HANDLING OF SECURITY REQUIREMENTS IN SOFTWARE DEVELOPMENT LIFECYCLE

DANIEL KEFER


@DKEFER


ISSUES

REPEATING MISTAKES

SECURITY DOCUMENTATION

SECURITY BEHIND DEV PROCESSES AND TOOLING

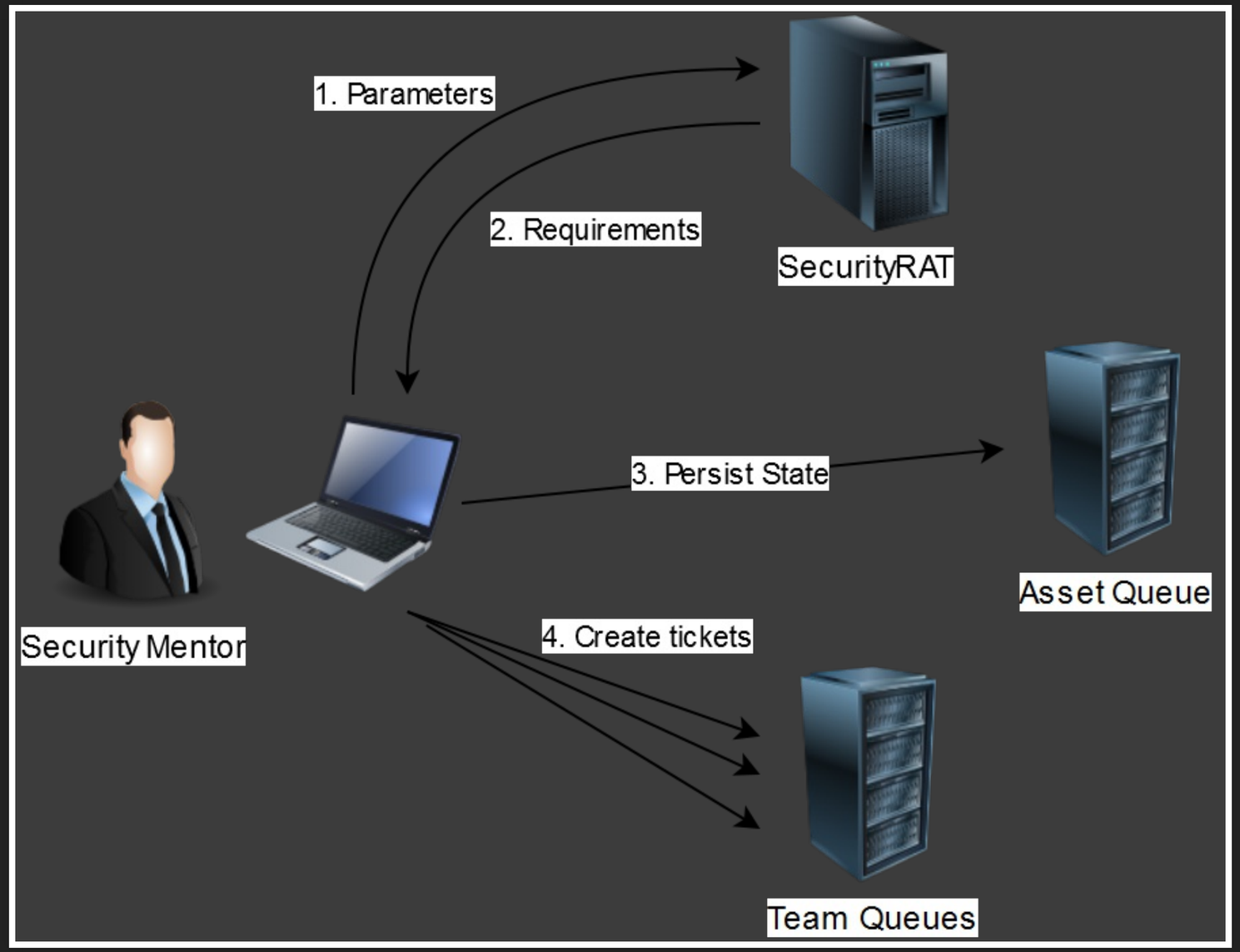
APPROACH

ALIGN THE PROCESS

