IRSS: Incident Response Support System

Ing. Gianluca Capuzzi
• Dipartimento di Ingegneria Informatica, Gestionale e dell’Automazione, Università Politecnica delle Marche

• Ing. Gianluca Capuzzi

• Ing. Egidio Cardinale

• Ing. Claudio Cilli

• Prof. Luca Spalazzi
Computer and network security can be improved by three kinds of tool:

- Tools dealing with prevention
- Tools dealing with detection
- Tools dealing with response

Several systems have been proposed for the first two kinds, the response is still left to the Security Manager
Current scenarios

- High volume of log messages (several different structures)
- Insufficiency of support systems: no integrated tools
- Timeliness of the Incident Response Activity
Aims of the work

- It creates an incident response system (we call IRSS) that supports the job of the Security Manager
- It gathers information from the other security systems (log messages)
- It correlates them to recognize attacks (set of events)
- It searches in a Knowledge Base for the closest past incident
- It returns the related plan solution
Cycle process

IRSS

Responses

Alerts

Attacks

Cases
Solutions can be applied to address some problems:

- Confortable Reading of logs
- Further Elaboration
- Response
- Network Forensic Analysis
Schema of the IRSS

Incident Assessment

Information Analysis → Correlations Search → Case Retrieval

Temporal sequence

Similar case → Plan Adaptation → Plan Execution

Plan executed → Plan Retain

CBR
It consists of two modules:

- A module of Incident Assessment that correlates the information in input outcoming attacks (sequences of events)

- A Reasoner (Case-Based Reasoner) that receives the new case (attack), searches the closest case in the KB and returns the corresponding response
There are many works related to this topic.

In particular, there is a paper that has a comprehensive approach to the problem of Alert Correlation: “A Comprehensive Approach to Intrusion Detection Alert Correlation”

The authors are F. Valeur, G. Vigna, C. Kruegel, and R. Kemmerer.

This paper describes a correlation algorithm, which considers results of previous publications.
Concerning the use of Case-Based Reasoning to network security, we have only few example

The most notable is “A Case-Based Approach to Network Intrusion Detection”

The authors are D. G. Schwartz, S. Stoecklin and E. Yilmaz”

This paper describes the possible application of CBR to the Intrusion Detection
Concerning Incident Response, we have:

- Tools which deal with Intrusion Prevention working in in-line mode to block malicious connections
- Tools dealing with Forensic Analysis
- Tools dealing with Restore previous state (backup)

But we have not a tool that supports the whole job of the Security Manager
Running example

Example to explain how this system works:

- Portscan
- Apache exploit
- Attempt to modify the linuxconf file

<table>
<thead>
<tr>
<th>ID</th>
<th>Type of attack</th>
<th>Sensor</th>
<th>Start/End</th>
<th>Source</th>
<th>Target</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IIS Exploit</td>
<td>N1</td>
<td>12.0/12.0</td>
<td>00.0.0.1</td>
<td>10.0.0.1.60</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Portscan</td>
<td>N2</td>
<td>10.1/14.8</td>
<td>31.3.3.7</td>
<td>10.0.0.1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Portscan</td>
<td>N1</td>
<td>10.0/15.0</td>
<td>31.3.3.7</td>
<td>10.0.0.1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>TFTP GET passwd</td>
<td>N1</td>
<td>11.3/11.3</td>
<td>192.168.10.41</td>
<td>192.168.10.52.80</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>TFTP GET passwd</td>
<td>N2</td>
<td>11.3/11.3</td>
<td>192.168.10.41</td>
<td>192.168.10.52.80</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Apache Exploit</td>
<td>N1</td>
<td>22.0/22.0</td>
<td>31.3.3.7</td>
<td>10.0.0.1.60</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Bad Request</td>
<td>A</td>
<td>22.1/22.1</td>
<td>10.0.0.1</td>
<td>10.0.0.1,Apache</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Local Exploit</td>
<td>H</td>
<td>24.6/24.6</td>
<td>10.0.0.1</td>
<td>10.0.0.1,linuxconf</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Local Exploit</td>
<td>H</td>
<td>24.7/24.7</td>
<td>10.0.0.1</td>
<td>10.0.0.1,linuxconf</td>
<td></td>
</tr>
</tbody>
</table>
Running example

- This is an example of an intrusion consisting of three steps.

- We have all the log messages of the attack.

- The first one is a non-relevant log event.

