# Risk Assessment and Security Testing

of Large Scale Networked Systems with RACOMAT

Johannes Viehmann 2015

Fraunhofer
FOKUS

## Overview

**Risk Assessment and Security Testing**
**of Large Scale Networked Systems with RACOMAT**

**Table of Content**

## Introduction – Risk Assessment and Security Testing

### Definition

- Risk assessment is a part of risk management and means to identify, analyze and evaluate risks
- Security testing is one possibility to analyze risks

### Why Risk Management is required

- In the real world, perfect security often cannot be achieved
  - There are residual risks for any complex ICT-System
- Risk assessment and risk treatment can help to create trust by:
  - Communicating residual risks
  - Help to implement safeguards and treatments for to high risks in order to reduce the risks

## Introduction – the Case Study

### The RASEN Research Project

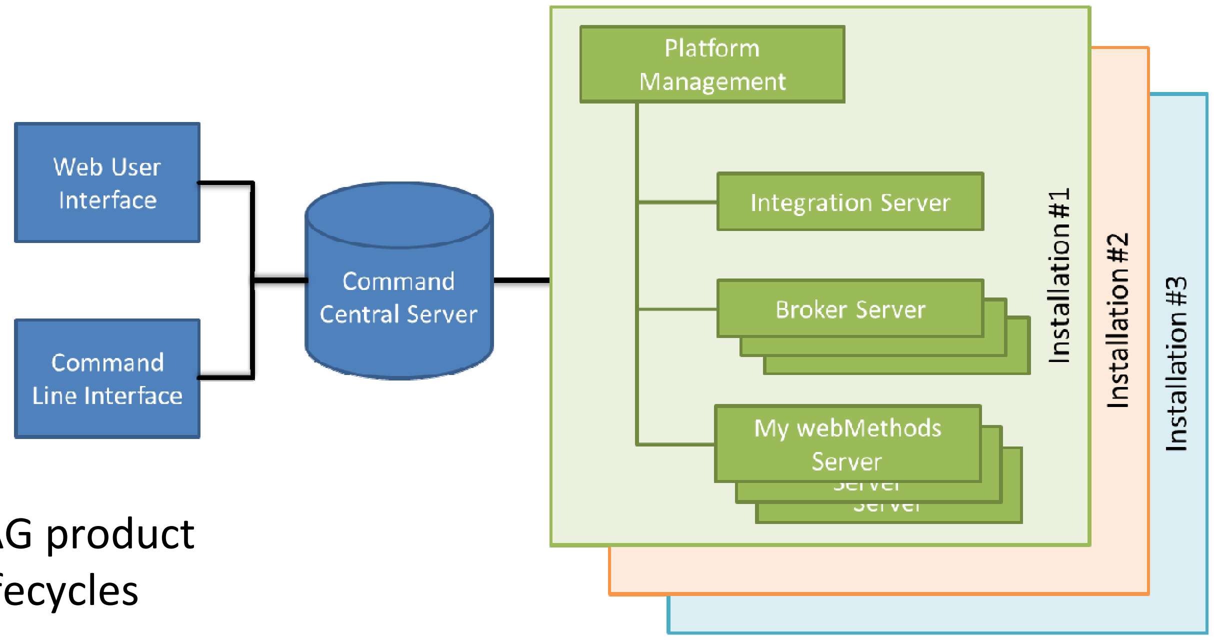
- European project with 7 partners in four countries
- Three industrial case studies

RASEN Compositional Risk Assessment and Security Testing of Networked Systems

SEVENTH FRAMEWORK PROGRAMME

### The Software AG Case Study

- Software under analysis is called Command Central
  - Part of webMethods tool suite by Software AG
  - Uses SAG Common Platform and OSGi framework
  - Intended to manage Software AG product installations throughout their lifecycles



Fraunhofer FOKUS

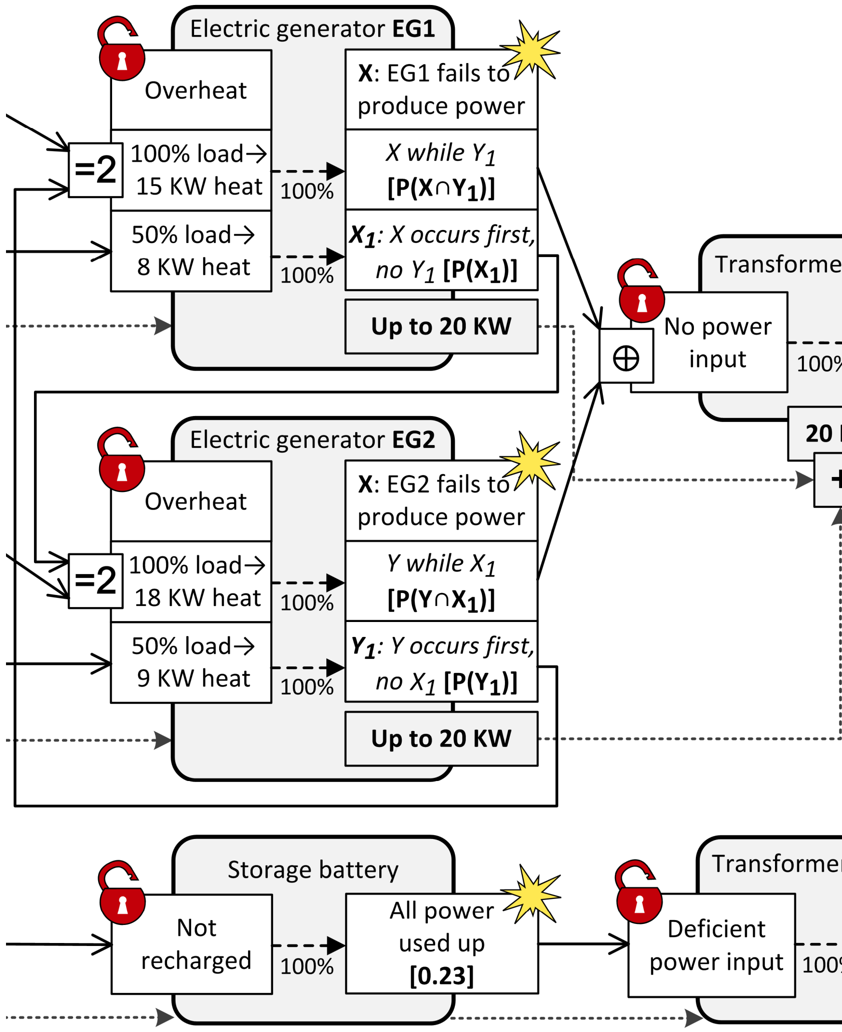
## State of the Art – Risk Assessment and Security Testing

