Link Fabrication Attack and Detection

A SDN COURSE PROJECT REPORT

Submitted by

Kaniganti Priyanka Saraswathi (201020422, DSAI)
Roopa Navya Muthi (201000045, CSE)

Under the guidance of

Dr. VENKANNA U.
(Assistant Professor, IIITNR)

Dr. Shyama Prasad Mukherjee International Institute of Information Technology, Naya Raipur

APRIL, 2023

ABSTRACT

A Link Fabrication Attack (LFA) in Software-Defined Networking (SDN) entails the creation of a new malicious link in the network by an attacker, giving them control over any traffic that passes via the connection. This attack can be carried out, for which there is no comprehensive defense, by relaying topology discovery trafficOur research offers a novel method to deal with this problem that compares the JSON file of the network topology with the original file to identify any modifications to the topology. The suggested approach involves testing the network, and a special controller is used to spot false links. This method, which allows for a thorough study of the network topology while minimizing the danger of false positives, offers an effective solution to the identification of LFAs in SDN. This study helps to safeguard the network's integrity and makes improvements to SDN security.

Keywords: SDN, Link Fabrication Attack, P4,LLDP

CHAPTER 1: INTRODUCTION

The innovative network paradigm known as "software-defined networking(SDN)" offers flexible, programmable, and all-encompassing network management. The SDN controller maintains a global perspective of the entire network, which is one of the key distinctions between SDN and traditional networks[1]. The quality of networking services is considerably increased when the controller uses its global perspective to create network-wide optimal solutions in response to network events. To provide this worldwide visibility, a crucial role is played by the topology discovery service[2]. In order to build a comprehensive image of the underlying network architecture in the control plane, it gathers information from the data plane. The global information is used by many network management programmes, including load balancing, packet forwarding, and network optimisation.

Unfortunately, the entire network may be impacted or entirely exposed to attackers if the topology discovery service offers poisoned topological information. Numerous attacks have been suggested due to the topology discovery service's significance[3]. For instance, a common topological attack that introduces phony links into the network is the LFA. In a LFA, the attacker inserts fake links into the network topology and manipulates the forwarding rules to send traffic through these fake links. This can cause network congestion, packet drops, and even denial-of-service (DoS) attacks, as the network traffic is redirected to non-existent or malicious links[4].

This paper presents a link creation attack that modifies the topology of a mininet network using a man-in-the-middle technique. By taking control of a host, the attacker can add fictitious linkages to the network topology using Python scripts. It makes use of flaws in the Link Layer Discovery Protocol(LLDP), a vital protocol for identifying internal links. By manipulating the load balancing programme to create a routing black hole or by injecting malicious packets to steal the identities of the network devices, attackers can use the bogus link to do significant harm. Researchers have discovered numerous relevant weaknesses in order to achieve this, and have created matching defenses to improve the security of the network topology.

This paper presents a link creation attack that modifies the topology of a mininet network using a man-in-the-middle technique. By taking control of a host, the attacker can add fictitious linkages to the network topology using Python scripts. The fake links start sending Link Layer Discovery Protocol (LLDP) packets once the attack and, Where routing-controller scripts have been run, and the custom controller treats them as real. By using this method, an attacker can change the network topology, which they can then use to conduct other assaults or sabotage network traffic. The aforementioned attack serves as a reminder of the value of putting strong network security measures in place to guard against unauthorized access and network topology manipulation[5].

1.1 RESEARCH PROBLEM

A severe security risk in Software-Defined Networking (SDN) is the Link Fabrication Attack (LFA), which enables attackers to build a malicious link in the network and take control of the data traveling over it. Although there could be disastrous effects, there is currently no all-encompassing defense against this kind of attack.

1.2 MAJOR CONTRIBUTIONS

The following are the paper's contributions:

- Overwriting the ‘P4app.json’ file which is used to automate the deployment of P4 programs on network topology using an **attack(named as ‘a’) python script.**
- A custom **routing-controller python script** for optimal path detection in a p4 mininet environment, which does not have a controller.
- Another customized python script named **routing-contoller_mitigation** is used for detection and mitigation of topology poisoning attack

CHAPTER 2: METHODOLOGY

2.1 Performing link fabrication attack

The goal of the LFA is to trick the SDN controller into thinking there is a direct inter-switch link when there actually isn't one. An attacker who gains access to the network infrastructure or the SDN controller can initiate this attack by changing the forwarding rules to produce fictitious or harmful links. The centralized SDN controller in an SDN network interacts with the switches to install forwarding rules and makes the forwarding choices. Based on the network architecture and the present state of the network, the switches employ these rules to forward packets to their destinations. To allow communication between the P4 Switch and the controller in P4-based Software-Defined Networking (SDN), The P4 runtime API is a standardized interface between a P4-programmable switch and a controller program. It allows the controller to dynamically configure and control the behavior of the switch at runtime, by sending P4-specific messages and commands over a network connection.

