Collaborative Incident Handling Based on the Blackboard-Pattern

Nadine Herold, Holger Kinkelin

November 25, 2016

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich
Foreword

• Presentation based on slides from 3rd Workshop on Information Sharing and Collaborative Security (WISCS 2016) held in conjunction with 23rd ACM Conference on Computer and Communications Security (CCS)

• Added for today: Future work on security and privacy aspects of the blackboard
Motivation and Background

Related Work and Problem Statement

System Design and Implementation

Evaluation

Future Work: Security and Privacy

Conclusion
Motivation and Background

Related Work and Problem Statement

System Design and Implementation

Evaluation

Future Work: Security and Privacy

Conclusion
Motivation

• Amount and variants of attacks on networks is growing
• Defending networks manually is impossible
• **Automated** incident handling is highly beneficial
  • Continuously defend the network
  • Respond quickly
  • Less error-prone
  • Systematical incident response
Background: Typical Intrusion Handling Steps

- **Network Monitoring (NMS)** and **Intrusion Detection Systems (IDS)** collect information about the network and its healthiness
  - NMS: collect infrastructure information
  - IDS: raise alerts when an intrusion is detected

- **Alert Processing Systems (APS)** aggregate, correlate and prioritize alerts
  - Gain more insights into the intrusion by analyzing the situation

- **Intrusion Response Systems (IRS)** counteract automatically
  - Identify suitable responses
  - Execute responses on the target network, e.g., block a rogue host
Motivation and Background

Related Work and Problem Statement

System Design and Implementation

Evaluation

Future Work: Security and Privacy

Conclusion
Execution Model: Pipelined Intrusion Handling

- NMS
- NIDS
- HIDS
- APS
- IRS

Info
Alert
Alert
Correlated or Aggregated Alerts
Response

Amount of Information

Holger Kinkelin – Collaborative Incident Handling Based on the Blackboard-Pattern
Problem Statement

• Significant effort has been made to improve each intrusion step individually

• No solution exists that interleaves steps and creates a comprehensive view on the target network
  • Information already collected/computed in previous steps is lost for being used by subsequent steps
  • Information and intermediate results cannot be shared efficiently between single steps
Motivation and Background

Related Work and Problem Statement

System Design and Implementation

Evaluation

Future Work: Security and Privacy

Conclusion
Introducing the Blackboard Pattern

• The blackboard pattern is applicable to problems that can be decomposed into smaller sub-problems / sub-tasks
  • Example: (distributed) incident handling / intrusion handling
• Sub-tasks solve their sub-problem and share their intermediate results with other sub-tasks
• Original information remains untouched
• Original information + intermediate results can be reused by sub-tasks to further tackle the problem
• Blackboard needs an Information Model specifically designed for the problem domain
Blackboard-based Intrusion Handling

- NMS
- NIDS
- HIDS
- Blackboard
- Alert Processing
- IRS

Intermediate Results (Aggregated or Correlated Alerts)
Original, Aggregated or Correlated Alerts and Info
Response
Information Model for Intrusion Response - Overview

Alert Processing
- Alert
- Consequences
- Attack
- Target
- Source
- Priority
- Alert Context

Infrastructure Information
- Network
- L3-Network
- IP-Address
- Port
- Service
- User
- MAC-Address
- Interface

Intrusion Response
- Response
- Network-Based
- Host-Based
- Service-Based
- User-Based
- Active
- Passive
- Metric
- Response Bundle
- Implementation

Holger Kinkelin – Collaborative Incident Handling Based on the Blackboard-Pattern
Infrastructure Information Model – Examples

- NMSes send their scanning results to specific interfaces which add the info to the Blackboard
- A Service runs at a Port opened on a NIC with an IP-Address belonging to a L3-Network
- A Device has a NIC with MAC-Address and assigned IP-Address
- A User is logged into Device
- A User uses Service
Implementation

• Python 3
• Object oriented implementation of Information Model
• Automatic translation of class structures to suitable database design
• Two different databases/database types used:
  • Relational: postgreSQL
  • Graph-based: OrientDB
Motivation and Background

Related Work and Problem Statement

System Design and Implementation

Evaluation

Future Work: Security and Privacy

Conclusion
Evaluation – Test Data Sets and Test Cases

→ Measure the prototype’s performance under varying conditions

• **Test data sets** simulate different attacks:
  - **DDoS** DDoS: many sources attack a small number of targets
  - **AP** Attack path: an attack spreads in the network
  - **F** Flooding: Multiple IDSes raise the same alert
  - Test data set size: from 1000 to 5000 alerts

• **Test cases** simulate typical tasks of the intrusion handling system
  - **ins** Node Insertion – Adding of Alert and Alert Context nodes
  - **prio** Node Prioritization – Updates Priority attribute of Alert and Alert Context nodes with random number
  - **comb** Node Combination – Combining related Alerts Context nodes
  - Test cases are cumulative, e.g., t3 contains t1 and t2
Measurement Results: Alerts per Second