There are lots of methods, libraries and tools for

- Risk Assessment
  - Standard: ISO 31000
  - FMEA/FMECA, FTA, ETA, CORAS …
  - Catalogues of common risk artifacts
    - CWE, CAPEC (Mitre), BSI IT-Grundschutz
- Testing and security testing
  - Standard: ISO 29119
  - Automated testing, fuzz testing …

There is less literature and support for the combination of Risk Assessment and Security testing

- Test-Based Risk Assessment (TBRA)
- Risk-Based Security Testing (RBST)
- Combination of TBRA and RBST



## Problems and Challenges

Risk assessment might be difficult and expensive

- Hard for large scale systems
- Is highly dependent on the skills and estimates of analysts

→ We have to find ways to make risk assessment more objective

➢e.g. with security testing

Security testing might be difficult and expensive, too

- Testing for unwanted behavior – there is no specification what to expect
- Even highly insecure system can produce lots of correct test verdicts if the “wrong” test cases have been created and executed
- Manual testing is error prone and infeasible for large scale systems

→ Automate security testing using risk assessment?

## Problems and Challenges – Combined Risk Assessment and Testing Process

1. Identification

What should be tested?

2. Prioritization

Spend how much effort for which tests?

3. Generation

Which test cases should be created?

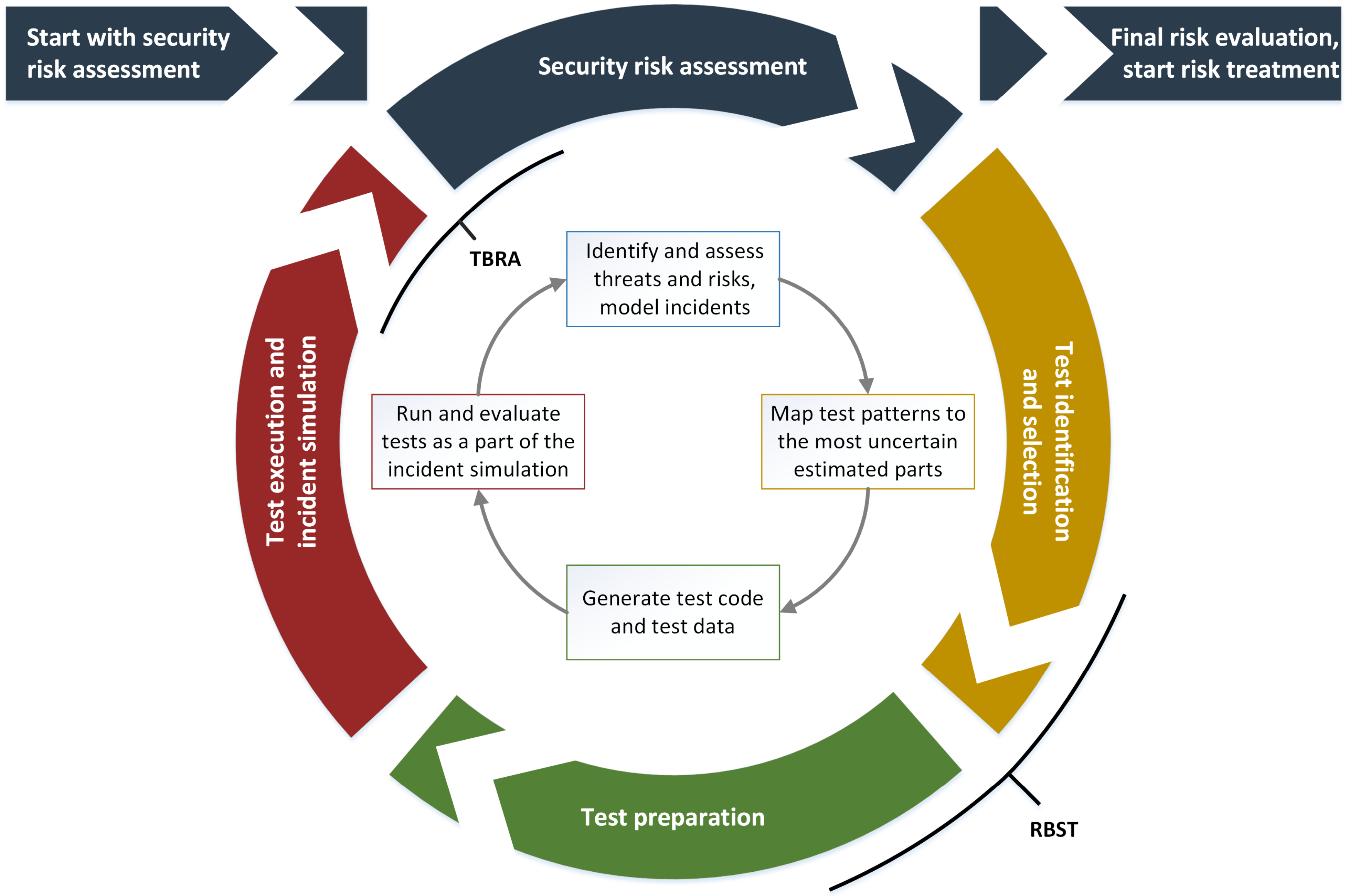
4. Execution

How to stimulate and observe? Where to stimulate and observe?