A variety of operations can be carried out on the switch using a set of well defined message types that are provided by the P4 runtime API. These actions include establishing the switch pipeline, adding and removing table entries, altering match-action tables, changing counters and meters, and more. A controller programme that runs on a different system and communicates with the switch over a network connection commonly uses the P4 runtime API. Using the P4 runtime API, the controller programme communicates with the switch and receives notifications and acknowledgements in return.

By importing topology from P4utils, which gives a framework for writing Python code to define and manage network topologies. The topology module enables programmers to construct intricate network topologies made up of numerous switches and hosts, and to define the P4-based forwarding rules that should be used to direct traffic through the network. The poisoned topology can be created by changing the links in the topology view, The P4 Runtime Server maintains a topology view of the network, which includes information about the switches, ports, and links in the network topology. This information is obtained through a combination of static configuration and dynamic discovery mechanisms, such as LLDP and topology discovery protocols. Here we considered a topology which contains 3 switches and 4 hosts, where the topology view is given the below fig 1:

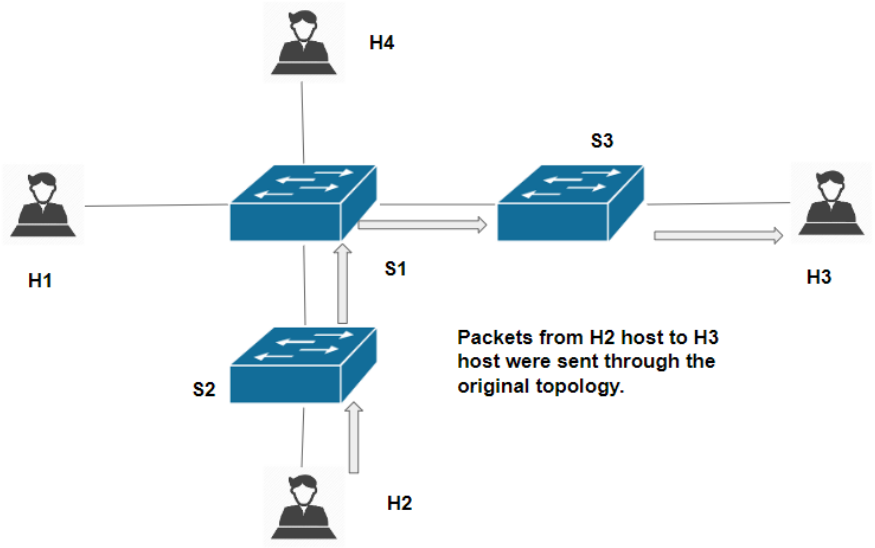

Fig 1: The Topology view before the attack.

In this topology, we will inject a fake link from Switch S2 to Switch S3 through a python script for changing the links in the P4app.json file, which specifies how a P4 program should be compiled and mapped onto a specific target architecture. In this file we can change links through python script, which changes the topology in topology.db file. Here the topology.db file is a database file used by the P4Utils library in Mininet to represent the topology of a network. It stores the configuration and connectivity information for all the switches, hosts, and links in the network. The topology.db file is generated by running the p4app run command, which reads the p4app.json file and generates the network topology based on the information provided in the file. And now in this topology the thrift ports are assigned through P4 runtime API. When a P4 switch starts up, it opens a set of Thrift ports that are used to communicate with the controller program. The number and port assignments for these Thrift ports are typically specified in the switch configuration file or in the P4 program itself. The controller program running on the host machine connects to the switch using the Thrift port number assigned to the switch by the P4 runtime API, and sends configuration and control messages to the switch using the P4 runtime API. Where the packets from host H2 to host H3 in the initial topology are sent through switch S1 but in poisoned topology the packets are sent from switch S2 to switch S3 link which is the fabricated link as given in the below fig 2:

