HANDLING OF SECURITY REQUIREMENTS IN SOFTWARE DEVELOPMENT LIFECYCLE

DANIEL KEFER
REPEATING MISTAKES
SECURITY BEHIND DEV PROCESSES AND TOOLING
APPROACH
ALIGN THE PROCESS
SCALE
KISS
SECURITYRAT
USE CASES

New assets

Production assets
INTERNALS
Based on JHipster
### Secure Architecture

<table>
<thead>
<tr>
<th>Short Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| **SA-01**  | 3rd party code is identified, checked for security vulnerabilities and its update process is defined. Implementation of automated tooling can support this task:  
- https://www.owasp.org/index.php/OWASP_Dependency_Check (mapping of dependencies to CVEs) 
- https://nodesecurity.io/tools (evaluation of vulnerable packages for npm) 
- [http://retrieve.github.io/retrieve.js/](http://retrieve.github.io/retrieve.js/) (JavaScript libraries with known vulnerabilities)  
Decrease the security risk being introduced by using vulnerable libraries. Be able to find out quickly if we're affected when new vulnerabilities are published. |
| **SA-02**  | No fundamentally different roles are present in the same application. Example:  
- internal employees and external customers should work on completely separated systems so that the privilege escalation probability and impact in case |

![Image of the interface with data](image-url)
### Optional Columns

<table>
<thead>
<tr>
<th>Short Name</th>
<th>Description</th>
<th>More Information</th>
<th>Motivation</th>
<th>Strategy</th>
<th>Comment</th>
</tr>
</thead>
</table>
| SA-01      | 3rd party code is identified, checked for security vulnerabilities and its update process is defined. Implementation of automated tooling can support this task:  
- https://www.owasp.org/index.php/OWASP_Dependency_Check (mapping of dependencies to CVEs)  
- https://node.security/tools (evaluation of vulnerable packages for npm)  
- http://retirejs.github.io/retire.js/ (JavaScript libraries with known vulnerabilities)  
Decrease the security risk being introduced by using vulnerable libraries. Be able to find out quickly if we're affected when new vulnerabilities are published. | | | | |
| SA-02      | No fundamentally different roles are present in the same application. Example:  
- Internal employees and external customers should work on completely separated systems so that the privilege escalation probability and impact in case | | | | |
Alternatives to Option Columns

OE-01

All untrusted data outputted to any interface are properly escaped for the particular context using a common and standardized approach.

These interfaces can include (but are not limited to):
- SQL
- NoSQL
- Web Services
- LDAP
- ...

Parametrized queries should be used in all cases.

JAVA Application

Example of a prepared statement for SQL queries:

```java
String selectSQL = "SELECT USER_ID, USERNAME FROM DBUSER WHERE USER_ID = ";
PreparedStatement preparedStatement = dbConnection.prepareStatement(selectSQL);
preparedStatement.setInt(1, 1000);
ResultSet rs = preparedStatement.executeQuery(selectSQL);
while (rs.next()) {
    String userid = rs.getString("USER_ID");
}
```
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Description</th>
<th>More Information</th>
<th>Motivation</th>
<th>Strategy</th>
<th>Comment</th>
</tr>
</thead>
</table>
| SA-01      | 3rd party code is identified, checked for security vulnerabilities and its update process is defined. Implementation of automated tooling can support this task:  
  - https://www.owasp.org/index.php/OWASP_Dependency_Check (mapping of dependencies to CVEs)  
  - https://nodesecurity.io/tools (evaluation of vulnerable packages for npm)  
  - http://retriejs.github.io/retrie.js/ (JavaScript libraries with known vulnerabilities)  
  Decrease the security risk being introduced by using vulnerable libraries. Be able to find out quickly if we're affected when new vulnerabilities are published. |                   |            |          |         |
| SA-02      | No fundamentally different roles are present in the same application. Example:  
  - Internal employees and external customers should work on completely separated systems so that the privilege escalation probability and impact in case | Task             |            |          |         |
# Implementation Type

## Artifact Properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Criticality</td>
<td><a href="#">Select</a></td>
</tr>
<tr>
<td>System Type</td>
<td><a href="#">Select</a></td>
</tr>
<tr>
<td>Authentication</td>
<td><a href="#">Select</a></td>
</tr>
<tr>
<td>Session Management</td>
<td><a href="#">Select</a></td>
</tr>
<tr>
<td>Reachability</td>
<td><a href="#">Select</a></td>
</tr>
</tbody>
</table>

**Implementation:**

<table>
<thead>
<tr>
<th>Implementation Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="#">Select</a></td>
<td></td>
</tr>
</tbody>
</table>
Collections

Artifact Properties:

- Criticality
- System Type
- Authentication
- Session Management
- Reachability

Implementation:

- Implementation Type
<table>
<thead>
<tr>
<th>Requirement Owner</th>
<th>Product Manager</th>
<th>Security Mentor</th>
<th>Project Manager</th>
<th>SCRUM Master</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase relevance</td>
<td>Initiation</td>
<td>Design</td>
<td>Coding</td>
<td>QA</td>
</tr>
<tr>
<td>QA</td>
<td>BlackBox</td>
<td>Functional Test</td>
<td>White box</td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td>Design</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
AUTHENTICATION

Own authentication scheme

CAS (Central Authentication Service)
ROLES

Frontend User
User
Admin
JIRA INTEGRATION

Cross Origin Request Sharing

SecurityRAT inherits user‘s rights in JIRA
FUTURE PLANS
REQUIREMENTS

Continuous development (quality vs quantity)

Language-specific information
INTEGRATION

Issue trackers

Other tooling in use by developers
AUTOMATED TESTING OF REQUIREMENTS

Speed up feedback loops for devs

Automated requirement audits
COMMUNITY

Issues

Pull requests

Derived projects
THANK YOU FOR YOUR ATTENTION!

https://securityrat.github.io

dan.kefer@gmail.com