5. Feedback

What do the test results mean for the overall risk picture?

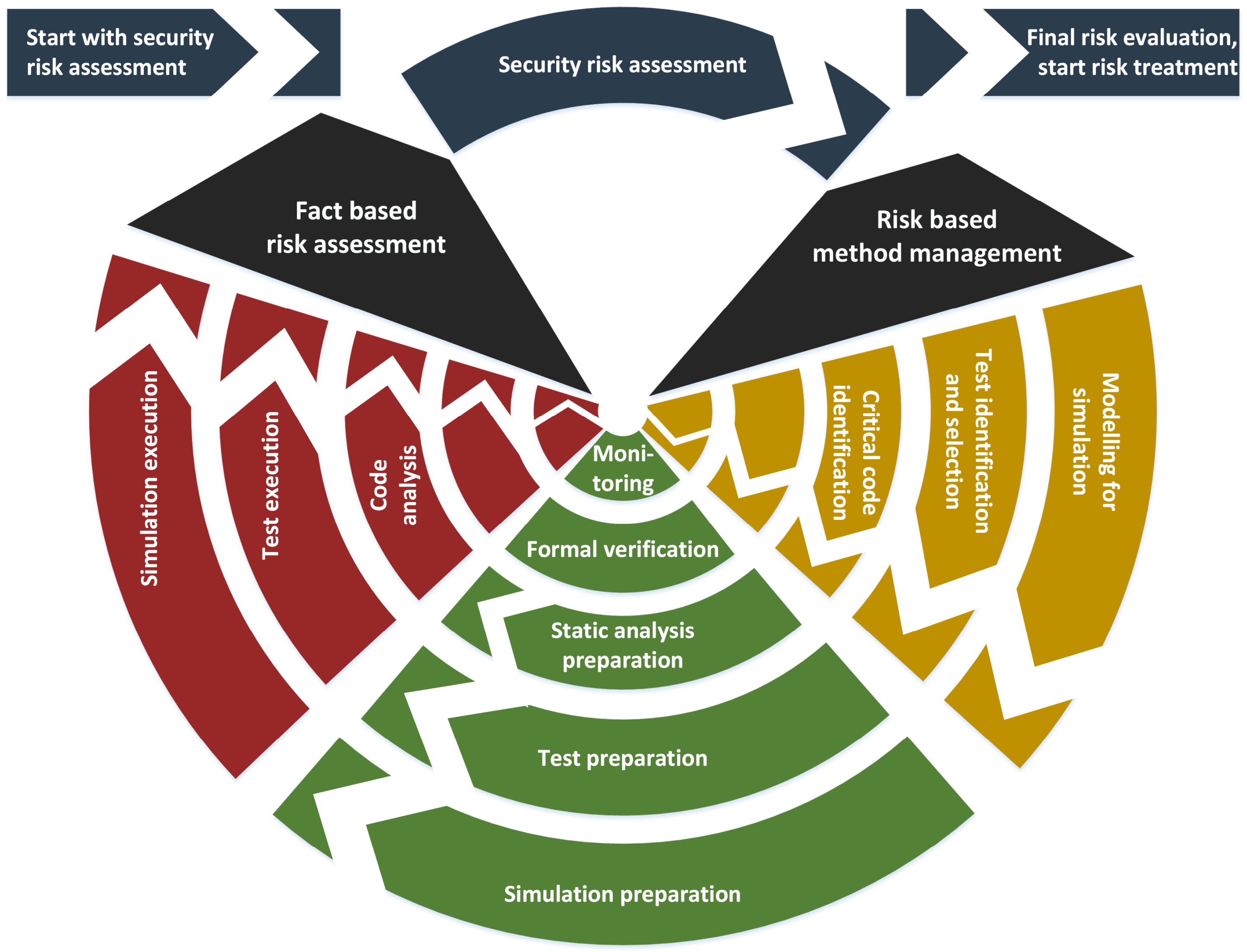
## Problems and Challenges – Iterative Risk Assessment Process



Start with security risk assessment
Security risk assessment
Final risk evaluation, start risk treatment
TBRA
Identify and assess threats and risks, model incidents
Test identification and selection
Map test patterns to the most uncertain estimated parts
Test execution and incident simulation
Run and evaluate tests as a part of the incident simulation
Generate test code and test data
Test preparation
RBST

Fraunhofer FOKUS

## Problems and Challenges – Iterative Risk Assessment Process


Start with security risk assessment
Security risk assessment
Final risk evaluation, start risk treatment
Fact based risk assessment
Risk based method management
Simulation execution
Test execution
Code analysis
Moni-toring
Formal verification
Static analysis preparation
Test preparation
Simulation preparation
Critical code identification
Test identification and selection
Modelling for simulation

## Problems and Challenges – The Case Study

Software AG wishes:

- Get a realistic picture of the overall risks associated with Command Central and the other Software AG products
  - Command Central is used in many different contexts for managing various systems
  - There should not be an expensive complete risks assessment required for each scenario
- Manual analysis methods are generally regarded to be not feasible
  - Software AG products are complex
  - There is only a limited budget

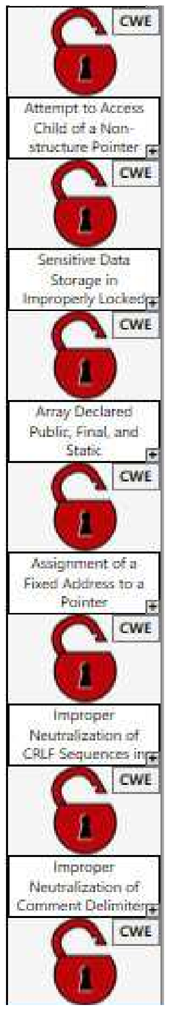
➢ As much automation and reusability as possible!

