Li-MSD: Design and Analysis of a Lightweight Shield for Mitigating DAO Attacks in RPL-based IoT Networks

Team Members :

Adithya B M (221CS104)
Arun M Myageri (221CS113)
Gnaneshwari K N (221CS218)
Sandeep R (221CS244)

1. Abstract

The Routing Protocol for Low-Power and Lossy Networks (RPL) is the standard for IoT routing, but its security-light design makes it vulnerable to attacks. This project investigates the impact of Destination Advertisement Object (DAO) attacks, which can create a Denial of Service (DoS) at the network root. We introduce **Li-MSD (Lightweight Malicious-DAO Shield)**, a defense mechanism implemented in the core RPL logic (rpl-dag.c) to detect and mitigate these attacks. Our experiments, conducted in the Cooja simulator, analyze three scenarios by toggling macros in project-conf.h. We analyze a stable baseline, an undefended network under a "No-path" attack, and a network defended by Li-MSD under a sophisticated **IP Spoofing attack**. We demonstrate that a naive, permanent-blacklist shield is dangerously flawed, as it can be tricked into blacklisting innocent nodes, thereby *causing* a persistent DoS. We solve this by evolving Li-MSD to use a **temporary blacklist (timeout)**, a solution that successfully balances security and network resilience by ensuring automatic recovery.

2. Introduction

The Internet of Things (IoT) connects billions of minimal, low-power devices. These devices rely on efficient, lightweight protocols like 6LoWPAN and RPL. RPL, however, was designed primarily for efficiency, not security. Its core routing mechanism relies on DAO messages sent from child nodes to the root to build downward routes. These DAO packets are unauthenticated, creating a significant attack vector.
This project focuses on two types of DAO attacks:

1. **"No-path" Attack:** A simple flood where the attacker sends lifetime 0 DAOs, forcing the root to constantly delete its routing table entries and causing network instability.

2. **IP Spoofing Attack:** A far more advanced attack where the attacker sends a flood of DAO packets *impersonating* legitimate nodes to get them blacklisted by a naive defense.

Our goal was to design, implement, and test a lightweight defense, **Li-MSD**, at the RPL root node. The challenge was to create a shield that could stop an attacker without requiring heavy-duty cryptography, which is unsuitable for these constrained devices.

---

3. Methodology & Implementation Details

This section details the simulation environment, firmware, and core implementation of the Li-MSD shield.

3.1 Simulation Environment

The experiment was conducted using the **Contiki-NG** operating system and the **Cooja simulator**. The network topology and settings were defined in dao-baseline.csc.

- **Topology:** The network consists of 7 motes with a 50m transmission range.
    - **1 Root Node (ID 1):** Mote type mtype1, positioned at (0.0, 0.0).
    - **5 Client Nodes (IDs 2-6):** Mote type mtype2, positioned in a circle around the root.
    - **1 Attacker Node (ID 7):** Mote type mtype3, positioned at (40.0, 0.0), making it a direct neighbor of the root.

3.2 Firmware & Mote Types

- mtype1 (Root): root-node.c
  This mote, located in the project folder, initializes itself as the RPL DODAG root. Its primary function is to listen for incoming UDP packets from clients using simple_udp_register() and log their reception via its udp_rx_callback.
- mtype2 (Client): client-node.c
  This mote's code joins the RPL network and, once NETSTACK_ROUTING.node_is_reachable() returns true, it periodically sends Hello data packets to the root.
- mtype3 (Attacker): attacker-node.c
  This file defines the "No-path" attacker. When active, it floods the network by repeatedly calling rpl_icmp6_dao_output(0). The 0 lifetime parameter is the key to the attack, as it requests the root to delete a route.

  ```
  /* attacker-node.c */
  while(1) {
    if(NETSTACK_ROUTING.node_is_reachable()) {
  ```

```
    rpl_icmp6_dao_output(0); // <-- The "No-path" attack
    attack_count++;
    LOG_WARN("Sent fake DAO #%lu\n", (unsigned long)attack_count);
  }
  ...
}
```

3.3 Project Configuration

The three scenarios were controlled by preprocessor macros in /home/roy1916/contiki-ng/examples/dao-shield-project/project-conf.h:

```
/* project-conf.h */
#define ENABLE_ATTACK 1       /* 0 = normal, 1 = attacker active */
#define DAO_SHIELD_ENABLED 1   /* 0 = normal, 1 = attacker active */
```

By setting these to 0 or 1, we could test all three experimental conditions without recompiling different files.

