



Title: Review of security testing tools

Version: 1.1
Date : 27.6.2011
Pages : 100

Author: Ilkka Uusitalo (VTT)

**Reviewers: Fredrik Seehusen, Michel Bourdelles,
 Jürgen Großmann/Florian Marienfeld**

To: DIAMONDS Consortium

The DIAMONDS Consortium consists of:

Codenomicon, Conformiq, Dornier Consulting, Ericsson, Fraunhofer FOKUS, FSCOM, Gemalto, Get IT, Giesecke & Devrient, Grenoble INP,itrust, Metso, Montimage, Norse Solutions, SINTEF, Smartesting, Secure Business Applications, Testing Technologies, Thales, TU Graz, University Oulu, VTT

Status:

[] Draft
 [] To be reviewed
 [] Proposal
 [X] Final / Released

Confidentiality:

[X] Public Intended for public use
 [] Restricted Intended for DIAMONDS consortium only
 [] Confidential Intended for individual partner only

Deliverable ID: D1_1

Title:

Review of Security Testing Tools

Summary / Contents:

Contributors:

Juha Matti Tirila, Tuomo Untinen, Rauli Kaksonen, Ari Takanen, Ami Juuso, Miia Vuontisjarvi (Codenomicon)

Bruno Legeard, Fabrice Bouquet, Julien Botella, Dooley Nsewolo Lukula (Smartesting)

Ilkka Uusitalo, Matti Mantere (VTT)

Peter Schmitting (FSCOM)

Stephan Schulz (Conformiq)

Ina Schieferdecker, Florian Marienfeld, Andreas Hinnerichs (Fraunhofer FOKUS)

Pekka Pietikäinen (OUSPG)

Wissam Mallouli, Gerardo Morales (Montimage)

Fredrik Seehusen (SINTEF)

Wolfgang Schlicker (Dornier Consulting)



	<p style="text-align: center;">Review of security testing tools</p> <p style="text-align: center;">Deliverable ID: D1_1</p>	<p>Page : 2 of 100</p> <hr/> <p>Version: 1.1 Date : 27.6.2011</p> <hr/> <p>Status : Final Confid : Public</p>
---	---	---

TABLE OF CONTENTS

1. Introduction	7
2. Behavioral MBT for security testing	7
2.1 Behavioral MBT - an introduction.....	7
2.2 Test Design with MBT	8
2.2.3 Automated Test Design with MBT in Standardization	12
2.3 Modeling For automated test generation	12
2.3.1 Modelling of Risk.....	13
2.3.2 Modelling of Functionality.....	13
2.3.3 Modelling of Security Aspects	25
2.3.4 Fokus!MBT.....	27
3. Extend test coverage using security-oriented test purposes	31
3.1 Conformiq approach for testing security properties.....	32
3.2 ETSI approach to security testbeds specific to IPv6 security testing	32
3.2.1 Organization of the work	33
3.2.2 Summary	38
4. Random, Block-based and Model-based fuzzing	39
5. Network Scanning	44
5.1 port scanners.....	45
6. Monitoring tools for detecting vulnerabilities	46
6.1 Intrusion detection systems	46
6.1.1 Network Based Intrusion Detection Systems.....	46
6.1.2 Host Based Intrusion Detection Systems	47
6.1.3 Scalability.....	47
6.1.4 Challenges.....	47
6.1.5 Examples of Current Intrusion Detection Systems.....	48
6.2 Network monitoring tools	51
6.2.1 Wireshark.....	52
6.2.2 OpenNMS	53
6.2.3 OmniPeek	54
6.2.4 Clarified Analyzer.....	54
6.2.5 Tcpxtract	55
6.3 Business Activity Monitoring	56
6.3.1 IBM Business Monitor	56
6.3.2 Oracle Business Activity Monitoring	57
6.4 database Activity Monitoring	58
6.4.1 IBM InfoSphere Guardium.....	58
6.4.2 dbWatch.....	60
6.4.3 DB Audit 4.2.29.....	61
6.5 Firewalls, Spam and Virus detection tool	63
6.5.1 Firewalls.....	63
6.5.2 Virus detection	65
6.5.3 Spam Detection and Filtering	70
7. Diagnosis and root-cause-analysis tools	73
7.1 Diagnosis tools for security testing	73
7.1.1 RCAT	73

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 3 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