Fraunhofer FOKUS

## Initial Risk Assessment

Manual high level analysis

- Establish the context
- Identify risks
  - Joint workshop of the Command Central product development team and security experts
  - Product under investigation and potential vulnerabilities modelled in ARIS tool
  - Used the existing Mitre CWE database
- Results:
  - Long lists of weaknesses for about 30 components
    - Not analyzed if the weaknesses actually exist
    - Not investigated how likely it is that the existing ones would actually be exploited or what the consequences might be


CWE
Attempt to Access Child of a Non-structure Pointer
CWE
Sensitive Data Storage in Improperly Locked
CWE
Array Declared Public, Final, and Static
CWE
Assignment of a Fixed Address to a Pointer
CWE
Improper Neutralization of CRLF Sequences in
CWE
Improper Neutralization of Comment Delimiter
CWE

## Refining the Initial Risk Picture

The initial risk identification contains not enough information to enable automated testing:

- Requires a low level risk assessment
  - Connection between risk analysis artefacts and system components
    - Where to stimulate?
    - Where to observe?
  - ➢ Create a model that has both system information and risk information
  - ➢ Lots of manual work to create such a model?

We decided to develop a tool for this step and the entire combined TBRA and RBST process in order to keep the manual effort as low as possible:

- The RACOMAT tool
  - Stand alone application
  - Also Visual Studio plug-in

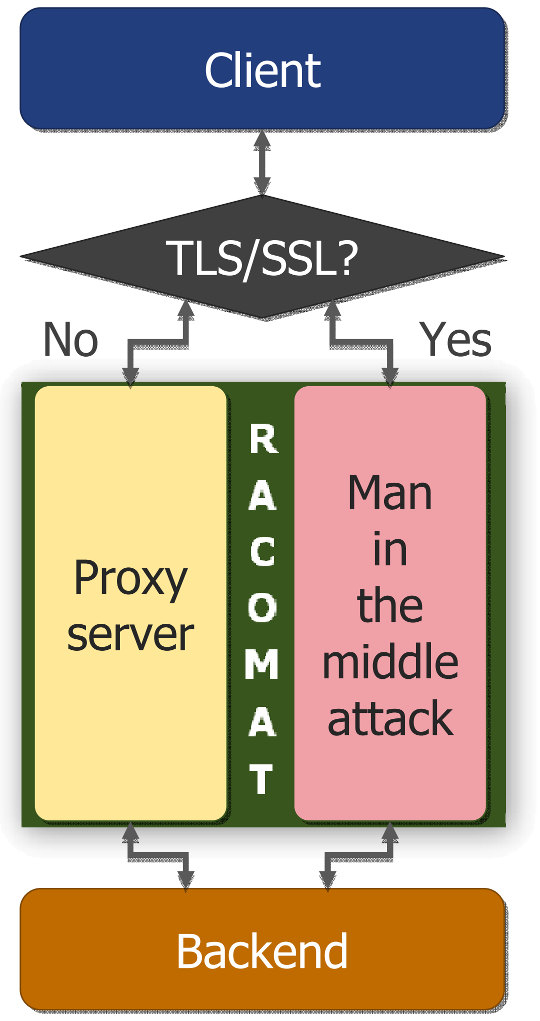
## Refining the Initial Risk Picture with RACOMAT

Generate system models with low level risk information

- Automated static analysis of components
  - Generate models for testable interfaces
    - HTML pages, source code, compiled libraries or programs …
  - Threat interfaces with input and output ports
    - Suggests typically related risk artefacts (e.g. vulnerabilities) for the identified interfaces

- For Command Central static analysis fails
  - Web interface has lots of scripts – hard to parse
  - The user interfaces are generated dynamically based on the session state

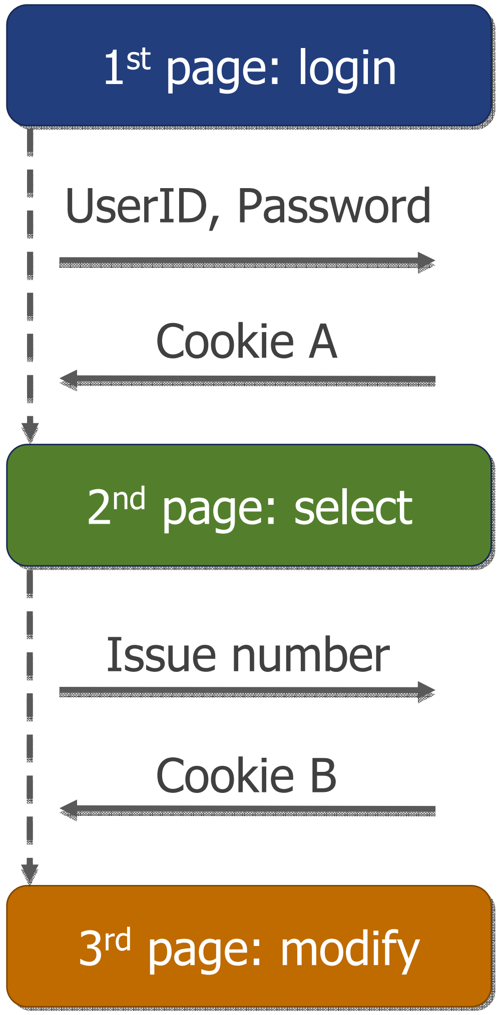
➢ Dynamical interface analysis required
  - Observe network traffic while the system is used and generate thread interface models
  - Semi automated



## Refining the Initial Risk Picture with RACOMAT

What RACOMAT dynamic analysis does

- Analyzes data exchange
  - Authentication
  - Cookies
  - Parameters (Url, multipart, JSON, SOAP …)
- Generates state dependent threat interface models
  - Input / output ports
    - Type information, Values
  - Suggests lists of typically related weaknesses for each port
    - From CWE database and type information
    - From initial risk assessment
- Models relations between threat interfaces
  - How to get to a certain state
    - e.g. authenticate, set cookie