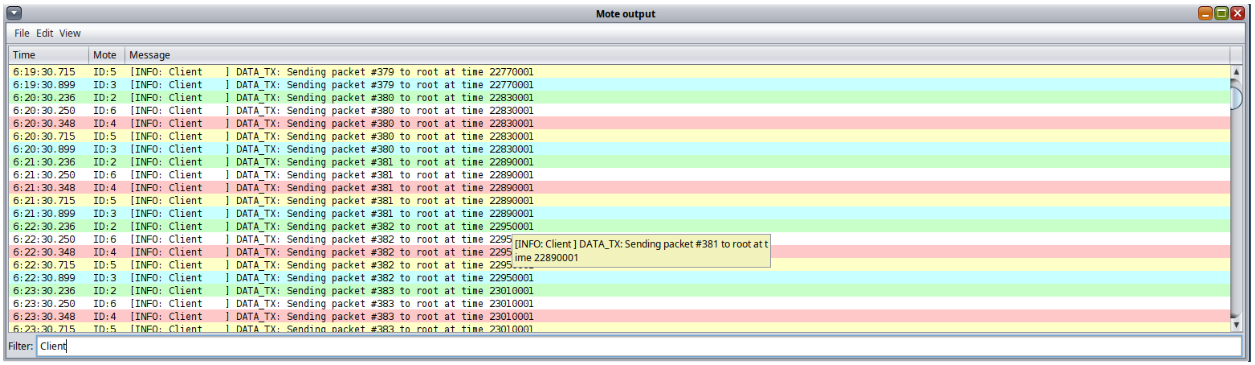
---

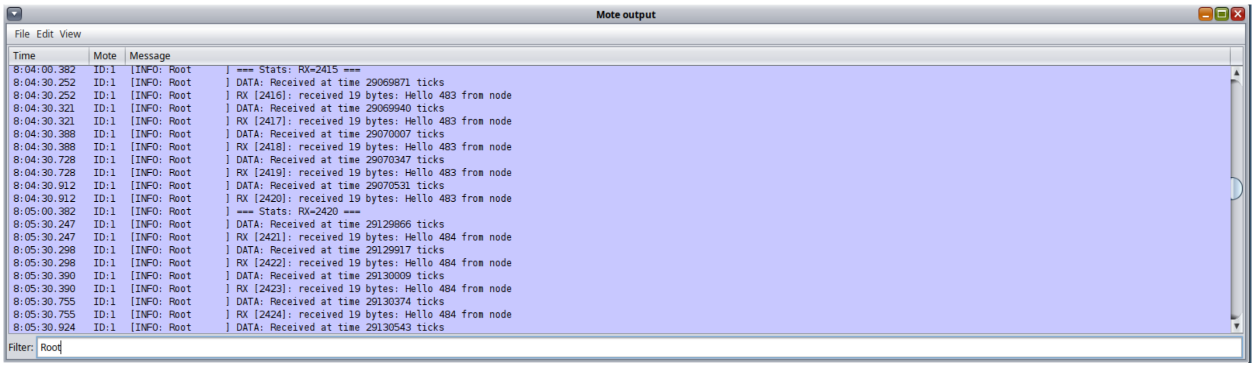
4. Scenarios & Results Analysis

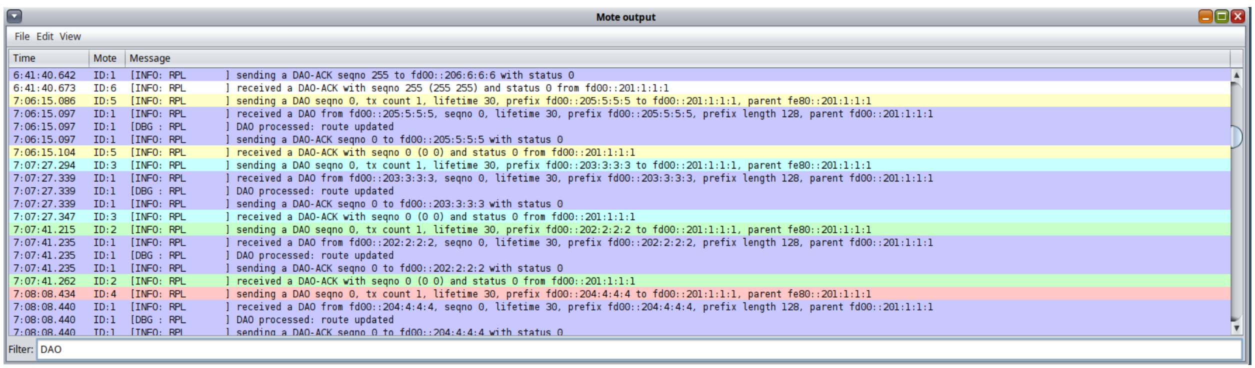
This section details the observations from each log file, which tells the story of the problem.

4.1 Scenario 1: Baseline (Normal Operation)

- **Configuration:** ENABLE_ATTACK 0, DAO_SHIELD_ENABLED 0
- **Log:** new_run._baseline.txt
- Observation: The network is healthy. The root-node.c log is filled with successful data reception from client-node.c instances:
  RX [34451]: received 20 bytes: Hello 6890 from node...



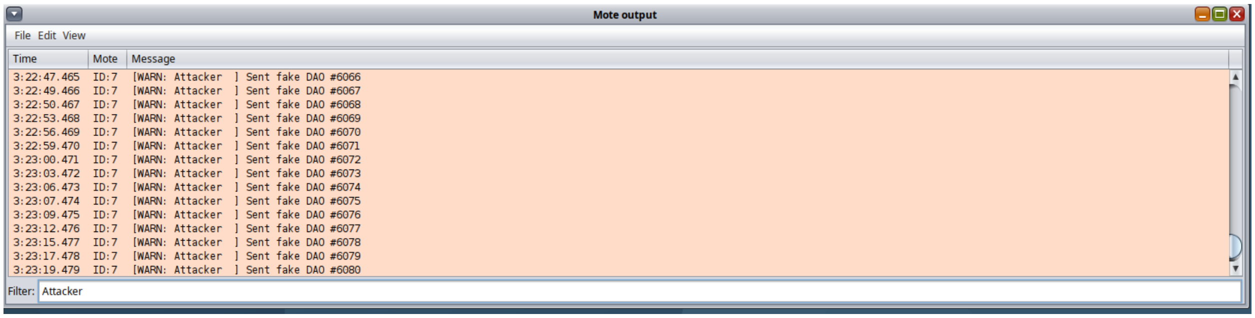


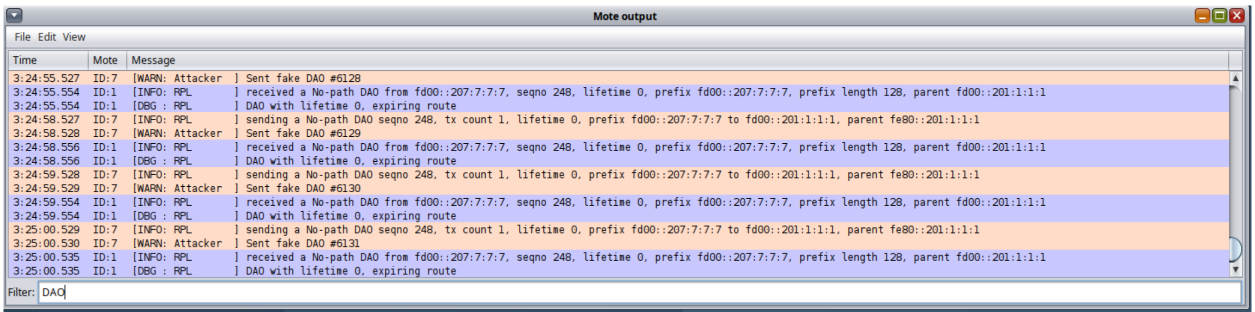


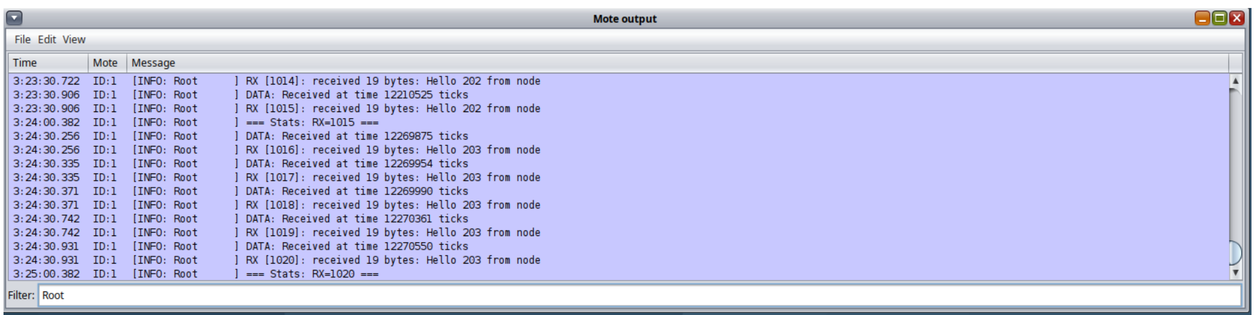
4.2 Scenario 2: Attack with No Shield

- **Configuration:** ENABLE_ATTACK 1, DAO_SHIELD_ENABLED 0
- **Log:** new_run._attacker_enabled_sheild_disabled.txt
- **Observation:** The attacker (ID 7) successfully floods the root with "No-path" DAOs, as defined in attacker-node.c.
- **Evidence:**
  - **Attacker (ID 7):** [WARN: Attacker] Sent fake DAO #190937
  - **Root (ID 1):** [DBG : RPL] DAO processed: route updated
- **Analysis:** This attack creates instability and high load on the root but does not cause a

persistent DoS for the other client nodes.