7.1.2	XFRACAS	74
7.2	Intrusion prevention systems	76
7.2.1	Cisco intrusion prevention system	76
8.	Tool integration platforms	77
8.1	MODELBUS	78
8.2	JAZZ	80
8.3	Connected Data Objects – CDO	81
8.4	EMF STORE	82
9.	Risk analysis and modeling tools	83
9.1	Microsoft THREAT MODELING	85
9.2	the coras tool	86
9.3	CRAMM - CCTA Risk Analysis and Management Method and Tool	88
9.4	MotOrbac	90
9.5	GOAT	92
9.5.1	VDC editor plugin for GOAT	93
9.5.2	TSM editor plugin for GOAT	94
9.6	SeaMonster	96
10.	References	97

FIGURES

Figure 1.	The Test Design process with MBT.	8
Figure 2.	Relationship between both repositories (tests and requirements)	10
Figure 3.	Main roles in the MBT process	11
Figure 4.	Model-Based Test Development [20]	12
Figure 5.	CORAS Tool Snapshot	13
Figure 6	UML state chart with QML action language	14
Figure 7.	BPMN Example	15
Figure 8.	Example UML statechart diagram	16
Figure 9.	Example Object diagram	17
Figure 10.	UTP example of a test component definition	19
Figure 11.	ADML and XAML Model infrastucture	20
Figure 12.	System model example	21
Figure 13.	Behavioural system model example	21
Figure 14.	Test Case Designer	22
Figure 15.	Model-editing in Defensics	24
Figure 16.	Sample UMLsec Deployment Diagram	26
Figure 17.	UMLsec stereotypes (excerpt) [4]	26
Figure 18.	secureUML meta model	27
Figure 18.	The FOKUS!MBT modelling and test generation approach	28
Figure 19.	The FOKUS!MBT infrastructure	29
Figure 20.	Conceptual Overview of TestingMM	30
Figure 21.	Smartesting approach based on security test schemas	Error! Bookmark not defined.
Figure 22.	Card status	Error! Bookmark not defined.
Figure 23.	Specification-based approach	41
Figure 24.	Test-case generation	42



	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 4 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

Figure 25. Test target information	43
Figure 26. Step 1. Load PCAP file	44
Figure 27. Step 2. Select protocol elements	44
Figure 28. SNORT IPS Console.....	49
Figure 29. OpenNMS.....	54
Figure 30. IBM Business Monitor configuration.....	57
Figure 31. Oracle BAM	58
Figure 32. IBM InfoSphere Guardium user interface	59
Figure 33. dbWatch Architecture	61
Figure 34. DB audit user interface.....	62
Figure 35. Comodo Firewall	64
Figure 36. BitDefender Dashboard.....	67
Figure 37. Antivirus Configuration	67
Figure 38. Norton AntiVirus Scan.....	68
Figure 39. Norton Insight Network.....	69
Figure 40. Kaspersky Antivirus.....	70
Figure 41. SPAMfighter Pro Screenshot	72
Figure 42. XFRACAS user interface	75
Figure 43. Cisco IPS 4270 Sensor	77
Figure 44. Model Bus	Error! Bookmark not defined.
Figure 45. Model Bus process.....	Error! Bookmark not defined.
Figure 46. Jazz tool integration	Error! Bookmark not defined.
Figure 47. Connected data objects	Error! Bookmark not defined.
Figure 48. EMF Store	Error! Bookmark not defined.
Figure 49. Test Purpose in TPLan	Error! Bookmark not defined.
Figure 50. IPv6 Test Method	Error! Bookmark not defined.
Figure 51. Tunnel Mode	Error! Bookmark not defined.
Figure 52. Transport Mode.....	Error! Bookmark not defined.
Figure 53. Testbed architecture	Error! Bookmark not defined.
Figure 54. Screenshot of SDL Threat Modeling Tool [52].....	85
Figure 55. Screenshot of the CORAS tool	87
Figure 56. Screenshot of CRAMM [53]	89
Figure 57. The Or-BAC model.....	91
Figure 58. MotOrBAC user interface	92
Figure 59. GOAT user interface	93
Figure 60. Vulnerability detection condition for “Use of tainted value to malloc” in GOAT	94
Figure 61. Graphical representation of IP blocking security rule.....	96
Figure 62. SeaMonster screen capture	97

TABLES

Table 3.1: Language key words	Error! Bookmark not defined.
Figure 22 Card status	Error! Bookmark not defined.


