A Trust-Based Benchmark for DBMS Configurations

Afonso Araújo Neto acaneto@dei.uc.pt
Marco Vieira mvieira@dei.uc.pt

University of Coimbra
Informatics Engineering Department

IEEE 15th Pacific Rim International Symposium on Dependable Computing – PRDC’09
Presentation outline

- The problem of security metrics
- Trust-based benchmarking
- Benchmark components and assumptions
- Experimental evaluation
- Final remarks
The problem of security metrics

- Security metric definition:
  “degree to which security goals are met in a given system allowing an administrator to make informed decisions”

- Quantifying security is a hard problem
  - No consensual quantitative security metric has been proposed so far
  - Enterprise Level security metrics is included in the 2005 Hard Problems List prepared by the INFOSEC Research Council
Why is it so hard?

- Actual security depends much more on what is *unknown* about the system
  - Hidden vulnerabilities
  - Configuration inconsistencies
  - Environment interaction

- Also, the adversaries are hard to model
  - How much do they know?
  - How much resources do they have?
The problem of security evaluation

- Available security evaluation methodologies
  - Not DBMS specific
  - Do not easily incorporate environments context
  - Too complex
  - Not comparison oriented

- Furthermore, usually demand security experts to be successfully applied
  - Out of reach of medium/small enterprises
Trust-based benchmark

- Administrator’s have certain level of confidence in the security of their systems
  - How much effort was applied
  - If important aspects were neglected
  - Complex interactions that create vulnerabilities

- This confidence, or trust, is intuitively assessed by the administrator
  - But is this trust warranted?
Trust-based benchmark

- Stop measuring security
  - We don’t know how to do that

- Measure justifiable trust that can be put on the system

- Transform the intuition into *justification* and *evidence* for action
Trust-based metrics

- **Input:**
  - System, Administrator’s efforts

- **Output:**
  - Level of justified trust

- **Trust-based metric**
  
  Systematizes and exposes the trustworthiness relationship between the administrator and his system
Trust-based benchmark

- Measure the “trustworthiness” relationship of the administrator and his system
  - We do not know how to measure how secure a DBMS configuration is
  - *But we can measure how evidently and justifiably untrustworthy it must be considered*

- Minimum Untrustworthiness (MU):
  
  *how much distrust should the DBA have (at least) that a given configuration will prevent the manifestation of relevant threats.*
A Trust-Based Benchmark for DBMS Configurations
DBMS Configurations

- Set of aspects directly/indirectly influenced by the DBAs decisions

- Examples
  - DBMS settings values
  - Privileges
  - How applications interact with the DBMS
  - Some network configurations
Benchmark assumptions

1. One DBMS/ One OS/ One Machine
2. DBA is the only administrator
3. Machine → LAN → Internet
4. Threats ← Real individuals
5. Interaction Classes: OS userid, DBMS userid, app. userid (or none)
6. Roles: one or more interaction classes
7. Custom code executes in webservers, remote hosts in the LAN or as stored procs
The metric

- **Minimum untrustworthiness**
  
  \[ \text{Level of effort employed in securing the system} \]
  
  \[ \times \]
  
  \[ \text{Consensual maximum level of effort possible} \]

- **Less evident effort indicates a more untrustworthy configuration**
  
  - Untrustworthiness suggests the relative likelihood of manifestation of threats
  
  - Maximum ➔ Comprehensive list of security best practices: provides a scale
Benchmark components

- Set of pessimistic scenarios defined through security best practices
- Set of relevant threats for the domain
- Mapping between threats and scenarios/practices
  - Mapping is done from the interaction classes perspective
- Guidelines on how to compute the minimum untrustworthiness
From best practices to threats

- Which practices are associated with which threats?
  
  *In other words, how to cross-reference security best practices with the active manifestation of threats which also depend on several other factors?*

- Solution: *assume the pessimistic consequences of not following the security recommendations*
  
  - Define pessimistic scenarios
### Security best practices

<table>
<thead>
<tr>
<th>#</th>
<th>Security best practice</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prevent physical access to the DBMS platform by unauthorized people</td>
<td>5.26%</td>
</tr>
<tr>
<td>2</td>
<td>Install and properly conf. a firewall on the network border</td>
<td>4.73%</td>
</tr>
<tr>
<td>3</td>
<td>Separate production and development platforms</td>
<td>4.21%</td>
</tr>
<tr>
<td>4</td>
<td>Enforce remote communication encryption with strong algorithms</td>
<td>4.21%</td>
</tr>
<tr>
<td>5</td>
<td>No userid should have privileges to change system properties or configurations</td>
<td>4.21%</td>
</tr>
<tr>
<td>6</td>
<td>Avoid platforms which also hosts critical network services (naming, authentication, etc)</td>
<td>3.68%</td>
</tr>
<tr>
<td>7</td>
<td>Keep the DBMS software updated</td>
<td>3.68%</td>
</tr>
<tr>
<td>8</td>
<td>Use strong encryption in password storage</td>
<td>3.68%</td>
</tr>
<tr>
<td>9</td>
<td>Enforce strong password policies</td>
<td>3.63%</td>
</tr>
<tr>
<td>10</td>
<td>Change/remove default userid/password pairs</td>
<td>3.15%</td>
</tr>
</tbody>
</table>
# Pessimistic scenarios

