy behavior of the overall network devices.

The OpenFlow is the commonly used and standardized protocol adopted for the southbound interface. It can use a security layer, namely Transport Layer Security (TLS), but its use it is not mandatory. Therefore, OpenFlow has a lot of vulnerabilities. Among all of them crucial are the ones related to the OpenFlow discovery procedure mainly due to the absence of mechanisms of authentication between controller and DP devices. So, this makes it easy to perform different typologies of security attacks (Iqbal et al., 2019). The main idea of this work consists in the study of one of the typical attacks against an SDN network related to the topology discovery mechanism, the Link Layer Discovery Protocol (LLDP) fake link fabrication attack. We propose to add integrity checks to LLDP protocol in order to mitigate the aforementioned attack against the LDS evaluating three possible types of security algorithms: Hash-based message authentication code (HMAC), Digital Signature Algorithm (DSA) using RSA and, lastly, Elliptic Curve DSA (ECDSA) (De Rango et al., 2020; Tropea et al., 2022).

The rest of the paper is organized as follows: Section 2 presents a brief overview of the research state-of-the-art about threats on topology discovery attacks; Section 3 introduces the LLDP protocol and Section 4 shows the attack due to the crafted LLDP injection. The proposed mitigations are described in Section 5, based on HMAC, DSA and ECDSA approaches. Lastly, performance evaluation and conclusions are presented in Section 6 and Section 7, respectively.

2 RELATED WORKS

The topic of security is very important for the scientific community as proven by the huge literature in different field of research starting from VPN (Gentile et al., 2021) and involving various mitigation techniques such as packet marking (Fazio et al., 2020). Recently, many works have been proposed on security mechanisms and adequate countermeasures for network architectures based on the new paradigm known as SDN. In this section, some of these papers are reviewed in order to show how the literature has dealt with the security topic concerning the link discovery service and in particular the LLDP protocol.

Authors in (Marin et al., 2019) found some vulnerabilities in LDS provided by Floodlight controller. They proposed some guidelines to mitigate vulnerabilities in LDS. One of these vulnerabilities is, indeed, Link Fabrication attack. To mitigate it they implemented a MAC tag based mitigation. We followed this principle and we considered modern and more resistant techniques such as ECDSA to implement at the same time a stronger and lightweight integrity check.

In (Chou et al., 2020) a topology anomaly detection module is proposed which is integrated in the SDN controller which constantly searches for abnormal LLDP packets that are flowing through the network. The anomaly detection is based on a correlation analysis on network traffic trying to avoid and mitigate topology discovery injection and flood attacks.

(Gao and Xu, 2022) proposed methods to detect host migration that are caused by host hijacking attacks and to thwart Link Fabrication attacks. Authors implemented integrity check on LLDP by means of a tag. However, they did not provide details on the cryptographic primitive used to compute this tag and, therefore, it is not possible to determine whether the integrity check is strong enough in terms of security.

Authors in (Baidya and Hewett, 2020) proposed a simple defence mechanism based on active port in order to detect host-based and switch-based discovery attacks. They based the defence mechanism on the principle that no active port can be used to simultaneously connect more than one link otherwise they are considered fake links. However, this does not prevent an attacker to forge malicious LLDP packets poisoning the topology seen by the controller.

(Al Salti and Zhang, 2022) proposed LINK-GUARD which is a framework used to make more secure the link discovery process in SDN environments. It is designed with the aim of detecting Link Fabrication Attack in order to reduce the damage that are caused by the poisoning of controller's topology view. The framework is not based on specific cryptographic primitives and the only mitigation action is blocking switch ports on which a LLDP attack is detected.

Authors in (Popic et al., 2020), in order to mitigate LLDP vulnerabilities proposed a new mechanism for LDS based on NETCONF (Network Configuration Protocol) protocol used as a Southbound Interface together with a modelling language called YANG (Yet Another New Generation). However, NETCONF is not secured, so it represents a possible point for performing attack to the network.