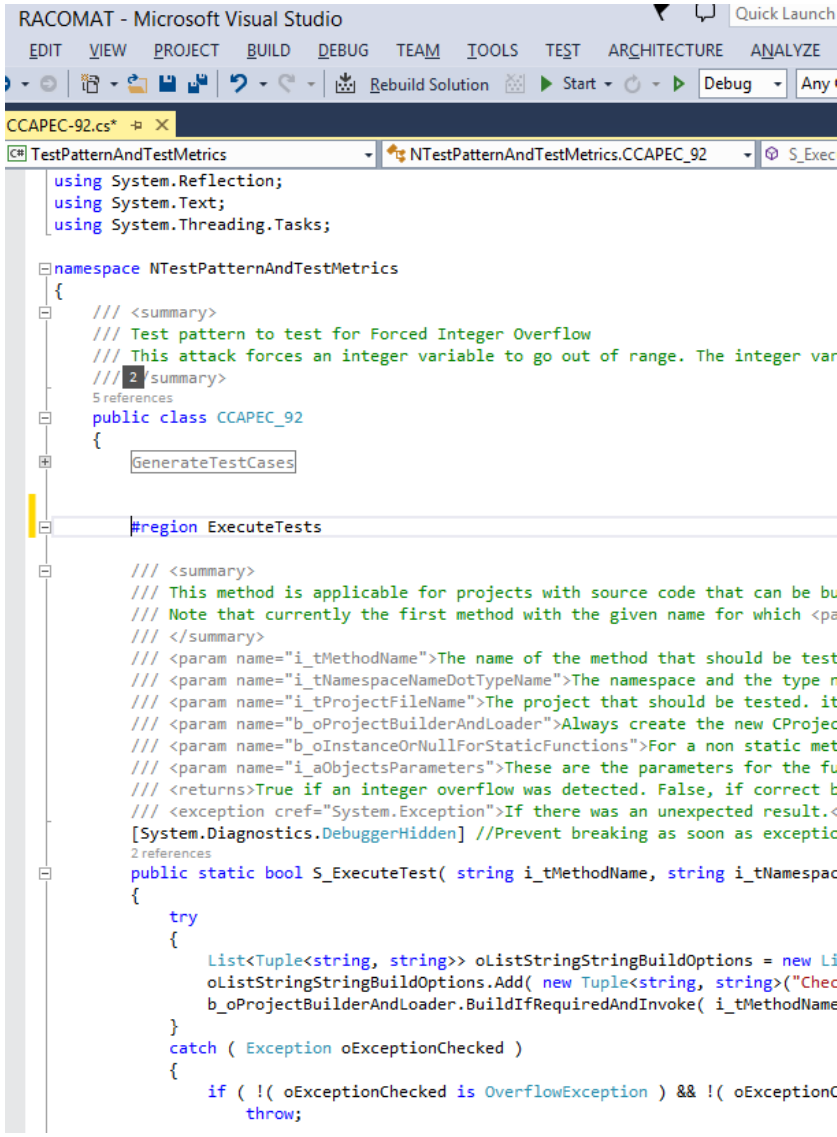


Fraunhofer FOKUS

# Automated Risk-Based Security Testing

Basic ideas

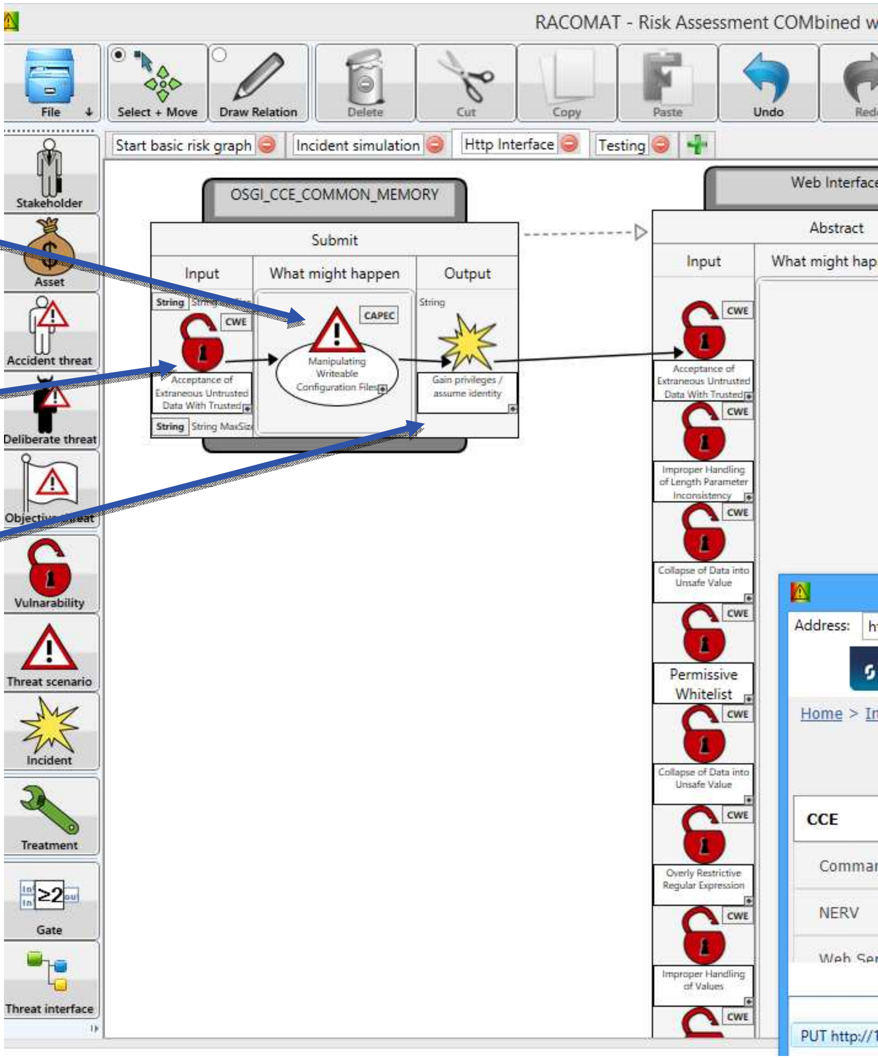
- Security testing means attacking the SUT
  - ➢ Attack patterns describe exactly, how such an attack could be made
  - ➢ CWE weaknesses contain links to typically related CAPEC attack patterns
    - ➢ Add CAPEC attack patterns to the system and risk model
- Problem: Attack patterns are designed for human beings
  - – Implementing them requires a lot of manual work
- ➢ Introduce reusable Security Test Patterns
  - ➢ Machine interpretable, executable
  - ➢ Attached to CAPEC attack patterns for minimal instantiation effort