	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 5 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

HISTORY

Vers.	Date	Author	Description
0.1	4/2011	Ilkka Uusitalo	Template created
0.2	4/2011	Ilkka Uusitalo	First inputs collected
0.3	5/2011	Ilkka Uusitalo	Second round of inputs collected, first integration
0.4	18.5.2011	Ilkka Uusitalo	First integration for review

APPLICABLE DOCUMENT LIST

Ref.	Title, author, source, date, status	DIAMONDS ID
1		


	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 6 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

EXECUTIVE SUMMARY

The DIAMONDS project focuses on model-based security testing of networked systems to validate their dependability in face of malice, attack, error or mishap. Testing is the main method to reliably check the functionality, robustness, performance, scalability, reliability and resilience of systems as it is the only method to derive objectively characteristics of a system in its target environment. In this document we discuss the state-of-the art in security testing tools.

Model- based Testing is the approach of deriving systematic tests on a system based on the formal description of system information. These models may describe the behaviour of the system, security constraints (for example access control), the security requirements, or information about possible security threats, faults or attacks. In chapter 2 we give an introduction to behavioral model-based testing tools, an umbrella of approaches that make use of models in context of testing. The Fokus!MBT tool is discussed in more detail in chapter 2. After this, in chapter 3, Smartesting and Conformiq describe their approaches for testing security properties.

In Section 4 both open-source and commercial random, block-based and model-based fuzzing tools are described. Sections 5 discusses network scanning while Section 6 focuses on tools for detecting vulnerabilities, such as IDS/IPS and network monitoring. Section 7 analyses diagnosis and root-cause analysis tools, and Section 8 is all about tool integration platforms. We conclude the document with a discussion on risk analysis and modeling tools in Section 9. of this document describe different models used in security testing.

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 7 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

1. INTRODUCTION

This document is a state of the art review on security testing tools for the Diamonds project. The goal of the document is to survey existing security tools from the project's point-of-view, that is, the emphasis is on model-based security testing tools.

We begin with an introduction to behavioral model-based testing, an umbrella of approaches that make use of models in context of testing. After this, Smartesting and Conformiq describe their approaches for testing security properties.

In Section 4 both open-source and commercial random, block-based and model-based fuzzing tools are described. Section 5 discusses network scanning while Section 6 focuses on tools for detecting vulnerabilities, such as IDS/IPS and network monitoring. Section 7 analyses diagnosis and root-cause analysis tools, and Section 8 is all about tool integration platforms. We conclude the document with a discussion on risk analysis and modeling tools in Section 9.

Static Application Security Testing (SAST) tools are out of the scope of this document.

2. BEHAVIORAL MBT FOR SECURITY TESTING

Testing of security aspects is heavily tied to the operation or behaviour exhibited by a system to be tested. Therefore – especially in the commercial model-based testing (MBT) tool landscape – MBT tools used for functional testing are the driver also for testing security aspects. The following sections provide a brief introduction in current MBT approaches.


2.1 BEHAVIORAL MBT - AN INTRODUCTION

Model-based testing (MBT) is an umbrella of approaches that makes use of models in the context of testing. In the area of test design, model-based testing tools for assessing system behaviour in functional testing can be categorized into the 3 main categories [1]:

- Test Data Generators – generate test data for the creation of logical and/or abstract test cases
- Test Case Editors – tools, which, based on an abstract model of a test-case, creates one or more test-cases for manual execution or test scripts for automated execution
- Test Case Generators - tools, which, automatically, create test-cases, test scripts or, even, complete test suites, using configurable coverage criteria, based on a model of the system behaviour, the system environment or of tests and specific control information

There are several reasons for the growing interest in using model-based testing:

- The complexity of software applications continues to increase, and the user's aversion to software defects is greater than ever, so our functional testing has to become more and more effective at detecting bugs;
- The cost and time of testing is already a major proportion of many projects (sometimes exceeding the costs of development), so there is a strong push to investigate methods like MBT that can decrease the overall cost of test by designing as well as executing tests automatically;

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 8 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

- The MBT approach and the associated tools are now mature enough to be applied in many application areas, and empirical evidence is showing that they can give a good ROI.

Model-based testing renews the whole process of functional software testing: from requirements to the test repository, with manual or automated test execution. It supports the phases of designing and generating tests, documenting the test repository, producing and maintaining the bi-directional traceability matrix between tests and requirements, and accelerating test automation.