The work in (Alimohammadifar et al., 2018) proposed a link fabrication attack detection based on a probing techniques, in which probing packets are sent across the network in order to find fake links. However, this defense mechanism comes at the cost of a very high bandwidth consumption caused by the additional probing packets.

Considering the reviewed literature, to the best

of our knowledge, this is the first attempt to employ modern cryptographic techniques, such as elliptic curve, to enhance LLDP with integrity checks without the need of deeply modify the protocol and its behavior, keeping also low the computational and protocol overhead imposed on it.

3 LINK LAYER DISCOVERY PROTOCOL (LLDP)

One of the crucial procedures that allows an SDN controller to retrieve information about the network topology and keep it constantly updated is the Link Discovery Service (LDS) (Hong et al., 2015). Exploiting the LDS, the controller can dynamically discover the data plane devices distribution, the existing links between them and other information that are useful to achieve a consistent management of the network (Jimenez et al., 2021). LDS implementations, which referring to OpenFlow is also denoted as OpenFlow Discovery Protocol (OFDP), rely on the Link Layer Discovery Protocol (LLDP) in order to retrieve information about switches and their links making able the switches themselves to embed this information inside specific control packets which encapsulate the LLDP frame. The LLDP protocol is formally defined in the IEEE 802.1AB standard (Committee et al., 2009) and it works at level 2 of the ISO/OSI stack, and its frame format is described in Figure 1.

				LLDP Data Unit (LLD-PDU)				
Preamble	Dst MAC	Src MAC	EtherType: **0x88CC**	Switch DPID	Port ID	TTL	Optional TLV	End TLV

Figure 1: LLDP frame format.

In the SDN environment the link discovery procedure is carried out using Open Flow control packets, namely Packet-In and Packet-Out. Indeed, the LLDP frame depicted in Figure 1 is encapsulated inside those specific packets that are exchanged between controller and switches (Nguyen and Yoo, 2017). Topology information are exchanged by filling up the LLDP Data Unit (LLD-PDU) when an LLDP packet reaches a switch. LLD-PDU is composed of the TLV - Type, Length, Value - which is used to denote the needed information to retrieve the topology structure, such as the Datapath ID (DPID), a 64-bit switch identifier, the Port ID used to specify the outgoing port of the switch that will be used to forward the packet to the next switch, and other useful information. To better understand how the link discovery is carried out we make an example depicted in Figure 2.

Let the considered topology be composed of a single SDN controller and two switches $s1$ and $s2$. For

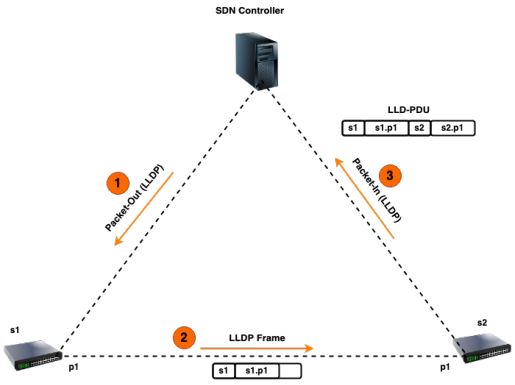

Figure 2: LLDP execution flow.

the sake of making easier to understand the example, we only consider the discovery of the unidirectional link from switch $s1$ to switch $s2$ since the reverse procedure is carried out in the same way. Initially, the SDN controller sends a LLDP packet encapsulated in a Packet-out message to switch $s1$ specifying in the LLDP payload the DPID of the switch $s1$ and its outgoing Port ID. The controller will send this message for each outgoing port of the switch $s1$. When $s1$ receives the packet-Out it decapsulates the LLDP frame and sends it to switch $s2$. Switch $s2$, upon the receipt of the LLDP message from $s1$, adds to the frame its DPID and the Port ID that identifies the port on which $s2$ has received the LLDP packet from $s1$. Then, switch $s2$ encapsulates the frame inside a Packet-In sending it to the controller that will get the information about the link between switch $s1$ and switch $s2$, i.e. the $(s1,s1.p1),(s2,s2.p1)$ link. It is worth noting that switches must be set with a pre-installed flow rule imposing that any LLDP packet received on a switch on a port which is not the port used to connect it with the controller must be forwarded, as a Packet-in, to the controller. This procedure will be repeated for each switch that is managed by the controller and it will be also performed periodically during time in order to get always fresh topology information making it able to get a holistic view of the whole network.