RACOMAT - Microsoft Visual Studio
Quick Launch
EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE
Rebuild Solution
Start
Debug
Any
CCAPEC-92.cs*
TestPatternAndTestMetrics
NTestPatternAndTestMetrics.CCAPEC_92
S_Exec

```
using System.Reflection;
using System.Text;
using System.Threading.Tasks;

namespace NTestPatternAndTestMetrics
{
    /// <summary>
    /// Test pattern to test for Forced Integer Overflow
    /// This attack forces an integer variable to go out of range. The integer var
    /// 2 /summary>
    5 references
    public class CCAPEC_92
    {
        GenerateTestCases


        #region ExecuteTests

        /// <summary>
        /// This method is applicable for projects with source code that can be bu
        /// Note that currently the first method with the given name for which <pa
        /// </summary>
        /// <param name="i_tMethodName">The name of the method that should be test
        /// <param name="i_tNamespaceNameDotTypeName">The namespace and the type n
        /// <param name="i_tProjectFileName">The project that should be tested. it
        /// <param name="b_oProjectBuilderAndLoader">Always create the new CProjec
        /// <param name="b_oInstanceOrNullForStaticFunctions">For a non static met
        /// <param name="i_aObjectsParameters">These are the parameters for the fu
        /// <returns>True if an integer overflow was detected. False, if correct b
        /// <exception cref="System.Exception">If there was an unexpected result.<
        [System.Diagnostics.DebuggerHidden] //Prevent breaking as soon as exceptic
        2 references
        public static bool S_ExecuteTest( string i_tMethodName, string i_tNamespac
        {
            try
            {
                List<Tuple<string, string>> oListStringStringBuildOptions = new Li
                oListStringStringBuildOptions.Add( new Tuple<string, string>("Chec
                b_oProjectBuilderAndLoader.BuildIfRequiredAndInvoke( i_tMethodName
            }
            catch ( Exception oExceptionChecked )
            {
                if ( !( oExceptionChecked is OverflowException ) && !( oExceptionC
                    throw;
```

## Automated Risk-Based Security Testing with RACOMAT

- RACOMAT uses the combined system and risk model to instantiate test patterns
  - Attack patterns indicate which test patterns should be used
    - Priority of tests can be calculated based on likelihood and consequence values
  - Vulnerabilities indicate where to stimulate the SUT
  - Unwanted Incidents can be introduced in order to determine what should be observed to get some verdict
  - ➢ Complete automation often achievable
- Problem: There are only a few test pattern available
  - Implementing generic reusable test pattern is challenging
  - Currently not really saving manual effort
  - ➢ Vision: create an open security test pattern library



# Test-Based Risk Assessment

There are basically two types of test based updates to the risk model

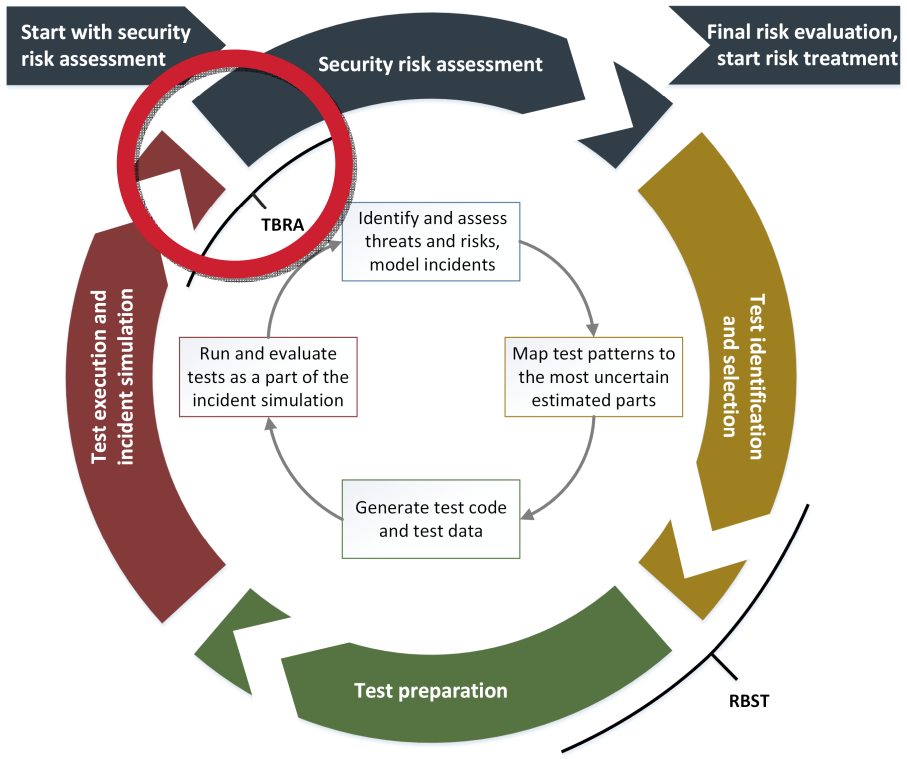
•Introduce new unwanted incidents and vulnerabilities discovered while testing

•Update likelihood values based on test results

- ➢ Use security testing metrics

➢RACOMAT supports both in a semi-automated fashion

- – Problem: how to deal with test results that did not find any unwanted incidents
- – Problem: There are only a few good security testing metrics available at the moment