This section addresses these points by giving a realistic overview of model-based testing and its expected benefits. It discusses **what** model-based testing is, **how** you have to organize a process and a team to use MBT, and **which** benefits it may expect from this software testing approach.

2.2 TEST DESIGN WITH MBT

Test design with model-based testing refers to the processes and techniques for the automatic derivation of abstract test cases from abstract formal models, via the generation of concrete tests from abstract tests, and to the manual or automated execution of the resulting concrete test cases. Therefore, the key points of automated test design with model-based testing are the modeling principles for test generation, the test generation strategies and techniques, and the implementation of abstract tests into concrete, executable tests. A typical deployment of MBT in industry goes through the four stages shown in Figure 1:

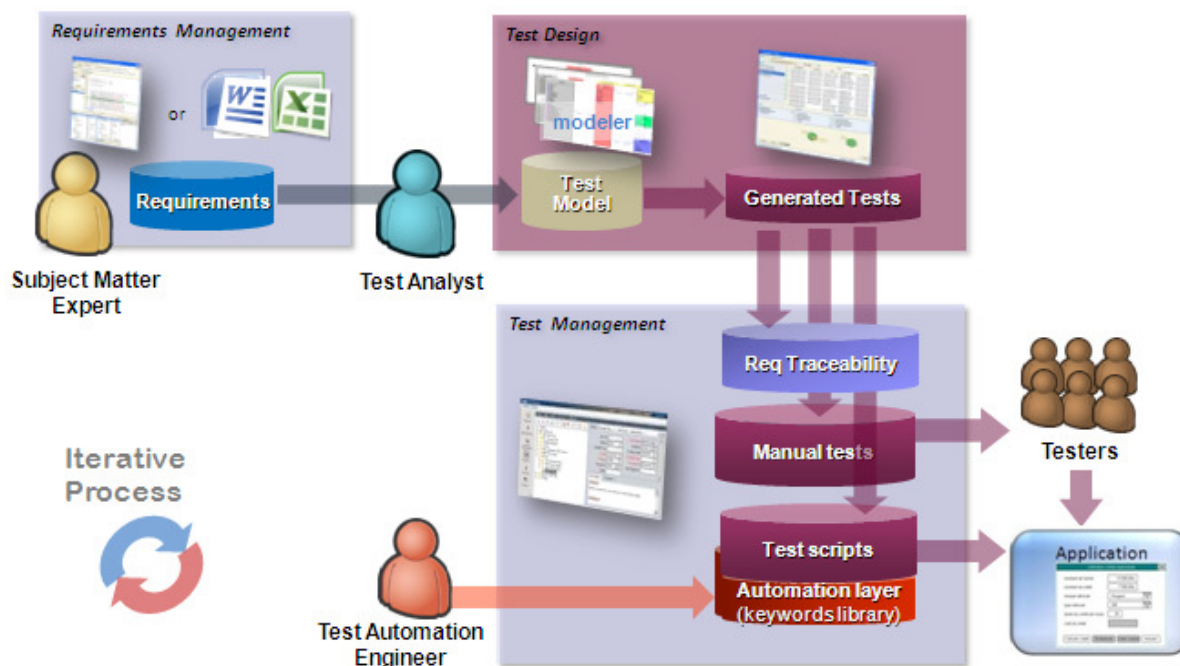




Figure 1. The Test Design process with MBT.

1. **Design a functional test model.** The model, sometimes called the *test model*, represents the expected operational behavior of the system under test (SUT) or the system environment or usage. The choice must be done by the controls and observations elements provided by the SUT. Standard modeling languages such as UML can be used to formalize the control points and observation points of the system, the expected dynamic behavior of the system, the entities associated with the test, and some data for the initial test configuration. Model elements such as transitions or decisions are linked to the requirements, in order to

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 9 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

ensure bi-directional traceability between the requirements and the model, and later to the generated test cases. Models must be precise and complete enough to allow automated derivation of tests from these models;