4 LINK FABRICATION ATTACKS: FORGERY AND RELAY

Due to the lack of adequate security measures in the LDS, an attacker can exploit this procedure to carry out several types of attack that could bring serious damage to the network: *Fake Link Fabrication*, *Denial of Service*, *Man-In-The-Middle*. We evaluate a scenario in which an attacker, which is inside of the SDN network or is able to hijack and control a host in-

side of it, injects fake topology information, exploiting the lack of integrity checks, leading the controller lose the right view of the network. The considered attack is the Fake Link Fabrication. It can be categorized in two types: *Relay* and *Forgery* (Gao and Xu, 2022). In the first case, a malicious user has to obtain a genuine LLDP packet that is flowing inside the network. This can be achieved by monitoring the OpenFlow traffic and sniffing one of the exchanged LLDP packet. Once the attacker obtains the genuine/legacy LLDP packet, he modifies its information by replacing the DPID of the legacy destination switch (L_s) with a DPID of a switch A_s that is not actually connected to L_s. So, this tampered packet is forwarded through the network waiting until the discovery procedure is executed and the controller save the fake link between L_s and A_s in its links database. The latter type of Link Fabrication is easier to achieve because an attacker can simply forge from scratch a new LLDP packet which will determine a fake link between two arbitrary switches in the network.

5 PROPOSED MITIGATION

In order to mitigate the LLDP fake link fabrication attack, the *openflow.discovery* module was modified enhancing it with a set of algorithms to provide the integrity feature to the LLDP protocol: HMAC, DSA-RSA, ECDSA. In each of these methods the integrity check is guaranteed used as data the DPID, Port ID and TTL of the LLDP packet generated by the controller. Before sending the LLDP packets to the switches in the network the controller computes a tag based on this data and attach it to the packet. When the LLDP packet comes back from the network to the controller filled with the data gathered by switches, the controller will compute again the tag using the same data and will check if the computed tag and the tag into the packet are the same. If it is true the controller will be sure that the packet is not been modified during the transmission otherwise it drops the packet. This procedure is depicted in Figure 3 in which there is an attempt of LLDP packet tampering in order to create a fake link.

5.1 Hash-Based Message Authentication Code (HMAC)

LLDP injection attack mitigation can be achieved through the use of HMAC, a hash-based authentication technique that allows to verify the integrity and authenticity of the exchanged messages. In this work, a HMAC with a SHA-256 hash function has been

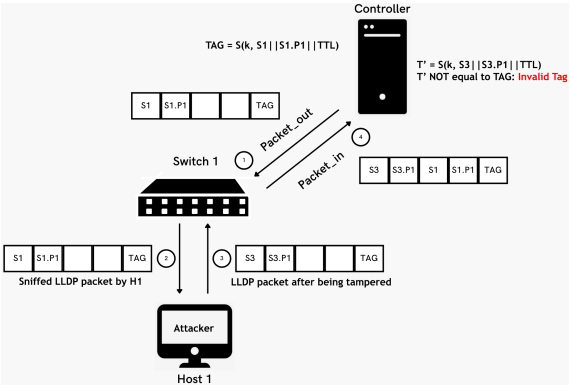

Figure 3: LLDP attack example.

used. The following formula describes how HMAC generates a tag $S(k,m)$ starting from a message m and a key k:

$$S(k,m) = H(k \oplus opad \,||\, H(k \oplus ipad \,||\, m)) \qquad (1)$$

The 128-bit key k, generated in a pseudo-random way, is updated every time the controller starts a new discovery cycle on the network. Message m is formed by the concatenation of $DPID$, $PortID$ and TTL.