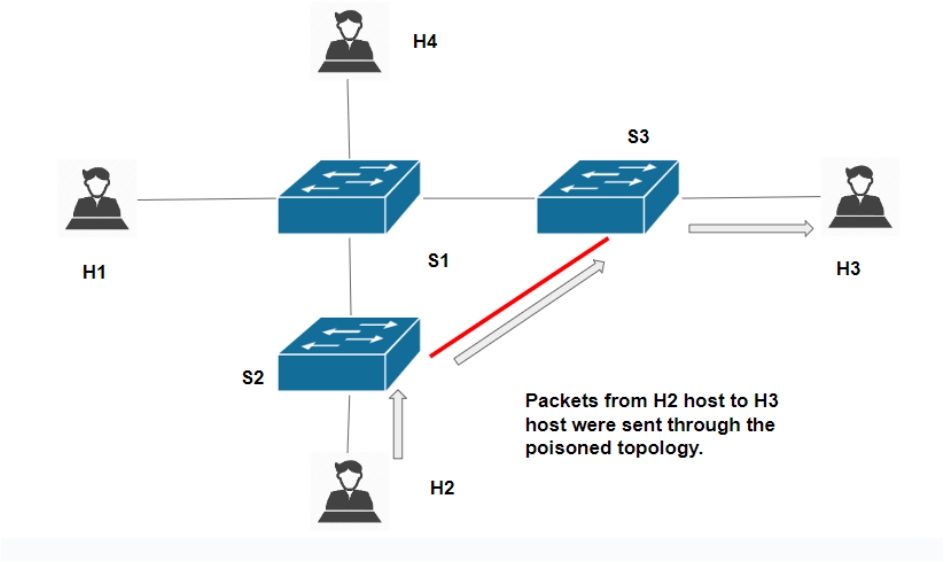

Fig 2: The Poisoned view of the topology with the LFA .

2.2 Detecting and Mitigation of LFA

We can detect and mitigate the attack through the following algorithms where the topo_file path p4app.json path. Where the original p4app.json file with the original topology is defined and if there are any changes in the topology then the controller detects and mitigates the attack and then the original topology is reverted.

ALGORITHMS

Algorithm 1: Link Fabrication Attack Detection Algorithm
Input:
Topology file path (topo_file)
Output:
A message indicating whether the topology has been changed or not

- Read the original topology file from topo_file1
- Read the modified topology file from topo_file
- If original_topo is equal to modified_topo, output "The topology is the same"
- Otherwise, output "The topology is changed"

Algorithm 2: Routing Controller Algorithm with Topology Change Detection

Input:

The network topology JSON file path

The modified network topology JSON file path

Output:

Configures the network switches with routing rules

- Create a new instance of the Topology class with the network topology JSON file path.
- Create an empty dictionary called "controllers" to store instances of the SimpleSwitchAPI class for each switch in the topology.
- Call the "detect" function to check if the network topology has been modified from the original version.
- If the topology has been modified, continue to the next step. Otherwise, exit the function.
- For each switch in the topology, create an instance of the SimpleSwitchAPI class and add it to the "controllers" dictionary.
- Call the "reset_states" function to reset the switch states to their default values.
- Call the "set_table_defaults" function to set the default routing rule for each switch to drop all traffic.
- Call the "route" function to configure the routing rules for the switches based on the network topology.
- Exit the function

CHAPTER 3: STIMULATION RESULTS AND DISCUSSION

In order to perform the attack we consider the p4app.json file which describes the topology that we want to create with the help of Mininet and the P4-Utils package. To create the topology created in p4app we need to call p4run which by default will check if the file p4app.json exists in the path:

```
sudo p4run
```

We will have 6 links in the original topology as in the fig 3.

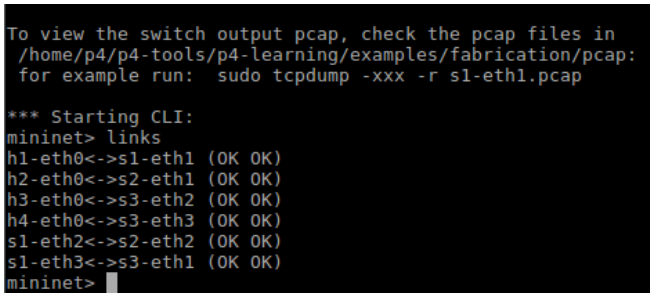

Fig 3: Links before the attack.

Here, in the p4app.json file we will change the links through a python script. After changing the links we will re-run the sudo p4run command. Where the topology gets poisoned and a new link from switch S2 to switch S3 will be injected as in fig 4.

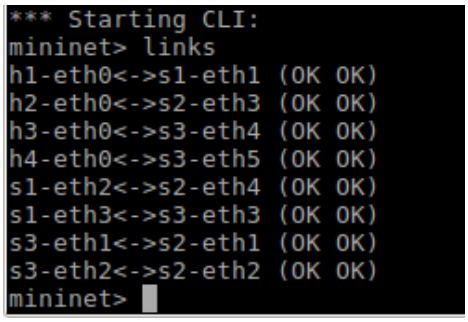

Fig 4: Links after the attack.

