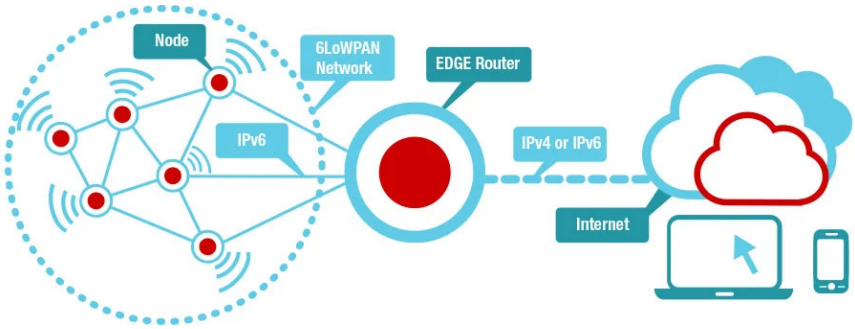
UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI INGEGNERIA INFORMATICA, MODELLISTICA, ELETTRONICA E SISTEMISTICA
DIMES

Progetto di Network Security



Studenti:

Danny DI CELLO 224527

Giuseppe SEMINARA 227540

Roberto MAGURNO 227501

Raffaele MIRIELLO 227564

Docente:

Floriano DE RANGO

Corso di Laurea Magistrale in Ingegneria Informatica

Anno Accademico 2021/2022

Indice

1 Introduzione **1**

1.1 Descrizione sintetica dell'idea progettuale 1

1.2 Organizzazione dell'idea progettuale 2

1.3 6LoWPAN (Ipv6 over Low-Power Wireless Personal Area Networks) 3

1.4 RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) 4

2 Ambiente di sviluppo **5**

2.1 Installazione e configurazione dell'ambiente 5

2.1.1 Requisiti 6

2.1.2 Installazione 6

2.1.3 Struttura Cartelle 7

3 Attacchi **8**

3.1 DoS (*Denial of Service*) 8

3.1.1 Variante 1 12

3.1.1.1 Mitigazione 14

3.1.2 Variante 2 19

3.1.2.1 Mitigazione 23

3.1.3 Variante 3 25

3.1.3.1 Mitigazione 29

3.2 Rank 31

3.2.1 Variante 1 33

3.2.1.1 Mitigazione 35
3.2.2 Variante 2 38
3.2.2.1 Mitigazione 41
3.2.3 Variante 3 42
3.2.3.1 Mitigazione 44
3.3 Wormhole 46
3.3.1 Mitigazione 53
3.4 Fragment Duplication 54
3.4.1 Mitigazione 58
3.5 Come si avvia una simulazione 62

Conclusione **63**

Bibliografia **65**

Elenco delle figure

1 Topologia scenario leggittimo DOS 9
2 Log scenario leggittimo DOS 10
3 Log attacco variante 1 12
4 Diagramma pacchetti inviati e ricevuti attacco variante 1 . . . 13
5 Confronto frequenza invio pacchetti attacco variante 1 13
6 Tempo al crash del simulatore attacco variante 1 14
7 Log mitigazione variante 1 15
8 Diagramma pacchetti inviati e ricevuti mitigazione variante 1 16
9 Confronto frequenza ricezione pacchetti mitigazione variante 1 16
10 Topologia variante 2 19
11 Log attacco variante 2 20
12 Diagramma pacchetti inviati e ricevuti variante 2 21
13 Confronto frequenza invio pacchetti attacco variante 2 21
14 Tempo al crash del simulatore attacco variante 2 22
15 Log mitigazione variante 2 23
16 Diagramma pacchetti inviati e ricevuti mitigazione variante 2 24
17 Confronto frequenza ricezione pacchetti mitigazione variante 2 24
18 Topologia variante 3 25
19 Log attacco variante 3 26
20 Diagramma pacchetti inviati e ricevuti variante 3 27
21 Confronto frequenza invio pacchetti attacco variante 3 27
22 Tempo al crash del simulatore attacco variante 3 28
23 Log mitigazione variante 2 29

24 Diagramma pacchetti inviati e ricevuti mitigazione variante 3 30

25 Confronto frequenza ricezione pacchetti mitigazione variante 3 30

26 Topologia scenario leggittimo RANK 31

27 Log scenario leggittimo RANK 32

28 Variante 1 – Messaggi DIO 33

29 Variante 1 – DAG 34

30 Log mitigazione Rank Attack variante 1 36

31 Mitigazione Rank Attack – DAG nodo vittima 36

32 Topologia variante 2 38

33 Variante 2 - Log 39

34 Variante 2 – DAG 40

35 Log mitigazione Rank Attack variante 2 41

36 Variante 3 – DAG 43

37 Log mitigazione Rank Attack variante 3 44

38 Topologia esempio attacco 46

39 Topologia Wormhole base 47

40 Output simulazione attacco su topologia base 47

41 Output simulazione attacco nodo 6 48

42 Topologia Wormhole small 51

43 Topologia Wormhole medium 51

44 Topologia Wormhole large 51

45 Grafico confronto simulazioni 52

46 Output simulazione mitigazione 53

47 Struttura Frame 6LowPan 54

48 Esempio attacco Fragment Duplication 55

49 Topologia Simulazione Fragment Duplication 55

50 Output simulazione in assenza di attacco 56

51 Output simulazione con fragment duplication 56

52 Esempio di content chain 58

53 Output simulazione con mitigazione 61

Capitolo 1

Introduzione

1.1 Descrizione sintetica dell'idea progettuale

L'obiettivo del seguente progetto è l'implementazione di un insieme di attacchi, e delle rispettive mitigazioni, in un ambiente IoT simulato. In questo tipo di contesti abbiamo a che fare con dispositivi dotati di bassa potenza di elaborazione che comunicano, ed è proprio a causa di tali caratteristiche che, i protocolli di comunicazioni devono essere leggeri in modo da poter essere eseguiti senza causare un overhead eccessivo. Per tale motivo spesso, tali protocolli, tralasciano alcuni aspetti di sicurezza che richiederebbero un elevato quantitativo di risorse, lasciando uno spazio per potenziali attacchi. Nel nostro caso di studio specifico sfrutteremo questi spiragli di sicurezza in riferimento ai protocolli *6LoWPAN* e *RPL*, implementando in particolare i seguenti attacchi:

- DoS Attack.
- Rank Attack.
- Wormhole Attack.
- Fragment Duplication Attack.

1.2 Organizzazione dell'idea progettuale

Dopo aver descritto brevemente gli obiettivi del nostro lavoro, concludiamo la seguente sezione fornendo una breve descrizione dei due protocolli analizzati, al fine di rendere maggiormente comprensibile la discussione affrontata nei capitoli successivi, in particolare nel:

- **capitolo 2** descriviamo l'ambiente di sviluppo utilizzato per l'implementazione degli scenari sopra citati;
- **capitolo 3** ci occupiamo di descrivere gli attacchi realizzati e le possibili mitigazioni in riferimento a diversi scenari applicativi.

Infine, concludiamo l'elaborato, traendo le conclusioni sul lavoro effettuato, analizzando e descrivendo i punti salienti affrontati ed eventuali effetti collaterali derivanti dagli scenari considerati.

1.3 6LoWPAN (Ipv6 over Low-Power Wireless Personal Area Networks)

È un protocollo che definisce meccanismi di incapsulamento e compressione dell'intestazione che consentono l'invio e la ricezione di pacchetti IPv6 su reti basate su IEE 802.15.4, ovvero Low-Rate Wireless Personal Aerea Networks. Ai nodi IPv6 vengono assegnati indirizzi IP gerarchici a 128 bit, attraverso l'uso di un prefisso di lunghezza arbitraria. 6LoWPAN permette di superare i problemi legati alle grandi dimensioni delle intestazioni IPv4 e IPv6, permettendo anche a dispositivi a bassa potenza, con risorse hardware limitate di comunicare. Le specifiche di 6LoWPAN sono definite nei seguenti documenti RFC: **RFC 4919** e **RFC 4944**.

I principali problemi legati ai dispositivi LoWPAN sono i seguenti:

- **Frammentazione e riassemblaggio**: i vincoli di dimensione del pacchetto imposti da IPv6 e 802.15.4 rappresentano un problema di eccessiva frammentazione/riassemblaggio.
- **Compressione dell'intestazione IPv6**: con l'attuale intestazione IPv6 di 40 byte, il payload è ridotto. La compressione dell'intestazione IPv6 rappresenta una necessità per ottimizzare i trasferimenti su una rete 6LoWPAN.
- **Routing**: le reti LoWPAN sono costituite da moltissimi nodi, è necessario adottare quindi un protocollo di instradamento per supportare tali reti, che al contempo soddisfi i vincoli hardware dei dispositivi stessi.

1.4 RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks)

È un protocollo di routing per reti wireless a bassa potenza e generalmente soggette a perdita di pacchetti, ottimizzato per la comunicazione multi-hop e la comunicazione molti-a-uno, ma supporta anche la comunicazione uno-a-uno. Il protocollo è definito nel RFC 6550.

RPL crea una topologia simile ad un albero, che prende il nome di DAG (Grafo Diretto Aciclico). Ogni nodo all'interno della rete ha un rank assegnato, che aumenta man mano che i nodi si allontanano dal nodo radice. I nodi scelgono dunque il rank più basso come criterio di selezione della route. I messaggi ICMPv6 sono definiti nel RFC 4443, e in particolare possiamo individuare i seguenti:

- **DIS (*Information Request DODAG*)**: Utilizzato per richiedere informazioni al DODAG nelle vicinanze. È analogo ai messaggi di richiesta inviati dal router per scoprire le reti esistenti.
- **DIO (*Object of Information of the DAG*)**: messaggio che condivide le informazioni del DAG, inviato in risposta ai messaggi DIS, e utilizzato per aggiornare periodicamente le informazioni sulla topologia della rete.
- **DAO (*Object of Update to the Destination*)**: messaggio inviato dai vari nodi per aggiornare le informazioni relative al loro nodo padre in tutto il DAG.

Capitolo 2

Ambiente di sviluppo

Per l'implementazione degli scenari descritti si è deciso di utilizzare Contiki-NG, come sistema operativo per i dispositivi IoT, e il simulatore COOJA per costruire l'ambiente virtualizzato. La principale motivazione che ha influito su tale scelta è che Contiki è il sistema operativo più utilizzato per l'implementazione del protocollo RPL, essenziale ad espletare gli scopi del progetto, inoltre il sistema è Open Source e supporta un elevato numero di dispositivi diversi, dai computer a 8 bit fino ai nodi che formano le reti di sensori.

2.1 Installazione e configurazione dell'ambiente

Sebbene sia possibile seguire diverse strade per l'installazione del sistema Contiki, si è deciso di seguire l'approccio consigliato nella documentazione ufficiale, utilizzando quindi un'immagine Docker. La scelta effettuata presenta diversi vantaggi, i cui due più rilevanti sono i seguenti:

- ***Facilità di installazione***: l'installazione richiede pochi passi al termine dei quali si dispone di un ambiente totalmente configurato e funzionante.
- ***Compatibilità***: l'immagine Docker fornisce una maggiore compatibilità ed un maggior numero di dispositivi simulati, rispetto ad un processo di installazione nativa sul sistema Host.

2.1.1 Requisiti

Se non è già presente sul sistema è necessario installare il pacchetto *Docker-CE*, con il seguente comando:

$ sudo apt-get install docker-ce

Successivamente è opportuno aggiungere l'utente corrente al gruppo Docker. Questo passaggio è facoltativo, ma permette di interagire col Docker senza l'utilizzo dei privilegi di amministratore.

$ sudo usermod -aG docker <nome_utente>

2.1.2 Installazione

Installato Docker-ce sul sistema host è possibile procedere al download dell'immagine di Contiki-ng digitando

$ docker pull contiker/contiki-ng

Tale comando scarica l'ultima versione di Contiki disponibile. Una volta completato il download è necessario clonare il repository GitHub di Contiki-ng e inizializzare tutti i sotto moduli con i seguenti comandi:

$ git clone https://github.com/contiki-ng/contiki-ng.git
$ cd contiki-ng
$ git submodule update –init –recursive

Infine, per semplificare il processo di avvio è stato creato il seguente alias:

export CNG_PATH=/home/ns/contiki-ng alias contiker="docker run –privileged –sysctl net.ipv6.conf.all.disable_ipv6=0 –mount type=bind,source=$CNG_PATH, destination=/home/user/contiki-ng -e DISPLAY=$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix -v /dev/bus/usb:/dev/bus/usb -ti contiker/contiki-ng"

A questo punto l'ambiente di sviluppo è installato ed è possibile lanciare il simulatore digitando il comando

$ contiker cooja

2.1.3 Struttura Cartelle

Per la riproduzione degli esperimenti sviluppati è necessario identificare due cartelle importanti:

- */contiki-ng*: contiene tutti i file di configurazione di Contiki-NG;
- */contiki-ng/<nome_attacco>*: contiene tutti i file modificati per implementare gli attacchi e le rispettive mitigazioni (che saranno descritti nel capitolo 3).

La cartella attacchi è suddivisa in sottocartelle ognuna delle quali contiene l'implementazione di un attacco e della rispettiva mitigazione, riferita a topologie differenti. Inoltre, è presente uno script Bash che permette di preparare tutti i file necessari e pulire l'ambiente prima di caricare una simulazione, utilizzando i file di backup presenti nella rispettiva cartella.

Capitolo 3

Attacchi

Nel seguente capitolo si descrivono i diversi attacchi implementati, esponendo una trattazione teorica degli stessi e allegando il relativo codice implementativo, opportunamente documentato.

3.1 DoS (*Denial of Service*)

L'attacco DoS mira a compromettere la disponibilità di un servizio, sovraccaricando una risorsa con richieste inutili, impedendole così di soddisfare le richieste legittime. L'attacco DoS è un attacco difficile da mitigare perché sfrutta il normale schema di funzionamento del sistema.
Nel nostro caso specifico si è implementato 3 varianti dell'attacco, nella prima abbiamo una topologia semplice composta da un solo nodo attaccante ed un solo nodo vittima; nella seconda, si è implementata una topologia più complessa dove si simula un DDoS (Distributed Denial of Service), in cui sono presenti più attaccanti che attaccano lo stesso nodo vittima. Infine, nell'ultimo scenario, si effettua sempre un DDoS su una nuova topologia dove son presenti diversi attaccanti e due server.

Prima di analizzare i casi specifichi, per poter comprendere a pieno gli effetti di tale attacco, si mostra con la seguente figura, un possibile scenario legittimo.

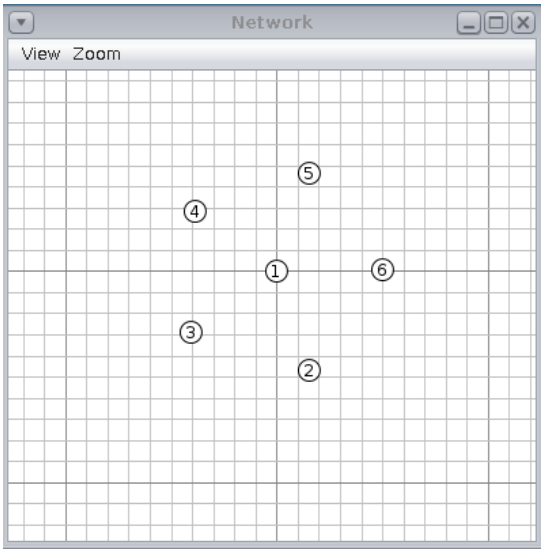
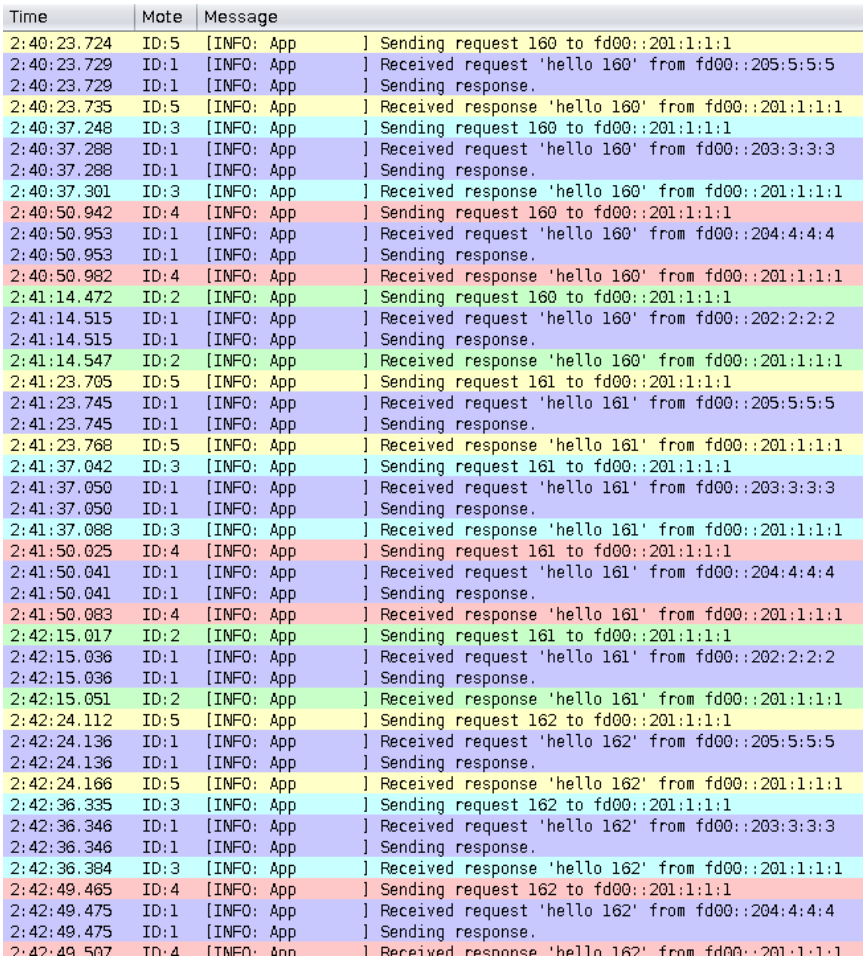


FIGURA 1: Topologia scenario leggittimo DOS

La figura mostra la topologia della rete considerata, che consiste in un server UDP, identificato da ID=1, e di cinque nodi client che inviano richieste ad intervalli casuali.

Di seguito viene riportato il log dei messaggi scambiati dove è possibile osservare come il server, dopo aver ricevuto la richiesta da parte da un client, invia una risposta che viene ricevuta correttamente dal client.



Time	Mote	Message
2:40:23.724	ID:5	[INFO: App] Sending request 160 to fd00::201:1:1:1
2:40:23.729	ID:1	[INFO: App] Received request 'hello 160' from fd00::205:5:5:5
2:40:23.729	ID:1	[INFO: App] Sending response.
2:40:23.735	ID:5	[INFO: App] Received response 'hello 160' from fd00::201:1:1:1
2:40:37.248	ID:3	[INFO: App] Sending request 160 to fd00::201:1:1:1
2:40:37.288	ID:1	[INFO: App] Received request 'hello 160' from fd00::203:3:3:3
2:40:37.288	ID:1	[INFO: App] Sending response.
2:40:37.301	ID:3	[INFO: App] Received response 'hello 160' from fd00::201:1:1:1
2:40:50.942	ID:4	[INFO: App] Sending request 160 to fd00::201:1:1:1
2:40:50.953	ID:1	[INFO: App] Received request 'hello 160' from fd00::204:4:4:4
2:40:50.953	ID:1	[INFO: App] Sending response.
2:40:50.982	ID:4	[INFO: App] Received response 'hello 160' from fd00::201:1:1:1
2:41:14.472	ID:2	[INFO: App] Sending request 160 to fd00::201:1:1:1
2:41:14.515	ID:1	[INFO: App] Received request 'hello 160' from fd00::202:2:2:2
2:41:14.515	ID:1	[INFO: App] Sending response.
2:41:14.547	ID:2	[INFO: App] Received response 'hello 160' from fd00::201:1:1:1
2:41:23.705	ID:5	[INFO: App] Sending request 161 to fd00::201:1:1:1
2:41:23.745	ID:1	[INFO: App] Received request 'hello 161' from fd00::205:5:5:5
2:41:23.745	ID:1	[INFO: App] Sending response.
2:41:23.768	ID:5	[INFO: App] Received response 'hello 161' from fd00::201:1:1:1
2:41:37.042	ID:3	[INFO: App] Sending request 161 to fd00::201:1:1:1
2:41:37.050	ID:1	[INFO: App] Received request 'hello 161' from fd00::203:3:3:3
2:41:37.050	ID:1	[INFO: App] Sending response.
2:41:37.088	ID:3	[INFO: App] Received response 'hello 161' from fd00::201:1:1:1
2:41:50.025	ID:4	[INFO: App] Sending request 161 to fd00::201:1:1:1
2:41:50.041	ID:1	[INFO: App] Received request 'hello 161' from fd00::204:4:4:4
2:41:50.041	ID:1	[INFO: App] Sending response.
2:41:50.083	ID:4	[INFO: App] Received response 'hello 161' from fd00::201:1:1:1
2:42:15.017	ID:2	[INFO: App] Sending request 161 to fd00::201:1:1:1
2:42:15.036	ID:1	[INFO: App] Received request 'hello 161' from fd00::202:2:2:2
2:42:15.036	ID:1	[INFO: App] Sending response.
2:42:15.051	ID:2	[INFO: App] Received response 'hello 161' from fd00::201:1:1:1
2:42:24.112	ID:5	[INFO: App] Sending request 162 to fd00::201:1:1:1
2:42:24.136	ID:1	[INFO: App] Received request 'hello 162' from fd00::205:5:5:5
2:42:24.136	ID:1	[INFO: App] Sending response.
2:42:24.166	ID:5	[INFO: App] Received response 'hello 162' from fd00::201:1:1:1
2:42:36.335	ID:3	[INFO: App] Sending request 162 to fd00::201:1:1:1
2:42:36.346	ID:1	[INFO: App] Received request 'hello 162' from fd00::203:3:3:3
2:42:36.346	ID:1	[INFO: App] Sending response.
2:42:36.384	ID:3	[INFO: App] Received response 'hello 162' from fd00::201:1:1:1
2:42:49.465	ID:4	[INFO: App] Sending request 162 to fd00::201:1:1:1
2:42:49.475	ID:1	[INFO: App] Received request 'hello 162' from fd00::204:4:4:4
2:42:49.475	ID:1	[INFO: App] Sending response.
2:42:49.507	ID:4	[INFO: App] Received response 'hello 162' from fd00::201:1:1:1

FIGURA 2: Log scenario leggittimo DOS

Inoltre, affinchè l'attacco rispecchiasse al meglio la realtà, si è pensato di aumentare il rate di invio dei pacchetti in maniera graduale inserendo il seguente codice all'interno di ogni *client*:

```
1 static float var = 60;
2 [... SNIPPET ... ]
3
4 if(var > 1.10) {
5     var = var - 0.03;
6     SEND_INTERVAL_DDOS = var * CLOCK_SECOND;
7 }
```

Infine, ogni l'attacco è stato implementato nei client cambiando solo l'id del/i nodo/i attaccante/i. Di seguito il codice generale:

```
[... SNIPPET ...]
static struct etimer periodic_timer;
static struct etimer periodic_timer_DDOS;
clock_time_t SEND_INTERVAL_DDOS = var * CLOCK_SECOND;
static unsigned count;
static char str[32];
uip_ipaddr_t dest_ipaddr;

[... SNIPPET ...]
etimer_set(&periodic_timer, random_rand() % SEND_INTERVAL);
etimer_set(&periodic_timer_DDOS, random_rand() % SEND_INTERVAL_DDOS
    );
while(1) {
    if(node_id == <id> || node_id == id>)
        PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&
    periodic_timer_DDOS));
    else
        PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&periodic_timer));

    [... SNIPPET ...]

    // Add some jitter
    if(node_id == <id> || node_id == id>)
        etimer_set(&periodic_timer_DDOS, SEND_INTERVAL_DDOS -
    CLOCK_SECOND + (random_rand() % (2 * CLOCK_SECOND)));
    else
        etimer_set(&periodic_timer, 2*SEND_INTERVAL - CLOCK_SECOND
    + (random_rand() % (2 * CLOCK_SECOND)));

    [... SNIPPET ...]
```

3.1.1 Variante 1

La prima variante è un semplice attacco DoS, funzionale per la comprensione dell'attacco, che riprende la topologia in figura 1 ed inserendo un nodo attaccante che inonda il server di richieste impedendogli di rispondere a quelle legittime.