<table>
<thead>
<tr>
<th></th>
<th>Prevent physical access to the DBMS platform by unauthorized people</th>
<th>The platform is physically stationed in a place where people that have nothing to do with the DBMS have regular unsupervised access</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Install and properly configure firewalls on the network</td>
<td>a) The platform does not have an operating system firewall, leaving all open local ports accessible to the local area network</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b) The network does not have a firewall separating the internal network (LAN) from the servers that provide services to the Internet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c) The network does not have a border firewall, leaving all network fully accessible to internet traffic</td>
</tr>
<tr>
<td>2</td>
<td>Separate production and development platforms</td>
<td>a) The development DBMS is installed in the same platform as the production DBMS, but use different DBMS instances with separate data and configurations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b) The development DBMS and the production DBMS are the same, and are only set apart by privileges within the database</td>
</tr>
</tbody>
</table>
DBMS configuration threats

- Typical DBMS threats are not easily identifiable
  - Orthogonal and representative
  - Defined based on several sources and discussions with industry practitioners

<table>
<thead>
<tr>
<th>Legitimate excessive privilege achievement</th>
<th>Illegitimate privilege elevation</th>
<th>Denial of Service</th>
<th>Communication weakness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication weakness</td>
<td>Side-channel data exposure</td>
<td>Audit trail weakness</td>
<td>SQL Injection enhancement</td>
</tr>
</tbody>
</table>
Mapping (1)

- For each combination of all
  - pessimistic scenarios
  - interaction classes (OS/DBMS/App userid or none)
  - threats

- Determine the following:
  
  *is there any attack possible unders these conditions which accomplishes the threat?*

- Yes identifies a relation between the practice that defines the scenario and the threat
  - For this specific interaction class
Mapping (2)

- Example

  - Recommendation: *separate development and production platforms*
  
  - Pessimistic scenario: *the development DBMS and the production DBMS are the same, and are only set apart by privileges within the database*
  
  - Threat: Denial of Service
  
  - Interaction classes
    - OS: a development based maintenance in the OS may crash the production DBMS
    - DBMS: the execution of an untested development SQL may crash the production DBMS
### Mapping (3)

<table>
<thead>
<tr>
<th>#</th>
<th><strong>Illegitimate privilege achievement</strong></th>
<th><strong>Denial of service</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N: Boot by a CD, copy all file system</td>
<td>N: Disconnect cables, turn off the server or simply destroy it physically.</td>
</tr>
<tr>
<td>2a</td>
<td>N: A LAN user connects to a vulnerable local listening service, causes a buffer overflow allowing arbitrary code execution</td>
<td>N: A LAN user connects to a local listening service and causes it to consume all CPU resources</td>
</tr>
<tr>
<td>2b</td>
<td>N: An attack on a server with internet applications may be used to launch another attack on a private network host, achieving access to all computers on the network, including the DBMS platform</td>
<td></td>
</tr>
<tr>
<td>2c</td>
<td>N: Internet port scans are free to find servers with vulnerabilities and which can be used as leverage to other attacks</td>
<td>N: Internet users may request all kinds of connections to any listening ports in any network server, flooding the network and hogging resources</td>
</tr>
</tbody>
</table>
Computing the metric (1)

- For each recommendation, the DBA evaluates if it is being applied
- Three results are possible
  - It is being correctly applied
  - It is not being applied
  - The DBA does not know if it is correctly being applied
    - In this case he should assume it is not (pessimistic approach)
- E.g., the production and development platforms are effectively separated?
Computing the metric (2)

- The computation may be done in regard to threats and interaction classes

- Threats:
  - For each threat, sum the weights of all not followed recommendations mapping to it
  - Divide the result by the sum of the weights of all recommendations that map to it
  - This results in the minimum untrustworthiness value for this threat

- The same goes for interaction classes
Computing the metric (3)

- The process provides the DBA with a fairly amount of structured information
  - He becomes aware of several security aspects related with his system
  - He becomes conscious of all recommendations he was not aware of
  - He learns how the set of not followed recommendations impacts threats
  - This allows the DBA to decide where more effort is needed and what are the next steps
Experimental evaluation

<table>
<thead>
<tr>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
<th>Case 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBMS</td>
<td>Oracle 10g</td>
<td>SQLServer 2005</td>
<td>MySQL 5.0</td>
</tr>
<tr>
<td>Apps.</td>
<td>3</td>
<td>54</td>
<td>3</td>
</tr>
<tr>
<td>DBAs</td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Developers</td>
<td>8</td>
<td>39</td>
<td>0</td>
</tr>
</tbody>
</table>
Computation of MU: Threats

 ![Diagram showing the computation of MU by scenario](image)
Computation of MU: Int. Classes

Analysys of the Untrustworthiness of each Interaction class

Minimum untrustworthiness

Case 1  Case 2  Case 3  Case 4

Scenario

- Non-System User
- DBMS UserID
- Application UserID
- Operating system UserID
Final remarks

- In this work we presented the idea of a “trust-based” security benchmark
  - Systematizes and exposes the trustworthiness relationship between the administrator and his system

- Minimum Untrustworthiness metric
  - How much distrust the DBA should have that his configuration will prevent manifestation of threats
  - Allows identifying interaction classes more prone to accomplish threats
Final remarks

- We presented a benchmark which allows computing this metric
  - Benchmark components and challenges involved

  More importantly, we believe that this benchmark can help non-security experts in understanding security and hardening their installations
Thank you for your attention

Marco Vieira  mvieira@dei.uc.pt
Afonso Araújo Neto  acaneto@dei.uc.pt