Start with security risk assessment
Security risk assessment
Final risk evaluation, start risk treatment
TBRA
Identify and assess threats and risks, model incidents
Run and evaluate tests as a part of the incident simulation
Map test patterns to the most uncertain estimated parts
Generate test code and test data
Test execution and incident simulation
Test identification and selection
Test preparation
RBST

## High Level Composition

Refining the risk picture and testing produce detailed risk models

•Required to get more objective picture, but too much information

•For risk management, typically more high level results are wanted

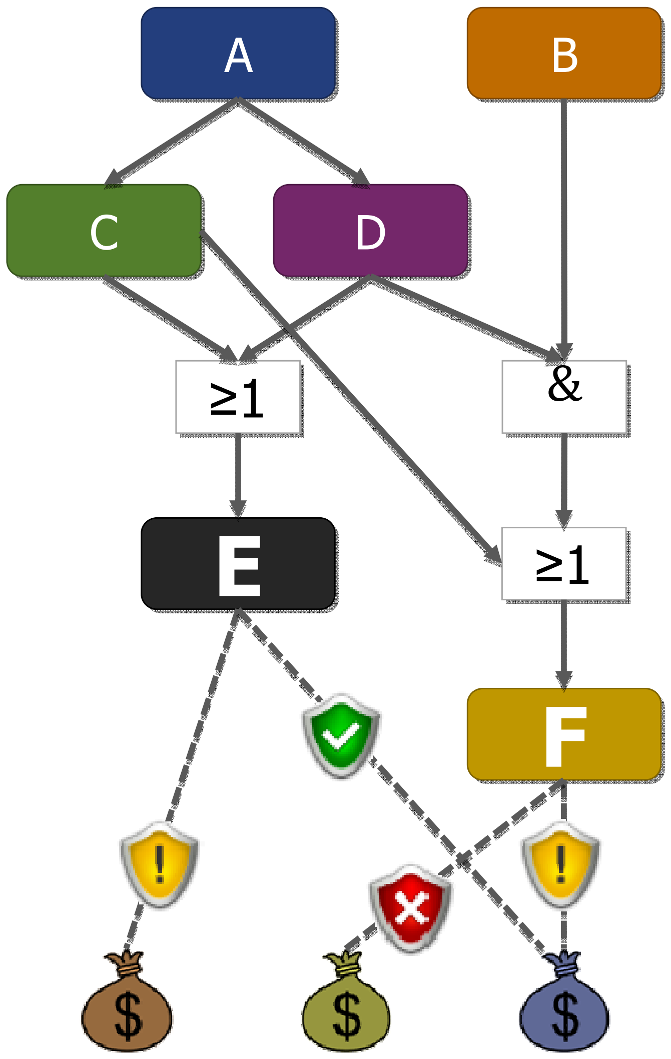
•The same components and systems may be used in different scenarios and contexts

➢ Aggregate risk analysis results

- ➢ RACOMAT uses simulations to calculate high level risk values

➢ Model the different contexts

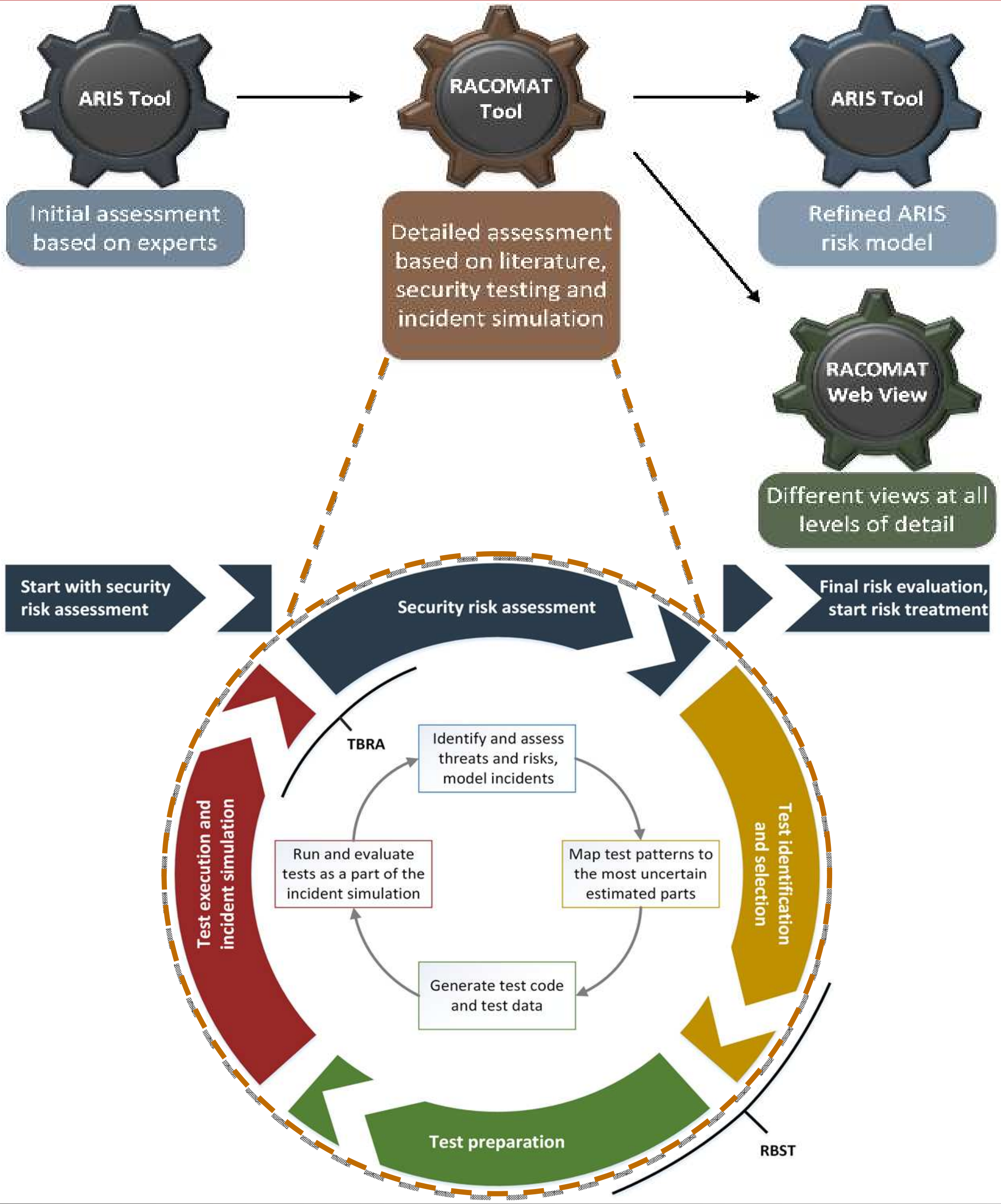
- ➢ Use CVE vulnerabilities database for common software components
- ➢ Do compositional risk assessment
  - Requires manual modelling of relations?