SCALE

KISS

SECURITYRAT


1. Parameters
2. Requirements
SecurityRAT
3. Persist State
Asset Queue
Security Mentor
4. Create tickets
Team Queues

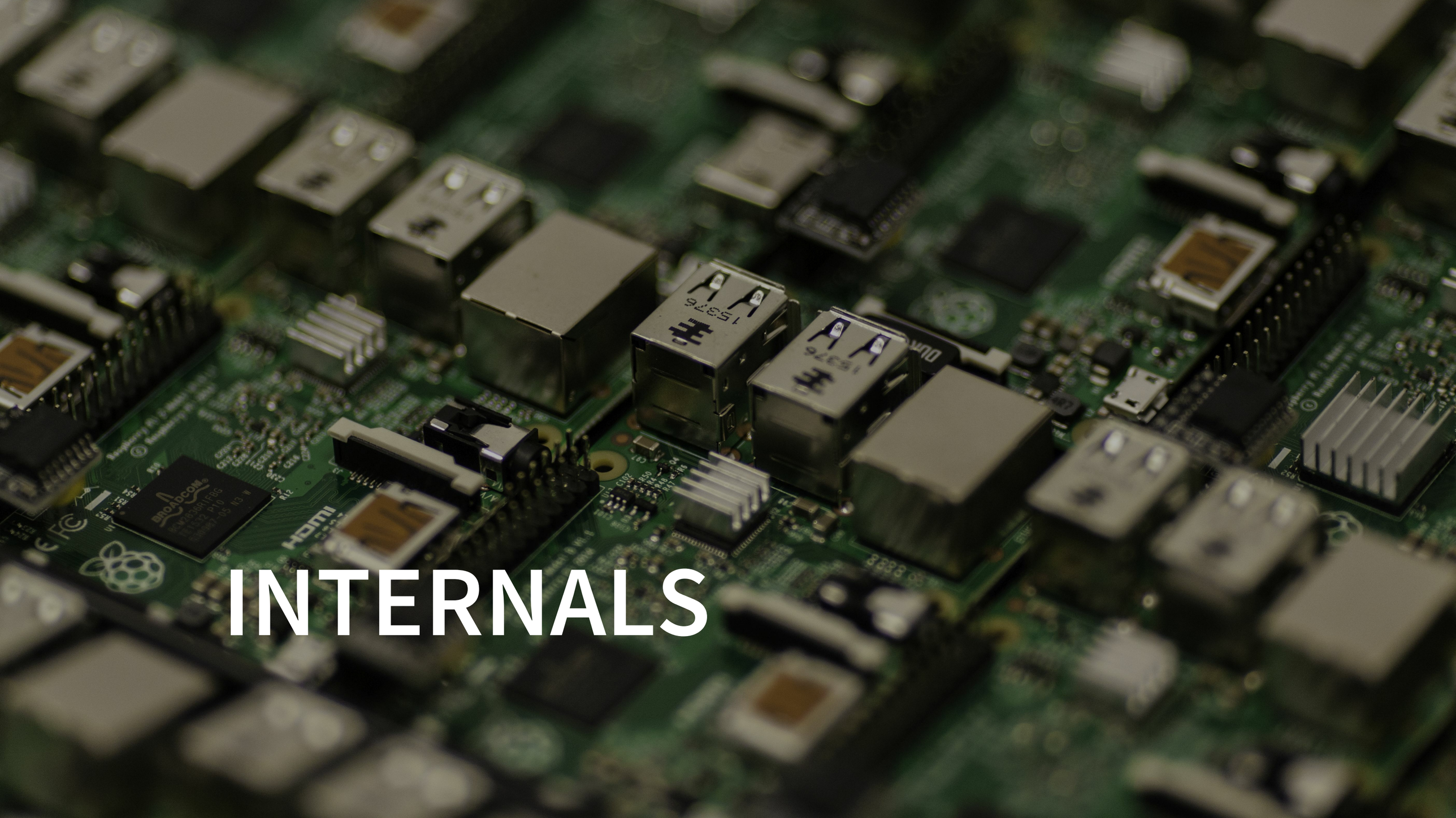
USE CASES

New assets

Production assets