- There are two log messages related to another attack.
System overview

Schema of the network used to test the IRSS
Design of the Correlator
The Correlation is carried out in several steps

Each step is realized by a submodule

The input data is a set of alerts, while the output data is a set of attacks

Each Buffer List allows transferring correlated alerts
The structure of Log message

- ID
- Message
- Sensor
- Start_time
- End_time
- Source
- Target
- Tag
Example: alert correlation

This is the result after the correlation

<table>
<thead>
<tr>
<th>ID</th>
<th>Sensor</th>
<th>Start/End</th>
<th>Source</th>
<th>Target</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>N1, N2</td>
<td>10.0/14.8</td>
<td>31.3.3.7</td>
<td>10.0.0.1</td>
<td>2,3</td>
</tr>
<tr>
<td>11</td>
<td>N1, N2</td>
<td>11.3/11.3</td>
<td>192.168.10.41</td>
<td>192.168.10.52.80</td>
<td>4,5</td>
</tr>
<tr>
<td>12</td>
<td>N1, A</td>
<td>22.0/22.1</td>
<td>31.3.3.7</td>
<td>10.0.0.1.80</td>
<td>6,7</td>
</tr>
<tr>
<td>13</td>
<td>H</td>
<td>24.6/24.7</td>
<td>10.0.0.1</td>
<td>10.0.0.1,linuxconf</td>
<td>8,9</td>
</tr>
<tr>
<td>14</td>
<td>N1, N2,A,H</td>
<td>10.0/24.7</td>
<td>31.3.3.7,10.0.0.1</td>
<td>10.0.0.1.80,Apache,linuxconf</td>
<td>10,12,13</td>
</tr>
<tr>
<td>15</td>
<td>N1, N2</td>
<td>11.3/11.3</td>
<td>192.168.10.41</td>
<td>192.168.10.52.80</td>
<td>11</td>
</tr>
</tbody>
</table>
The correlation schema consists of four steps:

- Fusion
- Session reconstruction
- Focus recognition
- Thread reconstruction
Normalization and preprocessing

- These are not real correlation steps
- The first one normalizes log messages giving them the same structure
- The second one marks alerts non-relevant: for example, if the target is not vulnerable to this attack
Fusion

- This step aims to merge alerts produced by the same event: for instance, those produced by two sensors detecting the same packet.
- It merges identical alerts whose timestamps differs no more than $\Delta t$.
- $\Delta t$ is the max delay of the network.
This step aims to merge alerts produced by different kinds of source.

For instance, alerts produced by Network-IDS, Host-IDS, O.S., etc.

The resulting alert includes more information.
Focus recognition

This step aims to merge alerts produced by attacks *one-to-many* and *many-to-one*

For instance, portscanning, DDoS attacks, etc.

The resulting alert as one source IP and several target IPs, or viceversa
Thread reconstruction

- This is the most important step

- It aims to link events related to the same attack

- It analyzes alerts which have the same source IP and target

- The result is a sequence of attack steps
Two classes of experiments: DARPA Data Sets, attacks launched by ourselves

The result of the first class:

<table>
<thead>
<tr>
<th>Input Alerts</th>
<th>Output Alerts</th>
<th>Reduction Volume Alerts</th>
</tr>
</thead>
<tbody>
<tr>
<td>45942</td>
<td>33</td>
<td>99.93%</td>
</tr>
</tbody>
</table>
Experiments and results

Alerts

Reduction 99.93%

Attacks
Experiments and results

- The results of the second class:

<table>
<thead>
<tr>
<th>Attack</th>
<th>$I_P$</th>
<th>$I_{corr}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>nmap</td>
<td>0.17</td>
<td>0.99</td>
</tr>
<tr>
<td>smurf</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ncsa</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>portscan</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>apache</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.73</strong></td>
<td><strong>0.998</strong></td>
</tr>
</tbody>
</table>
Design of the CBR

This is the schema of the CBR system
Incident Retrieval

Knowledge Base:
- Abstraction
- Structure of Case Memory
- Minimal Set of Cases (attacks)