Di seguito viene mostrato come, il nodo attaccante (ID = 3), inondi il server (ID = 1) di richieste consumando tutte le risorse e costringendolo a scartare le richieste provenienti dai nodi legittimi.

```
17:06.884    ID:3   [INFO: App       ] Received response '1004' from fd00::201:1:1:1
17:07.949    ID:3   [INFO: App       ] Sending request 1005 to fd00::201:1:1:1
17:07.987    ID:1   [INFO: App       ] Received request '1005' from fd00::203:3:3:3
17:07.987    ID:1   [INFO: App       ] Sending response.
17:08.028    ID:3   [INFO: App       ] Received response '1005' from fd00::201:1:1:1
17:08.675    ID:3   [INFO: App       ] Sending request 1006 to fd00::201:1:1:1
17:08.689    ID:1   [INFO: App       ] Received request '1006' from fd00::203:3:3:3
17:08.689    ID:1   [INFO: App       ] Sending response.
17:08.711    ID:3   [INFO: App       ] Received response '1006' from fd00::201:1:1:1
17:08.823    ID:3   [INFO: App       ] Sending request 1007 to fd00::201:1:1:1
17:08.858    ID:1   [INFO: App       ] Received request '1007' from fd00::203:3:3:3
17:08.858    ID:1   [INFO: App       ] Sending response.
17:08.885    ID:3   [INFO: App       ] Received response '1007' from fd00::201:1:1:1
17:10.423    ID:3   [INFO: App       ] Sending request 1008 to fd00::201:1:1:1
17:10.448    ID:1   [INFO: App       ] Received request '1008' from fd00::203:3:3:3
17:10.448    ID:1   [INFO: App       ] Sending response.
17:10.490    ID:3   [INFO: App       ] Received response '1008' from fd00::201:1:1:1
17:10.822    ID:3   [INFO: App       ] Sending request 1009 to fd00::201:1:1:1
17:10.851    ID:1   [INFO: App       ] Received request '1009' from fd00::203:3:3:3
17:10.851    ID:1   [INFO: App       ] Sending response.
17:10.862    ID:3   [INFO: App       ] Received response '1009' from fd00::201:1:1:1
17:10.960    ID:3   [INFO: App       ] Sending request 1010 to fd00::201:1:1:1
17:10.971    ID:1   [INFO: App       ] Received request '1010' from fd00::203:3:3:3
17:10.971    ID:1   [INFO: App       ] Sending response.
17:11.015    ID:3   [INFO: App       ] Received response '1010' from fd00::201:1:1:1
17:11.561    ID:3   [INFO: App       ] Sending request 1011 to fd00::201:1:1:1
17:11.592    ID:1   [INFO: App       ] Received request '1011' from fd00::203:3:3:3
17:11.592    ID:1   [INFO: App       ] Sending response.
17:11.615    ID:3   [INFO: App       ] Received response '1011' from fd00::201:1:1:1
17:12.351    ID:3   [INFO: App       ] Sending request 1012 to fd00::201:1:1:1
17:12.360    ID:1   [INFO: App       ] Received request '1012' from fd00::203:3:3:3
17:12.360    ID:1   [INFO: App       ] Sending response.
17:12.374    ID:3   [INFO: App       ] Received response '1012' from fd00::201:1:1:1
17:12.385    ID:3   [INFO: App       ] Sending request 1013 to fd00::201:1:1:1
17:12.421    ID:1   [INFO: App       ] Received request '1013' from fd00::203:3:3:3
17:12.421    ID:1   [INFO: App       ] Sending response.
17:12.442    ID:3   [INFO: App       ] Received response '1013' from fd00::201:1:1:1
17:12.890    ID:3   [INFO: App       ] Sending request 1014 to fd00::201:1:1:1
17:12.915    ID:1   [INFO: App       ] Received request '1014' from fd00::203:3:3:3
17:12.915    ID:1   [INFO: App       ] Sending response.
17:12.921    ID:3   [INFO: App       ] Received response '1014' from fd00::201:1:1:1
17:13.673    ID:3   [INFO: App       ] Sending request 1015 to fd00::201:1:1:1
17:13.713    ID:1   [INFO: App       ] Received request '1015' from fd00::203:3:3:3
17:13.713    ID:1   [INFO: App       ] Sending response.
17:13.734    ID:3   [INFO: App       ] Received response '1015' from fd00::201:1:1:1
17:14.161    ID:3   [INFO: App       ] Sending request 1016 to fd00::201:1:1:1
```

FIGURA 3: Log attacco variante 1

Il tutto viene messo in evidenza da un'analisi effettuata sul numero di pacchetti inviati e ricevuti, in cui si nota come, il nodo attaccante inondi la rete surclassando le richieste dei nodi leggittimi.

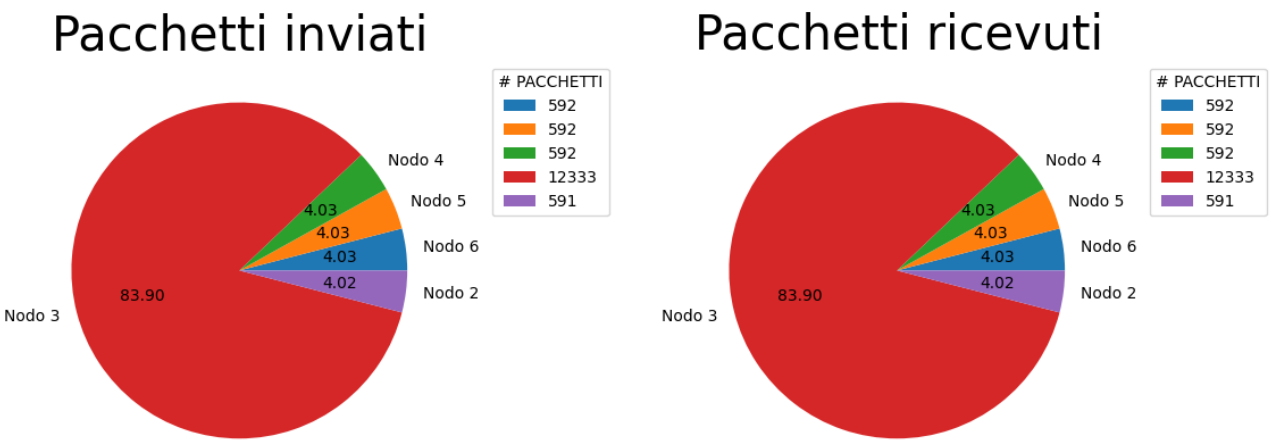


FIGURA 4: Diagramma pacchetti inviati e ricevuti attacco variante 1

L'effetto dell'attacco, nonostante sia effettuato su una topologia molto semplice, si evince ancor più analizzando il rate di invio dei pacchetti che cresce in maniera lineare fino a saturare l'intero canale.

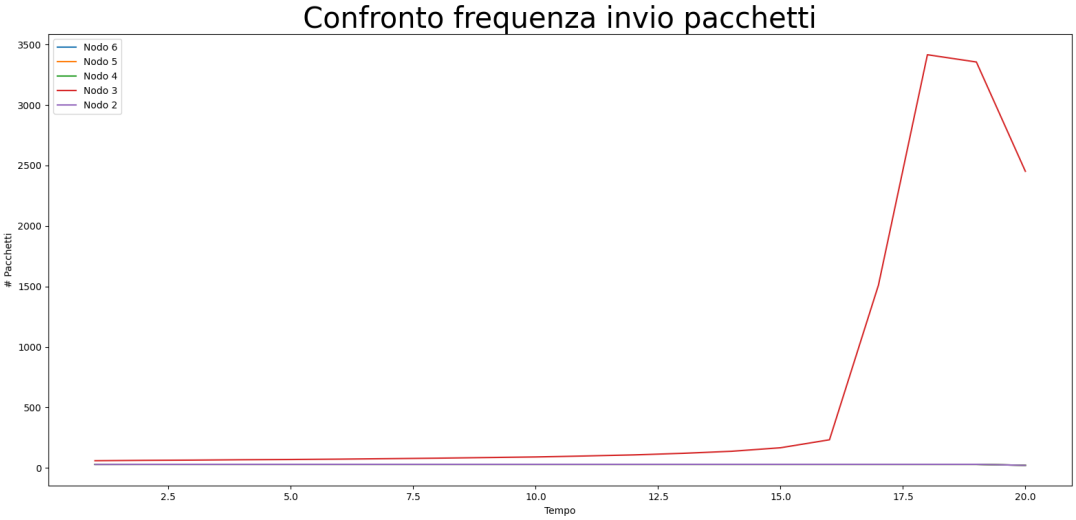


FIGURA 5: Confronto frequenza invio pacchetti attacco variante 1

Infine si nota un decremento dell'andamento causato proprio dall'assenza di risorse da parte del server che porta al crash del simulatore stesso.

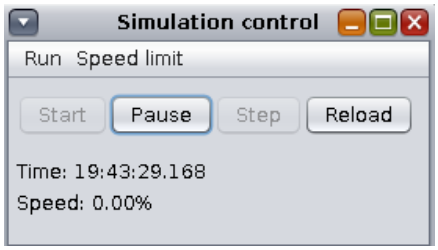
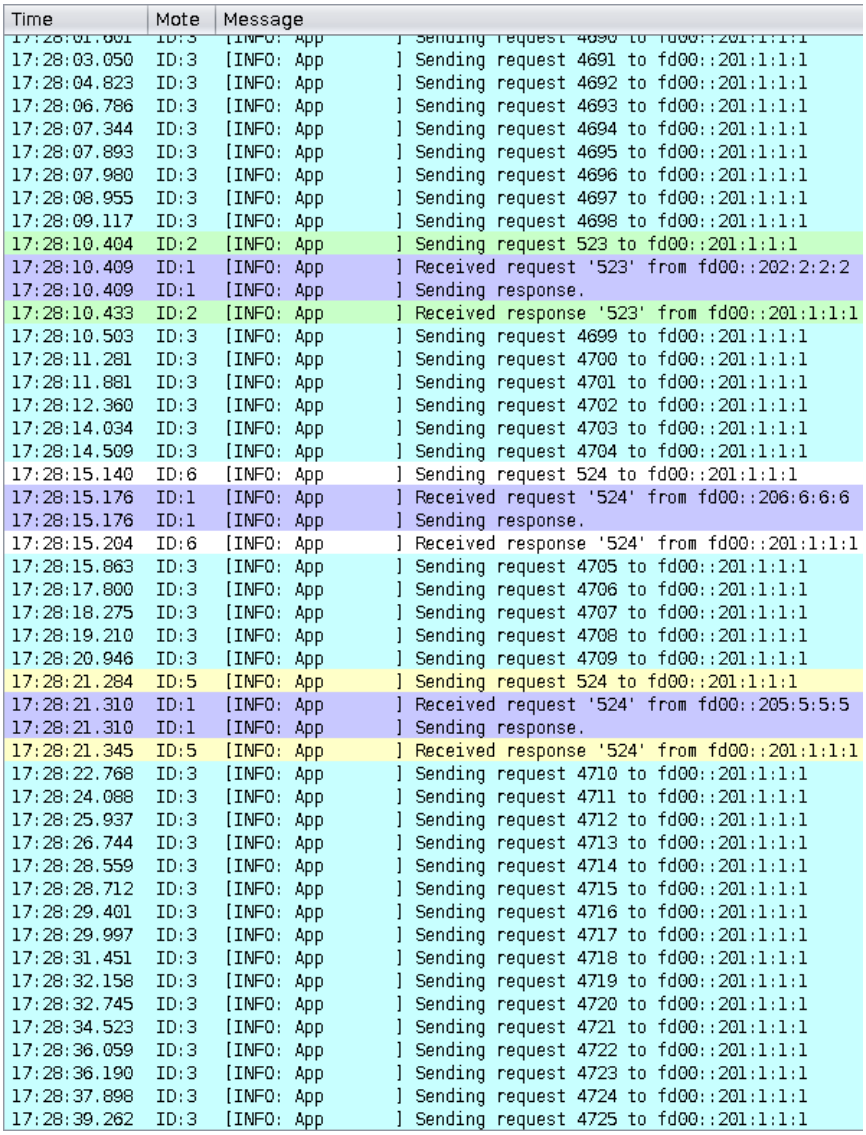


FIGURA 6: Tempo al crash del simulatore attacco variante 1

3.1.1.1 Mitigazione

La mitigazione dell'attacco si basa sul rate di ricezione dei messaggi da parte del server. In particolare, si fissa una soglia, stabilita in modo empirico sulla base della topologia della rete, andando a scartare i messaggi provenienti da nodi che hanno un rate più elevato rispetto alla soglia stabilita. Il valore della soglia ricopre un'importanza cruciale, fissando un valore troppo basso, infatti, anche i nodi legittimi potrebbero essere classificati come malevoli, viceversa fissando un valore troppo elevato potrebbe passare troppo tempo prima di accorgersi dell'attacco consumando tutte le risorse a disposizione e iniziando a scartare richieste legittime.

Per implementare tale mitigazione è stato creato un thread sul server che calcola e aggiorna periodicamente il rate di ricezione dei pacchetti per ogni nodo presente nella rete e che comunica con lo stesso. Sulla base di tale valore, ogni volta che arriva un pacchetto si decide se elaborarlo o scartarlo. L'efficacia della mitigazione è nettamente visibile già analizzando il log dove si evince come le richieste riceventi dal nodo attaccante non vengano elaborate.



Time	Mote	Message
17:28:01.001	ID:3	[INFO: App] Sending request 4690 to fd00::201:1:1:1
17:28:03.050	ID:3	[INFO: App] Sending request 4691 to fd00::201:1:1:1
17:28:04.823	ID:3	[INFO: App] Sending request 4692 to fd00::201:1:1:1
17:28:06.786	ID:3	[INFO: App] Sending request 4693 to fd00::201:1:1:1
17:28:07.344	ID:3	[INFO: App] Sending request 4694 to fd00::201:1:1:1
17:28:07.893	ID:3	[INFO: App] Sending request 4695 to fd00::201:1:1:1
17:28:07.980	ID:3	[INFO: App] Sending request 4696 to fd00::201:1:1:1
17:28:08.955	ID:3	[INFO: App] Sending request 4697 to fd00::201:1:1:1
17:28:09.117	ID:3	[INFO: App] Sending request 4698 to fd00::201:1:1:1
17:28:10.404	ID:2	[INFO: App] Sending request 523 to fd00::201:1:1:1
17:28:10.409	ID:1	[INFO: App] Received request '523' from fd00::202:2:2:2
17:28:10.409	ID:1	[INFO: App] Sending response.
17:28:10.433	ID:2	[INFO: App] Received response '523' from fd00::201:1:1:1
17:28:10.503	ID:3	[INFO: App] Sending request 4699 to fd00::201:1:1:1
17:28:11.281	ID:3	[INFO: App] Sending request 4700 to fd00::201:1:1:1
17:28:11.881	ID:3	[INFO: App] Sending request 4701 to fd00::201:1:1:1
17:28:12.360	ID:3	[INFO: App] Sending request 4702 to fd00::201:1:1:1
17:28:14.034	ID:3	[INFO: App] Sending request 4703 to fd00::201:1:1:1
17:28:14.509	ID:3	[INFO: App] Sending request 4704 to fd00::201:1:1:1
17:28:15.140	ID:6	[INFO: App] Sending request 524 to fd00::201:1:1:1
17:28:15.176	ID:1	[INFO: App] Received request '524' from fd00::206:6:6:6
17:28:15.176	ID:1	[INFO: App] Sending response.
17:28:15.204	ID:6	[INFO: App] Received response '524' from fd00::201:1:1:1
17:28:15.863	ID:3	[INFO: App] Sending request 4705 to fd00::201:1:1:1
17:28:17.800	ID:3	[INFO: App] Sending request 4706 to fd00::201:1:1:1
17:28:18.275	ID:3	[INFO: App] Sending request 4707 to fd00::201:1:1:1
17:28:19.210	ID:3	[INFO: App] Sending request 4708 to fd00::201:1:1:1
17:28:20.946	ID:3	[INFO: App] Sending request 4709 to fd00::201:1:1:1
17:28:21.284	ID:5	[INFO: App] Sending request 524 to fd00::201:1:1:1
17:28:21.310	ID:1	[INFO: App] Received request '524' from fd00::205:5:5:5
17:28:21.310	ID:1	[INFO: App] Sending response.
17:28:21.345	ID:5	[INFO: App] Received response '524' from fd00::201:1:1:1
17:28:22.768	ID:3	[INFO: App] Sending request 4710 to fd00::201:1:1:1
17:28:24.088	ID:3	[INFO: App] Sending request 4711 to fd00::201:1:1:1
17:28:25.937	ID:3	[INFO: App] Sending request 4712 to fd00::201:1:1:1
17:28:26.744	ID:3	[INFO: App] Sending request 4713 to fd00::201:1:1:1
17:28:28.559	ID:3	[INFO: App] Sending request 4714 to fd00::201:1:1:1
17:28:28.712	ID:3	[INFO: App] Sending request 4715 to fd00::201:1:1:1
17:28:29.401	ID:3	[INFO: App] Sending request 4716 to fd00::201:1:1:1
17:28:29.997	ID:3	[INFO: App] Sending request 4717 to fd00::201:1:1:1
17:28:31.451	ID:3	[INFO: App] Sending request 4718 to fd00::201:1:1:1
17:28:32.158	ID:3	[INFO: App] Sending request 4719 to fd00::201:1:1:1
17:28:32.745	ID:3	[INFO: App] Sending request 4720 to fd00::201:1:1:1
17:28:34.523	ID:3	[INFO: App] Sending request 4721 to fd00::201:1:1:1
17:28:36.059	ID:3	[INFO: App] Sending request 4722 to fd00::201:1:1:1
17:28:36.190	ID:3	[INFO: App] Sending request 4723 to fd00::201:1:1:1
17:28:37.898	ID:3	[INFO: App] Sending request 4724 to fd00::201:1:1:1
17:28:39.262	ID:3	[INFO: App] Sending request 4725 to fd00::201:1:1:1

FIGURA 7: Log mitigazione variante 1

Entrando nei particolari possiamo notare, dai diagrammi sottostanti, come i pacchetti inviati del nodo attaccante siano in numero molto minore rispetto a quelli che il server riceve ed elabora.

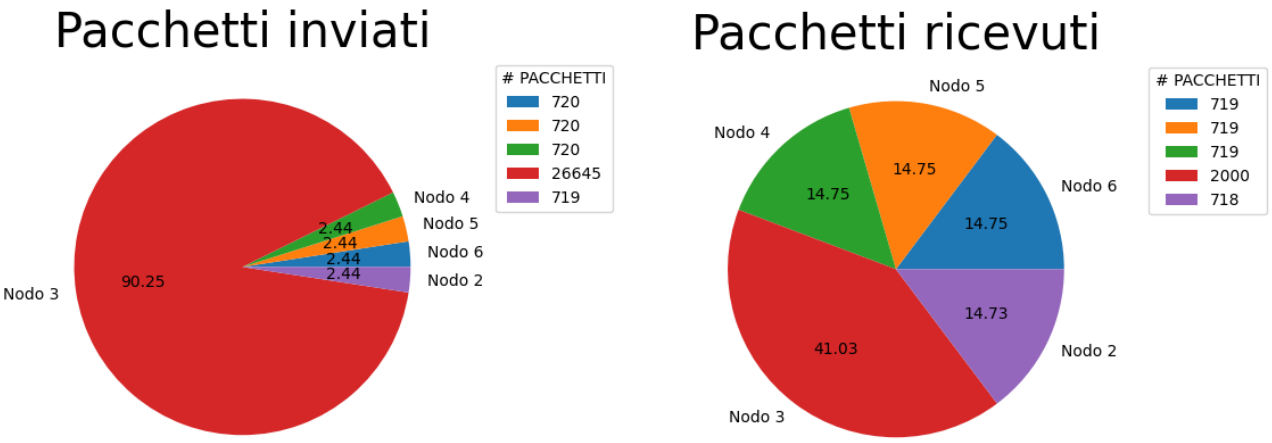


FIGURA 8: Diagramma pacchetti inviati e ricevuti mitigazione variante 1

L'effetto della mitigazione si evince ancor più analizzando il rate di ricezione dei pacchetti che, per il nodo attaccante, cresce in maniera lineare fino a quando il rate di invio supera la soglia predisposta per il rilevamento dell'attacco poichè dopo il rilevamento il nodo viene reso innocuo.

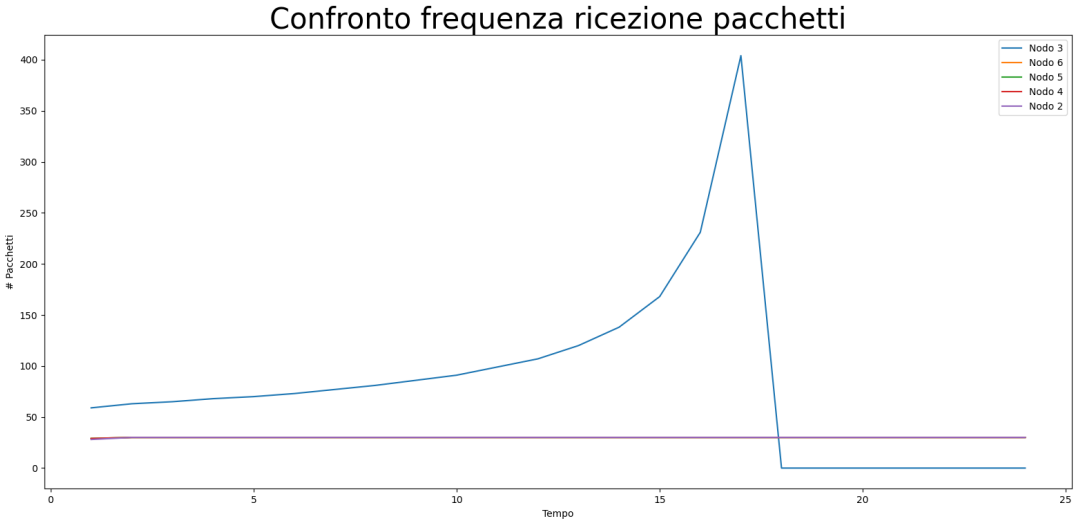


FIGURA 9: Confronto frequenza ricezione pacchetti mitigazione variante 1

Di seguito il codice della funzione *check_packet* utilizzata per valutare se elaborare o meno un pacchetto:

```
#define INTERVALLO (60* CLOCK_SECOND)

uip_ipaddr_t ip[5];
int conta[5] = {0,0,0,0,0};
int i = 0;
float rate[5] = {0,0,0,0,0};

static void check_packet(struct simple_udp_connection *c, const
    uip_ipaddr_t *sender_addr, uint16_t sender_port, const
    uip_ipaddr_t *receiver_addr, uint16_t receiver_port, const
    uint8_t *data, uint16_t datalen) {
     int trovato=0;
     for (int j = 0; j < i; j++) {
         if (uip_ip6addr_cmp(sender_addr, &ip[j])) {
             conta[j]++;
             trovato = 1;
             if (rate[j] < 0.0005)
                 udp_rx_callback(c, sender_addr, sender_port,
    receiver_addr, receiver_port, data, datalen);
         }
     }
     if (!trovato) {
         ip[i]=*sender_addr;
         conta[i]++;
         i++;
     }
}
```

Segue quindi il codice del thread implementato per il calcolo del rate:

```
1  PROCESS_THREAD(calcola, ev, data){
2    static struct etimer tempo;
3
4    PROCESS_BEGIN();
5    etimer_set(&tempo, INTERVALLO);
6
7    while (1) {
8      PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&tempo));
9      for (int j = 0; j < i; j++) {
10       rate[j]=(float)conta[j]/INTERVALLO;
11       conta[j]=0;
12     }
13     etimer_set(&tempo, INTERVALLO);
14   }
15   PROCESS_END();
16 }
```

Infine l'implementazione della mitigazione è replicata per le altre varianti dell'attacco, adattando solo gli array *ip*, *conta* e *rate* al numero di nodi.

3.1.2 Variante 2

Nel seguente si simula un attacco DDoS, si ha quindi una topologia (Figura 18) più complessa in cui abbiamo 4 nodi malevoli (IDs: 3, 6, 9, 16) che attaccano il nodo server.