The use of HMAC helps mitigate the threat of LLDP injection, as forged LLDP messages will not be able to pass HMAC-based authentication and will be rejected by the network if the attacker tries to perform the attack shown first would have no effect, as the integrity of the LLDP frame would not be verified. The complexity derived by the use of HMAC is not very high as there is no key management between the different hosts, but only the controller takes care of it.

5.2 Digital Signature Algorithm (DSA)

A second way to mitigate LLDP injection can be achieved through the use of a digital signature, an authentication technique based on public-key cryptography. The sender of an LLDP message uses their private key to create a digital signature of the message, which is then sent along with the message. The receiver gets the message and verifies the digital signature using the sender's public key. If the digital signature verification is successful, the receiver can be confident that the message comes from the legitimate sender and has not been modified during transmission. Using a digital signature therefore helps mitigate the threat of LLDP packets tampering, as forged or replayed LLDP messages will not be able to generate a valid digital signature using the legitimate sender's private key, and will be rejected by the network. Digital signature generation and verification

require cryptographic computation which can lead to increased processing load on network devices.

In our case, the digital signature based on the RSA algorithm with a 2048-bit key was used.

5.3 Elliptic Curve Digital Signature Algorithm (ECDSA)

LLDP injection mitigation can also be achieved through the use of a digital signature based on Elliptic Curves Cryptography (ECC), a public key cryptography method that uses elliptic curves (Saho and Ezin, 2020). In practice, the ECC allows the use of shorter keys than other encryption techniques, but with the same cryptographic strength. The use of a digital signature based on ECC therefore helps mitigate the threat of LLDP injection, as forged LLDP messages will not be able to generate a valid digital signature using the legitimate sender's private key, and will be rejected by the network. Furthermore, using ECC to generate digital signatures is also more secure than other public key cryptography techniques, as it is more resistant to some cryptographic attack techniques such as brute force attack and key lookup attack. Two types of curves have been used: random curves (r) and Koblitz curves (k) (Brown, 2010). Koblitz curves are characterized by their particular structure, while random curves are generated randomly. Koblitz curves type is characterized by some inherently mathematical properties that allow to make an efficient implementation of the group of operations that are used during the keys generation over the curve. However, it is common to believe that more randomly selected parameters (i.e. random curves) can provide a more entropic force and, thus more security strength. But there is the suspect that random coefficients can be chosen in order to provide a backdoor (Suárez-Albela et al., 2018). The choice among one curves type or the other is strictly related to the specific needs of the application in which they are used. If we need a computational effective implementation we can use Koblitz, otherwise if we want to give a more entropic feature to the key generation procedure we can choice random curves.

6 PERFORMANCE EVALUATION

In this section we provide the performance evaluation of the proposed mitigation varying the integrity algorithm. Our experimental setup is composed of a computer running a virtual machine with Ubuntu 18.04 operative system running the latest version of Mininet (Tropea and Palmieri, 2022) and the POX SDN controller. Different simulation scenarios have been considered during performance evaluation phase and they are described in Table 1.

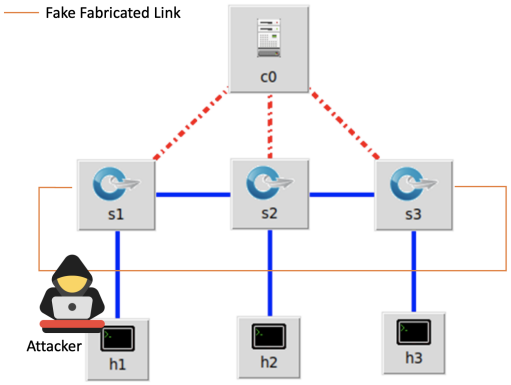

Figure 4: Considered network scenario.

Table 1: Considered scenarios for performance evaluation.

Scenario	Description
S1	Normal behavior
S2	HMAC mitigation
S3	REC-256 mitigation
S4	REC-384 mitigation
S5	KEC-283 mitigation
S6	KEC-409 mitigation
S7	DSA-RSA 2048 mitigation
S8	System under LLDP attack

6.1 Attack Scenario

To replicate the crafted fake link injection attack, we considered the topology depicted in Figure 4.

