Aggregation and Correlation of Intrusion-Detection Alerts

Hervé Debar
France Télécom R&D

Andreas Wespi
IBM Zurich Research Laboratory
Overview

- Introduction
- Architecture
- Representation of alerts
- Aggregation and correlation of alerts
- Future work
**Issues addressed**

- Aggregation and correlation of ID alerts generated by a diversity of sensors
- Flooding: operator gets overloaded by the large number of ID alerts
  - Context: grouping of related alerts
  - Elimination of false alerts
- Scalability
What's special about our project?

- Started about 3 years ago
- Influenced the IDWG work on IDMEF
- Reused existing framework to integrate the ID sensors and their alerts: Tivoli Enterprise Console
- Is available as a product: Tivoli Risk Manager
Overview

- Introduction
- Architecture
- Representation of alerts
- Aggregation and correlation of alerts
- Future work
Tivoli Enterprise Console

**TEC_tasks**
Alert reaction

**TEC_rules**
Alert correlation

**TEC_reception**
Alert reception

Pre-adapter (ISS RealSecure)

Pre-adapter (Cisco Secure IDS)

Probe (ISS RealSecure)

Probe (WebIDS)

Probe (Cisco Secure IDS)
Overview

- Introduction
- A generic, scalable ID Architecture
- Representation of alerts
- Aggregation and correlation of alerts
- Future work
Alert class hierarchy (1)

- Data-driven approach
  - represent in a structured way all information as provided by the probes
  - use same data structure for all types of probes (host- and network-based, knowledge- and behavior-based, ...)

- Unified data model:
  - independence of the correlation algorithms and the actual alerts generated by the probes
  - easy integration of any probe in the correlation framework
Alert class hierarchy (2)

- Data model corresponds to an early draft of the IDWG proposal for a standard message exchange format
- Abstract classes:
  - generic part of alerts
- Implementation classes:
  - inherit from abstract classes
  - carry specific alert information generated by specific probes
Layered data model

Sensor layer

Target layer

Source layer

Detailed target layer
Overview

- Introduction
- Architecture
- Representation of alerts
- Aggregation and correlation
- Conclusions and future work
Alert preprocessing

- Purpose:
  - error checking
  - unify information provided by different probes
- IP address (network-based IDS) vs. hostname (host-based IDS)
- Port number (network-based IDS) vs. service name (host-based IDS)
- Validity of timestamp is checked and adjusted if needed
Different sensors see the same attack, e.g.:

- `duplicate_event('NR_WWW_bat_File', 'RS_HTTP_IE_BAT', [source, dest, url, time], 4.0).

If source, dest, and url of the two events are the same and the timestamps are close, the two events are linked and further on treated as a single event.
Two events by the same sensor are related, e.g.:

- `duplicate_event('WW_Suspicious_Cgi', 'WW_Success', [ids, req_id], 25.0).

- Duplicate gets a high severity value due to a successful cgi attack.
Consequences

- A consequence chain is a set of alerts linked in a given order, where the link must occur within a given time interval, e.g.:
  - `consequence('RS_Http_Phf, 'rs.probe.org', 'WW_InsecCgiPhf', 'web.probe.org', 30, 10.0).
- If the WebIDS probe does not report the phf attack within 30 seconds, an internal event is generated with severity 10.0
- This feature may be used to check whether a probe is still operational.
Single alerts may not be significant, but the aggregation of single alerts may reveal the attack.

A situation is a set of alerts that have certain characteristics in common.

Three aggregation axes: source, target, and attack category.

Systematic combination of aggregation axes results in seven situations.
### Situations (2)

<table>
<thead>
<tr>
<th></th>
<th>attack/source/dest</th>
<th>attack</th>
<th>source</th>
<th>dest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-1</td>
<td>attack/source</td>
<td>attack</td>
<td>source</td>
<td>*</td>
</tr>
<tr>
<td>2-2</td>
<td>attack/dest</td>
<td>attack</td>
<td>*</td>
<td>dest</td>
</tr>
<tr>
<td>2-3</td>
<td>source/dest</td>
<td>*</td>
<td>source</td>
<td>dest</td>
</tr>
<tr>
<td>3-1</td>
<td>attack</td>
<td>attack</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>3-2</td>
<td>source</td>
<td>*</td>
<td>source</td>
<td>*</td>
</tr>
<tr>
<td>3-3</td>
<td>dest</td>
<td>*</td>
<td>*</td>
<td>dest</td>
</tr>
</tbody>
</table>
Situations (3)

- Situations are evaluated in parallel.
- Thresholds for different warning levels have to be specified, e.g.
  - `threshold('situation1', warning, minor, critical, source, dest, attack).
- Generic as well as specific situations, e.g.:
  - `threshold('situation1', 10.0, 20.0, 30.0, _, _, _)`.
  - `threshold('situation1', 5.0, 10.0, 20.0, 'susp.host.org, _, _)`.
- Different sizes of time windows are considered.
Overview

- Introduction
- Architecture
- Representation of alerts
- Correlation and aggregation
- Future work
Future work

- Improved correlation rules
  - Analysis of real data
  - Robust correlation rules
- Improved performance by outsourcing certain correlation rules from TEC's prolog engine
  - Zurich correlation engine
- Consider information about network topology and machine configurations
- Integration of new sensors
Operational issues

- Configuration
  - Severity and confidence value per alert
  - Default vs. site-specific values

- Performance
  - One alert per second
Conclusions

- Correlation is needed to reduce the number of alerts a system administrator has to handle.
- A standard message format will help to easily integrate new sensors in our framework.