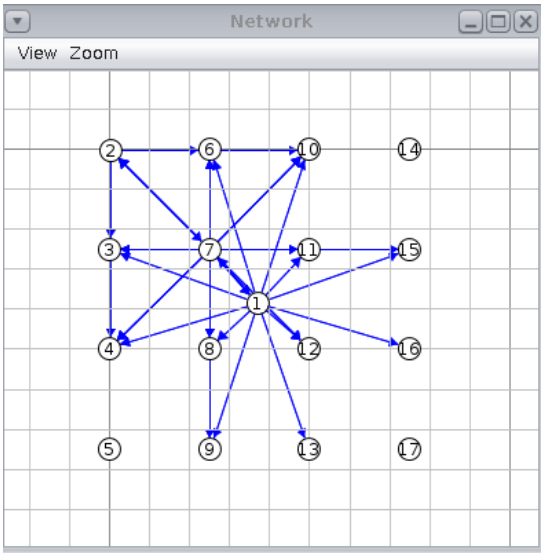


FIGURA 10: Topologia variante 2

Time	Mote	Message
16:42:08.605	ID:1	[INFO: App] Sending response.
16:42:08.613	ID:1	[INFO: App] Received request '500' from fd00::202:2:2:2
16:42:08.613	ID:1	[INFO: App] Sending response.
16:42:08.635	ID:13	[INFO: App] Received response '500' from fd00::201:1:1:1
16:42:08.652	ID:2	[INFO: App] Received response '500' from fd00::201:1:1:1
16:42:08.728	ID:3	[INFO: App] Sending request 2066 to fd00::201:1:1:1
16:42:08.760	ID:1	[INFO: App] Received request '2066' from fd00::203:3:3:3
16:42:08.760	ID:1	[INFO: App] Sending response.
16:42:08.801	ID:3	[INFO: App] Received response '2066' from fd00::201:1:1:1
16:42:09.272	ID:3	[INFO: App] Sending request 2067 to fd00::201:1:1:1
16:42:09.300	ID:1	[INFO: App] Received request '2067' from fd00::203:3:3:3
16:42:09.300	ID:1	[INFO: App] Sending response.
16:42:09.331	ID:3	[INFO: App] Received response '2067' from fd00::201:1:1:1
16:42:09.747	ID:9	[INFO: App] Sending request 2089 to fd00::201:1:1:1
16:42:09.778	ID:1	[INFO: App] Received request '2089' from fd00::209:9:9:9
16:42:09.778	ID:1	[INFO: App] Sending response.
16:42:09.799	ID:9	[INFO: App] Received response '2089' from fd00::201:1:1:1
16:42:09.809	ID:16	[INFO: App] Sending request 2101 to fd00::201:1:1:1
16:42:09.851	ID:1	[INFO: App] Received request '2101' from fd00::210:10:10:10
16:42:09.851	ID:1	[INFO: App] Sending response.
16:42:09.895	ID:16	[INFO: App] Received response '2101' from fd00::201:1:1:1
16:42:09.903	ID:6	[INFO: App] Sending request 2017 to fd00::201:1:1:1
16:42:09.908	ID:1	[INFO: App] Received request '2017' from fd00::206:6:6:6
16:42:09.908	ID:1	[INFO: App] Sending response.
16:42:09.926	ID:6	[INFO: App] Received response '2017' from fd00::201:1:1:1
16:42:10.030	ID:16	[INFO: App] Sending request 2102 to fd00::201:1:1:1
16:42:10.046	ID:1	[INFO: App] Received request '2102' from fd00::210:10:10:10
16:42:10.046	ID:1	[INFO: App] Sending response.
16:42:10.054	ID:16	[INFO: App] Received response '2102' from fd00::201:1:1:1
16:42:10.814	ID:9	[INFO: App] Sending request 2090 to fd00::201:1:1:1
16:42:10.846	ID:1	[INFO: App] Received request '2090' from fd00::209:9:9:9
16:42:10.846	ID:1	[INFO: App] Sending response.
16:42:10.866	ID:9	[INFO: App] Received response '2090' from fd00::201:1:1:1
16:42:10.987	ID:6	[INFO: App] Sending request 2018 to fd00::201:1:1:1
16:42:11.014	ID:1	[INFO: App] Received request '2018' from fd00::206:6:6:6
16:42:11.014	ID:1	[INFO: App] Sending response.
16:42:11.019	ID:6	[INFO: App] Received response '2018' from fd00::201:1:1:1
16:42:11.315	ID:3	[INFO: App] Sending request 2068 to fd00::201:1:1:1
16:42:11.320	ID:1	[INFO: App] Received request '2068' from fd00::203:3:3:3
16:42:11.320	ID:1	[INFO: App] Sending response.
16:42:11.332	ID:3	[INFO: App] Received response '2068' from fd00::201:1:1:1
16:42:11.542	ID:16	[INFO: App] Sending request 2103 to fd00::201:1:1:1
16:42:11.580	ID:1	[INFO: App] Received request '2103' from fd00::210:10:10:10
16:42:11.580	ID:1	[INFO: App] Sending response.
16:42:11.609	ID:16	[INFO: App] Received response '2103' from fd00::201:1:1:1
16:42:11.625	ID:6	[INFO: App] Sending request 2019 to fd00::201:1:1:1
16:42:11.642	ID:1	[INFO: App] Received request '2019' from fd00::206:6:6:6
16:42:11.642	ID:1	[INFO: App] Sending response.

FIGURA 11: Log attacco variante 2

Il tutto viene messo in evidenza dall'analisi effettuata sul numero di pacchetti inviati e ricevuti, in cui si nota come, i nodi malevoli inondino la rete surclassando le richieste dei nodi leggittimi.

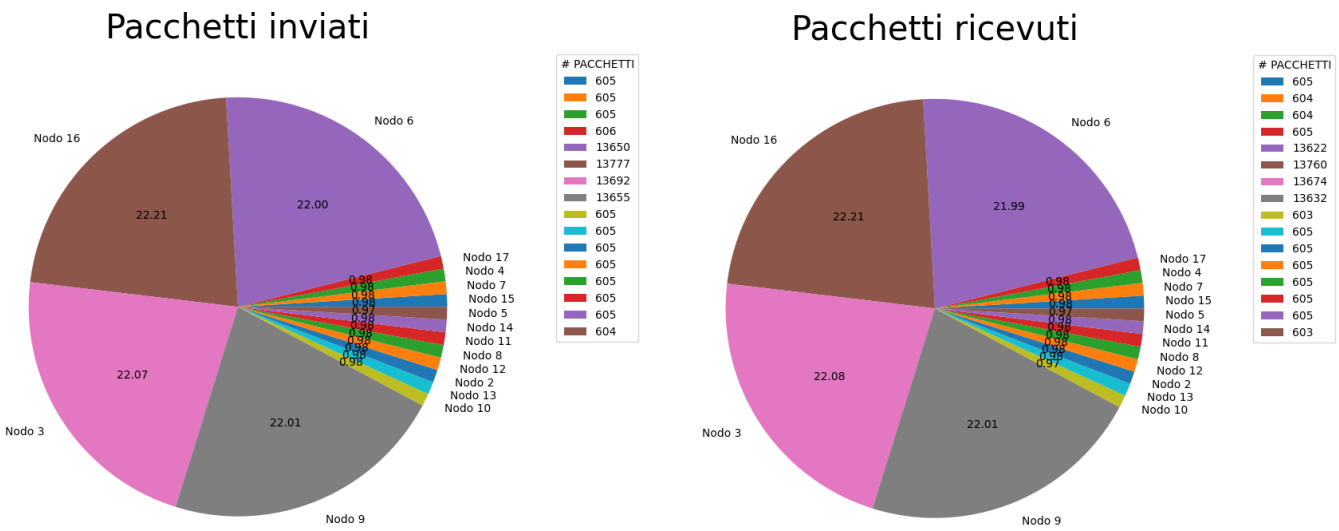


FIGURA 12: Diagramma pacchetti inviati e ricevuti variante 2

L'effetto dell'attacco si evince ancor più analizzando il rate di invio dei pacchetti che cresce in maniera lineare fino a saturare l'intero canale.

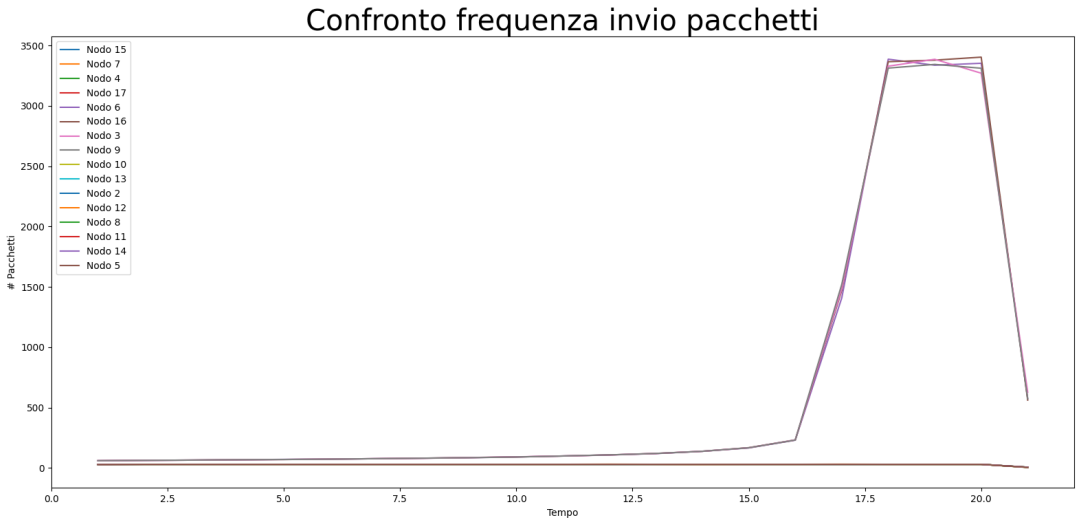


FIGURA 13: Confronto frequenza invio pacchetti attacco variante 2

Infine si nota un decremento dell'andamento causato proprio dall'assenza di risorse da parte del server che porta al crash del simulatore stesso.

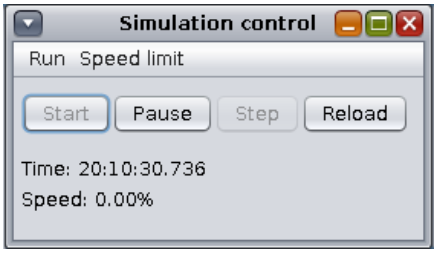


FIGURA 14: Tempo al crash del simulatore attacco variante 2

3.1.2.1 Mitigazione

Come già anticipato, la mitigazione applicata è uguale a quella analizzata nella prima variante dell'attacco (3.1.1.1).

Time	Mote	Message
39:26.296	ID:1	[INFO: App] Received request 'hello 39' from fd00::205:5:5:5
39:26.296	ID:1	[INFO: App] Sending response.
39:26.329	ID:5	[INFO: App] Received response 'hello 39' from fd00::201:1:1:1
39:35.620	ID:3	[INFO: App] Sending request 39 to fd00::201:1:1:1
39:35.652	ID:1	[INFO: App] Received request 'hello 39' from fd00::203:3:3:3
39:35.652	ID:1	[INFO: App] Sending response.
39:35.670	ID:3	[INFO: App] Received response 'hello 39' from fd00::201:1:1:1
40:02.591	ID:4	[INFO: App] Sending request 39 to fd00::201:1:1:1
40:02.612	ID:1	[INFO: App] Received request 'hello 39' from fd00::204:4:4:4
40:02.612	ID:1	[INFO: App] Sending response.
40:02.656	ID:4	[INFO: App] Received response 'hello 39' from fd00::201:1:1:1
40:04.396	ID:2	[INFO: App] Sending request 39 to fd00::201:1:1:1
40:04.425	ID:1	[INFO: App] Received request 'hello 39' from fd00::202:2:2:2
40:04.425	ID:1	[INFO: App] Sending response.
40:04.456	ID:2	[INFO: App] Received response 'hello 39' from fd00::201:1:1:1
40:20.211	ID:6	[INFO: App] Sending request 40 to fd00::201:1:1:1
40:20.247	ID:1	[INFO: App] Received request 'hello 40' from fd00::206:6:6:6
40:20.247	ID:1	[INFO: App] Sending response.
40:20.253	ID:6	[INFO: App] Received response 'hello 40' from fd00::201:1:1:1
40:25.761	ID:5	[INFO: App] Sending request 40 to fd00::201:1:1:1
40:25.797	ID:1	[INFO: App] Received request 'hello 40' from fd00::205:5:5:5
40:25.797	ID:1	[INFO: App] Sending response.
40:25.816	ID:5	[INFO: App] Received response 'hello 40' from fd00::201:1:1:1
40:35.556	ID:3	[INFO: App] Sending request 40 to fd00::201:1:1:1
40:35.573	ID:1	[INFO: App] Received request 'hello 40' from fd00::203:3:3:3
40:35.573	ID:1	[INFO: App] Sending response.
40:35.606	ID:3	[INFO: App] Received response 'hello 40' from fd00::201:1:1:1
41:01.835	ID:4	[INFO: App] Sending request 40 to fd00::201:1:1:1
41:01.846	ID:1	[INFO: App] Received request 'hello 40' from fd00::204:4:4:4
41:01.846	ID:1	[INFO: App] Sending response.
41:01.884	ID:4	[INFO: App] Received response 'hello 40' from fd00::201:1:1:1
41:04.304	ID:2	[INFO: App] Sending request 40 to fd00::201:1:1:1
41:04.314	ID:1	[INFO: App] Received request 'hello 40' from fd00::202:2:2:2
41:04.314	ID:1	[INFO: App] Sending response.
41:04.323	ID:2	[INFO: App] Received response 'hello 40' from fd00::201:1:1:1
41:21.000	ID:6	[INFO: App] Sending request 41 to fd00::201:1:1:1
41:21.022	ID:1	[INFO: App] Received request 'hello 41' from fd00::206:6:6:6
41:21.022	ID:1	[INFO: App] Sending response.
41:21.050	ID:6	[INFO: App] Received response 'hello 41' from fd00::201:1:1:1
41:25.562	ID:5	[INFO: App] Sending request 41 to fd00::201:1:1:1
41:25.585	ID:1	[INFO: App] Received request 'hello 41' from fd00::205:5:5:5
41:25.585	ID:1	[INFO: App] Sending response.
41:25.611	ID:5	[INFO: App] Received response 'hello 41' from fd00::201:1:1:1
41:35.729	ID:3	[INFO: App] Sending request 41 to fd00::201:1:1:1
41:35.741	ID:1	[INFO: App] Received request 'hello 41' from fd00::203:3:3:3
41:35.741	ID:1	[INFO: App] Sending response.
41:35.751	ID:3	[INFO: App] Received response 'hello 41' from fd00::201:1:1:1

FIGURA 15: Log mitigazione variante 2

Entrando nei particolari possiamo notare, dai diagrammi sottostanti, come i pacchetti inviati dei nodi attaccanti siano in numero molto minore rispetto a quelli che il server riceve ed elabora.

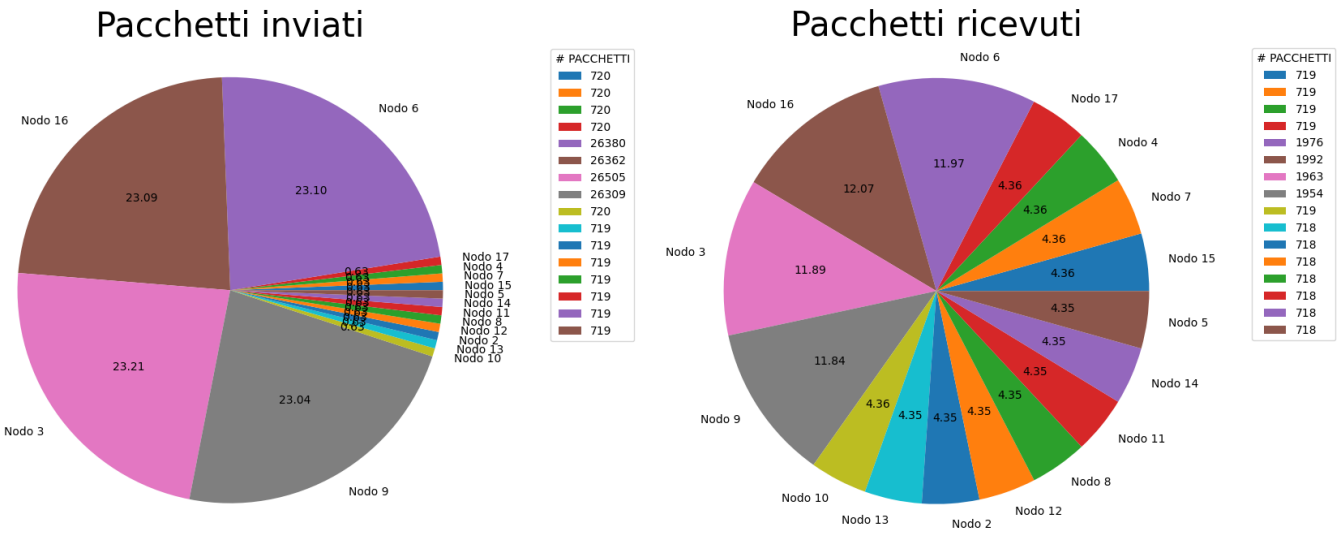


FIGURA 16: Diagramma pacchetti inviati e ricevuti mitigazione variante 2

L'effetto della mitigazione si evince ancor più analizzando il rate di ricezione dei pacchetti che, per i nodi attaccanti, cresce in maniera lineare fino a quando il rate di invio supera la soglia predisposta per il rilevamento dell'attacco poichè dopo il rilevamento i nodi vengono resi innocui.

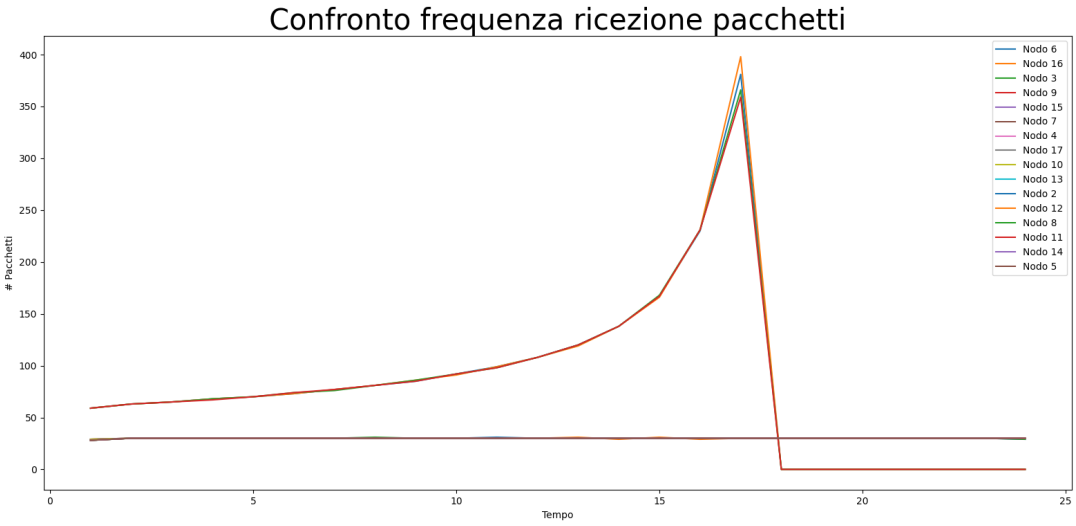


FIGURA 17: Confronto frequenza ricezione pacchetti mitigazione variante 2

3.1.3 Variante 3

In quest'ultima variante si simula un attacco DDoS con una topologia composta da due server ognuno dei quali è attaccato da tre nodi: uno in comune (ID = 13) e due che attaccano solo un server (IDs: 3, 5 per server con ID = 1, IDs: 9, 11 per server con ID = 2).

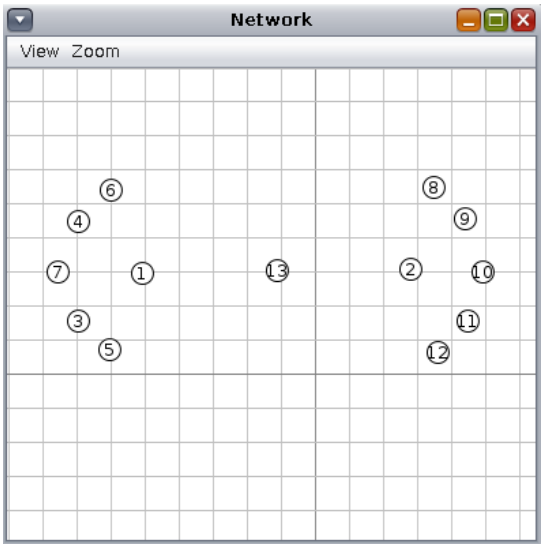
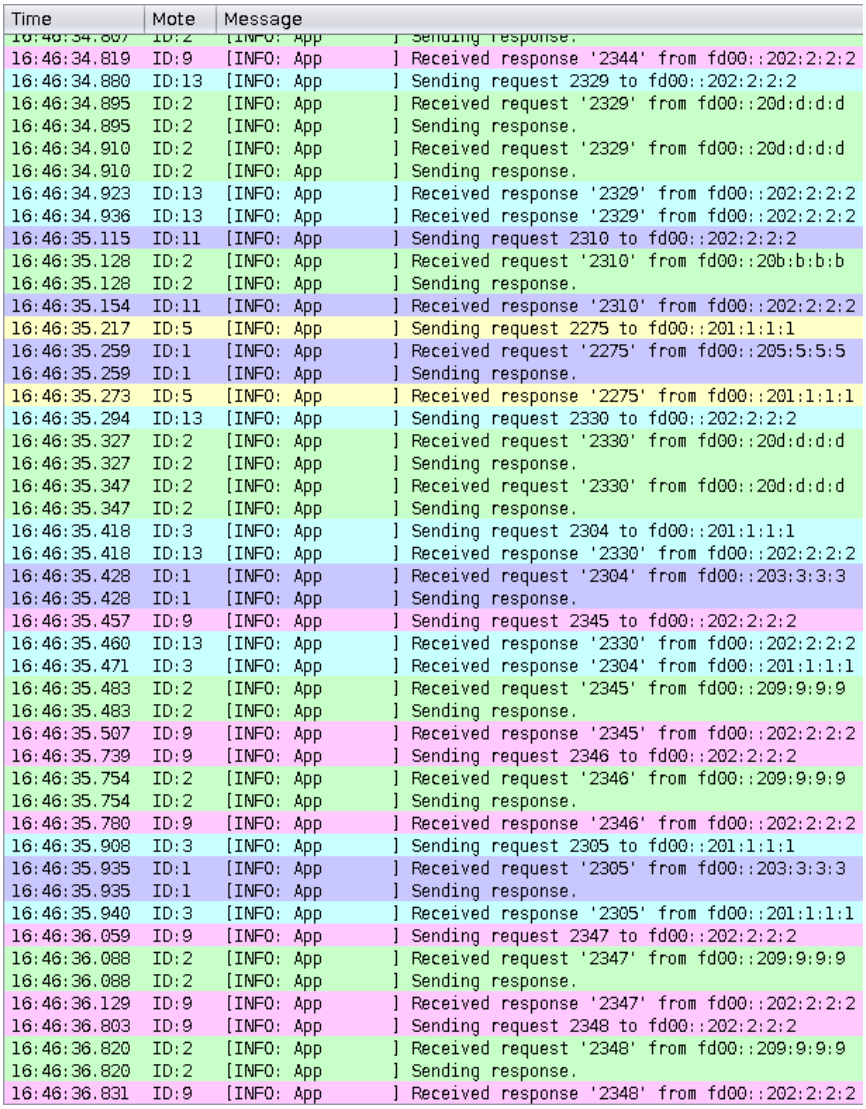


FIGURA 18: Topologia variante 3



Time	Mote	Message
16:46:34.807	ID:2	[INFO: App] Sending response.
16:46:34.819	ID:9	[INFO: App] Received response '2344' from fd00::202:2:2:2
16:46:34.880	ID:13	[INFO: App] Sending request 2329 to fd00::202:2:2:2
16:46:34.895	ID:2	[INFO: App] Received request '2329' from fd00::20d:d:d:d
16:46:34.895	ID:2	[INFO: App] Sending response.
16:46:34.910	ID:2	[INFO: App] Received request '2329' from fd00::20d:d:d:d
16:46:34.910	ID:2	[INFO: App] Sending response.
16:46:34.923	ID:13	[INFO: App] Received response '2329' from fd00::202:2:2:2
16:46:34.936	ID:13	[INFO: App] Received response '2329' from fd00::202:2:2:2
16:46:35.115	ID:11	[INFO: App] Sending request 2310 to fd00::202:2:2:2
16:46:35.128	ID:2	[INFO: App] Received request '2310' from fd00::20b:b:b:b
16:46:35.128	ID:2	[INFO: App] Sending response.
16:46:35.154	ID:11	[INFO: App] Received response '2310' from fd00::202:2:2:2
16:46:35.217	ID:5	[INFO: App] Sending request 2275 to fd00::201:1:1:1
16:46:35.259	ID:1	[INFO: App] Received request '2275' from fd00::205:5:5:5
16:46:35.259	ID:1	[INFO: App] Sending response.
16:46:35.273	ID:5	[INFO: App] Received response '2275' from fd00::201:1:1:1
16:46:35.294	ID:13	[INFO: App] Sending request 2330 to fd00::202:2:2:2
16:46:35.327	ID:2	[INFO: App] Received request '2330' from fd00::20d:d:d:d
16:46:35.327	ID:2	[INFO: App] Sending response.
16:46:35.347	ID:2	[INFO: App] Received request '2330' from fd00::20d:d:d:d
16:46:35.347	ID:2	[INFO: App] Sending response.
16:46:35.418	ID:3	[INFO: App] Sending request 2304 to fd00::201:1:1:1
16:46:35.418	ID:13	[INFO: App] Received response '2330' from fd00::202:2:2:2
16:46:35.428	ID:1	[INFO: App] Received request '2304' from fd00::203:3:3:3
16:46:35.428	ID:1	[INFO: App] Sending response.
16:46:35.457	ID:9	[INFO: App] Sending request 2345 to fd00::202:2:2:2
16:46:35.460	ID:13	[INFO: App] Received response '2330' from fd00::202:2:2:2
16:46:35.471	ID:3	[INFO: App] Received response '2304' from fd00::201:1:1:1
16:46:35.483	ID:2	[INFO: App] Received request '2345' from fd00::209:9:9:9
16:46:35.483	ID:2	[INFO: App] Sending response.
16:46:35.507	ID:9	[INFO: App] Received response '2345' from fd00::202:2:2:2
16:46:35.739	ID:9	[INFO: App] Sending request 2346 to fd00::202:2:2:2
16:46:35.754	ID:2	[INFO: App] Received request '2346' from fd00::209:9:9:9
16:46:35.754	ID:2	[INFO: App] Sending response.
16:46:35.780	ID:9	[INFO: App] Received response '2346' from fd00::202:2:2:2
16:46:35.908	ID:3	[INFO: App] Sending request 2305 to fd00::201:1:1:1
16:46:35.935	ID:1	[INFO: App] Received request '2305' from fd00::203:3:3:3
16:46:35.935	ID:1	[INFO: App] Sending response.
16:46:35.940	ID:3	[INFO: App] Received response '2305' from fd00::201:1:1:1
16:46:36.059	ID:9	[INFO: App] Sending request 2347 to fd00::202:2:2:2
16:46:36.088	ID:2	[INFO: App] Received request '2347' from fd00::209:9:9:9
16:46:36.088	ID:2	[INFO: App] Sending response.
16:46:36.129	ID:9	[INFO: App] Received response '2347' from fd00::202:2:2:2
16:46:36.803	ID:9	[INFO: App] Sending request 2348 to fd00::202:2:2:2
16:46:36.820	ID:2	[INFO: App] Received request '2348' from fd00::209:9:9:9
16:46:36.820	ID:2	[INFO: App] Sending response.
16:46:36.831	ID:9	[INFO: App] Received response '2348' from fd00::202:2:2:2

FIGURA 19: Log attacco variante 3

Il tutto viene messo in evidenza dall'analisi effettuata sul numero di pacchetti inviati e ricevuti, in cui si nota come, i nodi malevoli inondino la rete surclassando le richieste dei nodi leggittimi.

