Distributed Intrusion Detection in Wireless Mesh Networks

Anderson Morais and Ana Cavalli
anderson.morais@telecom-sudparis.eu

MTV2 Meeting
CEA LIST, Saclay, Nov 12, 2012
Topics

- Introduction and Motivation
- Distributed Intrusion Detection
- Mesh Network Testing Platform
- Performance Evaluation
Wireless Mesh Network (WMN) is comprised of radio devices **self-organized** in **multi-hop** architecture.

- The mesh nodes forward packets from others nodes to cooperatively reach a destination (multi-hop strategy).
- WMN do not rely on a dedicated wireless infrastructure (ex: APs), but the nodes trust on each other to transmit the traffic.
- WMN is **self-healed**, if a node crashes, then its neighbors establish new routes.
- Ex.: Wireless Community Networks and Municipal Wireless Networks.
Wireless Mesh Network Architecture

Internet

Internet router

Wireless mesh backbone (mesh cloud)

Gateway routers

Mesh routers

Mesh clients
Motivation

- The mesh nodes are vulnerable to:
  - **External Attacks** (ex: *Passive Eavesdropping, Jamming*) due to open medium access.
  - External Attacks can be avoided by encryption, authentication, control access, secure MAC, anti-jammers.
  - **Insider Attacks** (ex: *Selective/Random Packet Dropping, Selfishness, Flooding, other Routing Misbehaviors*).
  - A malicious node (with a valid key) can **intercept**, **modify**, **fabricate** and **drop** routing/data packets.
    - Disrupt the integrity of routes and network data services.
    - Consume valuable network resources (ex: bandwidth).
    - Exhaust the hardware resources (CPU, memory) of nodes.
Intrusion Detection

- **WMNs** need a **second line of defense** to deal with misbehaving nodes.

- **IDSs** are classified according to the **monitored events**:
  - *Host-based IDS* (HIDS): monitor the system logs and internals
  - *Network-based* (NIDS): monitor the network traffic

- **IDSs** are divided depending on the **detection engine**:
  - *Signature-based IDS*: use attack patterns (signatures/rules).
  - *Anomaly-based IDS*: based on normal behavior profile (statistical).

- **Our approach** is based on **specification-based IDS**: *Routing Constraints* optimized for common attacking techniques.
The Intrusion Detection Approach

- An IDS is installed at each node to passively capture and analyze the local traffic: generate *Routing Events*.
- The neighboring nodes exchange *Routing Events* for distributed and collaborative attack detection.
- Each IDS treats the *Routing Events* using *Routing Constraints* to calculate *Misbehaving Metrics*.
- If suspected routing behavior is found, the node triggers the **consensus mechanism** to track the **malicious node**:
  - The nodes share *Detection Results* and make a collective decision on the node culpability.
Intrusion Detection Architecture

- **Distributed Intrusion Detection Solution**
  - Response Mechanism
    - Passive Response
    - Active Response
  - Action
    - Cooperative Consensus Mechanism (CCM)
  - Exchange of Detection Results and Metrics

- **Distributed Intrusion Detection Engine (DIDE)**
  - Misbehaving Metrics
    - Packet Dropping Detection
      - Routing Event Handlers
    - Message Fabrication Detection
      - Routing Event Handlers
    - Flooding Detection
      - Routing Event Handlers
  - Exchange of Routing Events

- **Routing Protocol Analyzer (RPA)**
  - Routing Events
  - Exchange of Routing Events

- **Detection tool**
  - Routing Event Engine
  - Packets
    - Traffic Monitoring
  - Node Network Interface
Contributions

- A practical and efficient cooperative intrusion detection architecture for WMNs.
- The **Routing Protocol Analyzer** analyzes routing packets and produces *Routing Events* (protocol behavior).
- The **Distributed Intrusion Detection Engine** treats local and remote *Routing Events* and outputs *Misbehaving Metrics* based on *Routing Constraints* (protocol behavior).
- The **Cooperative Consensus Mechanism** exchanges *Intrusion Detection Results* to cooperatively detect attacks and the source of intrusion (malicious node).
- The approach is implemented into *Bro*\(^1\) IDS tool, and validated through extensive experiments in a virtual mesh network platform.