<table>
<thead>
<tr>
<th>CASE ID</th>
<th>ID</th>
<th>Type of Attack</th>
<th>Sensor</th>
<th>Source</th>
<th>Target</th>
<th>Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASE 1</td>
<td>C1.1</td>
<td>SQL Injection</td>
<td>N</td>
<td>ext</td>
<td>webserveraddress:httpports</td>
<td>PLAN 1</td>
</tr>
<tr>
<td></td>
<td>C1.2</td>
<td>SQL Injection Basic Union</td>
<td>N</td>
<td>ext</td>
<td>webserveraddress:httpports</td>
<td></td>
</tr>
<tr>
<td>CASE 2</td>
<td>C2.1</td>
<td>TFTP GET Passwd</td>
<td>N</td>
<td>int/ext</td>
<td>webserveraddress</td>
<td>PLAN 2</td>
</tr>
<tr>
<td>CASE 3</td>
<td>C3.1</td>
<td>Portscan</td>
<td>N</td>
<td>int/ext</td>
<td>webserveraddress</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C3.2</td>
<td>Apache Exploit, Bad Request</td>
<td>N,A</td>
<td>int/ext</td>
<td>webserveraddress:httpports,Apache</td>
<td>PLAN 3</td>
</tr>
<tr>
<td></td>
<td>C3.3</td>
<td>Local Exploit</td>
<td>H</td>
<td>int/ext</td>
<td>webserveraddress:linuxconf</td>
<td></td>
</tr>
<tr>
<td>CASE 4</td>
<td>C4.1</td>
<td>Local Exploit</td>
<td>H</td>
<td>int/ext</td>
<td>webserveraddress</td>
<td>PLAN 4</td>
</tr>
<tr>
<td></td>
<td>C4.2</td>
<td>IIS Exploit</td>
<td>N</td>
<td>int/ext</td>
<td>webserveraddress:httpports</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C4.3</td>
<td>Portscan</td>
<td>N</td>
<td>int/ext</td>
<td>webserveraddress</td>
<td></td>
</tr>
</tbody>
</table>
Incident Similarity Functions:

\[
F_1(I, I_h) = \|S\|
\]

\[
F_2(I, I_h) = \begin{cases} 
\|S\| & \text{Abs}(I) \subseteq \text{Abs}(I_h) \\
0 & \text{otherwise}
\end{cases}
\]

\[
F_3(I, I_h) = \begin{cases} 
\|S\| & \forall e_i, e_j \in I \\
\forall e_l, e_k \in I_h \\
\text{such that } \text{Abs}(e_i) = \text{Abs}(e_l) \in S \\
\text{and } \text{Abs}(e_j) = \text{Abs}(e_k) \in S \\
\text{then } T(e_i) < T(e_j) \text{ iff } T(e_l) < T(e_k) \\
0 & \text{otherwise}
\end{cases}
\]

\[
F_4(I, I_h) = \begin{cases} 
\|F_3(I, I_h)\| & \text{Abs}(I) \subseteq \text{Abs}(I_h) \\
0 & \text{otherwise}
\end{cases}
\]
For our experiments we used the Similarity Function 3

The result is the Case 3

An advantage of this kind of Reasoner is the possibility of manage new attacks

Therefore, it searches for past attacks, returning the closest Case
The adaptation activity is devoted to the Security Manager, who has the final decision.

Now, we follow the basic kind of adaptation, by replacing the abstract attributes with their current values.

After that, it is submitted for validation to the Security Manager.
Example of plan

- A plan consists of commands and messages
- In the example we use the Linux commands for:
  - Sending a message to Security Manager
  - Updating Systems (O.S., Security Systems, etc.)
  - Removing malicious code
  - Communicating to the operator

```
1) mail SecurityAdmin
2) apt-get update systems && apt-get upgrade systems
   echo "control update actions and configuration of security systems"
3) /usr/bin/removaltool
4) echo "restore previous status using backup copy and update it"
```
The output report is presented to the Security Manager who can change and execute it.

During the execution stage the Security Manager has the possibility of evaluating the effectiveness of the response plan.

He can correct the plan or add other instructions to complete it.
After the execution process, the plan has been corrected and evaluated.

Hence, the Security Manager can update the Knowledge Base (Case Memory).

This is an important aspect of the CBR, which is able to learn new cases.

In this way, it can help to solve the problem of managing new attacks.
Conclusions

- The IRSS provides a whole picture about what the Security Manager has to do, coordinating all activities (it’s original)
- It learns new cases with their responses (manage new attacks)
- We have implemented a prototype: first results
- We have planned:
  - to investigate new similarity metrics and more sophisticated adaptation algorithms
  - to perform more experiments