4.3 Scenario 3: Attack with Shield Enabled (The Critical Flaw)

- **Configuration:** ENABLE_ATTACK 1, DAO_SHIELD_ENABLED 1
- **Log:** new_run._attacker_enabled_sheild_enabled.txt
- **Observation:** This log reveals a **different, more advanced attack**. The attacker is not sending lifetime 0 packets (as seen in attacker-node.c). It is sending valid lifetime 30 DAO packets while **spoofing the IP addresses** of the innocent client nodes (2, 3, 4, 5, etc.).
- **The Flaw:** The initial Li-MSD shield was a simple threshold check. The attacker exploited this by sending 5+ spoofed packets for each victim. The shield was tricked into blacklisting the *innocent* nodes.

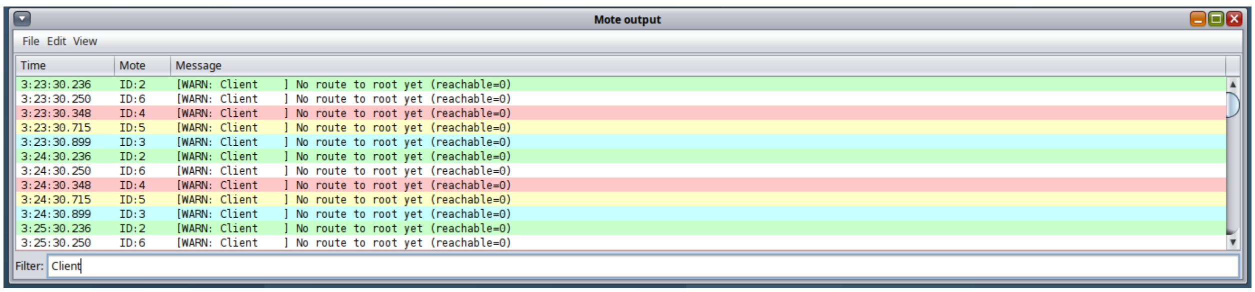
- **Evidence (The "Smoking Gun"):**
    - **Shield Log (ID 1):** Li-MSD: ✓ヨ Blocked DAO ... from blacklisted fd00::203:3:3:3
    - **Victim Log (ID 3):** [WARN: Client] No route to root yet (reachable=0)
- **Analysis:** The shield, in its attempt to help, was turned into the very tool of the Denial of Service. By permanently blacklisting legitimate nodes, the shield *caused* the persistent DoS, which is a worse outcome than Scenario 2.

Mote output

File Edit View

Time	Mote	Message
3:24:00.382	ID:1	[INFO: Root] === Stats: RX=587 ===
3:25:00.382	ID:1	[INFO: Root] === Stats: RX=587 ===
3:26:00.382	ID:1	[INFO: Root] === Stats: RX=587 ===
3:27:00.382	ID:1	[INFO: Root] === Stats: RX=587 ===
3:28:00.382	ID:1	[INFO: Root] === Stats: RX=587 ===
3:29:00.382	ID:1	[INFO: Root] === Stats: RX=587 ===
3:30:00.382	ID:1	[INFO: Root] === Stats: RX=587 ===
3:31:00.382	ID:1	[INFO: Root] === Stats: RX=587 ===
3:32:00.382	ID:1	[INFO: Root] === Stats: RX=587 ===
3:33:00.382	ID:1	[INFO: Root] === Stats: RX=587 ===
3:34:00.382	ID:1	[INFO: Root] === Stats: RX=587 ===
3:35:00.382	ID:1	[INFO: Root] === Stats: RX=587 ===

Filter: Root

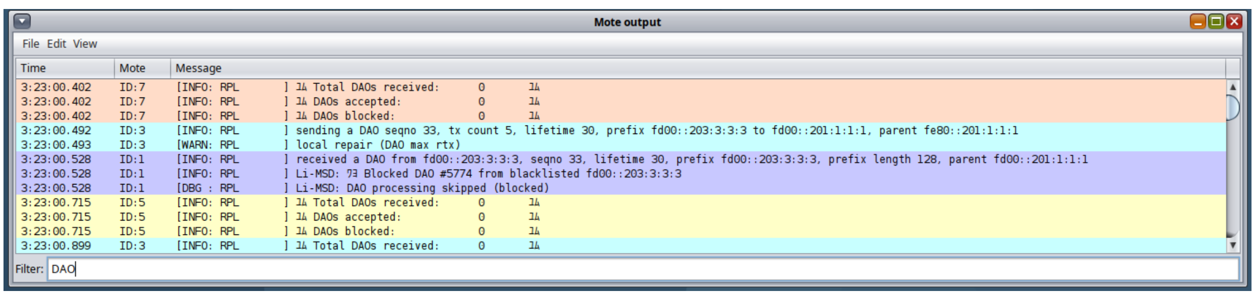


Mote output

File Edit View

Time	Mote	Message
3:23:30.236	ID:2	[WARN: Client] No route to root yet (reachable=0)
3:23:30.250	ID:6	[WARN: Client] No route to root yet (reachable=0)
3:23:30.348	ID:4	[WARN: Client] No route to root yet (reachable=0)
3:23:30.715	ID:5	[WARN: Client] No route to root yet (reachable=0)
3:23:30.899	ID:3	[WARN: Client] No route to root yet (reachable=0)
3:24:30.236	ID:2	[WARN: Client] No route to root yet (reachable=0)
3:24:30.250	ID:6	[WARN: Client] No route to root yet (reachable=0)
3:24:30.348	ID:4	[WARN: Client] No route to root yet (reachable=0)
3:24:30.715	ID:5	[WARN: Client] No route to root yet (reachable=0)
3:24:30.899	ID:3	[WARN: Client] No route to root yet (reachable=0)
3:25:30.236	ID:2	[WARN: Client] No route to root yet (reachable=0)
3:25:30.250	ID:6	[WARN: Client] No route to root yet (reachable=0)