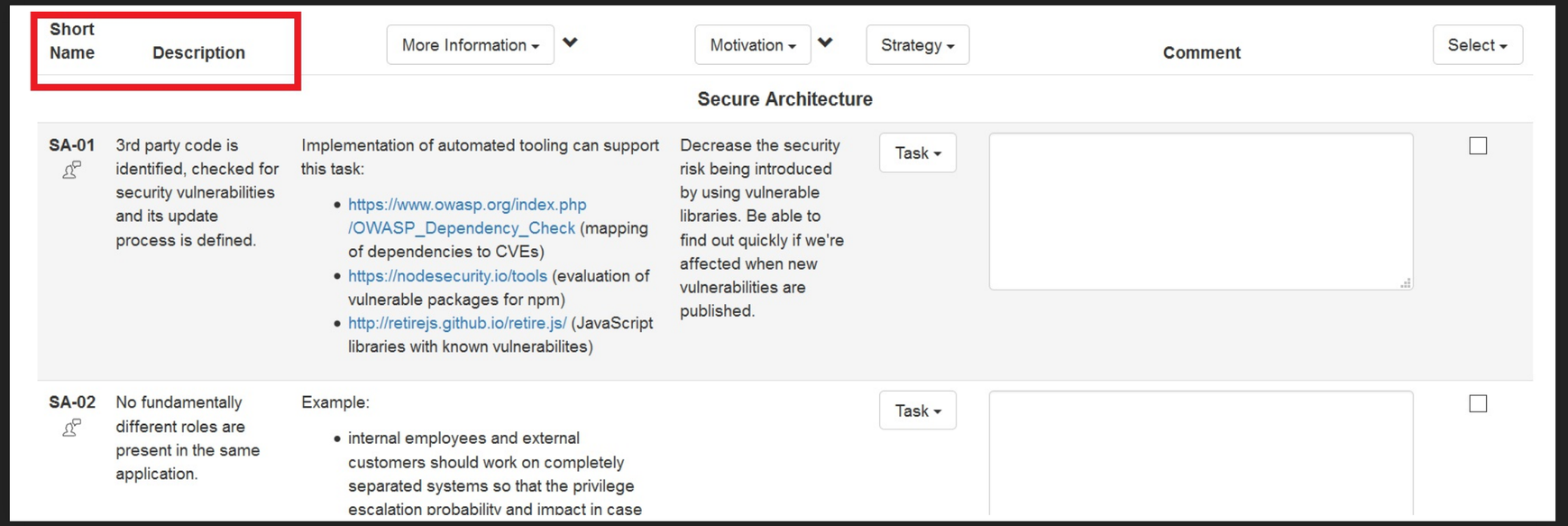

R
6h10m
P3.2K
69%
F4.0
NF89
Z31
BARS
EVF DTL
ZEBRA
LCD
WFM
COUNTER—RESET/TC SET
AUDIO
CH1 SELECT
CH2 SELECT
INT(L)
INPUT1
INPUT2
INT(R)
INPUT 2
CH 1
CH 2
AUTO
MANU
FUNCTION
SHTR/F.RATE
DIAL SEL
DISP/
MODE CHK
AUTO
MANU

