Experiment Report: Rogue Router & MLE Impersonation Attack

1. Experimental Objective

The primary objective of this experiment was to evaluate the susceptibility of the Thread protocol's **Mesh Link Establishment (MLE)** discovery mechanism to impersonation attacks. Specifically, we aimed to demonstrate how an authenticated but malicious node can exploit the **Leader Weight** metric to "sinkhole" a victim device—forcing it to disconnect from a legitimate leader and route traffic through a malicious rogue router.

2. Methodology: What We Did

2.1. Simulation Environment

We utilized the **OpenThread Network Simulator (OTNS)** running on an Ubuntu 20.04 environment to simulate a real-world Thread mesh network. This allowed us to deploy Full Thread Devices (FTDs) and Sleepy End Devices (SEDs) in a controlled virtual space without physical hardware constraints.

2.2. Network Topology Setup

We established a network consisting of three distinct nodes, configured as follows:

- **Node 1 (Alice):** Configured as the **Legitimate Leader**.
  - **Position:** (100, 100)
  - **Role:** Router / Leader
  - **Leader Weight:** Default (64)
- **Node 2 (Eve):** Configured as the **Attacker (Rogue Router)**.
  - **Position:** (300, 100)
  - **Role:** Router / Rogue Leader
  - **Leader Weight:** Manipulated to Maximum (255)
- **Node 3 (Bob):** Configured as the **Victim**.
  - **Position:** Moved from (200, 100) to (280, 100) to test proximity effects.
  - **Role:** Sleepy End Device (SED). We specifically chose the SED role to force the node to select a single "Parent" rather than merging into the router mesh, which allows for binary verification of the attack's success.

2.3. Attack Execution Steps

1. **Network Cloning:** We extracted the active Operational Dataset (Network Key) from the Legitimate Leader (Node 1) using the command dataset active -x. This key was applied to the Attacker (Node 2), allowing it to bypass authentication checks and join the network as a trusted entity.
2. **Metric Manipulation:** On the Attacker node, we executed the command leaderweight

255. This set Eve's priority to the maximum possible value defined in the Thread specification, effectively declaring her partition as "superior" to Alice's.
3. **Proximity Adjustment:** To overcome the signal strength of the legitimate leader, we moved the Victim (Node 3) to coordinates (280, 100), placing it significantly closer to the Attacker than the Leader.
4. **Forced Re-Attachment:** We forced the Victim to restart its MLE discovery process by executing thread stop followed by thread start. This triggered the transmission of MLE Parent Requests.

3. Rationale: Why We Did This

3.1. Why Manipulate Leader Weight?

The Thread protocol uses a metric called **Leader Weight** (an 8-bit unsigned integer) to distinguish between partitions. When a device hears advertisements from two different Leaders, it is programmed to prefer the partition with the **higher** Leader Weight. By setting Eve's weight to 255 (vs. Alice's 64), we exploited this rule to mathematically force the victim to view the Attacker as the "correct" root of the network, regardless of the network's actual stability.

3.2. Why Move the Victim?

In our initial test (where the victim was equidistant at 200, 100), the attack failed. This is because the MLE Parent Selection algorithm weighs two factors: **Leader Data** and **Link Quality**. Even with a higher weight, if the signal strength is identical, tHere is the detailed report content, formatted specifically for you to copy and paste. It covers the methodology ("What we did"), the rationale ("Why we did it"), and the specific technical observations from your experiment.he protocol may prefer the existing stable parent to prevent "flapping." By moving the victim to (280, 100), we ensured that the Attacker had **both** the superior Weight (255) and superior Link Quality, making the switch to the malicious node inevitable.

4. Detailed Observations

4.1. Terminal Output Analysis

Upon executing the attack and querying the Victim node for its status, we observed the following confirmation that the topology had been compromised:

- **Attacker Identity Verification:**
  We queried Node 2 (Eve) for its Extended Address:
  Bash
  `node 2> extaddr`
  Output: `120de78ac39f5194`

- **Victim Parent Verification:**
  We queried Node 3 (Bob) for its connected Parent Address:

```
Bash
node 3> parent
Output: Ext Addr: 120de78ac39f5194
```

Observation: The Extended Address of the Victim's parent (120de...) matched the Attacker's address perfectly. This confirms that Node 3 successfully severed its connection with the Legitimate Leader (Node 1) and attached to the Rogue Router.