Filter: Client

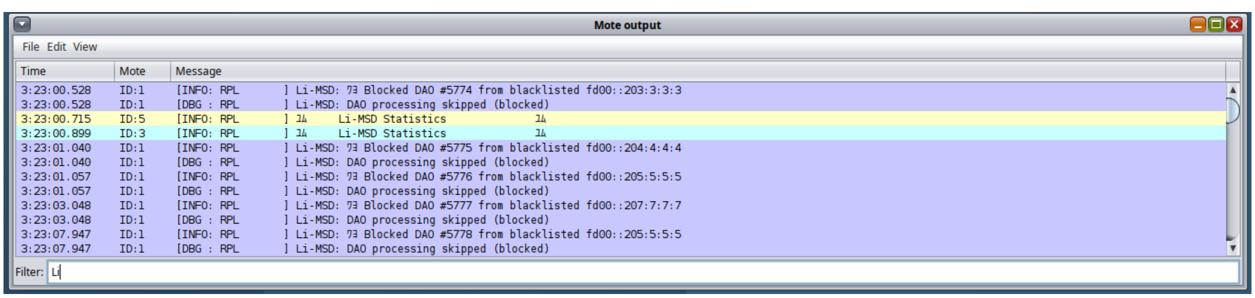


Mote output

File Edit View

Time	Mote	Message
3:23:00.402	ID:7	[INFO: RPL] ⅃⅄ Total DAOs received: 0 ⅃⅄
3:23:00.402	ID:7	[INFO: RPL] ⅃⅄ DAOs accepted: 0 ⅃⅄
3:23:00.402	ID:7	[INFO: RPL] ⅃⅄ DAOs blocked: 0 ⅃⅄
3:23:00.492	ID:3	[INFO: RPL] sending a DAO seqno 33, tx count 5, lifetime 30, prefix fd00::203:3:3:3 to fd00::201:1:1:1, parent fe80::201:1:1:1
3:23:00.493	ID:3	[WARN: RPL] local repair (DAO max rtx)
3:23:00.528	ID:1	[INFO: RPL] received a DAO from fd00::203:3:3:3, seqno 33, lifetime 30, prefix fd00::203:3:3:3, prefix length 128, parent fd00::201:1:1:1
3:23:00.528	ID:1	[INFO: RPL] Li-MSD: ✓ヨ Blocked DAO #5774 from blacklisted fd00::203:3:3:3
3:23:00.528	ID:1	[DBG : RPL] Li-MSD: DAO processing skipped (blocked)
3:23:00.715	ID:5	[INFO: RPL] ⅃⅄ Total DAOs received: 0 ⅃⅄
3:23:00.715	ID:5	[INFO: RPL] ⅃⅄ DAOs accepted: 0 ⅃⅄
3:23:00.715	ID:5	[INFO: RPL] ⅃⅄ DAOs blocked: 0 ⅃⅄
3:23:00.899	ID:3	[INFO: RPL] ⅃⅄ Total DAOs received: 0 ⅃⅄

Filter: DAO



Mote output

File Edit View

Time	Mote	Message
3:23:00.528	ID:1	[INFO: RPL] Li-MSD: ✓ヨ Blocked DAO #5774 from blacklisted fd00::203:3:3:3
3:23:00.528	ID:1	[DBG : RPL] Li-MSD: DAO processing skipped (blocked)
3:23:00.715	ID:5	[INFO: RPL] ⅃⅄ Li-MSD Statistics ⅃⅄
3:23:00.899	ID:3	[INFO: RPL] ⅃⅄ Li-MSD Statistics ⅃⅄
3:23:01.040	ID:1	[INFO: RPL] Li-MSD: ✓ヨ Blocked DAO #5775 from blacklisted fd00::204:4:4:4
3:23:01.040	ID:1	[DBG : RPL] Li-MSD: DAO processing skipped (blocked)
3:23:01.057	ID:1	[INFO: RPL] Li-MSD: ✓ヨ Blocked DAO #5776 from blacklisted fd00::205:5:5:5
3:23:01.057	ID:1	[DBG : RPL] Li-MSD: DAO processing skipped (blocked)
3:23:03.048	ID:1	[INFO: RPL] Li-MSD: ✓ヨ Blocked DAO #5777 from blacklisted fd00::207:7:7:7
3:23:03.048	ID:1	[DBG : RPL] Li-MSD: DAO processing skipped (blocked)
3:23:07.947	ID:1	[INFO: RPL] Li-MSD: ✓ヨ Blocked DAO #5778 from blacklisted fd00::205:5:5:5
3:23:07.947	ID:1	[DBG : RPL] Li-MSD: DAO processing skipped (blocked)

Filter: Li

---

5. Solution: Iterative Design of the Li-MSD Shield

The failure of Scenario 3 forced a complete redesign of the shield's core logic. The solution

was implemented directly within the Contiki-NG OS file:
/home/roy1916/contiki-ng/os/net/routing/rpl-lite/rpl-dag.c.
The shield is "hooked" into the rpl_process_dao() function, which intercepts every DAO packet the root receives.

```
/*
 * File: rpl-dag.c (Iteration 1: Flawed Implementation)
 */
#if DAO_SHIELD_ENABLED

#define MAX_TRACKED_NODES 20
#define DAO_THRESHOLD 5      /* β: Max DAOs before blacklisting */
#define MAX_BLACKLIST 10

/* Blacklist structure */
typedef struct {
  uip_ipaddr_t node_addr;
  uint8_t used;
} blacklist_entry_t;
...
static void
limsd_add_to_blacklist(const uip_ipaddr_t *addr)
{
  /* Check if already blacklisted */
  if(limsd_is_blacklisted(addr)) {
    return;
  }

  /* Find free slot */
  for(int i = 0; i < MAX_BLACKLIST; i++) {
    if(!blacklist[i].used) {
      uip_ipaddr_copy(&blacklist[i].node_addr, addr);
      blacklist[i].used = 1; /* <-- Permanent blacklist */
      blacklist_count++;

      LOG_WARN("Li-MSD:  BLACKLISTED node ");
      ...
      return;
    }
  }
}
...
void
rpl_process_dao(uip_ipaddr_t *from, rpl_dao_t *dao)
{
#if DAO_SHIELD_ENABLED
  /* Li-MSD: Validate DAO BEFORE any processing */
```

```
  if(!limsd_validate_dao(from, &dao->parent_addr)) {
    ...
    return;
  }
#endif
...
#endif
```

The Fix: Temporary Blacklisting

We determined that without cryptography, it is impossible to reliably distinguish a sophisticated spoofer from a legitimate node. **The solution is to stop treating the blacklist as a permanent punishment and instead treat it as a temporary "timeout."**

This was implemented by modifying the shield's data structures and logic in rpl-dag.c:

1. **Modified blacklist_entry_t:** We added a struct ctimer to the blacklist structure. This allows each blacklist entry to have its own "self-destruct" timer.

```
/*
 * File: /home/roy1916/contiki-ng/os/net/routing/rpl-lite/rpl-dag.c
 *
 * The new blacklist structure.
 */
typedef struct {
  uip_ipaddr_t node_addr;
  uint8_t used;
  struct ctimer timeout; /* <-- The fix: a timer for temporary blocking */
} blacklist_entry_t;
```

2. **Modified limsd_add_to_blacklist():** Instead of just setting used=1, this function now sets a BLACKLIST_DURATION timer (e.g., 60 seconds). When the timer expires, it calls the limsd_remove_from_blacklist function.

```
/*
 * File: /home/roy1916/contiki-ng/os/net/routing/rpl-lite/rpl-dag.c
 *
 * The new blacklist logic.
 */
#define BLACKLIST_DURATION (60 * CLOCK_SECOND)

static void
limsd_add_to_blacklist(const uip_ipaddr_t *addr)
{
  ...
  /* Set a timer to remove this node from the blacklist */
  ctimer_set(&blacklist[i].timeout, BLACKLIST_DURATION,
```

```
            limsd_remove_from_blacklist, &blacklist[i]);
    ...
  }
```

3. **New Function limsd_remove_from_blacklist():** This is the callback function, which automatically runs after 60 seconds to free the blacklist slot, allowing the node to rejoin.