In our scenario, the host h1 represents the malicious user which is sniffing on S1 to get a legacy LLDP packet with the DPID of S1. Once it gets this packet, it will replace the DPID of S1 with the DPID of S3. Then, forwarding this tampered packet it will make the controller to believe that there exists a direct link between S1 and S3, which actually does not exist in the legacy topology. This kind of attack can be the starting point to launch more harmful attacks such as Man-In-The-Middle or Denial of Service (DoS) (Potrino et al., 2019). In the first one, the attacker by letting the controller see a fake link between two arbitrary switches can sniff all the data exchanged on this fake link that is accurately installed and controlled by the attacker itself. In the case of the DoS, instead, the attacker can create fake links among switches leading to complex-loop topology that can cause a huge amount of extra data exchanged on the network leading to a high utilization of the bandwidth and, as a consequence, to an extra-overhead that can saturate the network itself. It has also to be mentioned, in the

context of DoS attack, the Spanning Tree Service that is implemented by Open Flow controllers. It is a service that is triggered each time that a topology update occurs in order to block switch ports that are redundant and that can lead to useless loop in the network. If the attacker can leverage the LDS, it can create and add a fake link between two legacy switches creating a loop that will be detected by the Spanning Tree Service that, as a consequence, will block the ports that connect those two switches making them not able to communicate anymore (Hong et al., 2015).

6.2 Network Performance Analysis

Using Wireshark, we have collected data relating to the bandwidth used for each type of mitigation, considering a fixed simulation time. As can be seen in Figure 5, the mitigation that uses the greatest bandwidth is the digital signature, since the key used by the algorithm is 2048 bits. For digital signatures based on elliptic curves, bandwidth consumption is lower because the keys used have a smaller length. In particular, on the digital signature based on ECC, it can be noted that an increase in the length of the key leads to an increase in bandwidth consumption. The mitigation with the use of HMAC turns out to be the least expensive compared to all the others. However, elliptic curve methods provide a higher degree of security, so there is a trade-off between security and network bandwidth consumption to be considered.

Regarding the generation and signing time of an LLDP frame, a series of timing data was collected and averaged. It can be seen from the Figure 6 that, due to the generation of a message for verifying the integrity of the data, the time increases with respect to the normal behavior where there is no verification of data integrity. In particular, as we can imagine, it can be observed that, using HMAC we got the less generation and verification time due to the use of high efficient hash function SHA-256 and the effective implementation of the HMAC algorithm. So we register a slight increase in time while providing integrity checks. When we deal with more complex methods the increasing in time is strongly related to the length of the key which is involved. Indeed, when elliptic curve is used we registered an increasing of time in generation and signature which is proportional to the size of the key. The signature verification procedure algorithm for elliptic curve is more computational expensive with respect to the one used in RSA and this turns out to be verified by our experiments as can be seen in Figure 6. The same consideration can be done for the LLDP frame processing phase which can be appreciated in Figure 7. Also in this case we have to face a trade-off between security degree (higher when using ECC and lower when using RSA) and processing time (lower when using RSA).

6.3 CPU and RAM Performance Analysis

CPU and RAM data usage was collected using a Python script which is able to interact with the underlying operative system. Following the aforementioned results about LLDP frame processing it can be noticed in Figure 8 that this trend is confirmed for the CPU usage. Indeed, when using the HMAC the amount of consumed CPU is negligible with respect to the normal behavior with the advantage of providing the needed integrity check. Instead, when using the ECC the consumed CPU increases following the trend of the LLDP frame processing time.