\(^1\) [http://bro-ids.org](http://bro-ids.org)
Packet Dropping Detection Module

- We use the IDS tool to monitor the traffic on the node’s network interface (libpcap).
  - We added a filter to select only BATMAN\(^1\) routing messages.

- A Routing Event is a message with parameters as the result of the analysis of a BATMAN routing packet.

Then, the Detection Engine module:
1. Treats local and remote Routing Events
2. Generates Packet Dropping Metrics that indicates an dropping attack is happening.
3. Exchanges the Packet Dropping Detection Results with the neighbors for making joint decisions.

\(^1\) Better Approach To Mobile Ad hoc Network (BATMAN) - http://www.open-mesh.org
BATMAN Routing Algorithm 1

1. Each node (Originator - $Orig$) periodically broadcasts Originator Messages (OGMs) to announce its existence.
   - An OGM has: an $Orig$ Address, a $Src$ Address, a TTL, and a Sequence Number ($Seqn$).

2. As the neighboring node receives an OGM:
   - Modify the $Src$ address to its own address
   - Re-broadcast the OGM according to BATMAN forwarding rules to tell its neighbors about the existence of the node, and so on and so forth.
3. The mesh network is flooded with OGMs until:
   - Every node has received it at least once.
   - They got lost because of link packet loss.
   - The TTL value has expired.

The best next-hop to an \textit{Orig} is the neighbor from which the node received the highest amount of OGMs (\textit{Seqn}) within a sliding window (route quality metric).
- The routing table of the node contains the best next-hop towards each \textit{Orig}.
Routing Protocol Analyzer

The **Routing Protocol Analyzer** analyzes OGMs and generates *Routing Events* according to the protocol behavior:

- $OGM\_reb\_rd(<params>)$: the node re-broadcasted a OGM received from a neighbor $Nb$.
- $OGM\_rcv(<params>)$: a OGM was received from a neighboring node $Nb$.
- $OGM\_broad(<params>)$: the node broadcasts an OGM to its neighbors $Nb$.

These *Routing Events* are treated by the node and sent to the neighboring nodes.
Distributed Intrusion Detection Engine

The Detection Engine module defines Routing Event Handlers that process local and remote Routing Events (exchanged with neighbors $Nb$):

- $EH_{OGM\_rebrd}()\{\$
  Rebrd_{Orig}^{Nb} =+ Seqn;
  Nb_{Rebrd}^{Src}_{Orig} =+ Seqn; \}$

- $EH_{OGM\_rcv}()\{\$
  Rcv_{Orig}^{Nb} =+ Seqn;
  Nb_{Rcv}^{Src}_{Orig} =+ Seqn; \}$
Routing Constraints

We define Routing Constraints based on the protocol behavior to detect malicious Packet Dropping PD:

- **EH_OGM_rcv( )**:
  - **C1** = “Check if the Seqn received by the node from Nb for Orig will be rebroadcasted by own node”:
    - **if Seqn ∉ Rebrd_{Orig}^{Nb} then PD_{Rebra}++**
  - **C2** = “Check if the Seqn received by its Nb from Src for Orig will be rebroadcasted by Nb”:
    - **if Seqn ∉ Nb_Rebrd_{Orig}^{Src} then PD_{Nb_Rebra}++**
Packet Dropping Metrics Calculation

Then we calculate the Rate of Packet Drop routing misbehavior $R$ for the Routing Constraints:

- For $C1$ at $EH\_OGM\_rcv( )$:
  \[ -R_{Rebrd} = \frac{PD_{Rebrd}}{|Rcv|} \]

- For $C2$ at $EH\_OGM\_rcv( )$:
  \[ -R_{Nb\_Rebrd} = \frac{PD_{Nb\_Rebrd}}{|Nb\_Rcv|} \]
Consensus Mechanism

- *Packet Dropping Metrics* are constantly monitored by the Consensus Mechanism module.