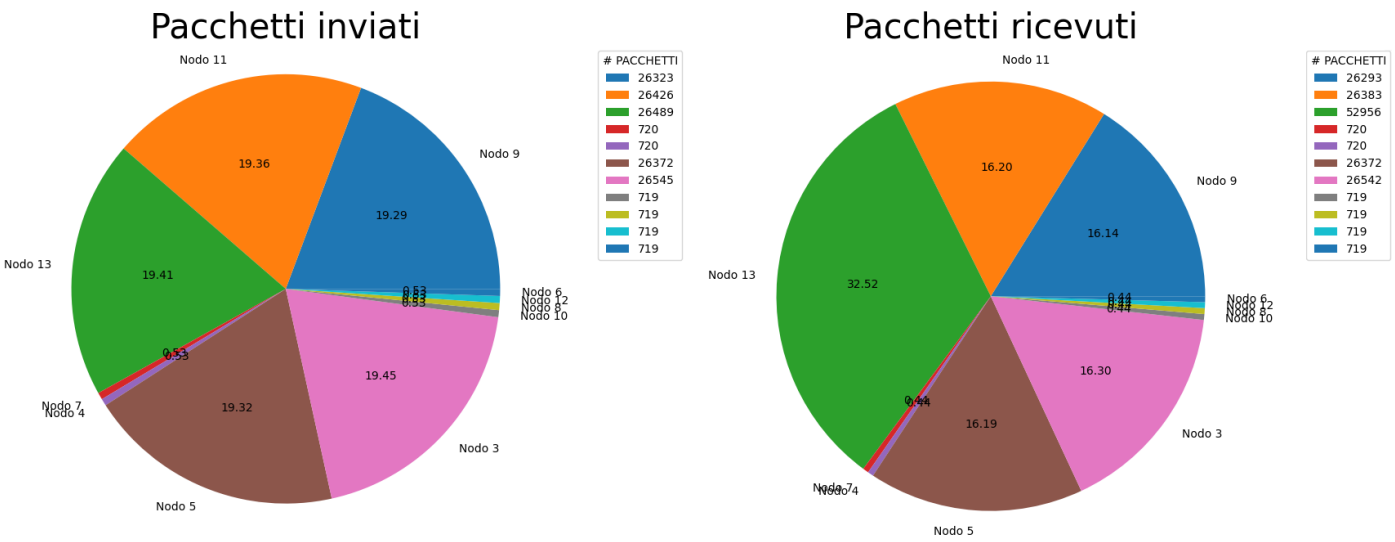


FIGURA 20: Diagramma pacchetti inviati e ricevuti variante 3

L'effetto dell'attacco si evince ancor più analizzando il rate di invio dei pacchetti che cresce in maniera lineare fino a saturare l'intero canale.

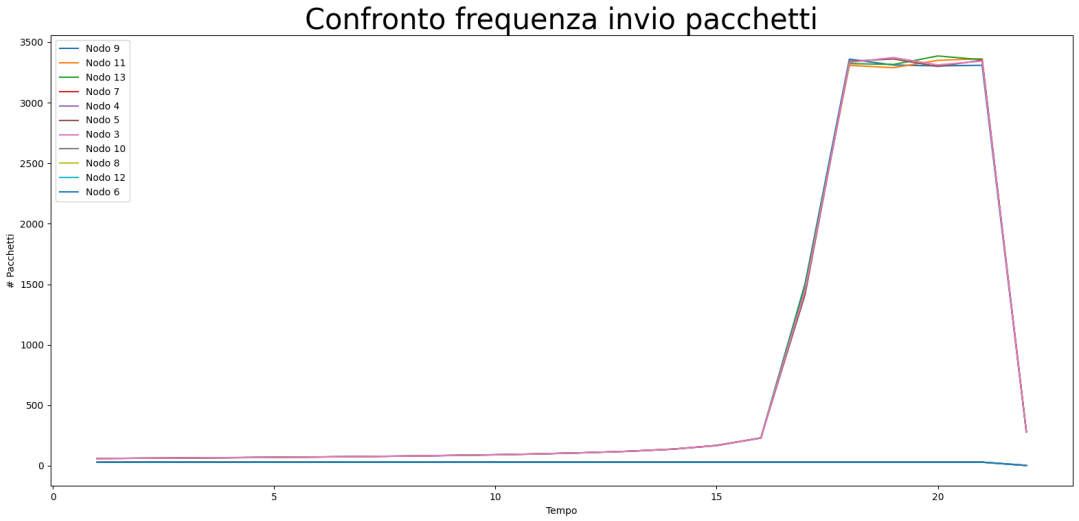


FIGURA 21: Confronto frequenza invio pacchetti attacco variante 3

Infine si nota un decremento dell'andamento causato proprio dall'assenza di risorse da parte del server che porta al crash del simulatore stesso.

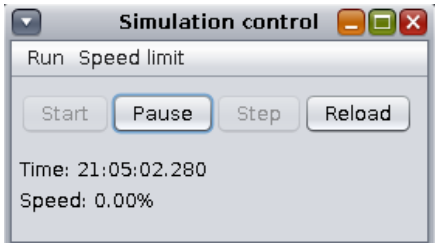


FIGURA 22: Tempo al crash del simulatore attacco variante 3

3.1.3.1 Mitigazione

Come già anticipato, la mitigazione applicata è uguale a quella analizzata nella prima variante dell'attacco (3.1.1.1).

Time	Mote	Message
39:26.296	ID:1	[INFO: App] Received request 'hello 39' from fd00::205:5:5:5
39:26.296	ID:1	[INFO: App] Sending response.
39:26.329	ID:5	[INFO: App] Received response 'hello 39' from fd00::201:1:1:1
39:35.620	ID:3	[INFO: App] Sending request 39 to fd00::201:1:1:1
39:35.652	ID:1	[INFO: App] Received request 'hello 39' from fd00::203:3:3:3
39:35.652	ID:1	[INFO: App] Sending response.
39:35.670	ID:3	[INFO: App] Received response 'hello 39' from fd00::201:1:1:1
40:02.591	ID:4	[INFO: App] Sending request 39 to fd00::201:1:1:1
40:02.612	ID:1	[INFO: App] Received request 'hello 39' from fd00::204:4:4:4
40:02.612	ID:1	[INFO: App] Sending response.
40:02.656	ID:4	[INFO: App] Received response 'hello 39' from fd00::201:1:1:1
40:04.396	ID:2	[INFO: App] Sending request 39 to fd00::201:1:1:1
40:04.425	ID:1	[INFO: App] Received request 'hello 39' from fd00::202:2:2:2
40:04.425	ID:1	[INFO: App] Sending response.
40:04.456	ID:2	[INFO: App] Received response 'hello 39' from fd00::201:1:1:1
40:20.211	ID:6	[INFO: App] Sending request 40 to fd00::201:1:1:1
40:20.247	ID:1	[INFO: App] Received request 'hello 40' from fd00::206:6:6:6
40:20.247	ID:1	[INFO: App] Sending response.
40:20.253	ID:6	[INFO: App] Received response 'hello 40' from fd00::201:1:1:1
40:25.761	ID:5	[INFO: App] Sending request 40 to fd00::201:1:1:1
40:25.797	ID:1	[INFO: App] Received request 'hello 40' from fd00::205:5:5:5
40:25.797	ID:1	[INFO: App] Sending response.
40:25.816	ID:5	[INFO: App] Received response 'hello 40' from fd00::201:1:1:1
40:35.556	ID:3	[INFO: App] Sending request 40 to fd00::201:1:1:1
40:35.573	ID:1	[INFO: App] Received request 'hello 40' from fd00::203:3:3:3
40:35.573	ID:1	[INFO: App] Sending response.
40:35.606	ID:3	[INFO: App] Received response 'hello 40' from fd00::201:1:1:1
41:01.835	ID:4	[INFO: App] Sending request 40 to fd00::201:1:1:1
41:01.846	ID:1	[INFO: App] Received request 'hello 40' from fd00::204:4:4:4
41:01.846	ID:1	[INFO: App] Sending response.
41:01.884	ID:4	[INFO: App] Received response 'hello 40' from fd00::201:1:1:1
41:04.304	ID:2	[INFO: App] Sending request 40 to fd00::201:1:1:1
41:04.314	ID:1	[INFO: App] Received request 'hello 40' from fd00::202:2:2:2
41:04.314	ID:1	[INFO: App] Sending response.
41:04.323	ID:2	[INFO: App] Received response 'hello 40' from fd00::201:1:1:1
41:21.000	ID:6	[INFO: App] Sending request 41 to fd00::201:1:1:1
41:21.022	ID:1	[INFO: App] Received request 'hello 41' from fd00::206:6:6:6
41:21.022	ID:1	[INFO: App] Sending response.
41:21.050	ID:6	[INFO: App] Received response 'hello 41' from fd00::201:1:1:1
41:25.562	ID:5	[INFO: App] Sending request 41 to fd00::201:1:1:1
41:25.585	ID:1	[INFO: App] Received request 'hello 41' from fd00::205:5:5:5
41:25.585	ID:1	[INFO: App] Sending response.
41:25.611	ID:5	[INFO: App] Received response 'hello 41' from fd00::201:1:1:1
41:35.729	ID:3	[INFO: App] Sending request 41 to fd00::201:1:1:1
41:35.741	ID:1	[INFO: App] Received request 'hello 41' from fd00::203:3:3:3
41:35.741	ID:1	[INFO: App] Sending response.
41:35.751	ID:3	[INFO: App] Received response 'hello 41' from fd00::201:1:1:1

FIGURA 23: Log mitigazione variante 2

Entrando nei particolari possiamo notare, dai diagrammi sottostanti, come i pacchetti inviati dei nodi attaccanti siano in numero molto minore rispetto a quelli che i server ricevono ed elaborano.

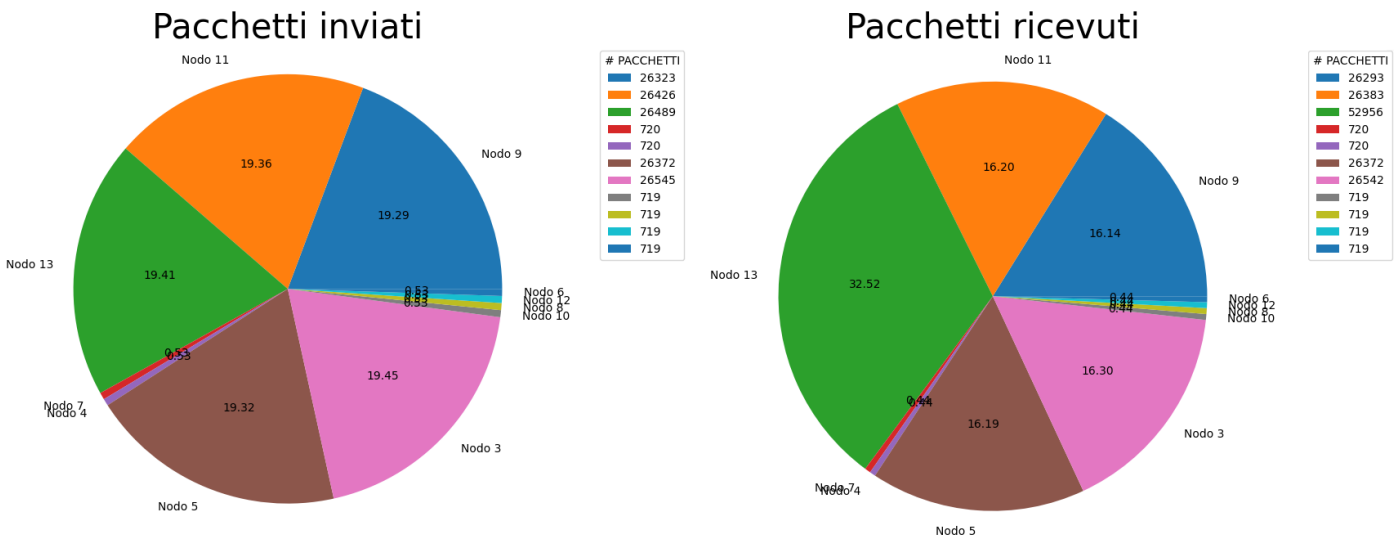


FIGURA 24: Diagramma pacchetti inviati e ricevuti mitigazione variante 3

L'effetto della mitigazione si evince ancor più analizzando il rate di ricezione dei pacchetti che, per i nodi attaccanti, cresce in maniera lineare fino a quando il rate di invio supera la soglia predisposta per il rilevamento dell'attacco poichè dopo il rilevamento i nodi vengono resi innocui.

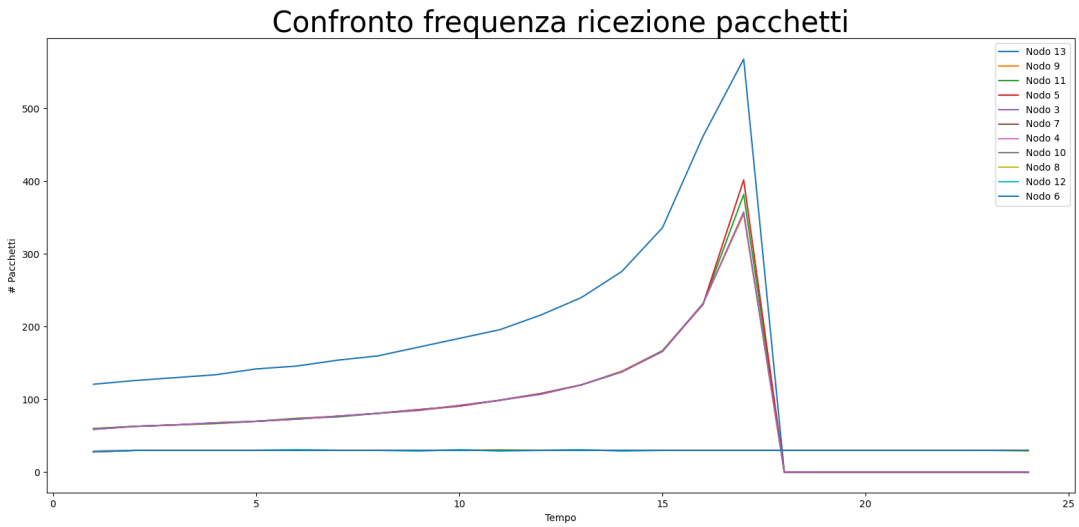


FIGURA 25: Confronto frequenza ricezione pacchetti mitigazione variante 3

3.2 Rank

L'introduzione del concetto di rank all'interno del protocollo RPL risulta utile a diverse applicazioni come l'ottimizzazione del percorso o la prevenzione di cicli, ma al contempo rappresenta un efficace vettore di attacco, che può incidere sulle performance della rete. Un nodo malevolo può infatti falsificare il valore del rank a discapito degli altri nodi che non hanno modo di verificare l'autenticità di tale valore. In questa tipologia di attacco un nodo malevolo modifica il proprio rank in modo da attirare e manipolare l'intera rete, causando, ad esempio, ritardi nella consegna dei messaggi, perdita dei pacchetti, creazione di loop che rendono la topologia instabile e rendendo necessario un elevato numero di messi di controllo per correggere la struttura della rete.

Di seguito si mostra un possibile scenario legittimo e successivamente si propongono tre varianti dell'attacco, con due topologie differenti.

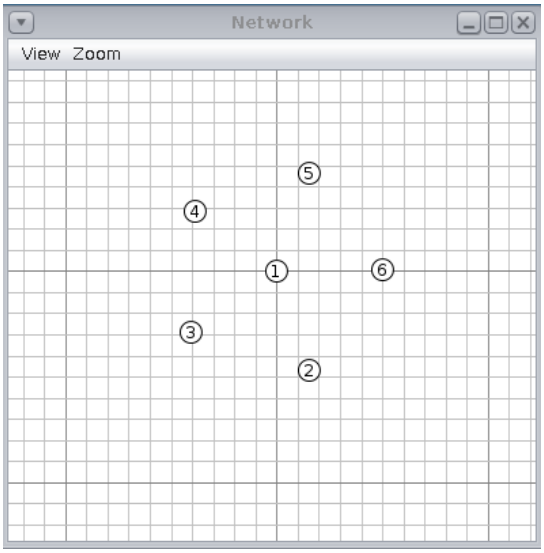


FIGURA 26: Topologia scenario leggittimo RANK

Nello scenario base è prevista la presenza di 7 nodi (Figura 26), di cui il nodo con ID = 1 svolge il ruolo di server, mentre i restanti svolgono il ruolo di client. Di seguito si mostrano i log derivanti dalla esecuzione di tale scenario, in cui vengono evidenziati, in blu, alcuni messaggi DIO inviati/ricevuti, e in rosso, i DAG dei nodi 3, 4, 7.

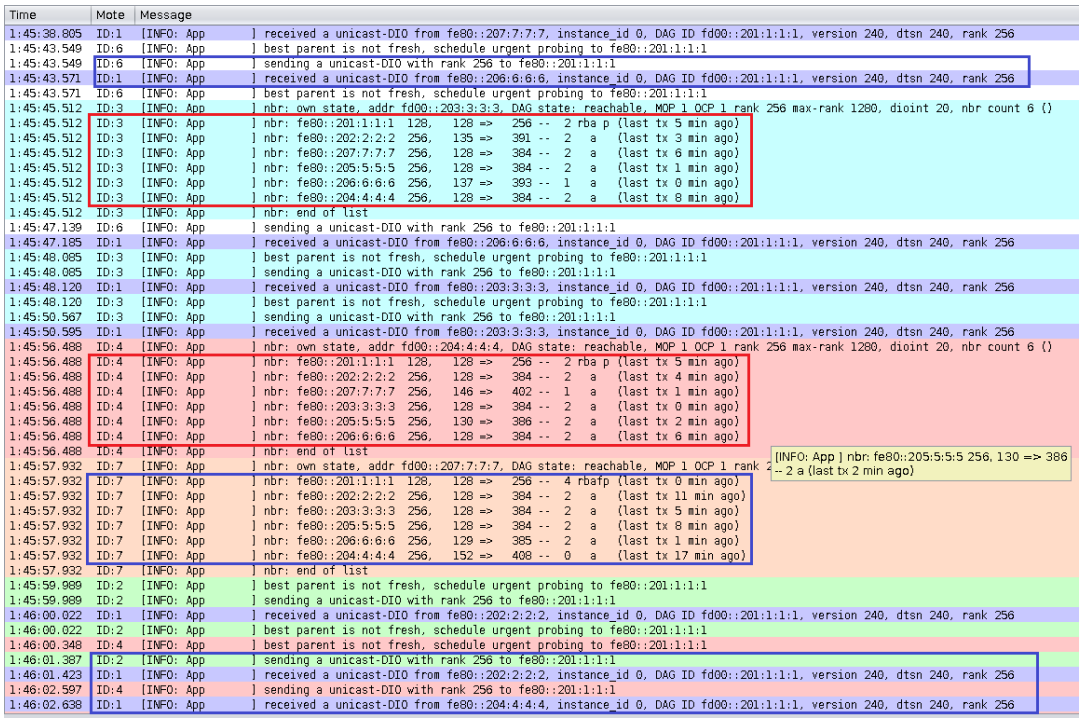


FIGURA 27: Log scenario leggittimo RANK

3.2.1 Variante 1

Nello scenario che verrà descritto, si utlizza la topologia rappresentata in figura 26, in cui si assume che il nodo 4 sia malevolo. Tale nodo effettuerà un poisoning del suo valore di rank nei confronti del solo nodo 3, designato come vittima. In particolare, dall'analisi dei log (Figura 28) è possibile osservare come il nodo attaccante con ID = 4 invii dei messaggi DIO con rank posto pari a 1000 solo al nodo 3, il quale riceve e registra tale valore di rank associato al nodo attaccante nel proprio DAG. Per semplificare la visualizzazione è stato cerchiato in blu il messaggio DIO indirizzato al nodo 3 con rank alterato, mentre in rosso è stato evidenziato i messaggi DIO che ricevono i nodi 1 e 5 contente il valore di rank reale del nodo 4 (Figura 28). Nella figura 29 invece in verde è stato evidenziato il DAG del nodo 3 in cui è possibile osservare la riuscita dell'attacco e in viola i DAG dei nodi che ricevono il valore di rank corretto (Le frecce indicano la riga del DAG riferita al nodo malevolo 4).

Time	Mote	Message
18:53:06.682	ID:4	[INFO: App] received a unicast-DIO from fe80::206:6:6:6, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:53:27.750	ID:4	[INFO: App] sending a unicast-DIO with rank 256 to fe80::205:5:5:5
18:53:27.758	ID:5	[INFO: App] received a unicast-DIO from fe80::204:4:4:4, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:53:29.293	ID:5	[INFO: App] sending a unicast-DIO with rank 256 to fe80::207:7:7:7
18:53:29.326	ID:7	[INFO: App] received a unicast-DIO from fe80::205:5:5:5, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:53:42.687	ID:2	[INFO: App] sending a unicast-DIO with rank 256 to fe80::205:5:5:5
18:53:42.707	ID:5	[INFO: App] received a unicast-DIO from fe80::202:2:2:2, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:53:52.165	ID:3	[INFO: App] sending a unicast-DIO with rank 256 to fe80::206:6:6:6
18:53:52.202	ID:6	[INFO: App] received a unicast-DIO from fe80::203:3:3:3, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:53:56.995	ID:7	[INFO: App] sending a unicast-DIO with rank 256 to fe80::206:6:6:6
18:53:57.030	ID:6	[INFO: App] received a unicast-DIO from fe80::207:7:7:7, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:54:04.691	ID:6	[INFO: App] sending a unicast-DIO with rank 256 to fe80::203:3:3:3
18:54:04.728	ID:3	[INFO: App] received a unicast-DIO from fe80::206:6:6:6, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:54:31.708	ID:2	[INFO: App] sending a unicast-DIO with rank 256 to fe80::206:6:6:6
18:54:31.749	ID:6	[INFO: App] received a unicast-DIO from fe80::202:2:2:2, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:54:34.775	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
18:54:34.793	ID:3	[INFO: App] received a unicast-DIO from fe80::204:4:4:4, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 1000
18:55:04.730	ID:3	[INFO: App] sending a unicast-DIO with rank 256 to fe80::205:5:5:5
18:55:04.737	ID:3	[INFO: App] received a unicast-DIO from fe80::205:5:5:5, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:55:13.319	ID:7	[INFO: App] sending a unicast-DIO with rank 256 to fe80::206:6:6:6
18:55:13.358	ID:6	[INFO: App] received a unicast-DIO from fe80::207:7:7:7, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:55:30.540	ID:6	[INFO: App] sending a unicast-DIO with rank 256 to fe80::205:5:5:5
18:55:30.563	ID:5	[INFO: App] received a unicast-DIO from fe80::206:6:6:6, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:55:32.209	ID:3	[INFO: App] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
18:55:32.252	ID:1	[INFO: App] received a unicast-DIO from fe80::203:3:3:3, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:55:32.422	ID:2	[INFO: App] sending a unicast-DIO with rank 256 to fe80::206:6:6:6
18:55:32.454	ID:6	[INFO: App] received a unicast-DIO from fe80::202:2:2:2, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:55:38.304	ID:4	[INFO: App] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
18:55:38.338	ID:1	[INFO: App] received a unicast-DIO from fe80::204:4:4:4, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:56:02.522	ID:2	[INFO: App] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
18:56:02.568	ID:1	[INFO: App] received a unicast-DIO from fe80::202:2:2:2, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:56:27.338	ID:4	[INFO: App] sending a unicast-DIO with rank 256 to fe80::205:5:5:5
18:56:27.363	ID:5	[INFO: App] received a unicast-DIO from fe80::204:4:4:4, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:56:29.621	ID:5	[INFO: App] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
18:56:29.668	ID:1	[INFO: App] received a unicast-DIO from fe80::205:5:5:5, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:56:45.490	ID:7	[INFO: App] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
18:56:45.505	ID:1	[INFO: App] received a unicast-DIO from fe80::207:7:7:7, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:56:51.242	ID:3	[INFO: App] sending a unicast-DIO with rank 256 to fe80::206:6:6:6
18:56:51.266	ID:6	[INFO: App] received a unicast-DIO from fe80::203:3:3:3, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:56:52.799	ID:6	[INFO: App] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
18:56:52.825	ID:1	[INFO: App] received a unicast-DIO from fe80::206:6:6:6, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:57:26.338	ID:5	[INFO: App] sending a unicast-DIO with rank 256 to fe80::203:3:3:3
18:57:26.375	ID:3	[INFO: App] received a unicast-DIO from fe80::205:5:5:5, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:57:40.259	ID:2	[INFO: App] sending a unicast-DIO with rank 256 to fe80::204:4:4:4
18:57:40.301	ID:4	[INFO: App] received a unicast-DIO from fe80::202:2:2:2, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
18:58:06.008	ID:4	[INFO: App] sending a unicast-DIO with rank 256 to fe80::202:2:2:2