<table>
<thead>
<tr>
<th>Exp.</th>
<th>pSQL&lt;sub&gt;min&lt;/sub&gt;</th>
<th>pSQL&lt;sub&gt;max&lt;/sub&gt;</th>
<th>pSQL&lt;sub&gt;avg&lt;/sub&gt;</th>
<th>Orient&lt;sub&gt;min&lt;/sub&gt;</th>
<th>Orient&lt;sub&gt;max&lt;/sub&gt;</th>
<th>Orient&lt;sub&gt;avg&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDoS&lt;sub&gt;ins&lt;/sub&gt;</td>
<td>287.09</td>
<td>354.72</td>
<td>320.75</td>
<td>11.4</td>
<td>19.72</td>
<td>14.73</td>
</tr>
<tr>
<td>DDoS&lt;sub&gt;prio&lt;/sub&gt;</td>
<td>228.61</td>
<td>307.27</td>
<td>257.8</td>
<td>8.4</td>
<td>16.24</td>
<td>11.55</td>
</tr>
<tr>
<td>DDoS&lt;sub&gt;comb&lt;/sub&gt;</td>
<td>64.97</td>
<td>125.44</td>
<td>86.15</td>
<td>1.37</td>
<td>6.75</td>
<td>3.12</td>
</tr>
<tr>
<td>AP&lt;sub&gt;ins&lt;/sub&gt;</td>
<td>299.4</td>
<td>355.76</td>
<td>324.76</td>
<td>12.5</td>
<td>19.35</td>
<td>15.13</td>
</tr>
<tr>
<td>AP&lt;sub&gt;prio&lt;/sub&gt;</td>
<td>230.36</td>
<td>287.86</td>
<td>250.71</td>
<td>8.91</td>
<td>16.23</td>
<td>11.62</td>
</tr>
<tr>
<td>AP&lt;sub&gt;comb&lt;/sub&gt;</td>
<td>30.80</td>
<td>85.12</td>
<td>49.59</td>
<td>0.51</td>
<td>3.01</td>
<td>1.1</td>
</tr>
<tr>
<td>F&lt;sub&gt;ins&lt;/sub&gt;</td>
<td>370.32</td>
<td>396.63</td>
<td>384.58</td>
<td>37.88</td>
<td>50.87</td>
<td>44.77</td>
</tr>
<tr>
<td>F&lt;sub&gt;prio&lt;/sub&gt;</td>
<td>318.1</td>
<td>330.31</td>
<td>325.04</td>
<td>15.4</td>
<td>35.29</td>
<td>23.38</td>
</tr>
<tr>
<td>F&lt;sub&gt;comb&lt;/sub&gt;</td>
<td>281.78</td>
<td>293.31</td>
<td>287.73</td>
<td>14.13</td>
<td>18.00</td>
<td>16.97</td>
</tr>
</tbody>
</table>

Table contains min, max and average rates of all test data set sizes
Motivation and Background

Related Work and Problem Statement

System Design and Implementation

Evaluation

Future Work: Security and Privacy

Conclusion
Authenticity, security and privacy

• ... of the information in the BB is important

• **Authenticity:** faked information might trigger IRS to counteract in a manner beneficial for the attacker
  • E.g.: Shut down VM, disconnect network, etc. → DoS-like effect

• **Security:** leaked information might provide helpful insights for an attacker
  • E.g.: Network structure, targets, weaknesses, defense mechanisms

• **Privacy:** information in the BB might be related to persons and needs sufficient protection
  • E.g.: MAC address of a personal device identifies person

→ We need to protect the BB’s data from rogue **Modules**
DB Security Orchestration by Blackboard Controller

- Authentication of Modules
  - Module obtains SSL certificate
  - Authenticates towards Controller
  - If needed: integrity checks possible (Remote Attestation)
  - BB Controller creates transient username/password for this Module
    → Generally applicable for each module

- Fine-grained DB access control:
  - Controller additionally sets specific DB permissions for a Module
  - R/W access to specific DB tables / DB table attributes
  - Creation of specific DB views for Module
  - Stored procedures, e.g., for querying aggregated values
    → Permissions/other options vary for different Modules and also the used DB
Can we additionally protect against server-side attacks?

- We still have a central collection of sensitive data on a server
- Server might be attacked → Can we use a cryptographic DB?
- **Example: ZeroDB**
  + Only encrypted information on DB server
  - Query logic shifted to clients
  - Decreases performance by some magnitudes (esp.: latency)
  - Only small subset of SQL features available, e.g., no views
  - Implementation so far only single user; no information sharing

→ Alternative: partially encrypt highly sensitive information with CP-ABE, etc.
Motivation and Background

Related Work and Problem Statement

System Design and Implementation

Evaluation

Future Work: Security and Privacy

Conclusion
Conclusion

• Related work has drawbacks: information sharing is difficult between intrusion handling steps, information loss, ...

• **Our contributions:**
  • Blackboard-pattern for intrusion handling
  • Suitable information model
  • → Enables Information sharing between intrusion handling steps
  • Proof-of-concept implementation using two different DBs

• **Future Work:**
  • Information security of the data on the Blackboard
  • Improving performance
Contact

Thank you for the audience!

Nadine Herold, Holger Kinkelin and Georg Carle

Technische Universität München
Department of Informatics
Chair of Network Architectures and Services
Boltzmann Straße 3
85748 Garching bei München
Germany

{lastname}@net.in.tum.de
https://github.com/Egomania/BlackboardIDRS