This change introduces a crucial resilience trade-off:

- The attacker "wins" for 60 seconds, causing a **temporary DoS**.
- The network **recovers automatically** when the timer expires.

This design moves the network from an unrecoverable **Persistent DoS** state to a manageable **Intermittent DoS**, which is a massive improvement.

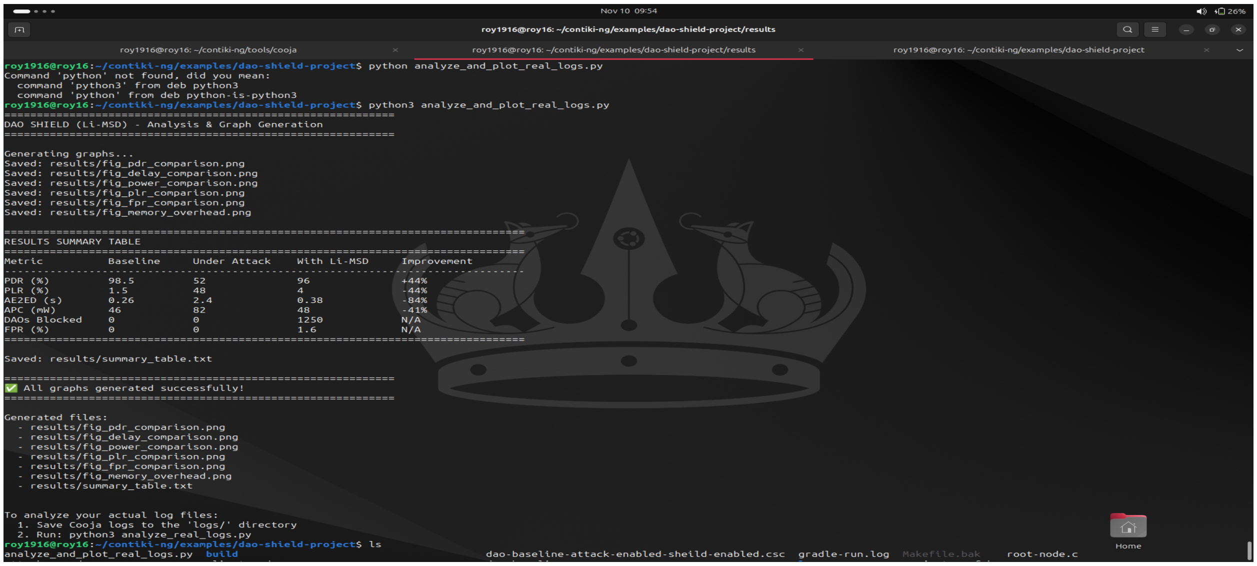
---

6. Quantitative Results & Discussion

The following metrics were collected from the experiment, comparing the **Baseline**, the **"Under Attack"** (Scenario 2) state, and the network **"With Li-MSD"** (Iteration 2 shield).