Since the signature generation and verification algorithm does not change when changing the curve type but it is related to the size of the employed key, the amount of CPU does not show an evident trend variation, see Figure 8. The high value registered for the CPU usage when using RSA is due to the most expensive computational overhead caused by the keys generation procedure. These results highlight the benefit of using elliptic curves since they provide a very efficient and lightweight keys generation procedure, which does not effect the overall overhead, and at the same time a more security strength degree. Therefore, in order to provide an integrity check for the LLDP protocol, it can be evaluated the use of HMAC or ECC considering a trade-off between the needed security degree and the computational overhead. Figure 9 compares the data collected on the use of memory (RAM) by the controller. In this case there is no difference between the digital signature and the elliptic curves, which use a slightly higher memory load than the base case and HMAC. This is because a 128-bit key is used in HMAC, while in other cases the key is 256 or 384 for digital signature based on random elliptic curves, 283 or 409 for digital signature based on Koblitz elliptic curves, and 2048 for RSA-based digital signature. Using Wireshark it was possible to analyze the network in the already considered scenarios. We have observed that the pattern of the exchanged packets in the network is almost identical in every situation since the attack is based on sending a single packet, which does not affect the network in any way. For this reason, identifying such type of attacks it is pretty complex since it can be considered as a very silent attack. Thus, the additional integrity check prevents the malicious user to carry out this kind of attacks.

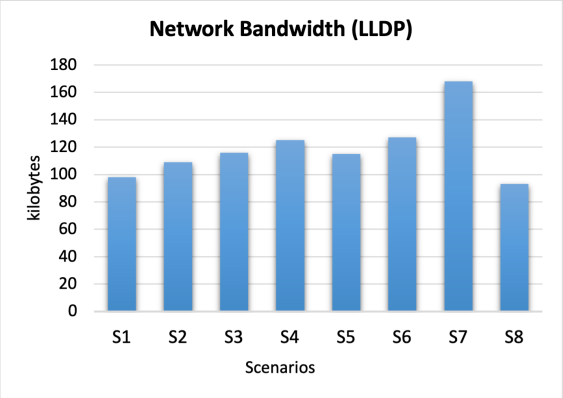

Figure 5: Network used bandwidth.

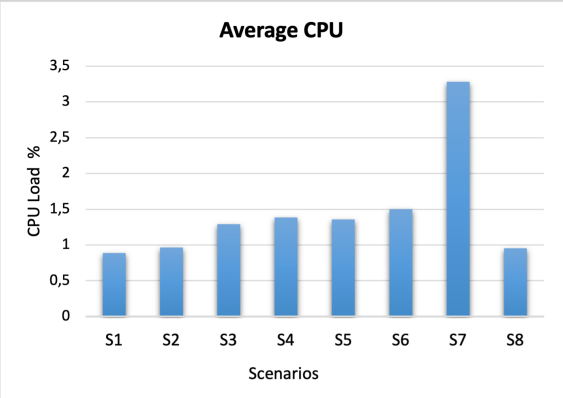

Figure 8: Average Used CPU.

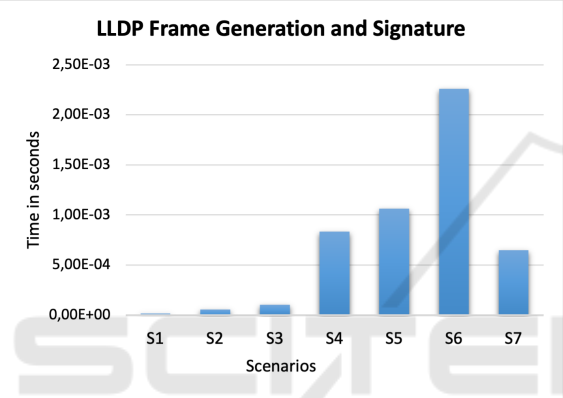

Figure 6: LLDP frame generation time.

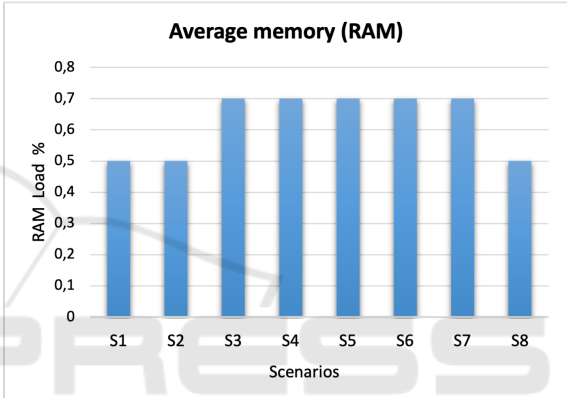

Figure 9: Average Used RAM memory.