We can check the reachability of all the hosts through the following command

```
pingall
```

This reachability as in fig 5 can be done by running the controller.py program which is a custom routing controller program used for routing the packets. Where we can manually add the commands for routing through commands.txt file in p4 program.

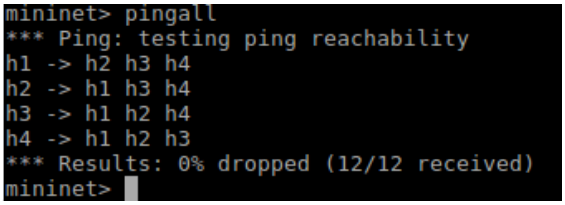

Fig 5: Reachability of all the hosts.

We set up the xterm nodes for testing the reachability of the packets between the hosts through the fabricated link and we will send the packets through python scripts which is IP forwarding of the packets as per in fig 6:

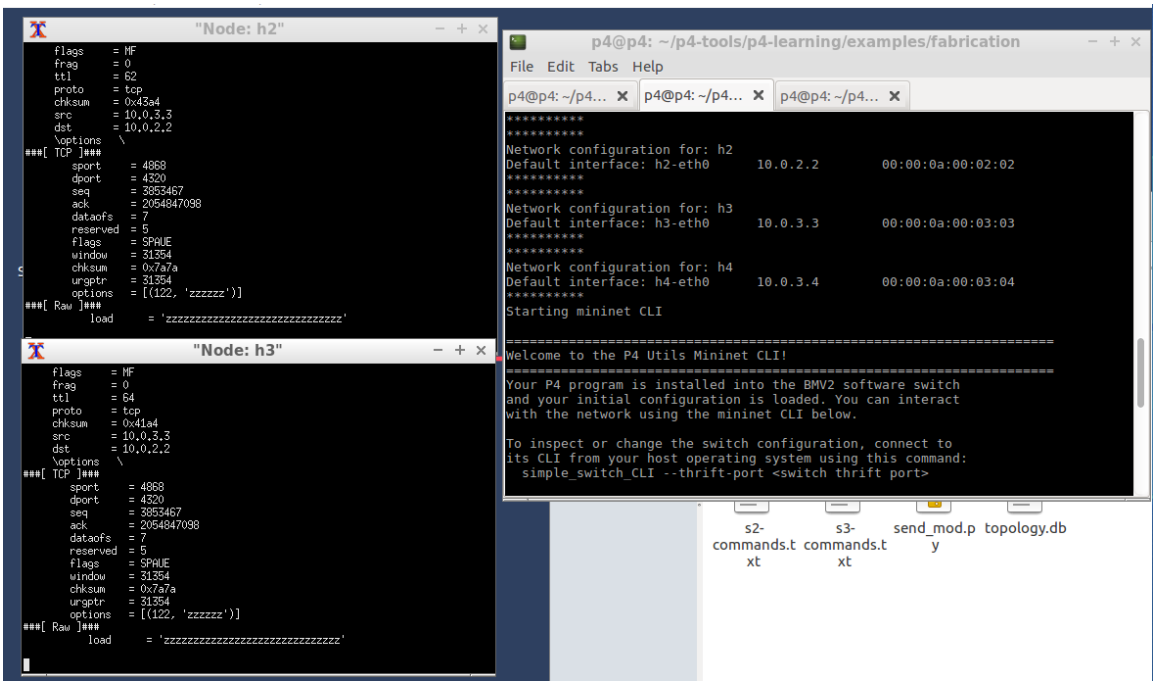

Fig 6: Setting up xterm nodes and sending the packets through python scripts.

And when we trace the packets through wireshark with following command in another tab

```
sudo wireshark
```

We can see the packets from host H2 are going through switch S2 and switch S3 via the fabricated link to the host H3 as in fig 7.

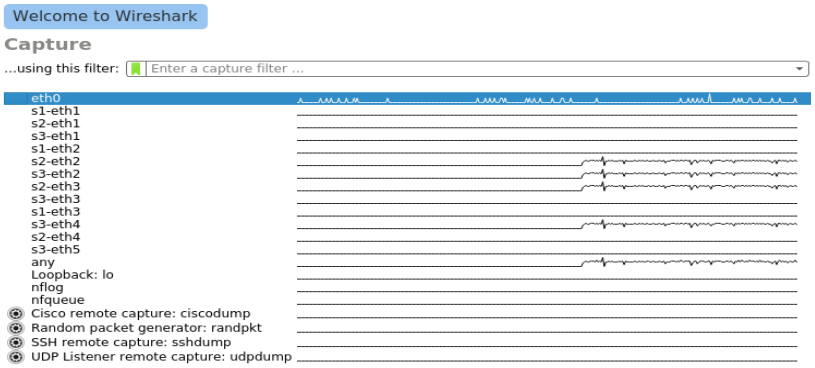

Fig 7: tracing the packets in the LFA environment.