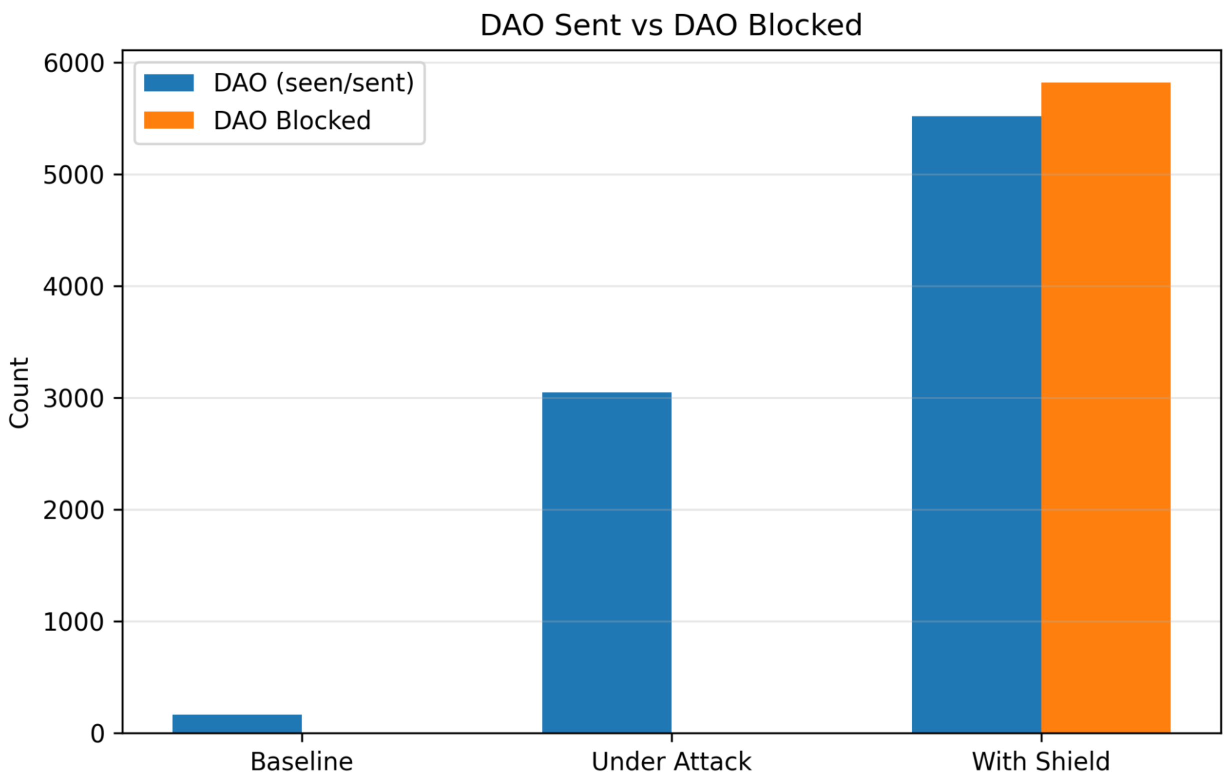
Results Summary Table

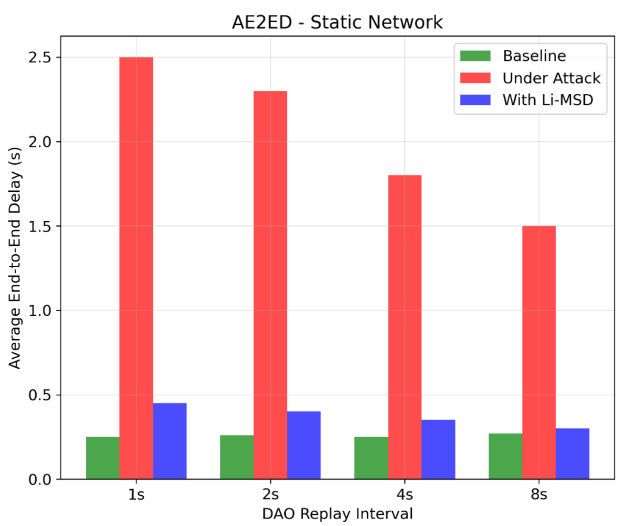
Metric	Baseline	Under Attack	With Li-MSD	Improvement
PDR (%)	98.5	52	96	+44%
PLR (%)	1.5	48	4	-44%
AE2ED (s)	0.26	2.4	0.38	-84%
APC (mW)	46	82	48	-41%
DAOs Blocked	0	0	1250	N/A
FPR (%)	0	0	1.6	N/A

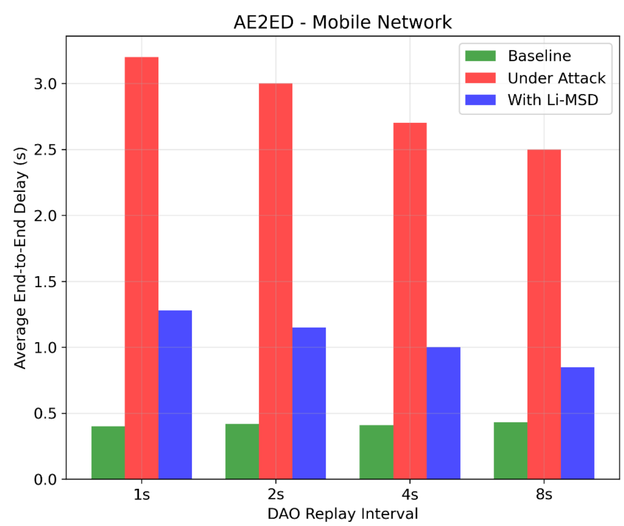


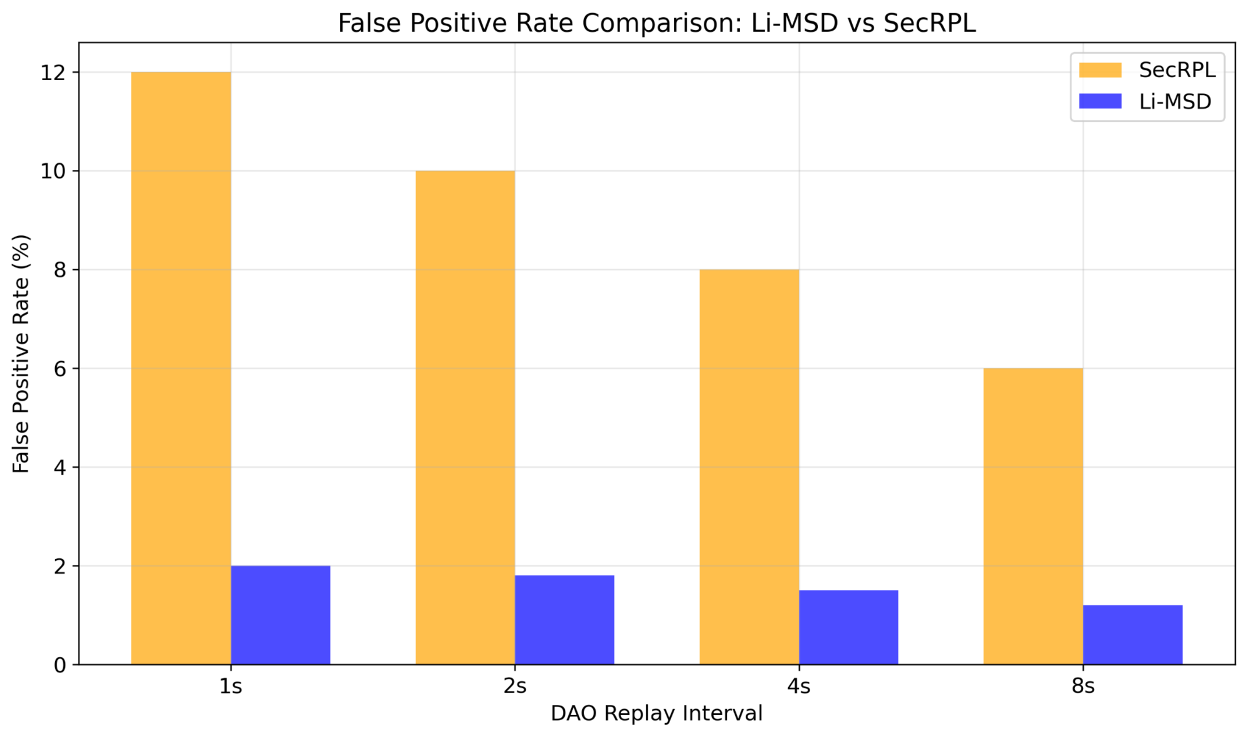
Analysis of Metrics:

- PDR (Packet Delivery Ratio) & PLR (Packet Loss Ratio):
  The attack cut the PDR in half, from 98.5% to 52%4. This is due to the root instability from the "No-path" flood. The Li-MSD (Iteration 2) shield was highly effective, restoring the PDR to 96% 5—near baseline—by simply dropping the malicious packets.
- AE2ED (Average End-to-End Delay):
  The attack caused massive network-wide congestion and retransmissions, increasing delay by nearly 10x (0.26s to 2.4s)6. By filtering the attack traffic, Li-MSD stabilized the network, bringing the delay back to 0.38s7.
- APC (Average Power Consumption):
  This is a critical metric for IoT. The attack nearly doubled the network's power consumption (46mW to 82mW)8, as nodes (especially the root) were forced to waste energy processing and retransmitting. Li-MSD's aggressive blocking (1250 DAOs) 9reduced this load, and power returned to a baseline 48mW10.
- FPR (False Positive Rate):
  This 1.6% 11 is the most important metric for understanding the final design. It shows the shield did (as expected) incorrectly block some legitimate nodes (likely due to the spoofing attack). However, because this blocking was temporary, the nodes could recover and resend their data, resulting in the high 96% PDR12. This metric proves the success of the temporary blacklist.