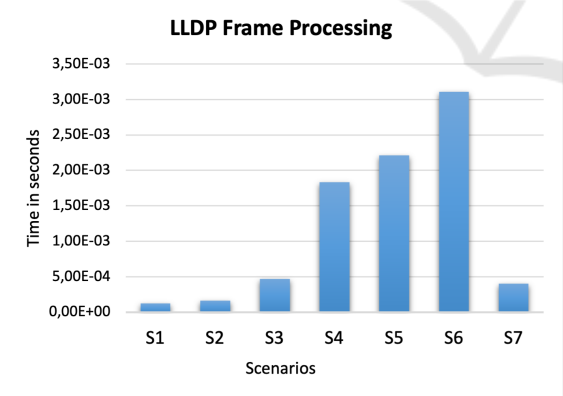

Figure 7: LLDP frame processing time.

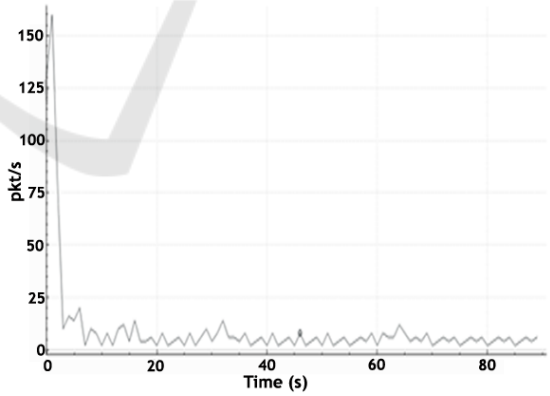

Figure 10.

7 CONCLUSIONS

This paper focuses on SDN network security, which introduces new risks due to control/data plane separation. The fake link fabrication attack is analyzed, and an integrity check is proposed using three cryptography primitives (HMAC, DSA-RSA, ECDSA) and evaluated using Mininet and a POX controller. Results show performance in terms of network and resource consumption. Choosing between cryptography primitives depends on network security needs and operational restrictions. ECDSA is best for security, while HMAC is a good alternative for a simpler and more efficient approach.

REFERENCES

Al Salti, I. and Zhang, N. (2022). LINK-GUARD: An Effective and Scalable Security Framework for Link Discovery in SDN Networks. *IEEE Access*, 10:130233–130252.

Alimohammadifar, A., Majumdar, S., Madi, T., Jarraya, Y., Pourzandi, M., Wang, L., and Debbabi, M. (2018). Stealthy Probing-Based Verification (SPV): An Active Approach to Defending Software Defined Networks Against Topology Poisoning Attacks. [Online; accessed 11. Mar. 2023].

Baidya, S. S. and Hewett, R. (2020). Link discovery attacks in software-defined networks: Topology poisoning and impact analysis. *J. Commun.*, 15(8):596–606.

Brown, D. R. (2010). Sec 2: Recommended elliptic curve domain parameters. *Standards for Efficient Cryptography*.

Chou, L.-D., Liu, C.-C., Lai, M.-S., Chiu, K.-C., Tu, H.-H., Su, S., Lai, C.-L., Yen, C.-K., and Tsai, W.-H. (2020). Behavior anomaly detection in sdn control plane: a case study of topology discovery attacks. *Wireless Communications and Mobile Computing*, 2020:1–16.

Committee, L. S. et al. (2009). Ieee standard for local and metropolitan area networks–station and media access control connectivity discovery.

De Rango, F., Potrino, G., Tropea, M., and Fazio, P. (2020). Energy-aware dynamic internet of things security system based on elliptic curve cryptography and message queue telemetry transport protocol for mitigating replay attacks. *Pervasive and Mobile Computing*, 61:101105.