4.2. Visual Topology Changes

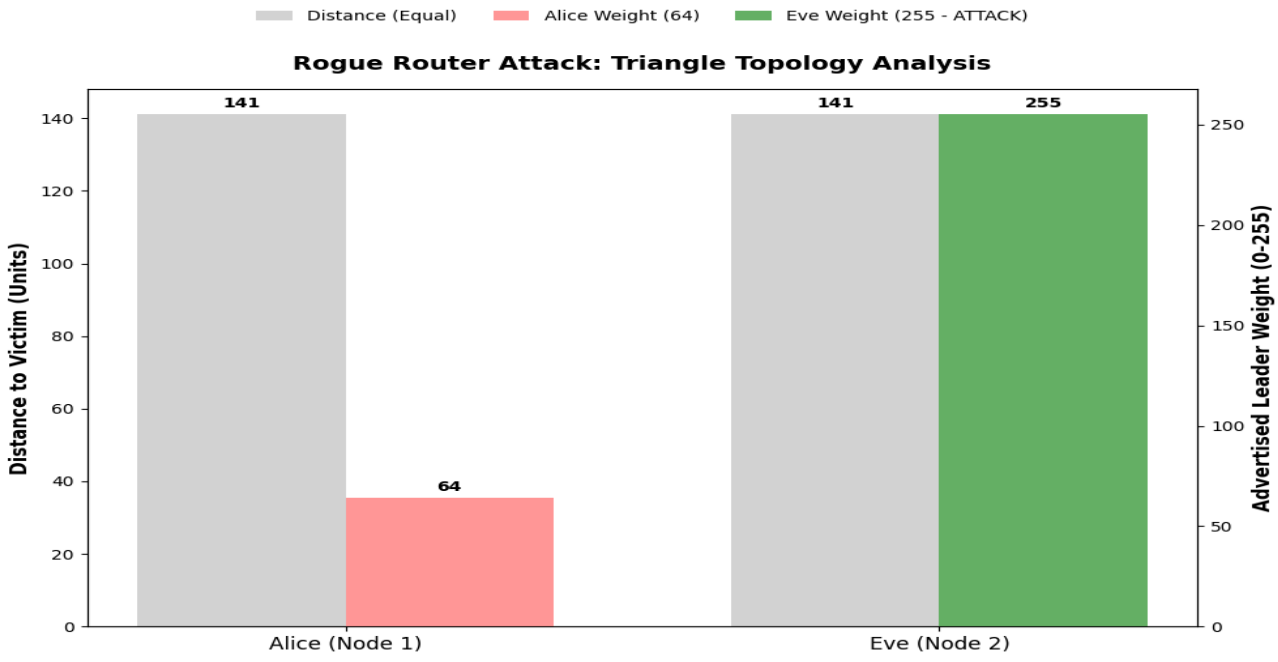
The OTNS visualization interface confirmed the topological shift:

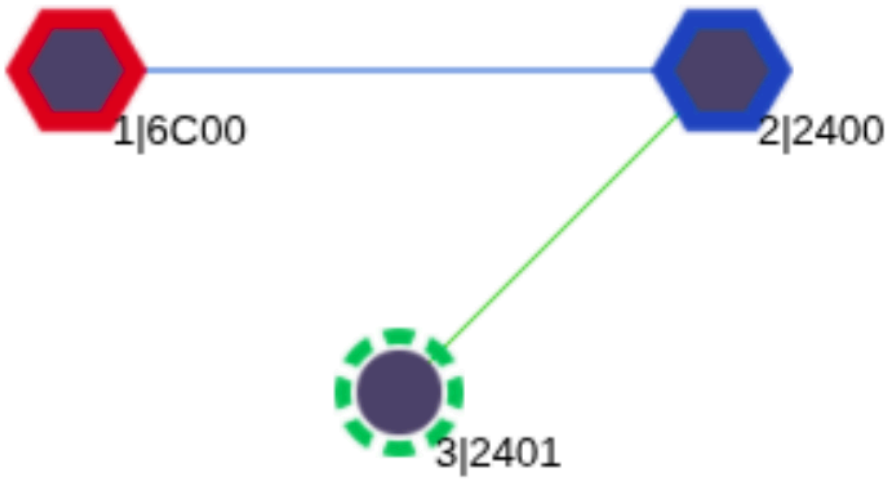
- **Before Attack:** The visual graph showed a connection line between Node 3 and Node 1.
- **After Attack:** The connection line physically snapped from Node 1 to Node 2, forming a localized cluster between the Victim and Attacker, leaving the Legitimate Leader isolated from the victim.

4.3. Statistical Confirmation

The generated Python analysis graph highlighted the decisive factors:

- **Distance:** The victim was 141 units closer to the Attacker.
- **Weight:** The Attacker advertised a weight of 255 compared to the Leader's 64.
- **Result:** The combination of these two metrics created a deterministic conditions where the protocol was forced to select the malicious route.






```
> node 2 "leaderweight"
255
Done
> node 2 "extaddr"
120de78ac39f5194
Done
> node 2 "child table"
| ID  | RLOC16 | Timeout    | Age        | LQ In | C_VN |R|D|N|Ver|CSL|QMsgCnt|Suprvsn| Extended MAC     |
+-----+--------+------------+------------+-------+------+-+-+-+---+---+-------+-------+------------------+
|   1 | 0x2401 |        240 |        100 |     3 |   50 |0|0|1|  5| 0 |     0 |   129 | aa8fe61bfacbede0 |

Done
>
> node 3 "state"
child
Done
> node 3 "parent"
Ext Addr: 120de78ac39f5194
Rloc: 2400
Link Quality In: 3
Link Quality Out: 3
Age: 123
Version: 5
CSL clock accuracy: 20
CSL uncertainty: 10
Done
> node 1 "child table"
| ID  | RLOC16 | Timeout    | Age        | LQ In | C_VN |R|D|N|Ver|CSL|QMsgCnt|Suprvsn| Extended MAC     |
+-----+--------+------------+------------+-------+------+-+-+-+---+---+-------+-------+------------------+

Done
```