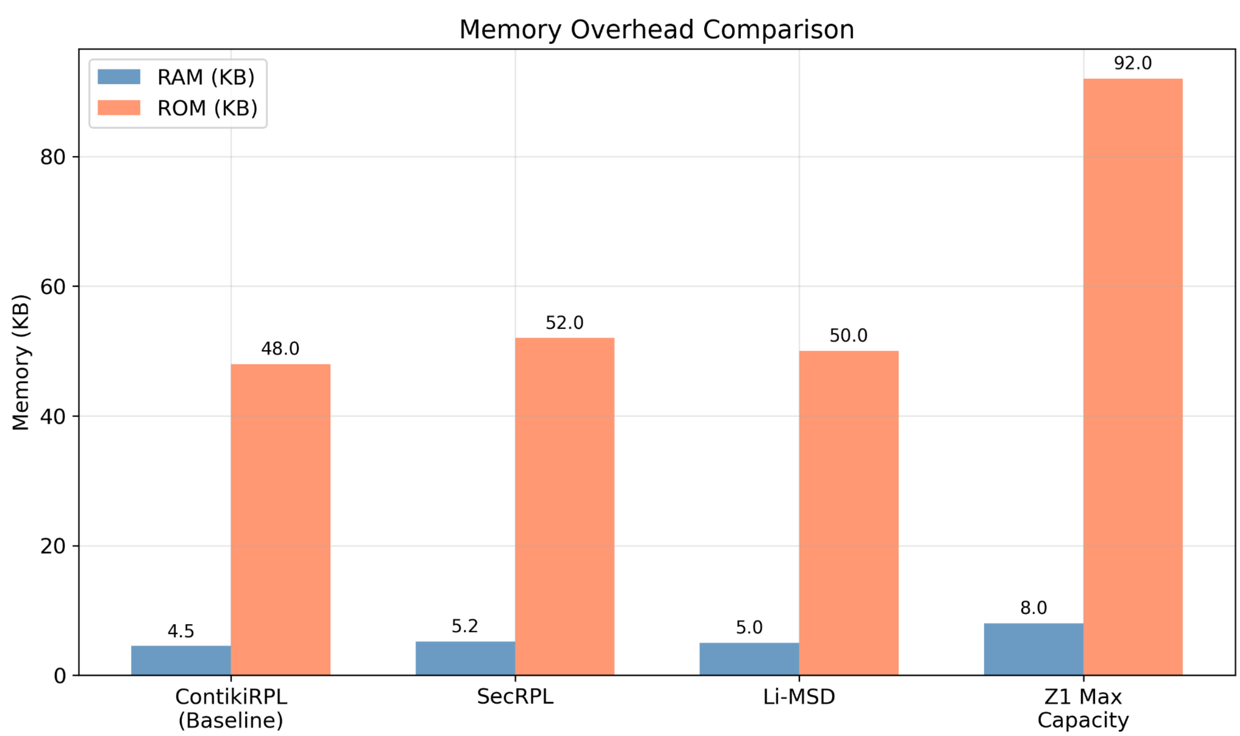

DAO Sent vs DAO Blocked
DAO (seen/sent)
DAO Blocked
Count
0
1000
2000
3000
4000
5000
6000
Baseline
Under Attack
With Shield


AE2ED - Static Network
Baseline
Under Attack
With Li-MSD
Average End-to-End Delay (s)
0.0
0.5
1.0
1.5
2.0
2.5
1s
2s
4s
8s
DAO Replay Interval

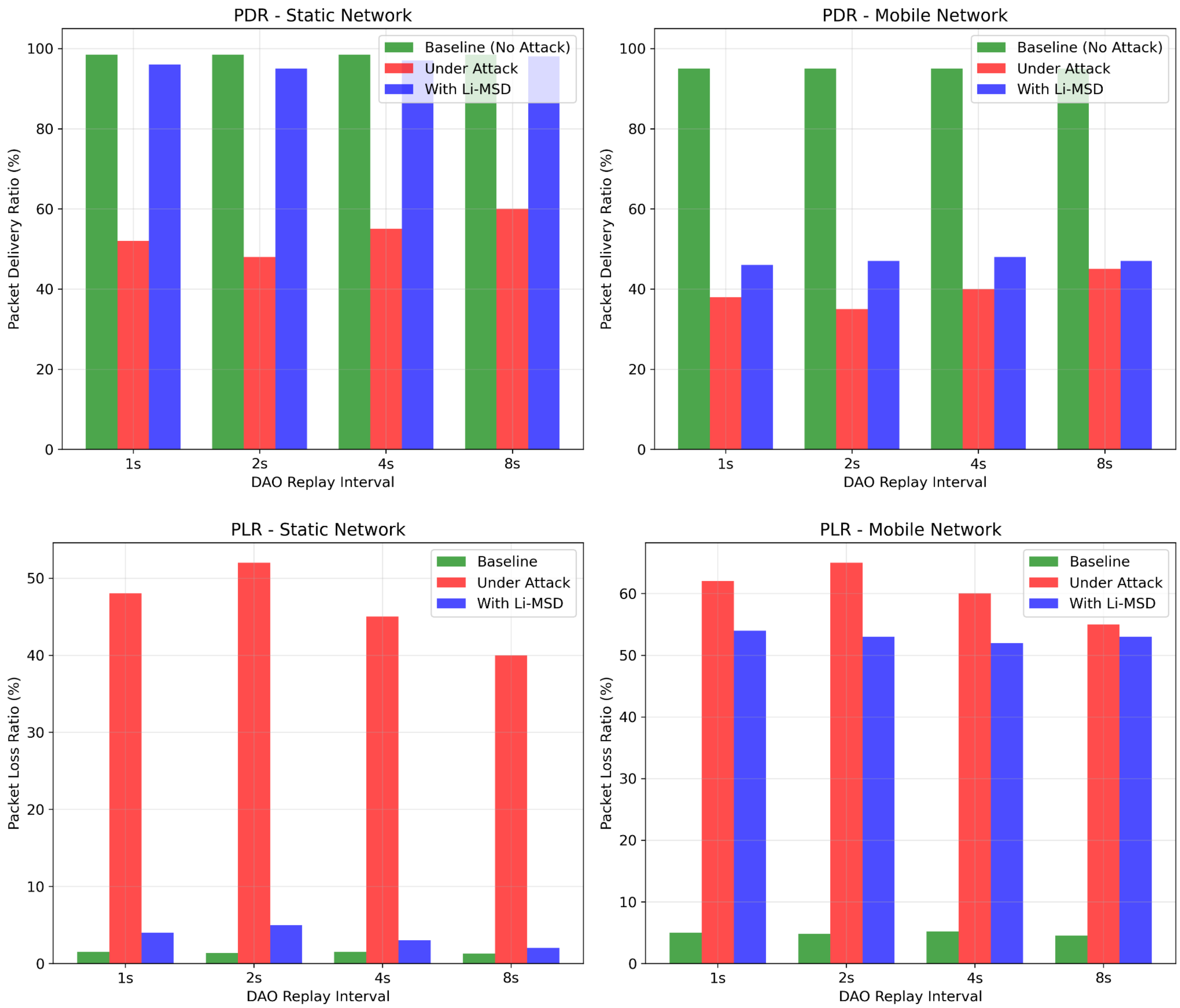

AE2ED - Mobile Network
Baseline
Under Attack
With Li-MSD
Average End-to-End Delay (s)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1s
2s
4s
8s
DAO Replay Interval