Fazio, P., Tropea, M., Voznak, M., and De Rango, F. (2020). On packet marking and markov modeling for ip traceback: A deep probabilistic and stochastic analysis. *Computer Networks*, 182:107464.

Fioravanti, G., Spina, M. G., and De Rango, F. (2023). Entropy based ddos detection in software defined networks. In *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, pages 636–639. IEEE.

Gao, Y. and Xu, M. (2022). Defense against software-defined network topology poisoning attacks. *Tsinghua Science and Technology*, 28(1):39–46.

Gentile, A. F., Fazio, P., and Miceli, G. (2021). A survey on the implementation and management of secure virtual private networks (vpns) and virtual lans (vlans) in static and mobile scenarios. In *Telecom*, volume 2, pages 430–445. MDPI.

Hong, S., Xu, L., Wang, H., and Gu, G. (2015). Poisoning network visibility in software-defined networks: New attacks and countermeasures. In *Ndss*, volume 15, pages 8–11.

Iqbal, M., Iqbal, F., Mohsin, F., Rizwan, M., and Ahmad, F. (2019). Security issues in software defined networking (sdn): risks, challenges and potential solutions. *International Journal of Advanced Computer Science and Applications*, 10(10).

Jimenez, M. B., Fernandez, D., Rivadeneira, J. E., Bellido, L., and Cardenas, A. (2021). A survey of the main security issues and solutions for the sdn architecture. *IEEE Access*, 9:122016–122038.

Li, W., Meng, W., and Kwok, L. F. (2016). A survey on openflow-based software defined networks: Security challenges and countermeasures. *Journal of Network and Computer Applications*, 68:126–139.

Liu, Y., Zhao, B., Zhao, P., Fan, P., and Liu, H. (2019). A survey: Typical security issues of software-defined networking. *China Communications*, 16(7):13–31.

Marin, E., Bucciol, N., and Conti, M. (2019). An in-depth look into sdn topology discovery mechanisms: Novel attacks and practical countermeasures. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1101–1114.

Melkov, D. and Paulikas, S. (2021). Security benefits and drawbacks of software-defined networking. In *2021 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, pages 1–4. IEEE.

Nguyen, T.-H. and Yoo, M. (2017). Analysis of link discovery service attacks in sdn controller. In *2017 International Conference on Information Networking (ICOIN)*, pages 259–261. IEEE.

Popic, S., Vuleta, M., Cvjetkovic, P., and Todorović, B. M. (2020). Secure topology detection in software-defined networking with network configuration protocol and link layer discovery protocol. In *2020 International Symposium on Industrial Electronics and Applications (INDEL)*, pages 1–5. IEEE.

Potrino, G., De Rango, F., and Santamaria, A. F. (2019). Modeling and evaluation of a new iot security system for mitigating dos attacks to the mqtt broker. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE.

Rahouti, M., Xiong, K., Xin, Y., Jagatheesaperumal, S. K., Ayyash, M., and Shaheed, M. (2022). Sdn security review: Threat taxonomy, implications, and open challenges. *IEEE Access*, 10:45820–45854.

Saho, N. J. G. and Ezin, E. C. (2020). Comparative study on the performance of elliptic curve cryptography algorithms with cryptography through rsa algorithm. In *CARI 2020-Colloque Africain sur la Recherche en Informatique et en Mathématiques Apliquées*.

Suárez-Albela, M., Fraga-Lamas, P., and Fernández-Caramés, T. M. (2018). A practical evaluation on rsa and ecc-based cipher suites for iot high-security energy-efficient fog and mist computing devices. *Sensors*, 18(11):3868.

Tropea, M. and Palmieri, N. (2022). Software defined networking emulator for network application testing. In *Open Architecture/Open Business Model Net-Centric Systems and Defense Transformation 2022*, volume 12119, pages 50–61. SPIE.

Tropea, M., Spina, M. G., De Rango, F., and Gentile, A. F. (2022). Security in wireless sensor networks: A cryptography performance analysis at mac layer. *Future Internet*, 14(5):145.