DEMO


INTERNALS

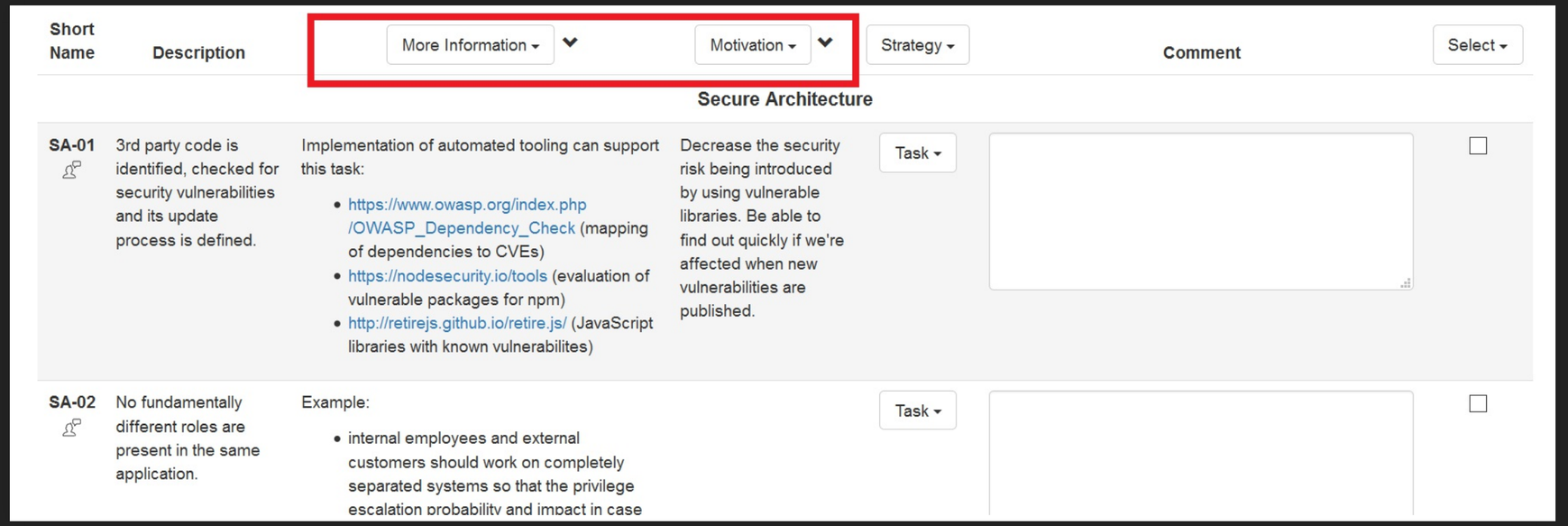
Based on JHipster

Requirement Skeletons



Short Name	Description	More Information	Motivation	Strategy	Comment	Select
		Secure Architecture				
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://retirejs.github.io/retire.js/ (JavaScript libraries with known vulnerabilites)	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.	Task		☐
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		☐

Optional Columns



Short Name	Description	More Information	Motivation	Strategy	Comment	Select
			Secure Architecture			
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://retirejs.github.io/retire.js/ (JavaScript libraries with known vulnerabilites)	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.	Task		☐
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		☐

Alternatives to Option Columns

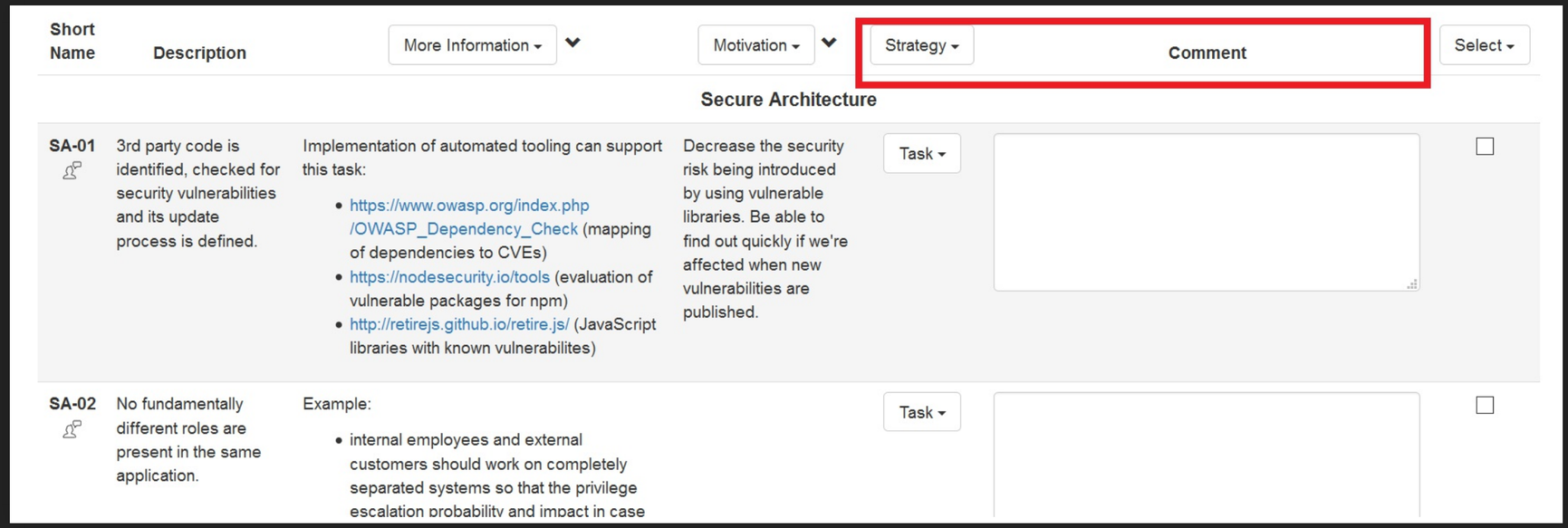
Short Name	Description	JAVA Application	Motivation	Strategy
				Output Encoding
OE-01	All untrusted data outputted to any interface are properly escaped for the particular context using a common and standardized approach.	These interfaces can include (but are not limited to): • SQL • NoSQL • Web Services • LDAP • ... Parametrized queries should be used in all cases.	Prevent injection attacks, e.g.: • SQL Injection • LDAP Injection	Task