FIGURA 28: Variante 1 – Messaggi DIO

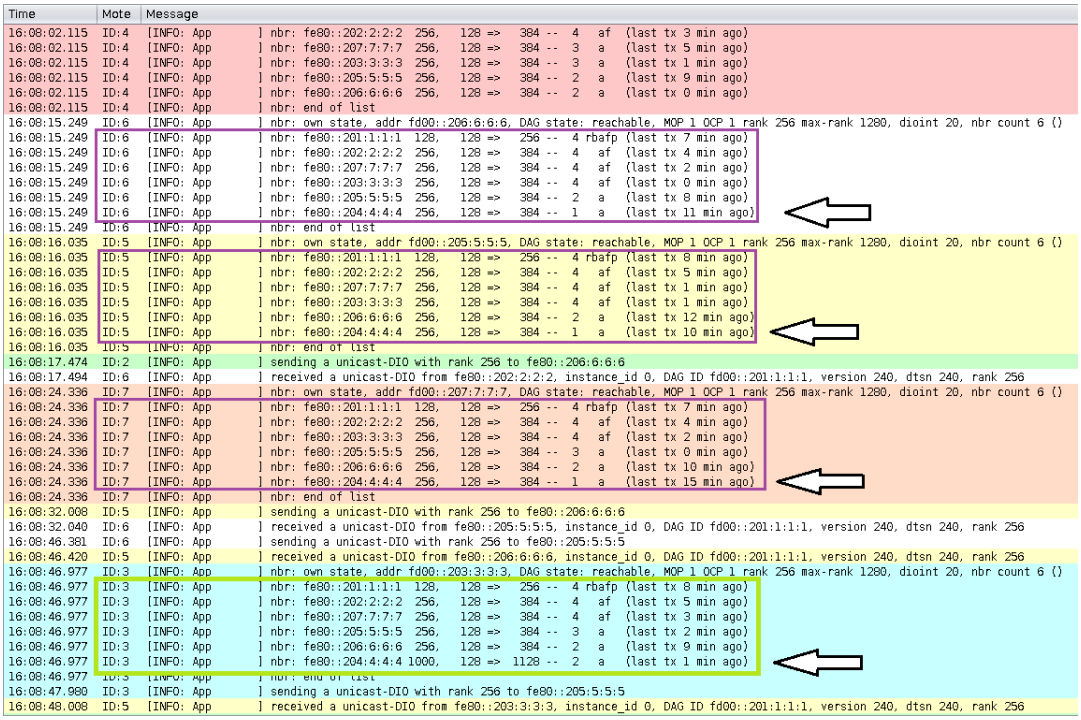


FIGURA 29: Variante 1 – DAG

Per implementare tale attacco è stato inserito il seguente codice all'interno del file *rpl-icmp6.c* nell'implementazione del metodo *rpl_icmp6_dio_output*.

```
1 char *vit="fe80::203:3:3:3";
2 uip_ipaddr_t ip;
3 uiplib_ip6addrconv(vit, &ip);
4
5 if(node_id == 4){ // attaccante
6     if(uip_ip6addr_cmp(&ip, addr))
7         curr_instance.dag.rank=1000; //rank poisoning
8 }
```

3.2.1.1 Mitigazione

Esistono diverse tecniche per mitigare tale tipologia di attacco, nel nostro caso di studio abbiamo deciso di adottare una tecnica di analisi statistica basata sul valore di rank dei nodi vicini. L'idea alla base di tale tecnica risiede nella convinzione che generalmente il valore di rank di un nodo, non può essere eccessivamente diverso rispetto a quello degli altri vicini. Il nostro obiettivo, quindi, è stabilire una soglia e considerare corretto il valore del rank solo se inferiore a tale soglia. In [Iuc+15] viene proposto di calcolare tale valore di soglia nel seguente modo:

$$soglia = MaxRank * (\frac{1}{n} - K) + RankMed$$

Dove:

- **MaxRank**: è il massimo valore di rank tra i vicini del nodo corrente;
- **n**: è il numero di vicini del nodo corrente;
- **K**: è una costante da determinare in modo sperimentale in funzione della topologia della rete;
- **RankMed**: è il rank medio dei vicini

La figura seguente mostra il log, in cui sono state filtrate tutte le righe che hanno valore 1000, come è possibile osservare il nodo 4 continua ad inviare alla vittima (3), i messaggi DIO con rank fasullo, ma il nodo 3 non dice di averli ricevuti perché vengono scartati e non elaborati. La conferma del corretto funzionamento della mitigazione lo si ha nella figura 5d in cui viene mostrato il DAG del nodo vittima, il quale associa al nodo 4 il valore di rank corretto propagato dai suoi vicini.

Time	Mote	Message
10:51:17.136	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
10:53:02.516	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
11:04:48.098	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
11:08:38.164	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
11:20:01.410	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
11:27:03.387	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
11:36:34.422	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
11:38:02.948	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
11:52:47.219	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
11:54:33.051	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
12:07:25.097	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
12:09:15.541	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
12:22:03.583	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
12:23:19.631	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
12:36:49.507	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
12:37:54.979	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
12:51:42.962	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
12:53:06.982	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
13:06:34.913	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
13:09:19.664	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
13:20:07.504	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
13:25:51.567	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
13:39:21.911	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
13:40:35.238	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
13:52:12.854	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
13:56:46.638	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
14:10:12.585	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
14:12:41.165	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
14:24:41.612	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
14:25:38.548	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
14:38:08.593	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
14:41:38.911	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
14:52:32.222	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
14:59:03.674	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
15:06:46.353	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
15:13:52.064	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
15:23:02.838	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
15:25:39.594	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
15:35:33.844	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
15:39:15.268	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
15:51:29.215	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
15:56:49.372	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
16:04:28.690	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
16:08:30.249	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
16:21:29.592	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
16:23:07.570	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3

FIGURA 30: Log mitigazione Rank Attack variante 1

Time	Mote	Message
02:20.512	ID:5	[INFO: App] nbr: fe80::204:4:4:4 256, 213 => 469 -- 0 a (no tx)
02:20.512	ID:5	[INFO: App] nbr: end of list
02:20.512	ID:7	[INFO: App] sending a multicast-DIO with rank 256 to ff02::1a
02:20.519	ID:1	[INFO: App] received a multicast-DIO from fe80::207:7:7:7, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
02:20.519	ID:2	[INFO: App] received a multicast-DIO from fe80::207:7:7:7, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
02:20.519	ID:3	[INFO: App] received a multicast-DIO from fe80::207:7:7:7, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
02:20.519	ID:4	[INFO: App] received a multicast-DIO from fe80::207:7:7:7, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
02:20.519	ID:5	[INFO: App] received a multicast-DIO from fe80::207:7:7:7, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
02:20.519	ID:6	[INFO: App] received a multicast-DIO from fe80::207:7:7:7, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
02:28.052	ID:7	[INFO: App] sending a unicast-DIO with rank 256 to fe80::202:2:2:2
02:28.061	ID:2	[INFO: App] received a unicast-DIO from fe80::207:7:7:7, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
02:35.452	ID:3	[INFO: App] sending a unicast-DIO with rank 256 to fe80::202:2:2:2
02:35.493	ID:2	[INFO: App] received a unicast-DIO from fe80::203:3:3:3, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
02:42.597	ID:3	[INFO: App] nbr: own state, addr fd00::203:3:3:3, DAG state: reachable, MOP 1 OCP 1 rank 256 max-rank 1280, dioint 17, nbr count 6 {}
02:42.597	ID:3	[INFO: App] nbr: fe80::201:1:1:1 128, 128 => 256 -- 5 rbafp (last tx 2 min ago)
02:42.597	ID:3	[INFO: App] nbr: fe80::202:2:2:2 256, 320 => 576 -- 1 a (last tx 0 min ago)
02:42.597	ID:3	[INFO: App] nbr: fe80::207:7:7:7 256, 213 => 469 -- 0 a (no tx)
02:42.597	ID:3	[INFO: App] nbr: fe80::205:5:5:5 256, 128 => 384 -- 1 a (last tx 1 min ago)
02:42.597	ID:3	[INFO: App] nbr: fe80::206:6:6:6 256, 320 => 576 -- 0 a (no tx)
02:42.597	ID:3	[INFO: App] nbr: fe80::204:4:4:4 256, 128 => 384 -- 0 a (no tx)
02:42.597	ID:3	[INFO: App] nbr: end of list
02:48.103	ID:2	[INFO: App] sending a unicast-DIO with rank 256 to fe80::207:7:7:7
02:48.122	ID:7	[INFO: App] received a unicast-DIO from fe80::202:2:2:2, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
02:51.297	ID:7	[INFO: App] nbr: own state, addr fd00::207:7:7:7, DAG state: reachable, MOP 1 OCP 1 rank 256 max-rank 1280, dioint 17, nbr count 6 {}
02:51.297	ID:7	[INFO: App] nbr: fe80::201:1:1:1 128, 128 => 256 -- 5 rbafp (last tx 2 min ago)
02:51.297	ID:7	[INFO: App] nbr: fe80::202:2:2:2 256, 128 => 384 -- 2 a (last tx 0 min ago)
02:51.297	ID:7	[INFO: App] nbr: fe80::203:3:3:3 256, 213 => 469 -- 0 a (no tx)
02:51.297	ID:7	[INFO: App] nbr: fe80::205:5:5:5 256, 128 => 384 -- 0 a (no tx)
02:51.297	ID:7	[INFO: App] nbr: fe80::206:6:6:6 256, 213 => 469 -- 0 a (no tx)
02:51.297	ID:7	[INFO: App] nbr: fe80::204:4:4:4 256, 320 => 576 -- 0 a (no tx)
02:51.297	ID:7	[INFO: App] nbr: end of list
02:52.124	ID:5	[INFO: App] sending a unicast-DIO with rank 256 to fe80::207:7:7:7
02:52.161	ID:7	[INFO: App] received a unicast-DIO from fe80::205:5:5:5, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
02:54.864	ID:4	[INFO: App] nbr: own state, addr fd00::204:4:4:4, DAG state: reachable, MOP 1 OCP 1 rank 256 max-rank 1280, dioint 17, nbr count 6 {}
02:54.864	ID:4	[INFO: App] nbr: fe80::201:1:1:1 128, 128 => 256 -- 5 rbafp (last tx 2 min ago)
02:54.864	ID:4	[INFO: App] nbr: fe80::202:2:2:2 256, 191 => 447 -- 1 a (last tx 0 min ago)
02:54.864	ID:4	[INFO: App] nbr: fe80::207:7:7:7 256, 320 => 576 -- 0 a (no tx)
02:54.864	ID:4	[INFO: App] nbr: fe80::203:3:3:3 256, 128 => 384 -- 0 a (no tx)
02:54.864	ID:4	[INFO: App] nbr: fe80::205:5:5:5 256, 213 => 469 -- 0 a (no tx)
02:54.864	ID:4	[INFO: App] nbr: fe80::206:6:6:6 256, 128 => 384 -- 0 a (no tx)
02:54.864	ID:4	[INFO: App] nbr: end of list
02:56.460	ID:6	[INFO: App] sending a unicast-DIO with rank 256 to fe80::202:2:2:2
02:56.482	ID:2	[INFO: App] received a unicast-DIO from fe80::206:6:6:6, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
03:04.515	ID:4	[INFO: App] sending a unicast-DIO with rank 256 to fe80::203:3:3:3
03:04.540	ID:3	[INFO: App] received a unicast-DIO from fe80::204:4:4:4, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
03:08.084	ID:2	[INFO: App] nbr: own state, addr fd00::202:2:2:2, DAG state: reachable, MOP 1 OCP 1 rank 256 max-rank 1280, dioint 17, nbr count 6 {}

FIGURA 31: Mitigazione Rank Attack – DAG nodo vittima

Di seguito il codice necessario per implementare la mitigazione. Il file di riferimento è sempre *rpl-icmp6.c*, mentre il metodo è *dio_input*.

```
1  rpl_nbr_t *nbr =rpl_parent_get_from_ipaddr(&UIP_IP_BUF->srcipaddr);
       //nodi vicini
2    rpl_rank_t somma=0, max=0;
3    int   count=0;
4
5    while (nbr != NULL){
6      rpl_rank_t t= rpl_neighbor_rank_via_nbr(nbr);
7      somma = somma + t;
8      count++;
9      if(t>max)
10       max=t;
11     nbr = nbr_table_next(rpl_neighbors, nbr);
12   }
13   if(count != 0){
14     rpl_rank_t med = (rpl_rank_t) somma/count;
15     rpl_rank_t soglia = (rpl_rank_t) max * (1/count - K) + med; //
      K = 0.3
16     if(dio.rank >= soglia)
17       goto discard;
18   }
```

Il file e/o metodo all'interno del quale sono implementati attacchi e mitigazioni del rank è comune a tutti e tre gli scenari rappresentati. Si eviterà quindi di riportalo nuovamente negli scenari successivi.

3.2.2 Variante 2

In questa seconda variante dell'attacco, si affronta uno scenario più complesso rispetto al precedente, troviamo infatti, due nodi malevoli (ID = 4 e 9) che inviano dai rank fasulli a tutti gli altri nodi della rete, con lo scopo di propagare il poisoning a tutta la rete. Al contrario del caso precedente, in tale contesto, si applica un approccio graduale per simulare meglio un ambiente reale. I due nodi, infatti, entrano nella rete come nodi legittimi che inviano valore di rank corretto e solo dopo un certo periodo di tempo iniziano ad effettuare il poisoning.

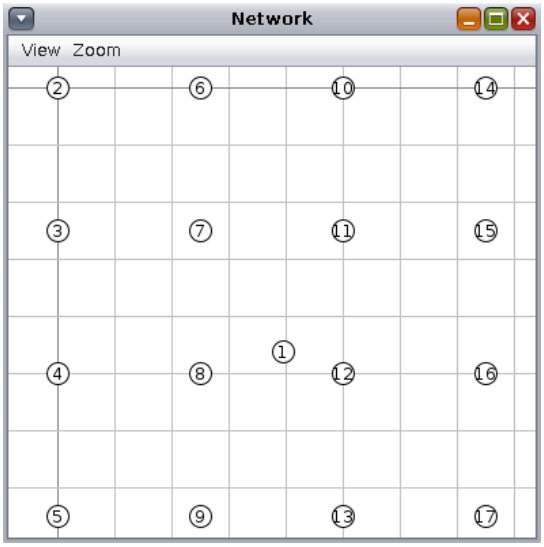
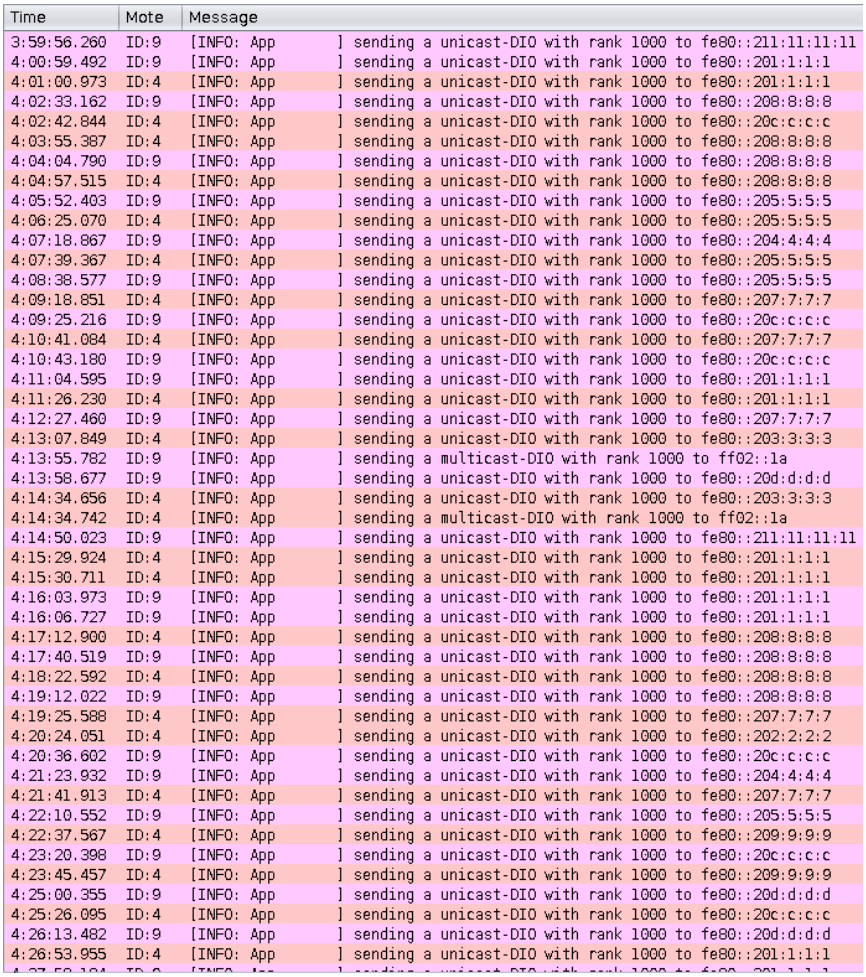


FIGURA 32: Topologia variante 2

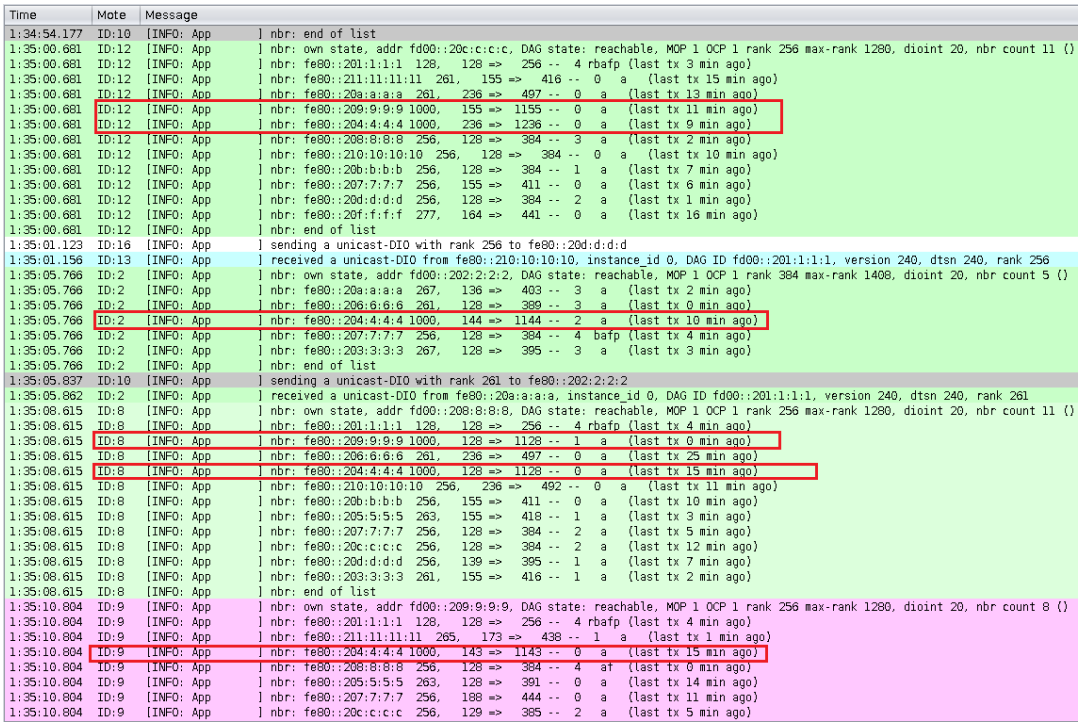
Nella figura sottostante si può osservare come tutti i messaggi DIO inviati dai nodi 4 e 9 contengano un rank falso.



Time	Mote	Message
3:59:56.260	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::211:11:11:11
4:00:59.492	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::201:1:1:1
4:01:00.973	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::201:1:1:1
4:02:33.162	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::208:8:8:8
4:02:42.844	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::20c:c:c:c
4:03:55.387	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::208:8:8:8
4:04:04.790	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::208:8:8:8
4:04:57.515	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::208:8:8:8
4:05:52.403	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::205:5:5:5
4:06:25.070	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::205:5:5:5
4:07:18.867	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::204:4:4:4
4:07:39.367	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::205:5:5:5
4:08:38.577	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::205:5:5:5
4:09:18.851	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::207:7:7:7
4:09:25.216	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::20c:c:c:c
4:10:41.084	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::207:7:7:7
4:10:43.180	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::20c:c:c:c
4:11:04.595	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::201:1:1:1
4:11:26.230	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::201:1:1:1
4:12:27.460	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::207:7:7:7
4:13:07.849	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
4:13:55.782	ID:9	[INFO: App] sending a multicast-DIO with rank 1000 to ff02::1a
4:13:58.677	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::20d:d:d:d
4:14:34.656	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::203:3:3:3
4:14:34.742	ID:4	[INFO: App] sending a multicast-DIO with rank 1000 to ff02::1a
4:14:50.023	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::211:11:11:11
4:15:29.924	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::201:1:1:1
4:15:30.711	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::201:1:1:1
4:16:03.973	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::201:1:1:1
4:16:06.727	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::201:1:1:1
4:17:12.900	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::208:8:8:8
4:17:40.519	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::208:8:8:8
4:18:22.592	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::208:8:8:8
4:19:12.022	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::208:8:8:8
4:19:25.588	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::207:7:7:7
4:20:24.051	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::202:2:2:2
4:20:36.602	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::20c:c:c:c
4:21:23.932	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::204:4:4:4
4:21:41.913	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::207:7:7:7
4:22:10.552	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::205:5:5:5
4:22:37.567	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::209:9:9:9
4:23:20.398	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::20c:c:c:c
4:23:45.457	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::209:9:9:9
4:25:00.355	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::20d:d:d:d
4:25:26.095	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::20c:c:c:c
4:26:13.482	ID:9	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::20d:d:d:d
4:26:53.955	ID:4	[INFO: App] sending a unicast-DIO with rank 1000 to fe80::201:1:1:1