- If suspected *Packet Dropping Attack* is found, the neighbors exchange respective *Dropping Metrics* and share *Detection Results* before a final decision is taken.
  - To avoid inaccurate detection results because of poor link quality due to interference and congested links.

- If a *consensus* is reached with majority agreement, then the *malicious node* is detected.
  - The *decision rule* is “unanimous agreement minus one vote or two votes”.
Cooperative Consensus Mechanism

- A threshold $Th$ is defined to separate the detection rate from the false positive rate (caused by the loss of messages):
  - if ($R_{Rebrd} \geq Th_{Rebrd}$)
    1. Node 1 ($O_1$) detected the malicious dropping $PD_{Rebrd}$;
    2. $O_1$ triggers the consensus;
    3. $Sync (R_{Rebrd}, R_{Nb_{Rebrd}})$ with the Neighbors $N_i$; $1 \leq i \leq k$
    4. for all $N_i$ { if $R_{Nb_{Rebrd}}^{N_i} \geq Th_{Nb_{Rebrd}}^{N_i}$};
      1. $Sync (MN_{O_1}, MN_{N_i})$;
      2. if ($MN_{O_1} = MN_{N_i}$) then consensus is done.

$^1 MN$: Malicious Node
Virtualized Mesh Network Platform

- We use a **virtual network environment** to emulate the mesh network topology:
  - Each node is a VM\(^1\) running BATMAN\(^2\) proactive routing protocol as Linux Kernel module.
  - The VMs are interconnected by a **virtual switch**:
    - To emulate *Bi-directional Links* between nodes;
    - Realistic Link Limitations: packet loss rate, lost burstiness.
  - We collect log files and pcap files at each node that are synchronized by the *global clock* of main machine.

- This platform represents the performance requirements of the IDS in terms of hardware resources and accuracy.

---

\(^1\) http://wiki.qemu.org  
\(^2\) http://www.open-mesh.org
Performance Evaluation: Scenario

Before the attack:
- Node $O_3$’s Routing Table:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Best Next-hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O_2$</td>
<td>$O_1$</td>
</tr>
</tbody>
</table>

Before the attack:
- Node $O_3$’s Routing Table:

- $O_3$’s Routing Table:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Best Next-hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O_2$</td>
<td>$O_1$</td>
</tr>
</tbody>
</table>

Network Events + Detection Results

OGM
Packet Dropping Attack

- The **malicious node** $O_1$ drops the OGMs rebroadcasted by the own node $O_1$ for node $O_2$.
  - Node $O_3$ will then choose node $O_4$ as best next-hop to gateway node $O_2$.
  - Then if node $O_4$ is compromised, it can execute black hole or gray hole attacks by dropping the data packets.
- The **Packet Dropping Attack** violates the integrity of the Routing Table of the target node.
- The attack is implemented in the **virtual link** between nodes $O_1$ and $O_3$ by using a filtering scheme.
Dropping Detection Evaluation

• After the attack:
  - Node $O_3$’s Routing Table:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Best Next-hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O_2$</td>
<td>$O_4$</td>
</tr>
</tbody>
</table>
Packet Dropping Detection at node $O_1$

- Rate of Packet Drop $R_{Rebrd}$ detected at node $O_1$.
  - The malicious node $O_1$ drops 5% OGMs at 180 secs, 10% OGMs at 360 secs, 20% at 540 secs, until 100%.

![Graph showing rate of packet drop over time](image)
Packet Dropping Detection at nodes $O_2$ and $O_3$

- Rate of Packet Drop $R_{Nb\_Rebrd}^{O_2}$ detected at node $O_2$ and node $O_3$.

![Graph showing packet dropping detection at nodes $O_2$ and $O_3$.]
Future work

- Implement Active Response Mechanisms to mitigate and/or block the attacks.
- Reduce the Communication Overhead without impacting the attack detection accuracy.
- Implement mobility models in the platform to evaluate the approach.
Thank you!