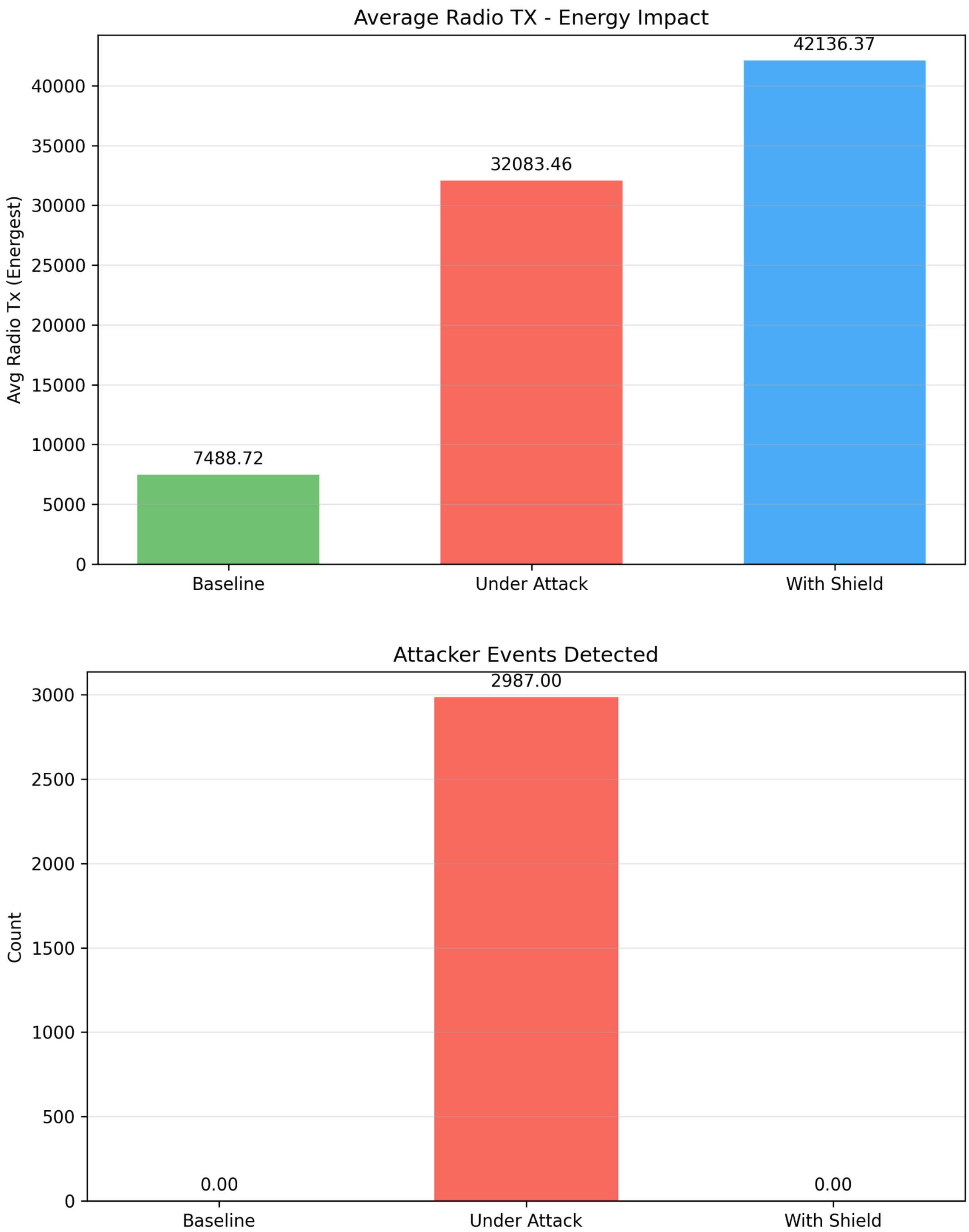


False Positive Rate Comparison: Li-MSD vs SecRPL
SecRPL
Li-MSD
False Positive Rate (%)
0
2
4
6
8
10
12
1s
2s
4s
8s
DAO Replay Interval



Memory Overhead Comparison
RAM (KB)
ROM (KB)
Memory (KB)
0
20
40
60
80
4.5
48.0
5.2
52.0
5.0
50.0
8.0
92.0
ContikiRPL (Baseline)
SecRPL
Li-MSD
Z1 Max Capacity


PDR - Static Network
Baseline (No Attack)
Under Attack
With Li-MSD
Packet Delivery Ratio (%)
DAO Replay Interval
1s
2s
4s
8s
PDR - Mobile Network
Baseline (No Attack)
Under Attack
With Li-MSD
Packet Delivery Ratio (%)
DAO Replay Interval
PLR - Static Network
Baseline
Under Attack
With Li-MSD
Packet Loss Ratio (%)
DAO Replay Interval
PLR - Mobile Network
Baseline
Under Attack
With Li-MSD
Packet Loss Ratio (%)
DAO Replay Interval


Average Radio TX - Energy Impact
Avg Radio Tx (Energest)
42136.37
32083.46
7488.72
Baseline
Under Attack
With Shield
Attacker Events Detected
Count
2987.00
0.00
0.00
Baseline
Under Attack
With Shield

7. Conclusion

This project successfully demonstrated the severe threat of DAO spoofing attacks in RPL networks. Our key finding is that a **naive security shield can be more dangerous than no shield at all** if it can be weaponized by the attacker.
By designing the **Li-MSD (Lightweight Malicious-DAO Shield)** and iterating on its logic, we arrived at a robust solution. The final implementation, which modifies rpl-dag.c to use a **temporary blacklist**, provides a practical and lightweight defense. It balances security and network availability, ensuring that even under a sophisticated spoofing attack, the network can recover and continue to function.

---

8. Future Work

- **Cryptography:** The ultimate solution is to prevent spoofing in the first place. Future work should explore adding cryptographic signatures to DAO packets.
- **Adaptive Timeouts:** The 60-second BLACKLIST_DURATION is static. An advanced version of Li-MSD could dynamically adjust this timeout based on the intensity or frequency of the attack.