FIGURA 33: Variante 2 - Log

Nella figura 34 è possibile osservare come l'attacco vada a buon fine. Sono state evidenziate in rosso le righe dei DAG dei diversi nodi alterate dall'attacco.



```
Time          Mote  Message
1:34:54.177   ID:10 [INFO: App       ] nbr: end of list
1:35:00.681   ID:12 [INFO: App       ] nbr: own state, addr fd00::20c:c:c:c, DAG state: reachable, MOP 1 OCP 1 rank 256 max-rank 1280, dioint 20, nbr count 11 ()
1:35:00.681   ID:12 [INFO: App       ] nbr: fe80::201:1:1:1  128,   128 =>   256 --  4 rbafp (last tx 3 min ago)
1:35:00.681   ID:12 [INFO: App       ] nbr: fe80::211:11:11:11  261,   155 =>   416 --  0   a   (last tx 15 min ago)
1:35:00.681   ID:12 [INFO: App       ] nbr: fe80::20a:a:a:a  261,   236 =>   497 --  0   a   (last tx 13 min ago)
1:35:00.681   ID:12 [INFO: App       ] nbr: fe80::209:9:9:9 1000,   155 =>  1155 --  0   a   (last tx 11 min ago)
1:35:00.681   ID:12 [INFO: App       ] nbr: fe80::204:4:4:4 1000,   236 =>  1236 --  0   a   (last tx 9 min ago)
1:35:00.681   ID:12 [INFO: App       ] nbr: fe80::208:8:8:8  256,   128 =>   384 --  3   a   (last tx 2 min ago)
1:35:00.681   ID:12 [INFO: App       ] nbr: fe80::210:10:10:10  256,   128 =>   384 --  0   a   (last tx 10 min ago)
1:35:00.681   ID:12 [INFO: App       ] nbr: fe80::20b:b:b:b  256,   128 =>   384 --  1   a   (last tx 7 min ago)
1:35:00.681   ID:12 [INFO: App       ] nbr: fe80::207:7:7:7  256,   155 =>   411 --  0   a   (last tx 6 min ago)
1:35:00.681   ID:12 [INFO: App       ] nbr: fe80::20d:d:d:d  256,   128 =>   384 --  2   a   (last tx 1 min ago)
1:35:00.681   ID:12 [INFO: App       ] nbr: fe80::20f:f:f:f  277,   164 =>   441 --  0   a   (last tx 16 min ago)
1:35:00.681   ID:12 [INFO: App       ] nbr: end of list
1:35:01.123   ID:16 [INFO: App       ] sending a unicast-DIO with rank 256 to fe80::20d:d:d:d
1:35:01.156   ID:13 [INFO: App       ] received a unicast-DIO from fe80::210:10:10:10, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
1:35:05.766   ID:2  [INFO: App       ] nbr: own state, addr fd00::202:2:2:2, DAG state: reachable, MOP 1 OCP 1 rank 384 max-rank 1408, dioint 20, nbr count 5 ()
1:35:05.766   ID:2  [INFO: App       ] nbr: fe80::20a:a:a:a  267,   136 =>   403 --  3   a   (last tx 2 min ago)
1:35:05.766   ID:2  [INFO: App       ] nbr: fe80::206:6:6:6  261,   128 =>   389 --  3   a   (last tx 0 min ago)
1:35:05.766   ID:2  [INFO: App       ] nbr: fe80::204:4:4:4 1000,   144 =>  1144 --  2   a   (last tx 10 min ago)
1:35:05.766   ID:2  [INFO: App       ] nbr: fe80::207:7:7:7  256,   128 =>   384 --  4  bafp (last tx 4 min ago)
1:35:05.766   ID:2  [INFO: App       ] nbr: fe80::203:3:3:3  267,   128 =>   395 --  3   a   (last tx 3 min ago)
1:35:05.766   ID:2  [INFO: App       ] nbr: end of list
1:35:05.837   ID:10 [INFO: App       ] sending a unicast-DIO with rank 261 to fe80::202:2:2:2
1:35:05.862   ID:2  [INFO: App       ] received a unicast-DIO from fe80::20a:a:a:a, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 261
1:35:08.615   ID:8  [INFO: App       ] nbr: own state, addr fd00::208:8:8:8, DAG state: reachable, MOP 1 OCP 1 rank 256 max-rank 1280, dioint 20, nbr count 11 ()
1:35:08.615   ID:8  [INFO: App       ] nbr: fe80::201:1:1:1  128,   128 =>   256 --  4 rbafp (last tx 4 min ago)
1:35:08.615   ID:8  [INFO: App       ] nbr: fe80::209:9:9:9 1000,   128 =>  1128 --  1   a   (last tx 0 min ago)
1:35:08.615   ID:8  [INFO: App       ] nbr: fe80::206:6:6:6  261,   236 =>   497 --  0   a   (last tx 25 min ago)
1:35:08.615   ID:8  [INFO: App       ] nbr: fe80::204:4:4:4 1000,   128 =>  1128 --  0   a   (last tx 15 min ago)
1:35:08.615   ID:8  [INFO: App       ] nbr: fe80::210:10:10:10  256,   236 =>   492 --  0   a   (last tx 11 min ago)
1:35:08.615   ID:8  [INFO: App       ] nbr: fe80::20b:b:b:b  256,   155 =>   411 --  0   a   (last tx 10 min ago)
1:35:08.615   ID:8  [INFO: App       ] nbr: fe80::205:5:5:5  263,   155 =>   418 --  1   a   (last tx 3 min ago)
1:35:08.615   ID:8  [INFO: App       ] nbr: fe80::207:7:7:7  256,   128 =>   384 --  2   a   (last tx 5 min ago)
1:35:08.615   ID:8  [INFO: App       ] nbr: fe80::20c:c:c:c  256,   128 =>   384 --  2   a   (last tx 12 min ago)
1:35:08.615   ID:8  [INFO: App       ] nbr: fe80::20d:d:d:d  256,   139 =>   395 --  1   a   (last tx 7 min ago)
1:35:08.615   ID:8  [INFO: App       ] nbr: fe80::203:3:3:3  261,   155 =>   416 --  1   a   (last tx 2 min ago)
1:35:08.615   ID:8  [INFO: App       ] nbr: end of list
1:35:10.804   ID:9  [INFO: App       ] nbr: own state, addr fd00::209:9:9:9, DAG state: reachable, MOP 1 OCP 1 rank 256 max-rank 1280, dioint 20, nbr count 8 ()
1:35:10.804   ID:9  [INFO: App       ] nbr: fe80::201:1:1:1  128,   128 =>   256 --  4 rbafp (last tx 4 min ago)
1:35:10.804   ID:9  [INFO: App       ] nbr: fe80::211:11:11:11  265,   173 =>   438 --  1   a   (last tx 1 min ago)
1:35:10.804   ID:9  [INFO: App       ] nbr: fe80::204:4:4:4 1000,   143 =>  1143 --  0   a   (last tx 15 min ago)
1:35:10.804   ID:9  [INFO: App       ] nbr: fe80::208:8:8:8  256,   128 =>   384 --  4  af   (last tx 0 min ago)
1:35:10.804   ID:9  [INFO: App       ] nbr: fe80::205:5:5:5  263,   128 =>   391 --  0   a   (last tx 14 min ago)
1:35:10.804   ID:9  [INFO: App       ] nbr: fe80::207:7:7:7  256,   188 =>   444 --  0   a   (last tx 11 min ago)
1:35:10.804   ID:9  [INFO: App       ] nbr: fe80::20c:c:c:c  256,   129 =>   385 --  2   a   (last tx 5 min ago)
```

FIGURA 34: Variante 2 – DAG

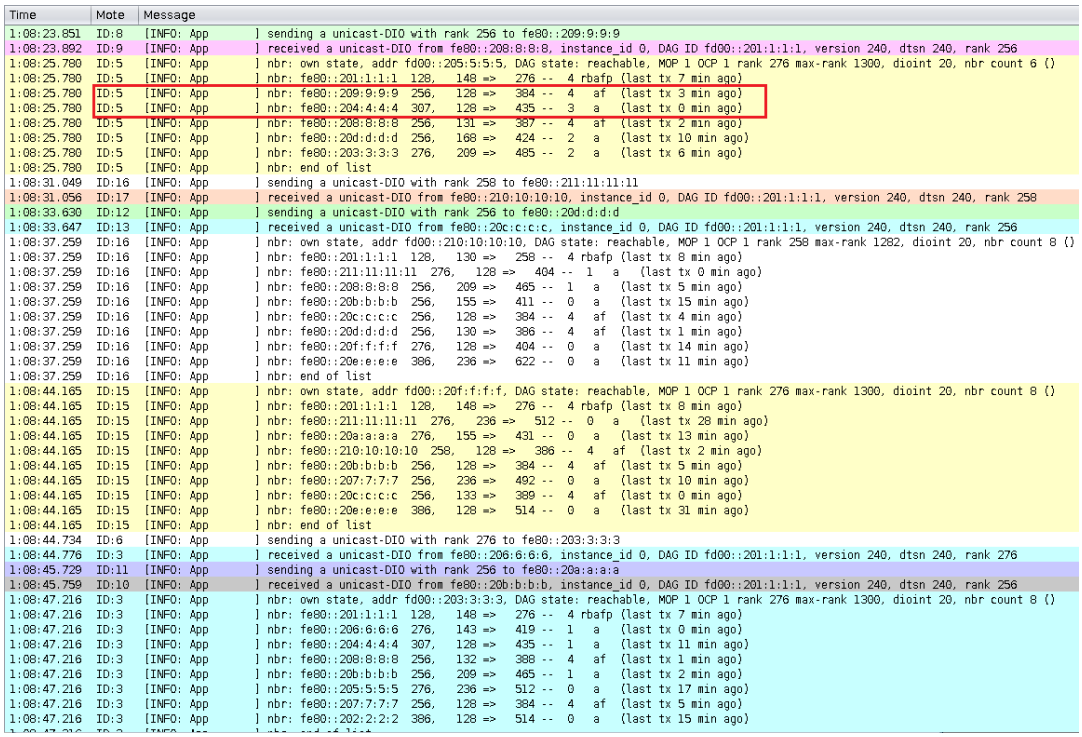
Di seguito il codice necessario per l'implementazione dell'attacco.

```
1 if(var <=0){//var inizializzato a 50. Simula attesa prima dell'
    attacco
2     if(node_id == 4 || node_id == 9){ // attaccanti
3         curr_instance.dag.rank=1000; //rank poisoning
4     }
5 } else{
6         var--;
7     }
```

3.2.2.1 Mitigazione

La mitigazione è comune a quella descritta nello scenario precedente.

Nella figura 6d è possibile osservare il corretto funzionamento della mitigazione, infatti nei DAG dei diversi nodi, non compare mai il valore di rank 1000, ma è presenta il valore reale che i nodi malevoli hanno iniviato nella prima fase.



```
Time          Mote    Message
1:08:23.851   ID:8    [INFO: App       ] sending a unicast-DIO with rank 256 to fe80::209:9:9:9
1:08:23.892   ID:9    [INFO: App       ] received a unicast-DIO from fe80::208:8:8:8, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
1:08:25.780   ID:5    [INFO: App       ] nbr: own state, addr fd00::205:5:5:5, DAG state: reachable, MOP 1 OCP 1 rank 276 max-rank 1300, dioint 20, nbr count 6 ()
1:08:25.780   ID:5    [INFO: App       ] nbr: fe80::201:1:1:1  128,   148 =>   276 --  4 rbafp (last tx 7 min ago)
1:08:25.780   ID:5    [INFO: App       ] nbr: fe80::209:9:9:9  256,   128 =>   384 --  4   af  (last tx 3 min ago)
1:08:25.780   ID:5    [INFO: App       ] nbr: fe80::204:4:4:4  307,   128 =>   435 --  3   a   (last tx 0 min ago)
1:08:25.780   ID:5    [INFO: App       ] nbr: fe80::208:8:8:8  256,   131 =>   387 --  4   af  (last tx 2 min ago)
1:08:25.780   ID:5    [INFO: App       ] nbr: fe80::20d:d:d:d  256,   168 =>   424 --  2   a   (last tx 10 min ago)
1:08:25.780   ID:5    [INFO: App       ] nbr: fe80::203:3:3:3  276,   209 =>   485 --  2   a   (last tx 6 min ago)
1:08:25.780   ID:5    [INFO: App       ] nbr: end of list
1:08:31.049   ID:16   [INFO: App       ] sending a unicast-DIO with rank 258 to fe80::211:11:11:11
1:08:31.056   ID:17   [INFO: App       ] received a unicast-DIO from fe80::210:10:10:10, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 258
1:08:33.630   ID:12   [INFO: App       ] sending a unicast-DIO with rank 256 to fe80::20d:d:d:d
1:08:33.647   ID:13   [INFO: App       ] received a unicast-DIO from fe80::20c:c:c:c, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
1:08:37.259   ID:16   [INFO: App       ] nbr: own state, addr fd00::210:10:10:10, DAG state: reachable, MOP 1 OCP 1 rank 258 max-rank 1282, dioint 20, nbr count 8 ()
1:08:37.259   ID:16   [INFO: App       ] nbr: fe80::201:1:1:1  128,   130 =>   258 --  4 rbafp (last tx 8 min ago)
1:08:37.259   ID:16   [INFO: App       ] nbr: fe80::211:11:11:11  276,   128 =>   404 --  1   a   (last tx 0 min ago)
1:08:37.259   ID:16   [INFO: App       ] nbr: fe80::208:8:8:8  256,   209 =>   465 --  1   a   (last tx 5 min ago)
1:08:37.259   ID:16   [INFO: App       ] nbr: fe80::20b:b:b:b  256,   155 =>   411 --  0   a   (last tx 15 min ago)
1:08:37.259   ID:16   [INFO: App       ] nbr: fe80::20c:c:c:c  256,   128 =>   384 --  4   af  (last tx 4 min ago)
1:08:37.259   ID:16   [INFO: App       ] nbr: fe80::20d:d:d:d  256,   130 =>   386 --  4   af  (last tx 1 min ago)
1:08:37.259   ID:16   [INFO: App       ] nbr: fe80::20f:f:f:f  276,   128 =>   404 --  0   a   (last tx 14 min ago)
1:08:37.259   ID:16   [INFO: App       ] nbr: fe80::20e:e:e:e  386,   236 =>   622 --  0   a   (last tx 11 min ago)
1:08:37.259   ID:16   [INFO: App       ] nbr: end of list
1:08:44.165   ID:15   [INFO: App       ] nbr: own state, addr fd00::20f:f:f:f, DAG state: reachable, MOP 1 OCP 1 rank 276 max-rank 1300, dioint 20, nbr count 8 ()
1:08:44.165   ID:15   [INFO: App       ] nbr: fe80::201:1:1:1  128,   148 =>   276 --  4 rbafp (last tx 8 min ago)
1:08:44.165   ID:15   [INFO: App       ] nbr: fe80::211:11:11:11  276,   236 =>   512 --  0   a   (last tx 28 min ago)
1:08:44.165   ID:15   [INFO: App       ] nbr: fe80::20a:a:a:a  276,   155 =>   431 --  0   a   (last tx 13 min ago)
1:08:44.165   ID:15   [INFO: App       ] nbr: fe80::210:10:10:10  258,   128 =>   386 --  4   af  (last tx 2 min ago)
1:08:44.165   ID:15   [INFO: App       ] nbr: fe80::20b:b:b:b  256,   128 =>   384 --  4   af  (last tx 5 min ago)
1:08:44.165   ID:15   [INFO: App       ] nbr: fe80::207:7:7:7  256,   236 =>   492 --  0   a   (last tx 10 min ago)
1:08:44.165   ID:15   [INFO: App       ] nbr: fe80::20c:c:c:c  256,   133 =>   389 --  4   af  (last tx 0 min ago)
1:08:44.165   ID:15   [INFO: App       ] nbr: fe80::20e:e:e:e  386,   128 =>   514 --  0   a   (last tx 31 min ago)
1:08:44.165   ID:15   [INFO: App       ] nbr: end of list
1:08:44.734   ID:6    [INFO: App       ] sending a unicast-DIO with rank 276 to fe80::203:3:3:3
1:08:44.776   ID:3    [INFO: App       ] received a unicast-DIO from fe80::206:6:6:6, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 276
1:08:45.729   ID:11   [INFO: App       ] sending a unicast-DIO with rank 256 to fe80::20a:a:a:a
1:08:45.759   ID:10   [INFO: App       ] received a unicast-DIO from fe80::20b:b:b:b, instance_id 0, DAG ID fd00::201:1:1:1, version 240, dtsn 240, rank 256
1:08:47.216   ID:3    [INFO: App       ] nbr: own state, addr fd00::203:3:3:3, DAG state: reachable, MOP 1 OCP 1 rank 276 max-rank 1300, dioint 20, nbr count 8 ()
1:08:47.216   ID:3    [INFO: App       ] nbr: fe80::201:1:1:1  128,   148 =>   276 --  4 rbafp (last tx 7 min ago)
1:08:47.216   ID:3    [INFO: App       ] nbr: fe80::206:6:6:6  276,   143 =>   419 --  1   a   (last tx 0 min ago)
1:08:47.216   ID:3    [INFO: App       ] nbr: fe80::204:4:4:4  307,   128 =>   435 --  1   a   (last tx 11 min ago)
1:08:47.216   ID:3    [INFO: App       ] nbr: fe80::208:8:8:8  256,   132 =>   388 --  4   af  (last tx 1 min ago)
1:08:47.216   ID:3    [INFO: App       ] nbr: fe80::20b:b:b:b  256,   209 =>   465 --  1   a   (last tx 2 min ago)
1:08:47.216   ID:3    [INFO: App       ] nbr: fe80::205:5:5:5  276,   236 =>   512 --  0   a   (last tx 17 min ago)
1:08:47.216   ID:3    [INFO: App       ] nbr: fe80::207:7:7:7  256,   128 =>   384 --  4   af  (last tx 5 min ago)
1:08:47.216   ID:3    [INFO: App       ] nbr: fe80::202:2:2:2  386,   128 =>   514 --  0   a   (last tx 15 min ago)
```

FIGURA 35: Log mitigazione Rank Attack variante 2

3.2.3 Variante 3

Nell'ultimo scenario, pur mantenendo la topologia mostrata nella figura 32, si mostra un utilizzo alternativo del rank attack rispetto al semplice poisoning visto nei casi precedenti. Il rank attack può infatti essere utilizzato come attacco di apertura per ulteriori attacchi come ad esempio un man in the middle, che ha sua volta può portare ad molti altri scenari come drop dei pacchetti o lo sniffing. Di seguito vogliamo utilizzare il rank attack per realizzare un man in the middle, quindi vogliamo far in modo che il nodo vittima scelga il nodo malevolo come suo padre nel DAG, in questo modo quando la vittima deve inoltrare un pacchetto sulla rete, lo farà attraverso il nodo malevolo che quindi potrà controllare tale pacchetto. Per implementare ciò è necessario che il nodo malevolo effettui un rank poisoning nei confronti della vittima, inserendo all'interno del messaggio DIO un valore di rank ottimale, quindi più basso, rispetto a quello di tutti i vicini del nodo vittima.
In questo scenario abbiamo due nodi malevoli i nodi 4 e 15. In figura 36 possiamo vedere come come il poisoning riesca correttamente ed in più è possibile vedere come i nodi vittima scelgano i nodi malevoli come loro padri, attraverso la stringa "***bafp***" riportata nelle righe dei DAG in riferimento ai nodi malevoli. La stringa "***bafp***" va interpretata nel seguente modo:

- **b**: Best rank. Indica che il nodo ha il miglior valore di rank rispetto agli altri vicini;
- **a**: il rank è accettabile (valido) e il nodo è un possibile candidato ad essere scelto come genitore nel DAG;
- **f**: indica che il collegamento è aggiornato;
- **p**: indica che il nodo è stato scelto come padre nel DAG.

La presenza quindi dei due caratteri *b* e *p* nei DAG dei nodi vittima 14 e 2, mostrano come l'attacco sia riuscito correttamente. In particolare, si puo osservare come il nodo 2 scelga il nodo malevolo 4 come suo padre nel DAG, e come il nodo 14 scelga il nodo malevolo 15.

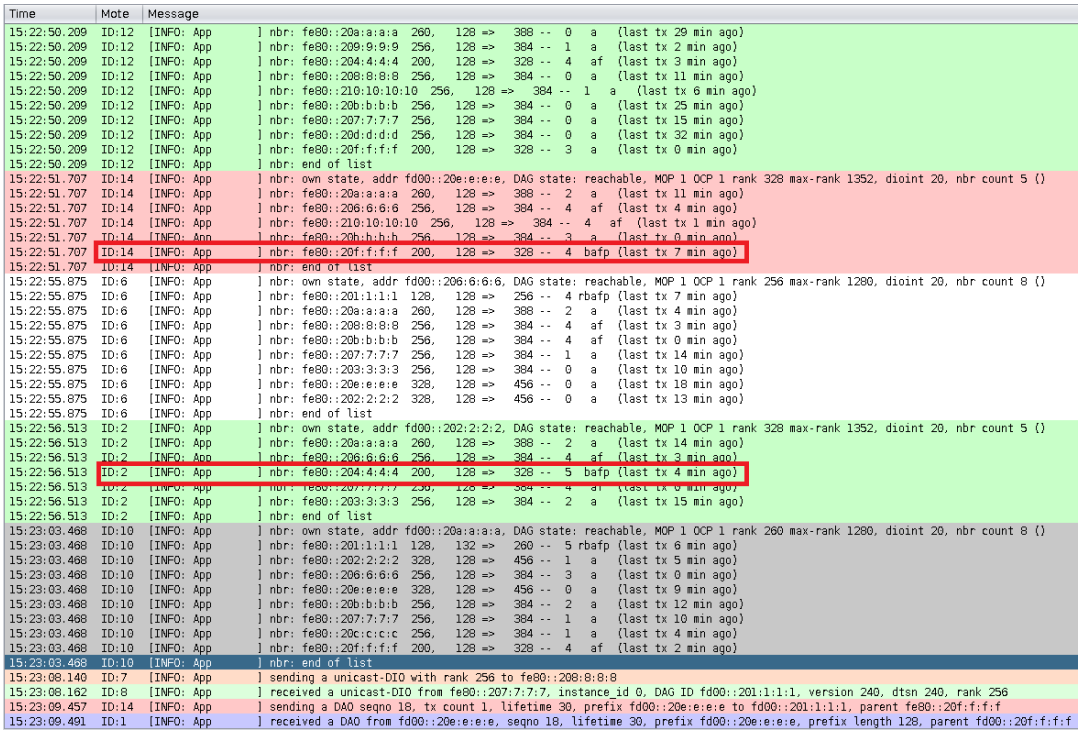


FIGURA 36: Variante 3 – DAG

Di seguito il codice necessario per l'implementazione di tale attacco.

```
1 if(var <=0){ //Inizializzato a 50. Simula attesa prima dell'attacco
2     if(node_id == 4 || node_id == 15){ // attaccanti
3         curr_instance.dag.rank=200; //rank poisoning
4     }
5 } else{
6       var--;
7     }
```

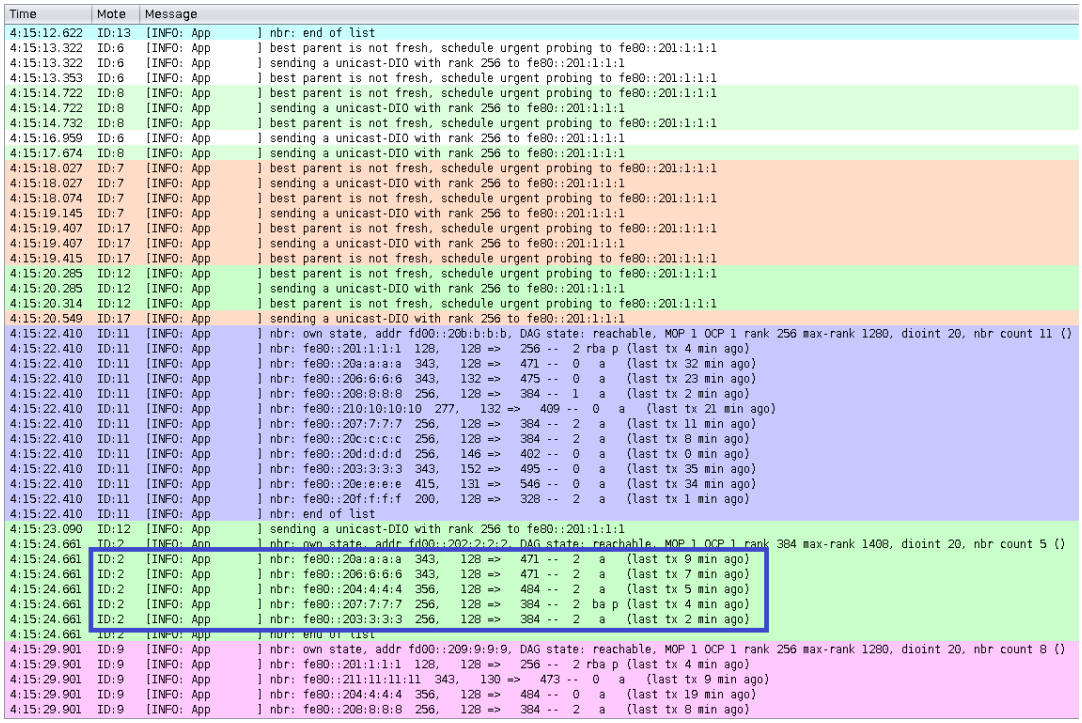
3.2.3.1 Mitigazione

La mitigazione è comune ai due casi trattati in precedenza, l'unica differenza è che la soglia in questo caso essa funge da lower bound, rispetto al ruolo di upper bound ricoperta nei due casi precedenti. La formula per il calcolo della soglia viene modificata nel seguente modo:

$$soglia = RankMed - MinRank * (\frac{1}{n} - K)$$

I cui i vari termini hanno lo steso significato del caso precedente, ad eccezione di MinRank che rappresenta il valore di rank minino tra i vicini del nodo in considerazione.