2. **Select some test generation criteria.** There are usually an infinite number of possible tests that could be generated from a model, so the test analyst chooses some test selection criteria to select the highest-priority tests, or to ensure good coverage of the system behaviors. One common kind of test selection criteria is based on structural model coverage, using well known test design strategies such as equivalence partitioning, cause-effect testing, pairwise testing, process cycle coverage, or boundary value analysis (see [1] for more details on these strategies). Another useful kind of test generation criteria ensures that the generated test cases cover all the requirements, possibly with more tests generated for requirements that have a higher level of risk. In this way, model-based testing can be used to implement a requirement and risk-based testing approach. For example, for a non-critical application, the test analyst may choose to generate just one test for each of the nominal behaviors in the model and each of the main error cases; but for one of the more critical requirements, she/he could apply more demanding coverage criteria to ensure that this part of the test model is more thoroughly tested;
3. **Generate the tests.** This is a fully automated process that generates the required number of (abstract) test cases from the test model. Each generated test case is typically a sequence of high-level SUT actions, with input parameters and expected output values for each action. These generated test sequences are similar to the high-level test sequences that would be designed manually in action-word testing [2]. They are easily understood by humans and are complete enough to be directly executed on the SUT by a manual tester. The test model allows computing the expected results and the input parameters. Data tables may be used to link some abstract value from the model with some concrete test value. To make them executable using a test automation tool, a further concretization phase automatically translates each abstract test case into a concrete (executable) script [3], using a user-defined mapping from abstract data values to concrete SUT values, and a mapping from abstract operations into test adaptation API calls. For example, if the test execution is via the GUI of the SUT, then the action words are linked to the graphical object map using a test robot – in this case the test adaptation. If the test execution of the SUT is API-based, then the action words need to be implemented on this API. This can be a direct mapping or a more complex adaptation layer. The expected results part of each abstract test case is translated into oracle code that will check the SUT outputs and decides automatically on a test pass/fail verdict. The tests generated from the test model may be structured into multiple test suites, and published into standard test management tools. Maintenance of the test repository is done by updating the test model, then automatically regenerating and republishing the test suites into the test management tool;
4. **Execute the tests.** The generated concrete tests are typically executed either manually or within a standard automated test execution environment. Either way, the result is that the tests are executed on the SUT, and we find that some tests pass and some tests fail. The failing tests indicate a discrepancy between the SUT and the expected results designed in the test model, which then needs to be investigated to decide whether the failure is caused by a bug in the SUT, or by an error in the model and/or the requirements. Experience shows that model-based testing is good at finding SUT errors, but is also highly effective at exposing requirements errors [1], even way before executing a single test (thanks to the modeling phase).

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 10 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

2.2.1.1 Requirements traceability

The automation of bidirectional traceability between requirements and test cases is a key aspect of the added value of MBT. *Bidirectional traceability* is the ability to trace links between two parts of the software development process with respect to each other. The starting points of the MBT process are, as usual, the informal functional requirements, use cases, descriptions of business processes and all other factors that provide the functional description of the application being tested. To be effective, requirements traceability implies that the requirements repository should be structured enough so that each individual requirement can be uniquely identified. It is desirable to link these informal requirements to the generated tests, and to link each generated test to the requirements that it tests.

A best practice in MBT, supported by most of the tools on the market, consists of linking model elements such as decision points and transitions to the relevant requirements. From these links in the test model, test generation tools ensure the automatic generation and maintenance of the traceability matrix between requirements and test cases.

2.2.1.2 Test repository and test management tools

The purpose of generating tests from the test model is to produce the test repository. This test repository is typically managed by a test management tool. The goal of such a tool is to help organize and execute test suites (groups of test cases), both for manual and automated tests.

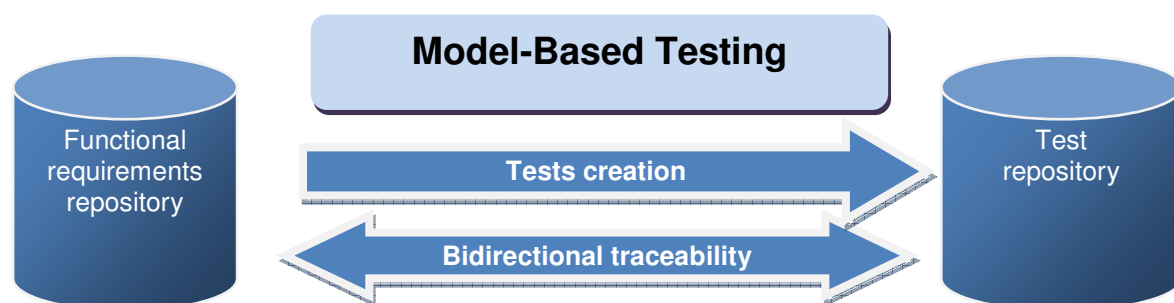



Figure 2. Relationship between both repositories (tests and requirements).

