Neighborhood Monitoring in Ad Hoc Networks

Axel Krings
Computer Science Dept.
University of Idaho
krings@uidaho.edu

Based on:
Axel Krings, and Stephan Muehlbacher-Karrer,
"Neighborhood Monitoring in Ad Hoc Networks",
Proc. Annual Cyber Security and Information Intelligence Research Workshop (CSIIRW'10),
Oak Ridge National Laboratory,
ACM International Conference Proceeding Series,
April 21-23, 2010
Wireless Networks

- Wireless Networks have gained great popularity
- Special focus: ad hoc nets, MANETs, sensor nets
- Wireless has many potential problems w.r.t.
  - **Security**: broadcast, “everybody can see”, nodes may be captured/impersonated/... many flavors
  - **Reliability**: nodes may be mobile, links and nodes have reliability/availability constraints, external interference, faults range from benign to malicious
  - **Mobility**: dynamic topology
Fault Models

“Problems” and “Faults”, “Errors” and “Failures”

in the end it boils down to Fault Models

What are the assumptions about faults?

- crash faults, omission faults, etc.
- independence of faults
- dependence of faults => common mode fault
- recovery differs greatly depending on the fault model

Fault Model Overview

[Diagram showing the classification of faults into Malicious and Benign categories, further divided into Asymmetric, Symmetric, Transmissive, Omissive, and Strictly Omissive subcategories.]
Recovery needs Redundancy

- Time redundancy
- Information redundancy
- Spatial redundancy

E.g. if one considers $s$ symmetric and $b$ benign faults, then one needs a redundancy level of $N > 2s + b$ to mask the faults.

Network Graph

- General Communication Model

Network Graph $G$ is a digraph.
Network Graph

- General network graph is a flow-graph (packet flow)
- In wireless networks this is different
  - broadcast is NOT point-to-point
  - broadcast implies flow on all outgoing edges of a node
  - if network consists of wireless and wired, then colored graph can be used

Graph Join Operation

- Join graph of two graphs

  Given $G_i = (V_i, E_i)$ and $G_j = (V_j, E_j)$
  
  $G = (V, E) = G_i + G_j$ where
  
  $V = V_i \cup V_j$
  $E = E_i \cup E_j$

  and $\forall v_i \in V_i, v_j \in V_j \quad e_{ij} \in E$

  ![Graph Join Operation Diagram]
General Join Graph (GJG)

- A path between $v_S$ and $v_D$ defines the primary communication path.
- Let $C_1$ be a clique of all vertices $v_i$ that is incident from $v_S$, i.e., for each $v_i \in C_1$ there exists $e_{S_i}$.
- For each $v_j$ in the primary communication path define $C_j$ as a clique of all vertices $v_i$, for which there exists an edge $e_{h_i}$ from all $v_h \in C_{j-1}$.
- Let $C_D$ be a trivial clique containing only $v_D$.

Attacks in Ad Hoc Nets

Behind any attack there is an action on the packet which involves

- Delaying packets
- Dropping packets
- Modification of packets
- Fabrication of packets
- and then of course “sniffing” packets
Related Work

- One-dimensional monitoring, e.g.,
  - Marti et. al. [2000] Watchdog + Pathrater,
  - Patcha [2003] Watchdog groups, dealing with collusions
  - Buchegger [2004] Limitations on Watchdogs

- Multi-dimensional monitoring, e.g.,
  - Krings and Ma [2006] MILCOM’06 (Join-Graphs)
  - Huang [2008] JICS (Extended Watchdog)
  - Khalil [2009] Ad Hoc Networks (Neighbor Monitoring)
**Watchdog a la Marti**

- Simple watchdog

**Extended Watchdog**

- packet ->A-B-C
- is packet for C?
- is B destination?
- what if A is malicious and does not send data to make B look bad?
- what if B drops packet due to congestion?

*from Huang & Liu JICS [2008]*

**Fig. 1: System Model**
Two-Dimensional Monitoring

Krings & Ma 2006

- Horizontal and orthogonal cross-monitoring
- mainly a topology-based argument

Horizontal and orthogonal cross-monitoring as depicted in the diagram.