La figura sottostante mostra come la mitigazione funzioni correttamente, si può osservare infatti, come nel dag del nodo vittima 2 non sia più presente la stringa bafp, si riconosce dunque il nodo 4 come attaccante e si impedisce l'attacco man in the middle.



```
Time          Mote  Message
4:15:12.622   ID:13 [INFO: App       ] nbr: end of list
4:15:13.322   ID:6  [INFO: App       ] best parent is not fresh, schedule urgent probing to fe80::201:1:1:1
4:15:13.322   ID:6  [INFO: App       ] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
4:15:13.353   ID:6  [INFO: App       ] best parent is not fresh, schedule urgent probing to fe80::201:1:1:1
4:15:14.722   ID:8  [INFO: App       ] best parent is not fresh, schedule urgent probing to fe80::201:1:1:1
4:15:14.722   ID:8  [INFO: App       ] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
4:15:14.732   ID:8  [INFO: App       ] best parent is not fresh, schedule urgent probing to fe80::201:1:1:1
4:15:16.959   ID:6  [INFO: App       ] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
4:15:17.674   ID:8  [INFO: App       ] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
4:15:18.027   ID:7  [INFO: App       ] best parent is not fresh, schedule urgent probing to fe80::201:1:1:1
4:15:18.027   ID:7  [INFO: App       ] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
4:15:18.074   ID:7  [INFO: App       ] best parent is not fresh, schedule urgent probing to fe80::201:1:1:1
4:15:19.145   ID:7  [INFO: App       ] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
4:15:19.407   ID:17 [INFO: App       ] best parent is not fresh, schedule urgent probing to fe80::201:1:1:1
4:15:19.407   ID:17 [INFO: App       ] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
4:15:19.415   ID:17 [INFO: App       ] best parent is not fresh, schedule urgent probing to fe80::201:1:1:1
4:15:20.285   ID:12 [INFO: App       ] best parent is not fresh, schedule urgent probing to fe80::201:1:1:1
4:15:20.285   ID:12 [INFO: App       ] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
4:15:20.314   ID:12 [INFO: App       ] best parent is not fresh, schedule urgent probing to fe80::201:1:1:1
4:15:20.549   ID:17 [INFO: App       ] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
4:15:22.410   ID:11 [INFO: App       ] nbr: own state, addr fd00::20b:b:b:b, DAG state: reachable, MOP 1 OCP 1 rank 256 max-rank 1280, dioint 20, nbr count 11 ()
4:15:22.410   ID:11 [INFO: App       ] nbr: fe80::201:1:1:1  128,   128 =>   256 --  2 rba p (last tx 4 min ago)
4:15:22.410   ID:11 [INFO: App       ] nbr: fe80::20a:a:a:a  343,   128 =>   471 --  0   a   (last tx 32 min ago)
4:15:22.410   ID:11 [INFO: App       ] nbr: fe80::206:6:6:6  343,   132 =>   475 --  0   a   (last tx 23 min ago)
4:15:22.410   ID:11 [INFO: App       ] nbr: fe80::208:8:8:8  256,   128 =>   384 --  1   a   (last tx 2 min ago)
4:15:22.410   ID:11 [INFO: App       ] nbr: fe80::210:10:10:10  277,   132 =>   409 --  0   a   (last tx 21 min ago)
4:15:22.410   ID:11 [INFO: App       ] nbr: fe80::207:7:7:7  256,   128 =>   384 --  2   a   (last tx 11 min ago)
4:15:22.410   ID:11 [INFO: App       ] nbr: fe80::20c:c:c:c  256,   128 =>   384 --  2   a   (last tx 8 min ago)
4:15:22.410   ID:11 [INFO: App       ] nbr: fe80::20d:d:d:d  256,   146 =>   402 --  0   a   (last tx 0 min ago)
4:15:22.410   ID:11 [INFO: App       ] nbr: fe80::203:3:3:3  343,   152 =>   495 --  0   a   (last tx 35 min ago)
4:15:22.410   ID:11 [INFO: App       ] nbr: fe80::20e:e:e:e  415,   131 =>   546 --  0   a   (last tx 34 min ago)
4:15:22.410   ID:11 [INFO: App       ] nbr: fe80::20f:f:f:f  200,   128 =>   328 --  2   a   (last tx 1 min ago)
4:15:22.410   ID:11 [INFO: App       ] nbr: end of list
4:15:23.090   ID:12 [INFO: App       ] sending a unicast-DIO with rank 256 to fe80::201:1:1:1
4:15:24.661   ID:2  [INFO: App       ] nbr: own state, addr fd00::202:2:2:2, DAG state: reachable, MOP 1 OCP 1 rank 384 max-rank 1408, dioint 20, nbr count 5 ()
4:15:24.661   ID:2  [INFO: App       ] nbr: fe80::20a:a:a:a  343,   128 =>   471 --  2   a   (last tx 9 min ago)
4:15:24.661   ID:2  [INFO: App       ] nbr: fe80::206:6:6:6  343,   128 =>   471 --  2   a   (last tx 7 min ago)
4:15:24.661   ID:2  [INFO: App       ] nbr: fe80::204:4:4:4  356,   128 =>   484 --  2   a   (last tx 5 min ago)
4:15:24.661   ID:2  [INFO: App       ] nbr: fe80::207:7:7:7  256,   128 =>   384 --  2  ba p (last tx 4 min ago)
4:15:24.661   ID:2  [INFO: App       ] nbr: fe80::203:3:3:3  256,   128 =>   384 --  2   a   (last tx 2 min ago)
4:15:24.661   ID:2  [INFO: App       ] nbr: end of list
4:15:29.901   ID:9  [INFO: App       ] nbr: own state, addr fd00::209:9:9:9, DAG state: reachable, MOP 1 OCP 1 rank 256 max-rank 1280, dioint 20, nbr count 8 ()
4:15:29.901   ID:9  [INFO: App       ] nbr: fe80::201:1:1:1  128,   128 =>   256 --  2 rba p (last tx 4 min ago)
4:15:29.901   ID:9  [INFO: App       ] nbr: fe80::211:11:11:11  343,   130 =>   473 --  0   a   (last tx 9 min ago)
4:15:29.901   ID:9  [INFO: App       ] nbr: fe80::204:4:4:4  356,   128 =>   484 --  0   a   (last tx 19 min ago)
4:15:29.901   ID:9  [INFO: App       ] nbr: fe80::208:8:8:8  256,   128 =>   384 --  2   a   (last tx 8 min ago)
```

FIGURA 37: Log mitigazione Rank Attack variante 3

Di seguito il codice necessario all'implementazione della mitigazione.

```
rpl_nbr_t *nbr =rpl_parent_get_from_ipaddr(&UIP_IP_BUF->srcipaddr);
  rpl_rank_t somma=0, min=10000;
  int   count=0;

  while (nbr!= NULL){
    rpl_rank_t t= rpl_neighbor_rank_via_nbr(nbr);
    somma =somma+t;
    count++;
    if(t<min)
      min=t;
    nbr = nbr_table_next(rpl_neighbors, nbr);
  }
  if(count!=0){
   rpl_rank_t med= (rpl_rank_t) somma/count;
    rpl_rank_t soglia= (rpl_rank_t) (med -  (min * (1/count - K)));

    if(dio.rank <= soglia)
      goto discard;
  }
```

Il codice sorgente e le simulazioni avviabili in cooja di questo attacco sono presenti all'interno della cartella rank.

3.3 Wormhole

L'attacco wormhole è uno degli attacchi più comuni e severi che vengono effettuati nelle reti RPL. È basato sulla presenza di due nodi malevoli, M1 e M2, i quali si adoperano per creare un tunnel tra nodi appartenenti alla rete, A e B, e distanti tra loro; la presenza di questo tunnel punta ad ingannare i due nodi A e B facendo credere loro di essere più vicini di quanto non lo siano in realtà. Ciò mira a modificare il normale transito dei pacchetti all'interno della rete; prendendo in considerazione la topologia mostrata nella Figura 38, il wormhole cerca di ingannare il nodo B facendogli credere di essere vicino al nodo radice A; questo porta il nodo B a scegliere come best parent la radice anziché il nodo N.

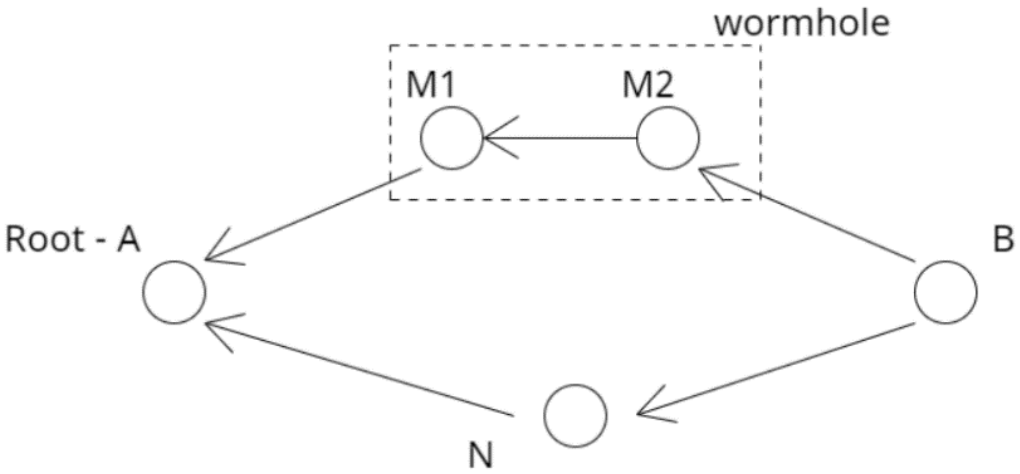
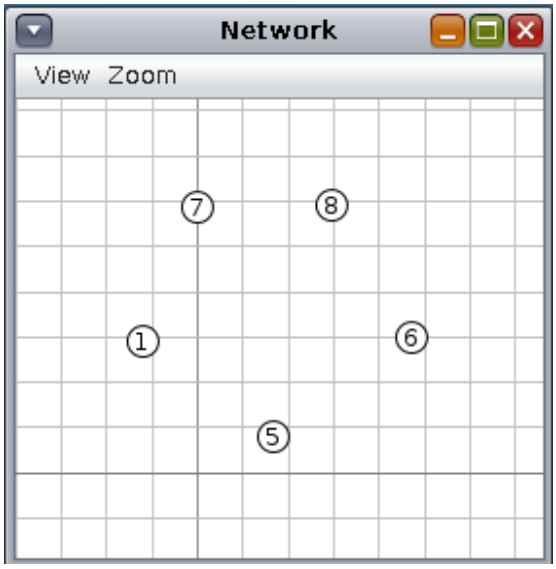


FIGURA 38: Topologia esempio attacco

Per questo attacco sono state eseguite quattro simulazioni. La prima è stata eseguita a puro scopo esplicativo prendendo come riferimento una semplice rete, mostrata nella Figura 39, composta da 5 sensori: un nodo radice contrassegnato dall'ID 1 e quattro nodi client con IDs 5, 6, 7 e 8:



Id Nodo	Id Best Parent
5	1
6	5
7	1
8	7

FIGURA 39: Topologia Wormhole base

In questa simulazione i nodi client provvedono ad inviare, ogni minuto, un pacchetto alla radice facendolo passare per il loro best parent. Di seguito vengono mostrati gli output dei nodi:

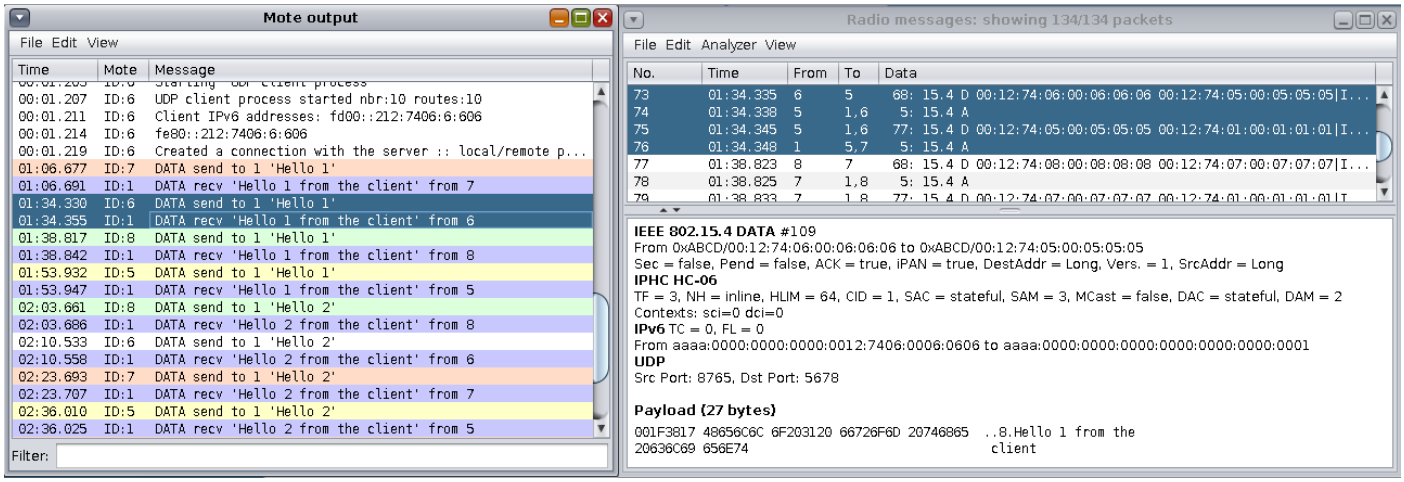


FIGURA 40: Output simulazione attacco su topologia base

In particolare, si vuole porre l'attenzione sul nodo con ID 6; in questo caso, il suo best parent, come già anticipato, risulta essere il nodo con ID 5. Il nodo 6, quindi, nel tentativo di recapitare il pacchetto alla radice lo trasmette inizialmente al nodo 5 il quale risponde con un ack; successivamente, il pacchetto viene recapitato direttamente al server; ciò si evince dalla serie di

pacchetti, mostrati nella Figura 40 (Radio messages), che vanno dal numero 73 al numero 76 e dall'output del nodo server (Figura 40 - Mote output) che conferma di aver ricevuto il pacchetto del nodo 6.

Viene considerata, adesso, una variante della precedente topologia in cui i due nodi contrassegnati dagli IDs 7 e 8 assumono ruolo di nodi malevoli e vengono incaricati di eseguire l'attacco wormhole; i nodi client, come sempre, provvedono ad inviare, ogni minuto, un pacchetto alla radice facendolo passare dal loro best parent. Riponendo l'attenzione sul nodo con ID 6 si evince come il suo best parent sia cambiato; infatti, a causa del wormhole creato dai nodi 7 e 8 il nodo 6 sceglie come best parent proprio il nodo 8 anziché il nodo 5:

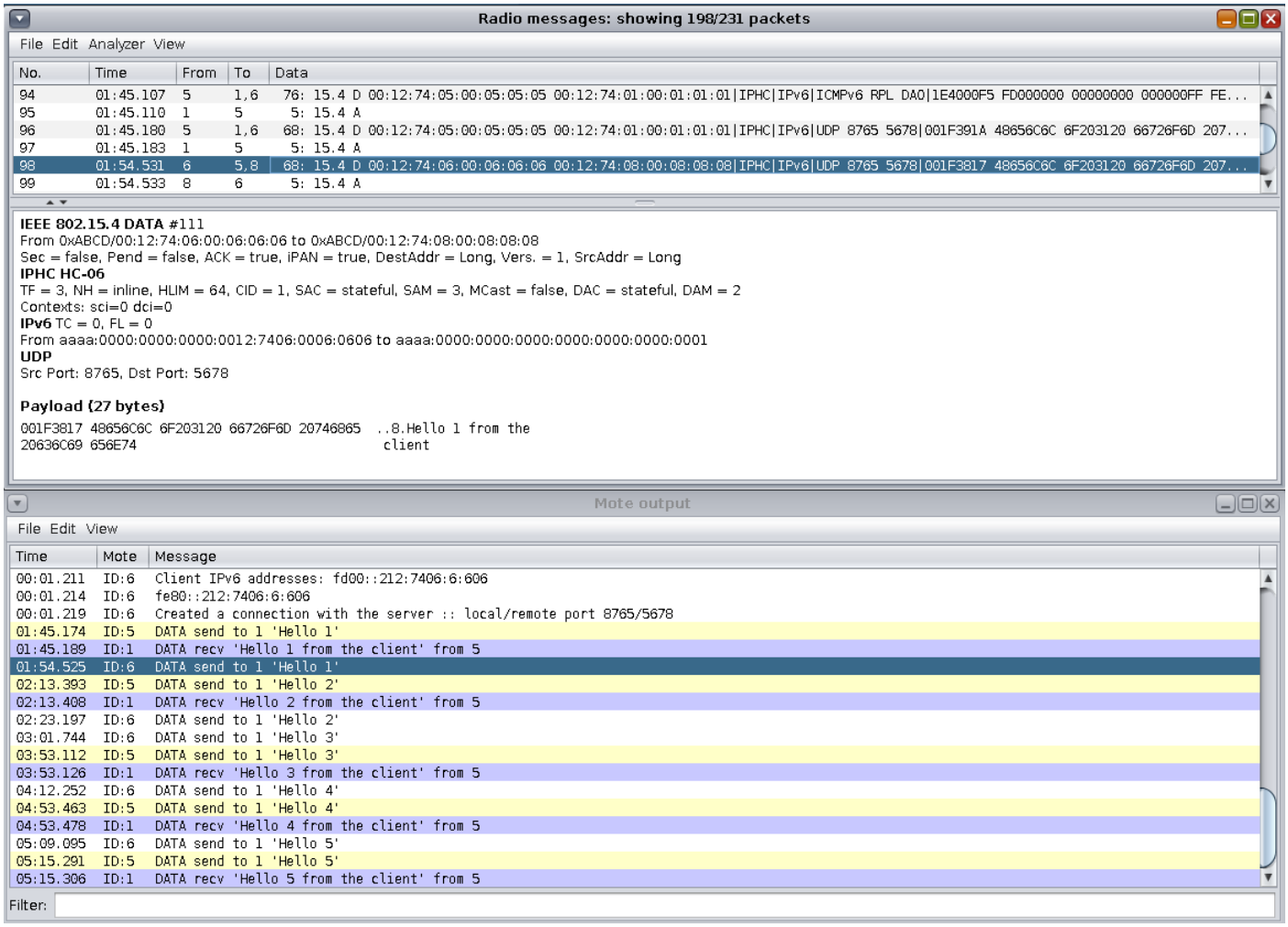


FIGURA 41: Output simulazione attacco nodo 6

Nel dettaglio, il nodo 7 si adopera per trasmettere il rank estrapolato dal messaggio dio della radice al nodo 8 che, a sua volta, lo trasmette al nodo con ID 6. L'effetto del wormhole è quello di far credere al nodo 6 di essere vicino alla radice; a questo punto il nodo 6 sceglie come best parent il nodo malevolo con ID 8 anziché il nodo 5. Il nodo 6, quindi, invia normalmente i suoi pacchetti alla radice facendoli passare, inconsapevolmente, per il wormhole. Il nodo 8 riceve il pacchetto dal nodo 6 e risponde inviandogli un ACK ed illudendolo, così, di aver preso in carico il suo pacchetto; a questo punto il wormhole esegue il drop del pacchetto del nodo 6 il quale è del tutto ignaro dell'accaduto [Per+18]. La radice, quindi, a causa dei drops eseguiti dal wormhole, non riceve nessuno dei pacchetti del nodo 6; ciò si evince dagli output dei nodi mostrati nella Figura 41 (Radio messages).
Di seguito viene mostrato il codice, inserito nel file *rpl-icmp6.c*, eseguito del nodo 8 per ingannare il nodo 6:

```
1  int node_id_attaccante = 8;
2  // il rank della radice viene ricevuto dal nodo 7
3  int rank_dio_radice = ROOT_RANK(instance);
4
5  // modifica del dio che il nodo 8 inviera' ai suoi nodi vicini. Il
       wormhole effettuera' il replay del dio creato dalla radice, in
       questo modo i nodi lontani crederanno di essere vicini alla
       radice
6  if (node_id == node_id_attaccante) {
7      // modifica del rank del dio che verra' inviato
8      dag->rank = rank_dio_radice;
9      set16(buffer, pos, dag->rank);
10
11     // invio del dio ai nodi vicini
12     goto finalize_and_send_dio;
13 }
```

Per le simulazioni vere e proprie dell'attacco sono state prese in considerazione tre topologie composte, rispettivamente, da 10, 50 e 100 nodi; in ogni simulazione, così come nell'esempio, ogni nodo client aveva il compito di inviare alla radice un pacchetto ogni minuto; la durata di ogni simulazione è stata impostata a 10 minuti.
L'impatto dell'attacco è stato misurato, per ogni simulazione, in termini di pacchetti persi e di wormholed nodes, definiti come il numero di nodi che inviano pacchetti che passano per il wormhole.

Di seguito vengono mostrate le topologie, denominate: small (10 nodi), medium (50 nodi) e large (100 nodi):

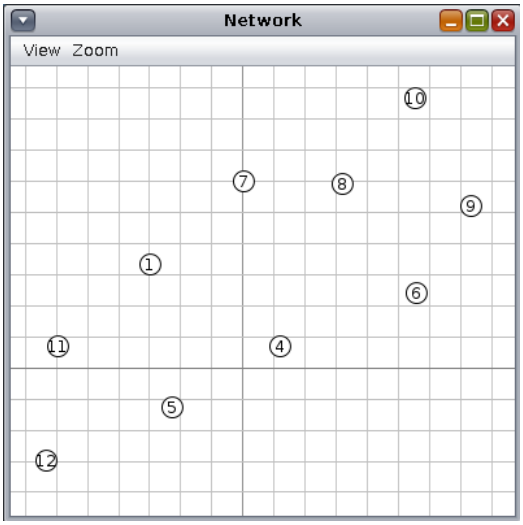


FIGURA 42: Topologia Wormhole small

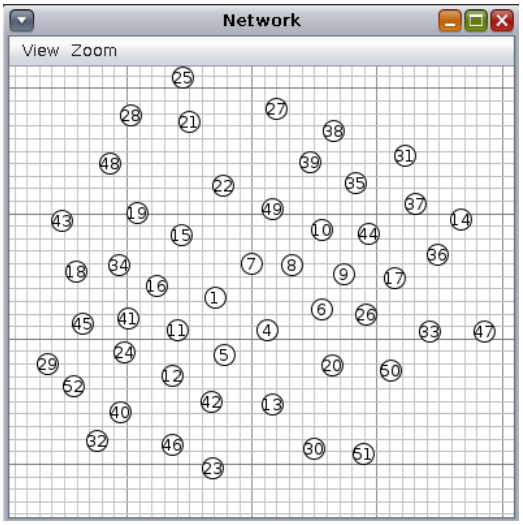


FIGURA 43: Topologia Wormhole medium

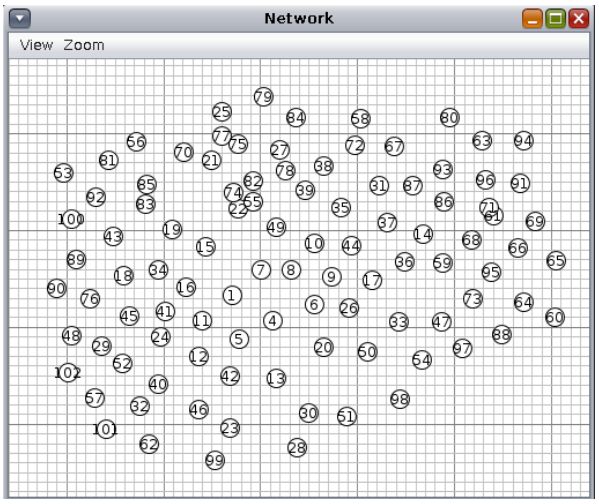


FIGURA 44: Topologia Wormhole large

Topology	Nodes	Wormholed Nodes	% packets lose	% packets delivered
small	10	3	42 %	58 %
medium	50	24	46 %	54 %
large	100	59	58 %	42 %

Da questi dati si evince come all'aumentare dei wormholed nodes aumenti anche la percentuale dei pacchetti persi e diminuisca la percentuale di pacchetti consegnati [Per+18]:

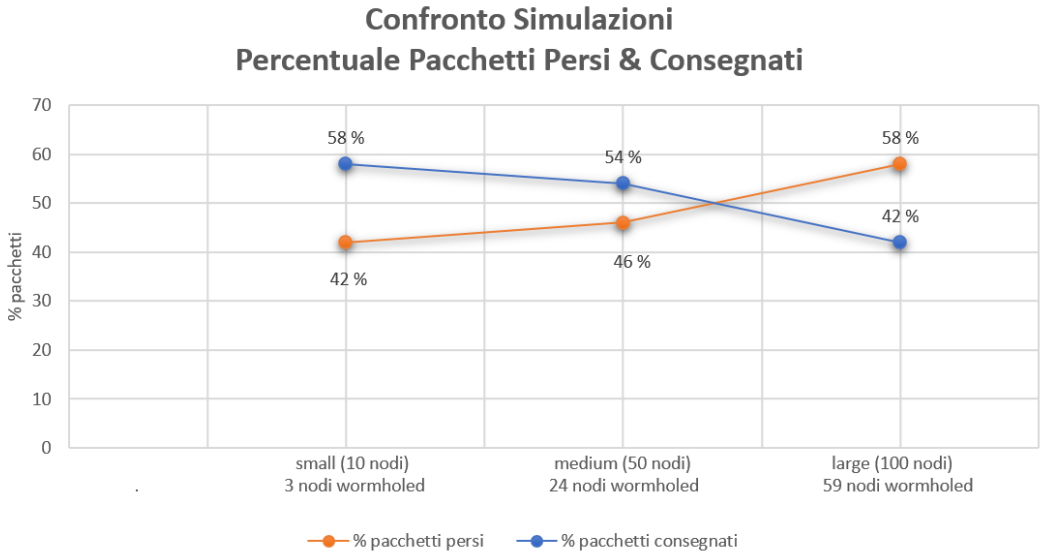


FIGURA 45: Grafico confronto simulazioni

3.3.1 Mitigazione

La mitigazione attuata come contromisura per questo attacco è stata quella basata sulla conoscenza delle posizioni dei nodi ritenuti sicuri; questa soluzione risulta essere economica ed efficiente quando si ha a che fare con nodi caratterizzati da posizioni fisse.
L'implementazione della mitigazione è stata eseguita creando una whitelist di nodi ritenuti sicuri; ogni nodo, quindi, accettava messaggi dio solo dai nodi contenuti in questa whitelist. Dal punto di vista del codice, la mitigazione è stata inserita nel file *rpl-dag.c*.

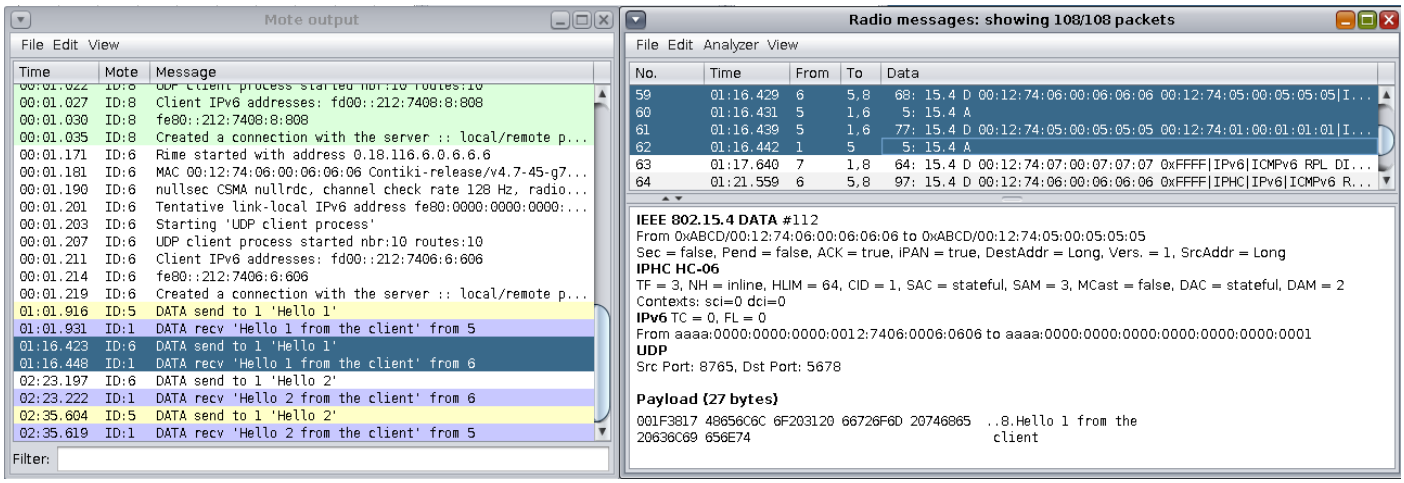


FIGURA 46: Output simulazione mitigazione

Come mostrato dagli output dei nodi (Figura 46), in cui viene considerata la stessa topologia rappresentata in Figura 39, nonostante il tentativo di attacco dei nodi 7 e 8, il nodo 6 sceglie come best parent il nodo con ID 5 e il pacchetto viene correttamente ricevuto dal nodo radice.