But as per the original topology the packets should go through switch S2, switch S1 and then through switch S3 like in fig 8.

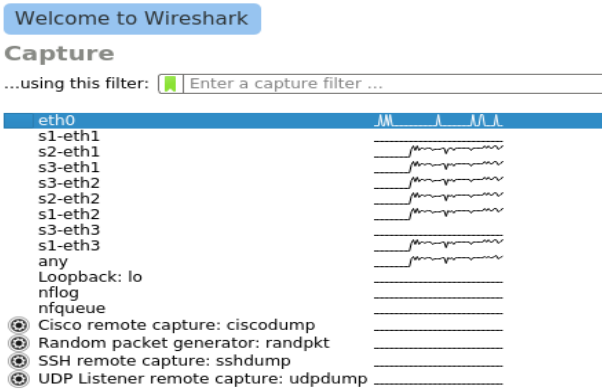

Fig 8: Tracing the packets through the original topology.

And when we run the controller with a detection algorithm the output will be given as “topology is changed” as per fig 9.

```
p4@p4:~/p4-tools/p4-learning/examples/attack/p$ sudo python routing-controller.py
Setting default action of ipv4_lpm
action:              drop
runtime data:
Setting default action of ipv4_lpm
action:              drop
runtime data:
Setting default action of ipv4_lpm
action:              drop
runtime data:
The topology is changed
```

Fig 8: Detection of the fabrication attack.

CHAPTER 4: CONCLUSION

Here in this project we performed LFA in p4 through mininet emulation. In conclusion, LFA is a serious security threat in P4-based networks deployed using Mininet. This attack involves creating fake links or modifying the properties of existing links in the network topology, in order to manipulate the routing decisions made by the network controller. By adopting a proactive approach to network security, it is possible to reduce the risk of LFA s and other security threats in P4-based networks deployed using Mininet. The security mechanism for the attack can be improved even more through lightweight encryption and other security measures.

REFERENCES

[1] M. -K. Shin, K. -H. Nam and H. -J. Kim, "Software-defined networking (SDN): A reference architecture and open APIs," 2012 International Conference on ICT Convergence (ICTC), Jeju, Korea (South), 2012, pp. 360-361, doi: 10.1109/ICTC.2012.6386859.

[2] D. Smyth, S. McSweeney, D. O'Shea and V. Cionca, "Detecting Link Fabrication Attacks in Software-Defined Networks," 2017 26th International Conference on Computer Communication and Networks (ICCCN), Vancouver, BC, Canada, 2017, pp. 1-8, doi: 10.1109/ICCCN.2017.8038435.

[3] F. Keti and S. Askar, "Emulation of Software Defined Networks Using Mininet in Different Simulation Environments," 2015 6th International Conference on Intelligent Systems, Modelling and Simulation, Kuala Lumpur, Malaysia, 2015, pp. 205-210, doi: 10.1109/ISMS.2015.46.

[4] Joseph, K.; Eyobu, O.S.; Kasyoka, P.; Oyana, T.J. A Link Fabrication Attack Mitigation Approach (LiFAMA) for Software Defined Networks. Electronics 2022, 11, 1581.

[5] S. Soltani, M. Shojafar, H. Mostafaei, Z. Pooranian and R. Tafazolli, "Link Latency Attack in Software-Defined Networks," 2021 17th International Conference on Network and Service Management (CNSM), Izmir, Turkey, 2021, pp. 187-193, doi: 10.23919/CNSM52442.2021.9615598.

[6] Baidya, Sonali Sen, and Rattikorn Hewett. "Link Discovery Attacks in Software-Defined Networks: Topology Poisoning and Impact Analysis." J. Commun. 15, no. 8 (2020): 596-606.

[7] Huang, Xinli, Peng Shi, Yufei Liu, and Fei Xu. "Towards trusted and efficient SDN topology discovery: A lightweight topology verification scheme." Computer Networks 170 (2020): 107119.

[8] F. Hauser, M. Schmidt, M. Häberle and M. Menth, "P4-MACsec: Dynamic Topology Monitoring and Data Layer Protection With MACsec in P4-Based SDN," in IEEE Access, vol. 8, pp. 58845-58858, 2020, doi: 10.1109/ACCESS.2020.2982859.