## Case Study Workflow

Final results are exported from the RACOMAT tool in two formats:

- ARIS exchange format (JSON) at the same level of detail which the initial risk assessment provided
- XHTML format at different level of details
  - Risk graphs
  - Test results
  - Dashboards (basically intended to support management decisions)

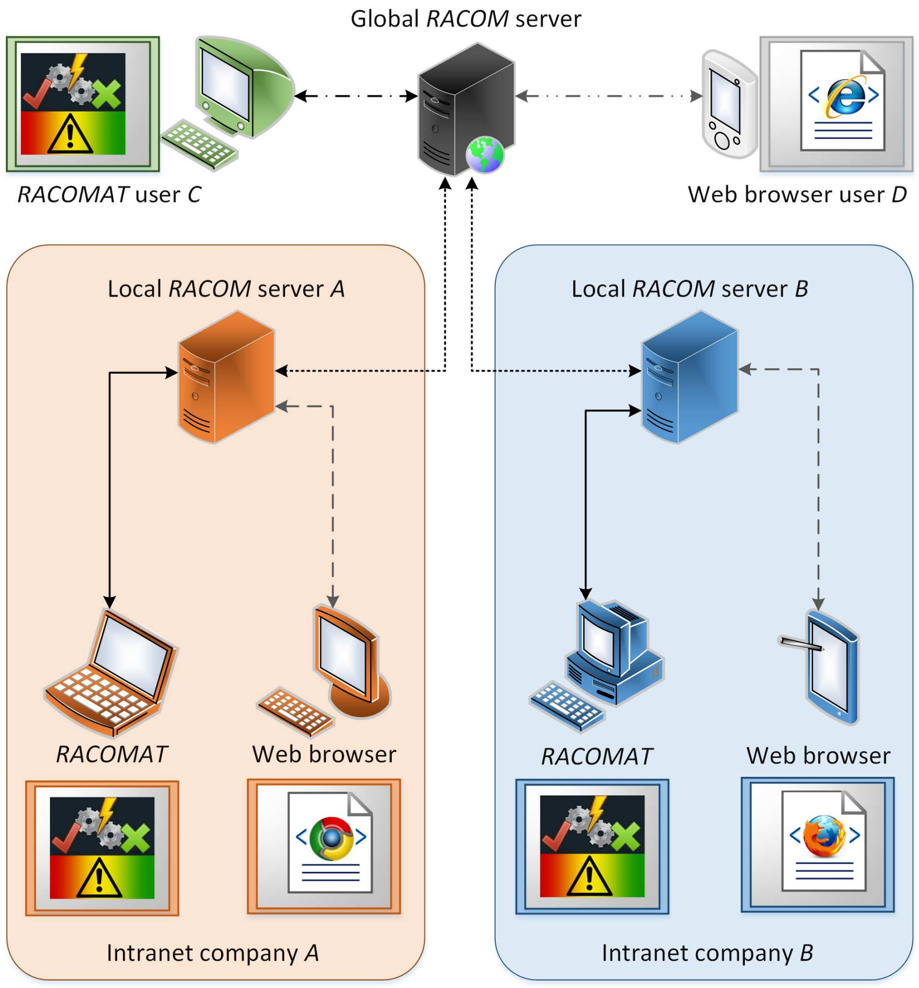

ARIS Tool
Initial assessment based on experts
RACOMAT Tool
Detailed assessment based on literature, security testing and incident simulation
ARIS Tool
Refined ARIS risk model
RACOMAT Web View
Different views at all levels of detail
Start with security risk assessment
Final risk evaluation, start risk treatment
Security risk assessment
TBRA
Identify and assess threats and risks, model incidents
Map test patterns to the most uncertain estimated parts
Generate test code and test data
Run and evaluate tests as a part of the incident simulation
Test identification and selection
Test preparation
Test execution and incident simulation
RBST

## Conclusion and Future Work

Observations

- Combined risk and system models are a good base for automated security testing
  - Creating such models does not require much manual work
  - Automation highly depends und good reusable existing artefacts
  - Problem: No adequate databases of test pattern and testing metrics available

- Future work
  - Complete the Software AG case study within the next five months
  - Development of RACOM server
    - Sharing test patterns, testing metrics
    - Sharing reusable threat interfaces for entire components or programs



## Questions, Remarks?

Thanks a lot for the attention!

Johannes Viehmann 2015

## Contact

**Fraunhofer Institute for Open Communication Systems FOKUS**

Kaiserin-Augusta-Allee 31
10589 Berlin, Germany

www.fokus.fraunhofer.de

Johannes Viehmann
Researcher
johannes.viehmann@fokus.fraunhofer.de

**System Quality Center SQC**

http://s.fhg.de/sqc

Dr. Tom Ritter
Head of competence center SQC
tom.ritter@fokus.fraunhofer.de

Friedrich Schön
Head of competence center SQC
friedrich.schoen@fokus.fraunhofer.de

Fraunhofer
FOKUS