3.4 Fragment Duplication

Il layer 6LowPan offre meccanismi di compressione degli header IP e di frammentazione di pacchetti. A differenza dei classici frame IP, l'header è contenuto esclusivamente nel primo frame. Sono presenti comunque in ogni frame informazioni utili al riassemblaggio anche ricevuti in ordine diverso.

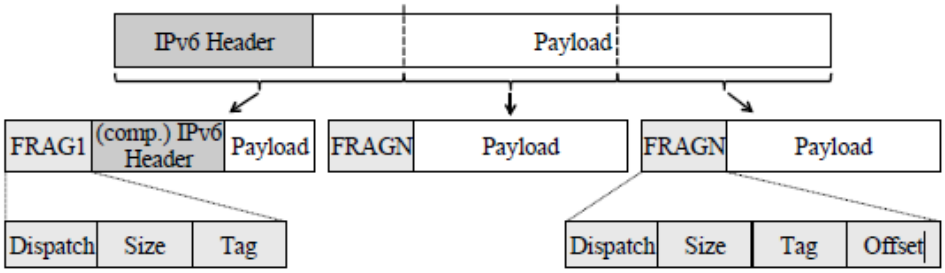


FIGURA 47: Struttura Frame 6LowPan

In Figura 47 viene riportata la struttura dei frame. Notiamo come ognuno di essi si porti dietro un tag, il quale è utile per ricondurre ogni singolo frame al primo (FRAG1) in cui sono contenute le informazioni di routing. Il fragment duplication attack sfrutta il fatto che il ricevente non è in grado di verificare che un frame provenga dalla stessa sorgente dei frame ricevuti in precedenza dello stesso pacchetto IP. Così facendo un attaccante potrebbe duplicare dei frame sulla rete andando a modificare esclusivamente il payload e di conseguenza la vittima non sarebbe in grado di verificare quale dei due frame è quello leggittimo. Lo standard 6LowPan suggerisce in questi casi di droppare il pacchetto IP corrotto, di conseguenza un malintenzionato potrebbe bloccare la comunicazione costringendo la vittima a scartare ogni pacchetto.

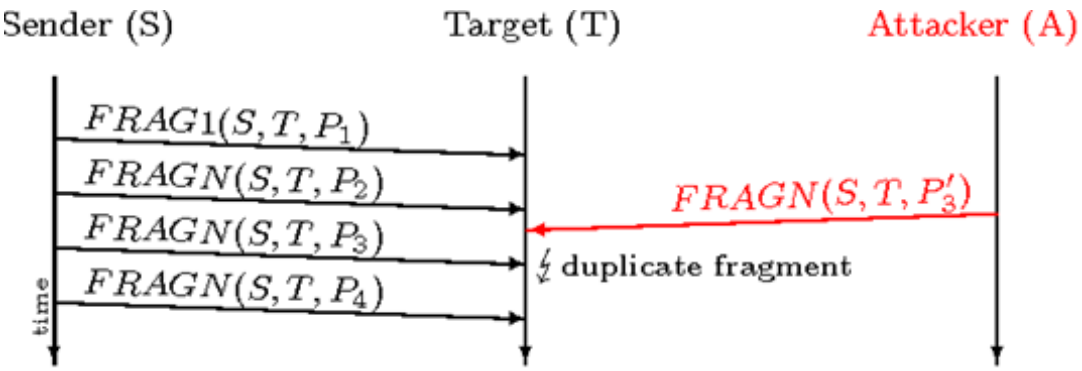


FIGURA 48: Esempio attacco Fragment Duplication

In Figura 48 si delinea la struttura di un esempio di attacco in cui l'attaccante duplica uno dei pacchetti sulla rete poiché ne basta semplicemente uno per ottenere il risultato voluto.

Per questa tipologia di attacco ci si è basati su una semplice simulazione in cui esiste un nodo che invia un costante flusso di pacchetti UDP frammentati. Per simulare il comportamento di un attaccante il nodo in questione oltre ai normali frame ne invia uno duplicato al nodo ricevente.

Ogni pacchetto ha la dimensione di 512 bytes che comporta la suddivisione in 6 frammenti leggittimi ed uno duplicato.

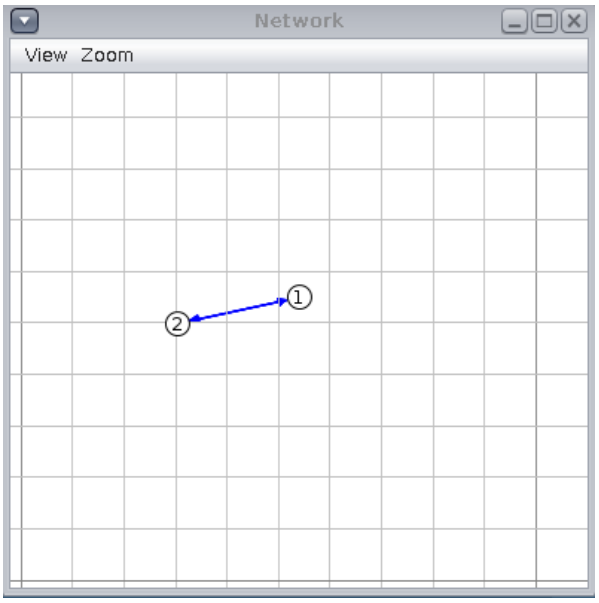


FIGURA 49: Topologia Simulazione Fragment Duplication

In particolare, come mostrato in Figura 49, il nodo con id 2 svolge il ruolo di attaccante ed il nodo 1 invece registra i pacchetti ricevuti correttamente. Non è interessante vedere diverse tipologie poiché l'attacco prende di mira un nodo in particolare e la situazione all'interno della rete non varia in base al numero di nodi coinvolti.

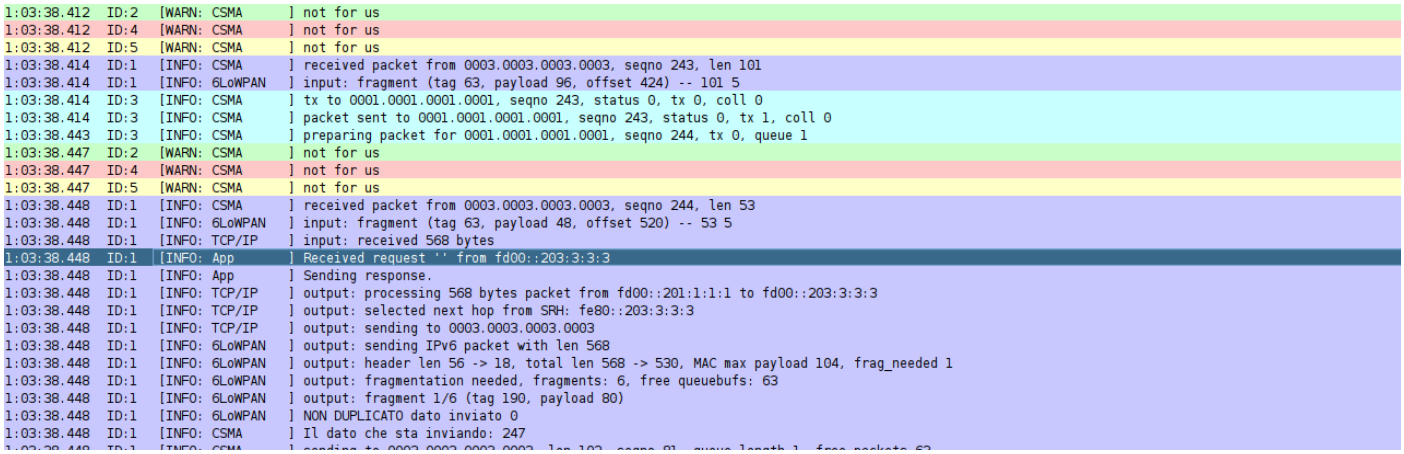


FIGURA 50: Output simulazione in assenza di attacco

In condizioni normali, come mostrato in Figura 50 i pacchetti vengono ricevuti correttamente dal nodo 1 che li processa e risponde al mittente.

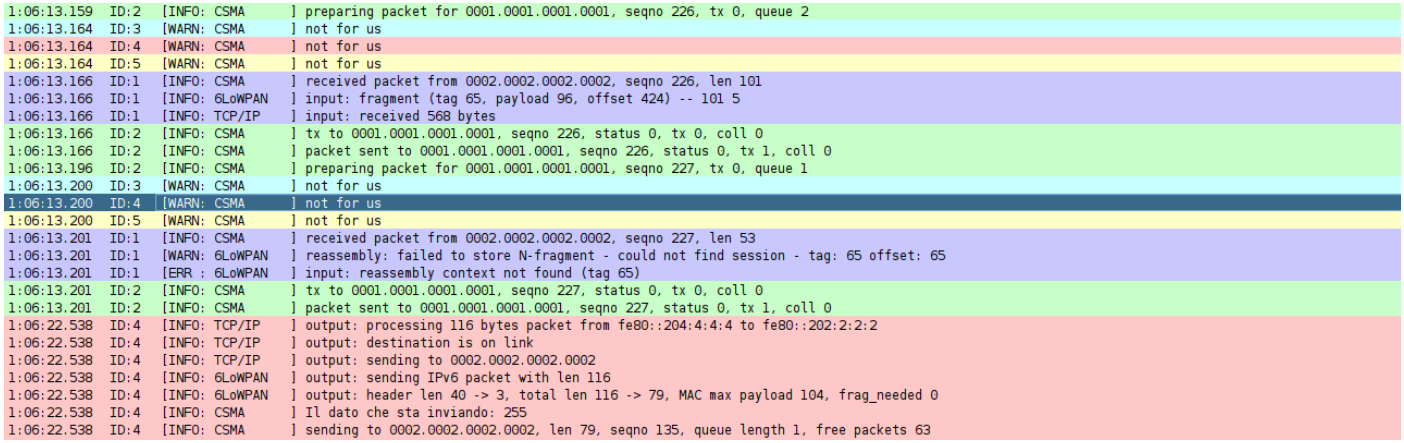


FIGURA 51: Output simulazione con fragment duplication

Durante la fase di attacco il nodo con id=1 non è in grado di selezionare i frammenti ricevuti in modo tale da scartare quello duplicato, il quale ha lo stesso tag e mittente degli altri ma differisce per il contenuto del payload. Il risultato della simulazione viene mostrato in Figura 51, in cui si nota che il pacchetto viene droppato.

3.4.1 Mitigazione

Una possibile mitigazione all'attacco di duplicazione dei frammenti è l'utilizzo di una coppia di chiavi pubblica e privata, ma a causa della complessità nella gestione di scambio delle chiavi non è conveniente implementarla.
A tal proposito è stata attuata una strategia differente basata sul *content-chaining scheme* che si basa sull'aggiunta di un token di autenticazione aggiunto dal mittente in coda ad ogni fragment durante il processo di frammentazione effettuato in 6LoWPAN.

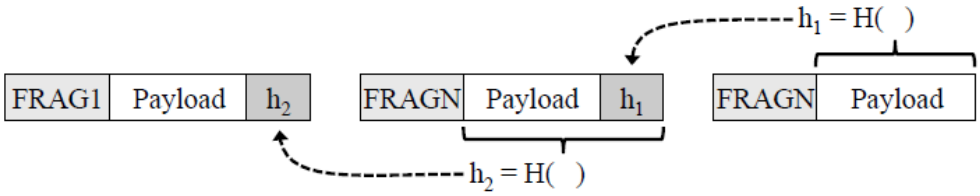


FIGURA 52: Esempio di content chain

Attraverso questo processo il destinatario può verificare il collegamento tra ogni fragment ricevuto, come mostrato nell'esempio di Figura 52 in cui sono presenti tre fragment e droppare gli eventuali fragment duplicati.
Le *content chain* si basano sul concetto delle catene di hash, un meccanismo leggero ed efficente per l'autenticazione del mittente di un flusso di dati di lunghezza finita. Gli elementi h_i di una catena di hash sono generati applicando iterativamente una funzione di hash $H(\cdot)$ all'output dell'iterazione precedente: $h_i = H(h_{i-1})$. Per la prima iterazione viene utilizzato un seed random come input, nel caso specifico della mitigazione attuata viene utilizzato come seed iniziale il payload dell'ultimo FRAGN, l'operazione viene ripetuta iterativamente per gli altri fragment con l'unica differenza che nel calcolo dell'hash viene utilizzato il payload più l'hash presente in coda finché non si arriva al primo fragment. Una volta che la costruzione della catena è conclusa il mittente può inoltrare ogni fragment con il relativo token.

La funzione hash utilizzata per la realizzazione della catena è md5 troncando ogni token a 8 bytes per ridurre gli overhead di comunicazione. Di seguito viene riportato il codice di generazione della catena:

```
1  memcpy(fragArray[fragment_count-1],(uint8_t *)UIP_IP_BUF +
       offsetFrag[fragment_count-1],uip_len-offsetFrag[fragment_count
       -1]);
2   for(int i = fragment_count-2; i >=0; i--){
3     //hashArray[i] = malloc(8);
4     uint8_t *result;
5     if(i == fragment_count-2){
6       result = md5String(fragArray[i+1],uip_len-offsetFrag[i+1]);
7       memcpy(fragArray[i],(uint8_t *)UIP_IP_BUF + offsetFrag[i],
       fragn_max_payload);
8       memcpy(fragArray[i]+fragn_max_payload,result,8);
9     }
10    else if(i==0){
11      result = md5String(fragArray[i+1],fragn_max_payload+8);
12      memcpy(fragArray[i],(uint8_t *)UIP_IP_BUF + offsetFrag[i],
       frag1_payload);
13      memcpy(fragArray[i]+frag1_payload,result,8);
14    }
15    else{
16      result = md5String(fragArray[i+1],fragn_max_payload+8);
17      memcpy(fragArray[i],(uint8_t *)UIP_IP_BUF + offsetFrag[i],
       fragn_max_payload);
18      memcpy(fragArray[i]+fragn_max_payload,result,8);
19    }
20   }
```

In fase di ricezione il destinatario quando riceve il primo fragment di un pacchetto lo processa normalmente e salva il token contenuto all'interno per la verifica dei fragment successivi. Per ogni FRAGN ricevuto viene effettuato il processo di validazione andando a calcolare l'hash sul contenuto del fragment ricevuto. Nel caso in cui risulti uguale al token salvato, la verifica ha avuto esito positivo e il fragment viene processato, altrimenti viene scartato immediatamente. La verifica lato destinatario viene effettuata attraverso la funzione di seguito mostrata:

```
1  static int duplicationAttack(int offset){
2    if(offset == 0){
3      //LOG_INFO("Primo pacchetto: hash iniziale impostato %d\n",
       packetbuf_datalen());
4      memcpy(lastHash,(packetbuf_ptr+packetbuf_datalen()-8),8);
5    }
6    else{
7      uint8_t *curHash = md5String((packetbuf_ptr+packetbuf_hdr_len),
       packetbuf_payload_len);
8      LOG_INFO("Hash calcolato: ");
9      for(int i = 0; i<8;i++){
10     LOG_INFO_("%02x vs %02x ",curHash[i], lastHash[i]);
11       if(curHash[i]!=lastHash[i]){
12         LOG_INFO_("\n");
13         return 1;
14       }
15     }
16     LOG_INFO_("\n");
17     memcpy(lastHash,(packetbuf_ptr+packetbuf_datalen()-8),8);
18     return 0;
19   }
20   return 0;
21 }
```

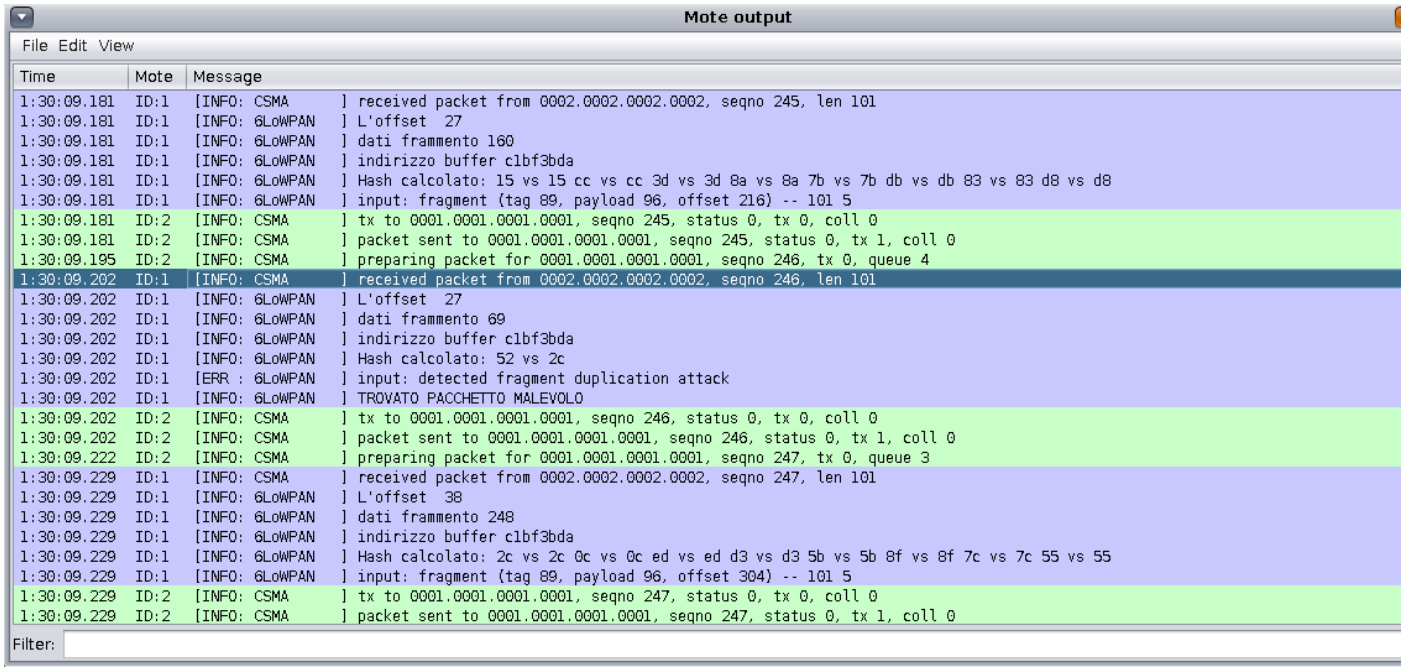


FIGURA 53: Output simulazione con mitigazione

L'utilizzo di questa mitigazione comporta l'individuazione del fragment duplicato e il successivo drop di quest'ultimo come mostrato in Figura 53.

Per la valutazione dell'efficacia della mitigazione sono state lanciate due diverse configurazioni di simulazioni, una in cui viene utilizzata una versione di Contiki senza modifiche ed una in cui è presente la mitigazione presentata. Nel caso della prima versione di Contiki il ricevente ha riscontrato una perdita completa dei pacchetti dovuta alla sovrascrittura del contenuto dal nodo attaccante tramite il *fragment duplication attack*. Al contrario, nella versione di Contiki con la mitigazione attiva si è registrato un PDR (*Packet Delivery Rate*) del 100% come mostrato nella tabella sottostante.

Mitigazione Attiva	**% packets lose**	**% packets delivered**
FALSE	100 %	0 %
TRUE	0 %	100 %

3.5 Come si avvia una simulazione

Affinchè il tutto funzioni correttamente è stato implementato uno script bash. Il suo funzionamento è molto semplice:

```
1 Usage: ./$0 <opts>
2     -A attack (DOS, RANK, WORMHOLE, FRAGMENT)
3     -M mitigation (DOS, RANK, WORMHOLE, FRAGMENT)
4     -S number of scenario:
5         * DOS [1,3]
6         * RANK [1,3]
```

Una volta avviato, l'ambiente verrà configurato in base a cosa viene scelto (attacco, mitigazione, scenario). Dopo di che, per poter avviare le simulazioni, è necessario avviare *contiker* e poi *cooja* dal terminale con il comando:

$ contiker cooja

Una volta avviata la GUI è necessario caricare la simulazione da

"File" > "Open simulation" > "Browse"

Infine bisogna dirigersi nella cartella

"contiki-ng" > <nome_attacco>

e scegliere la simulazione da avviare entrando nella corrispondente cartella e aprendo il file "*.csc".

Conclusione

Con il seguente elaborato sono stati implementati i seguenti attacchi: DoS Attack, Rank Attack, Fragment Duplication Attack e Warmhole Attack; in un ambiente IoT simulato attraverso l'uso del sistema operativo Contiki-NG e del simulatore Cooja. È stata fornita una breve trattazione dei principali protocolli trattati, evidenziando la loro importanza ed il ruolo assunto nei diversi scenari applicativi in riferimento sia ad un ambiente sicuro che ad un ambiente esposto a rischi. È stata descritta inoltre la procedura di installazione e configurazione dell'ambiente di sviluppo. Per concludere l'elaborato è bene fare presente ad eventuali assunzioni effettuate riguardo l'implementazione delle soluzioni proposte e l'ambiente di sviluppo:

- Per quanto riguarda l'attacco DoS, la mitigazione assume che la rete, intesa in numero di nodi, rimanga costante. Quest'assunzione è dovuta dall'impossibilità di passare argomenti al simulatore a tempo di esecuzione. Quest'assunzione non mira però in alcun modo l'efficacia o la generalità della soluzione proposta, in un ambiente reale, infatti, è sufficiente sostituire le allocazioni di memoria statiche, con delle allocazioni dinamiche (esempio usando la funzione malloc) in modo da poter allocare nuovo spazio, per memorizzare le informazioni suoi nuovi nodi, qualora fosse necessario.

- Sempre in riferimento al DDoS con mitigazione, è bene precisare che sono state effettuate simulazioni anche su reti più complesse, formate da 30+ nodi con diversi attaccanti e diverse vittime. Sebbene il tutto continui a funzionare correttamente, è stato scelto di non presentare

tale scenario. L'elevato numero di messaggi scambiati, infatti, rende poco chiaro il contesto e richiede un'attenta analisi del log completo, che per questioni di leggibilità, si è deciso di non allegare vista la sua eccessiva lunghezza.

- Il rank attack può essere utilizzato come vettore di partenza per la costruzione di altri attacchi, come ad esempio il Sinkhole o il DoS. Analizzando quest'ultimo esempio, infatti, un nodo malevolo combinando Rank Attack e IP Spoofing, potrebbe inviare pacchetti DIO sulla rete, contenenti un valore di rank ottimale e l'indirizzo IP del nodo vittima. In tal modo gli altri nodi della rete sceglieranno il nodo vittima come nodo di inoltro dei pacchetti e questo potrebbe causare un DoS su tale nodo, se il numero di pacchetti è eccessivamente elevato.

- Il wormhole attack risulta essere molto potente e può essere eseguito anche in reti che fanno uso di tecniche crittografiche in quanto è un attacco che non genera nuovi pacchetti ma si limita ad effettuarne il replay; quindi, l'attacco, permette di effettuare un denial of service senza la necessità di rubare chiavi private. Per quanto concerne la mitigazione, esistono molte contromisure per mitigare l'attacco ma nessuna di queste sembra essere definitiva. La contromisura scelta per questo progetto, basata sulla conoscenza delle posizioni dei nodi ritenuti sicuri, risulta essere praticabile e non dispendiosa per dispositivi "fissi" ma risulta inefficace nel caso in cui i nodi della rete siano dispositivi mobili.

- Il Fragment Duplication Attack risulta molto efficace nelle reti permettendo ad un nodo di poter selettivamente bloccare i pacchetti relativi a particolari protocolli non consentendo di conseguenza la comunicazione. Le mitigazioni possibili hanno come obiettivo comune quello di fornire un meccanismo di identificazione del mittente sulla base delle informazioni contenute nei frame.

Bibliografia

[Hum+13] René Hummen et al. «6LoWPAN fragmentation attacks and mitigation mechanisms». In: *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*. 2013.

[Iuc+15] Kenji Iuchi et al. «Secure parent node selection scheme in route construction to exclude attacking nodes from RPL network». In: *2015 21st Asia-Pacific Conference on Communications (APCC)*. IEEE. 2015.

[Per+18] Pericle Perazzo et al. «Implementation of a wormhole attack against a rpl network: Challenges and effects». In: *2018 14th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*. IEEE. 2018.

[RA17] Krishna Kumar Rai e Krishna Asawa. «Impact analysis of rank attack with spoofed IP on routing in 6LoWPAN network». In: *2017 Tenth International Conference on Contemporary Computing (IC3)*. IEEE. 2017.