JAVA Application

Example of a prepared statement for SQL queries:

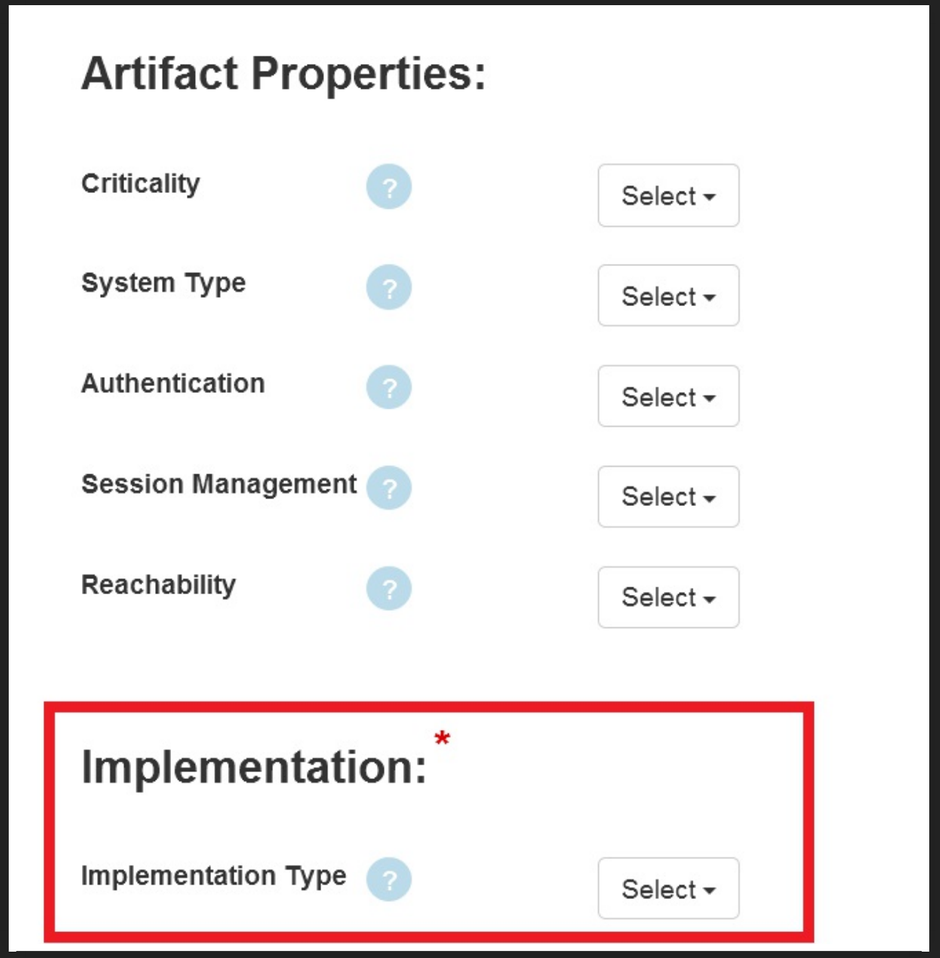
```
String selectSQL = "SELECT USER_ID, USERNAME FROM DBUSER
WHERE USER_ID = ?";
PreparedStatement preparedStatement = dbConnection.prepar
eStatement(selectSQL);
preparedStatement.setInt(1, 1001);
ResultSet rs = preparedStatement.executeQuery(selectSQL )
;
while (rs.next()) {
    String userid = rs.getString("USER_ID");
```