[Khalil et.al. 2009] UnMask

- Attack-free environment during neighborhood discovery
- Static wireless network
- Discovery is once in a lifetime only

The diagram illustrates a unmasking attack in a static wireless network environment.
General Neighborhood Monitoring

Attack Model

- Attack may originate within a node that is part of authenticated neighborhood or not
- Attacks
  - from outside of authenticated neighborhood
  - from good node gone bad
  - from malicious node that joint neighborhood
Evolution of Monitoring

a) watchdog
b) extended watchdog
c) UnMask
d) new approach

Neighborhood Discovery

- 1-Hop Discovery
  - arriving node $v_i$ broadcasts HELLO message
  - each $v_k$ receiving the message replies to $v_i$
  - $v_i$ collects the neighbors in neighborhood list $N_i$

- 2-Hop Discovery
  - each $v_k$ receiving the message replies with its neighborhood list $N_k$
  - can be used if adopting authentication under the assumptions of [Khalil et.al. 2009]
Neighborhood Discovery

1-Hop Discovery

![Graph showing nodes and edges]

<table>
<thead>
<tr>
<th>$N_u$</th>
<th></th>
<th>$N_v$</th>
<th></th>
<th>$N_m$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>1</td>
<td>s</td>
<td>1</td>
<td>r</td>
<td>1</td>
</tr>
<tr>
<td>s</td>
<td>1</td>
<td>t</td>
<td>1</td>
<td>s</td>
<td>1</td>
</tr>
</tbody>
</table>

2-Hop Discovery

![Graph showing nodes and edges]

<table>
<thead>
<tr>
<th>$N_u$</th>
<th></th>
<th>$N_v$</th>
<th></th>
<th>$N_m$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>1</td>
<td>s</td>
<td>1</td>
<td>r</td>
<td>1</td>
</tr>
<tr>
<td>s</td>
<td>1</td>
<td>t</td>
<td>1</td>
<td>s</td>
<td>1</td>
</tr>
<tr>
<td>m</td>
<td>2</td>
<td>m</td>
<td>2</td>
<td>t</td>
<td>1</td>
</tr>
<tr>
<td>v</td>
<td>2</td>
<td>r</td>
<td>2</td>
<td>u</td>
<td>2</td>
</tr>
<tr>
<td>t</td>
<td>2</td>
<td>u</td>
<td>2</td>
<td>v</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Multi-hop Monitoring

- Example 1: Omission at node \( r_1 \)
- Example 2: Manipulation at node \( r_1 \)

Example 3: Manipulation at node \( r_1 \), collaborating node \( s_2 \)

![Diagram of multi-hop monitoring with nodes and connections]

![Diagram of multi-hop monitoring with nodes and connections]
Multi-hop Monitoring

- Generalization as Join Graph

![Join Graph Diagram]

Neighborhood Threshold

- Thresholds for fault detection and correction depend on neighborhood awareness or lack thereof
- Type of fault considered: $\mathcal{F} = \{o, d, r, f, m\}$
  - o: omission
  - d: delay
  - r: routing
  - f: fabrication
  - m: manipulation
Neighborhood Threshold

1. \( \mathcal{F} = \{m\} \): Assume the only faults are in \( C_r \). Then a node \( s \in C_s \) can recover if it receives \( N > e \) identical notifications from clique \( C_r \). \( N = e + 1 \).

True for the topology-aware and topology-unaware cases.

Neighborhood Threshold

2. \( \mathcal{F} = \{m\} \): Assume there are \( e_r \) faults in \( C_r \) and \( e_s, e_t \) passive colluders in \( C_s \) and \( C_t \) respectively.

Topology-aware case: \( s \in C_s \) needs to deal only with faults in \( C_r \), as the others can be ignored. Thus \( s \) can recover if it receives \( N > e \) identical notifications from clique \( C_r \).

Topology-unaware case: the passive colluders have to be considered. Therefore, it is the burden of \( C_r \) to produce enough notifications to compensate for the notifications for the \( e_s + e_t \) colluders. This is only guaranteed if \( C_r \) is of size \( N > 2e \), with \( e = e_r + e_s + e_t \), i.e., at least \( e + 1 \) monitors in \( C_r \) respond.
Neighborhood Threshold

- Threshold for each fault type can be derived
  \[ \mathcal{F} = \{o, d, r, f, m\} \]
- Big issue becomes buffer size
  - packet ID
  - packet
  - header
  - signature
  - ...

General Neighborhood Watch

- There is no snake oil!
- The final model of a neighborhood watch depends on
  - the desired fault model
  - the overhead tolerated by the application
- This is a trade-off space!
Conclusions

A cross-monitoring method was presented that:

- Makes no assumptions about behavior of malicious nodes
- Establishes the thresholds for detection/correction; any node can be monitor
- Includes previous work as special case
- Establishes the connection between fault models and the associated monitoring overhead

Current efforts focus on overhead analysis

Questions?
Join Graph Example

Assume nodes are moved to implement the GJG below