Status Columns

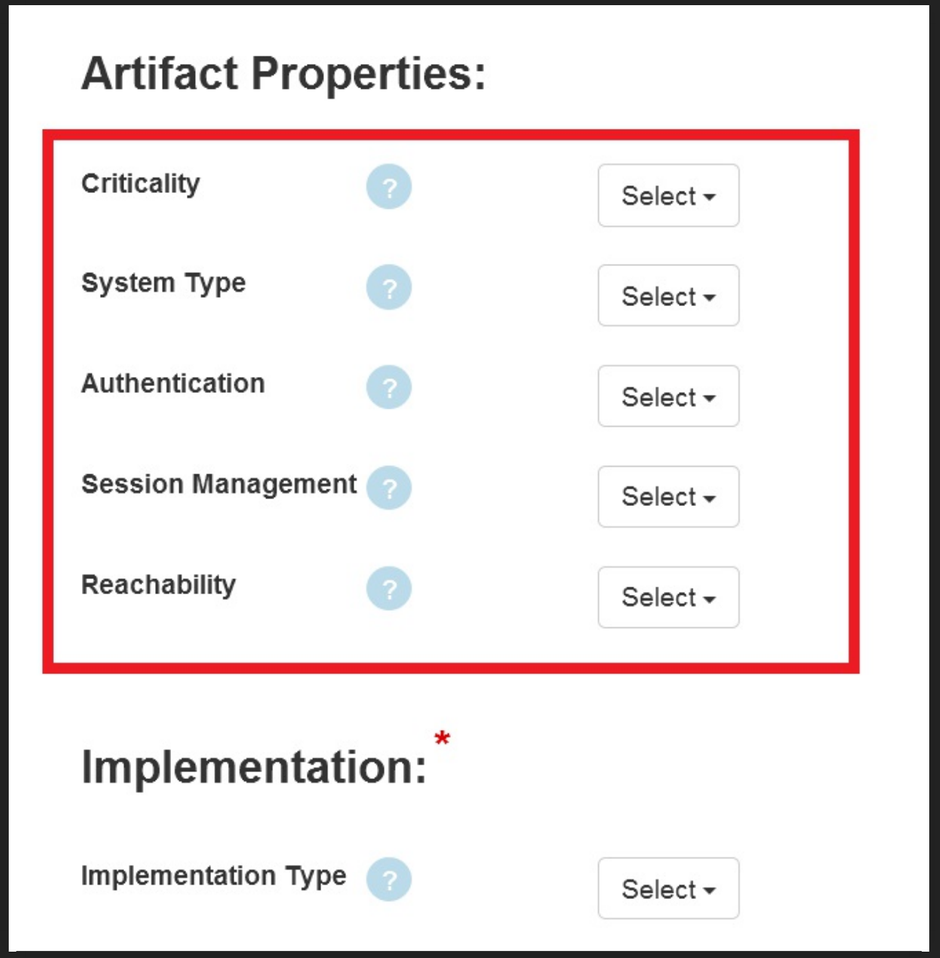


Short Name	Description	More Information	Motivation	Strategy	Comment	Select
				Secure Architecture		
SA-01	3rd party code is identified, checked for security vulnerabilities and its update process is defined.	Implementation of automated tooling can support this task: • https://www.owasp.org/index.php/OWASP_Dependency_Check (mapping of dependencies to CVEs) • https://nodesecurity.io/tools (evaluation of vulnerable packages for npm) • http://retirejs.github.io/retire.js/ (JavaScript libraries with known vulnerabilites)	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.	Task		☐
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		☐

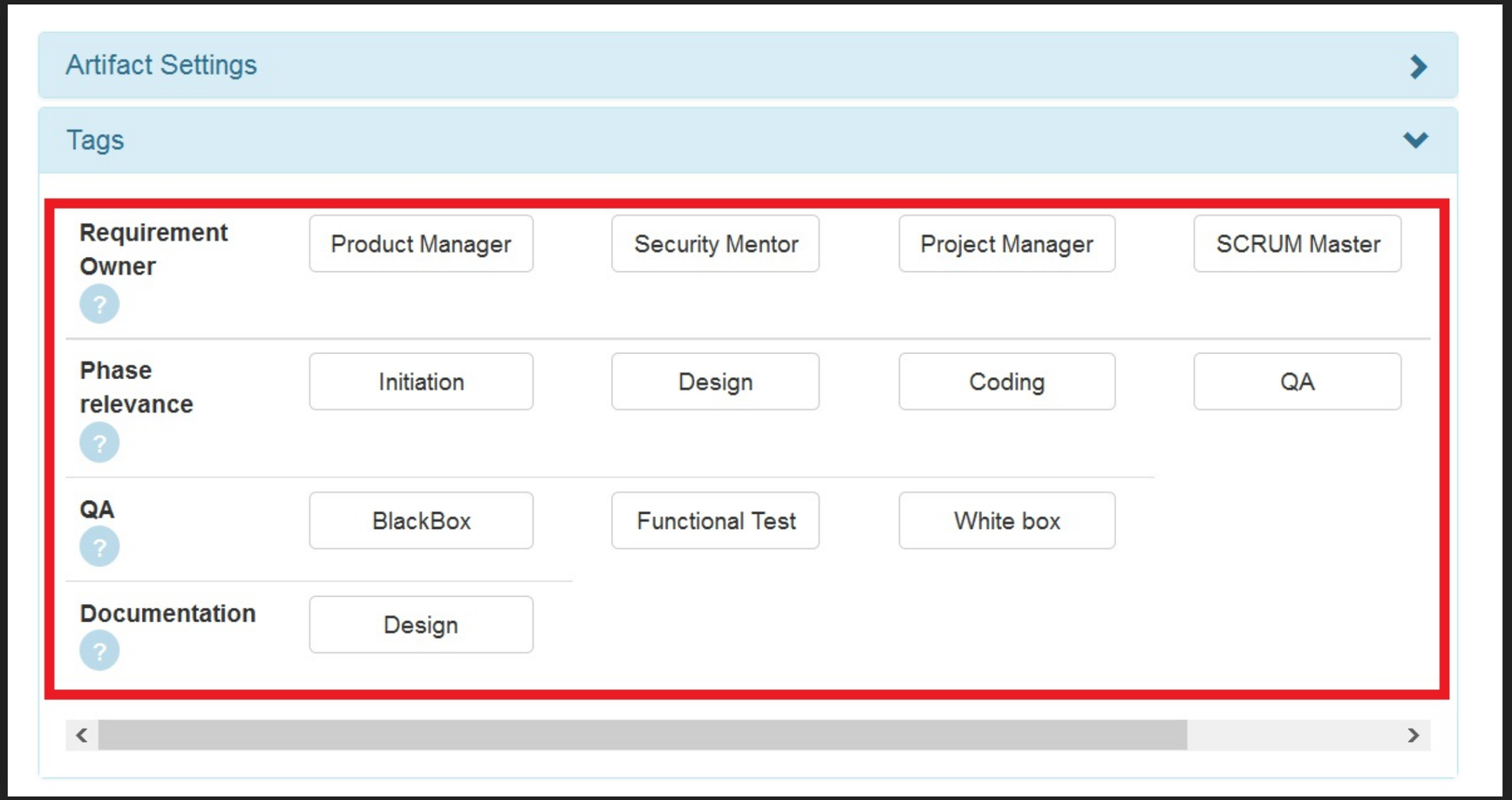
Implementation Type



Collections



Tags


Artifact Settings
Tags
Requirement Owner
Product Manager
Security Mentor
Project Manager
SCRUM Master
Phase relevance
Initiation
Design
Coding
QA
QA
BlackBox
Functional Test
White box
Documentation
Design

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