In the MBT process, the test repository documentation is fully managed by automated generation (from the test model): documentation of the test design steps, requirements traceability links, test scripts and associated documentation are automatically provided for each test case. Therefore, the maintenance of the test repository needs to be done in the test model.

2.2.1.3 Roles in the test design process with MBT

One way to define processes for test design with MBT is to think of it as involving three main roles (see Figure 3).

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 11 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

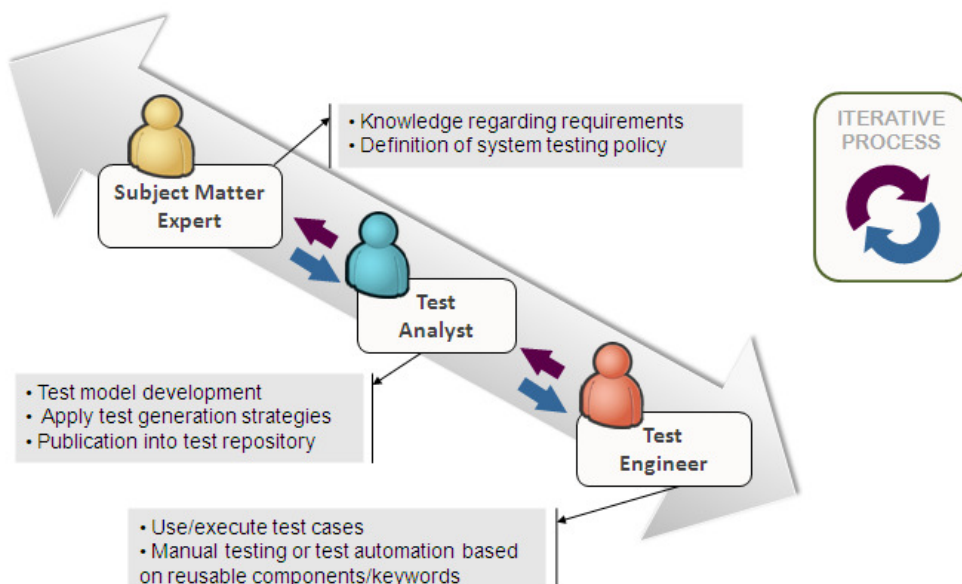


Figure 3. Main roles in the MBT process.

1. The **Test Analyst** interacts with the customers and subject matter experts regarding the requirements to be covered, and then develops the test model. He/she then uses the test generation tool to automatically generate tests and produce a repository of test suites that will satisfy the project test objectives.
2. The **Subject Matter Expert** is the reference person for the SUT requirements and business needs, and communicates with the test analyst to clarify the specifications and testing needs.
3. The **Test Engineer** may be a tester for manual test execution or a test automation engineer, who is responsible for connecting the generated tests to the SUT so that the tests can be executed automatically. The input for the test engineer is the test repository generated automatically by the test analyst from the test model.


The test analyst is responsible of the quality of the test repository in terms of coverage of the requirements and fault detection capability. So the quality of his/her interaction with the subject matter expert is crucial. In the other direction, the test analyst interacts with the test engineer to facilitate manual test execution or test automation (implementation of keywords or the action words). This interaction process is highly iterative.

2.2.1.4 Testing nature and levels

MBT is mainly used for functional black-box testing, where the SUT is tested against a model, and any differences in behaviour are reported as test failures. The model formalizes the functional requirements, representing the expected operational behaviour at a given level of abstraction.

Regarding the testing level, the current mainstream focus of MBT practice for automated test design is system testing (including end-to-end and component testing) and acceptance testing, rather than unit or module testing. Integration testing is considered at the level of the integration of subsystems. In the case of a large chain of systems, MBT may address test generation of detailed test suites for each subsystem, and manage end-to-end testing for the whole chain.

2.2.2 Key factors for success when deploying MBT

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 12 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

The key factors for effective use of MBT are the choice of MBT methods to be used, the organization of the team, the qualification of the people involved, and a mature tool chain.

1. **Requirements and risks-based method:** MBT is built on top of current best practices in functional software testing. It is important that the SUT requirements are clearly defined, so that the test model can be designed from those requirements, and the product risks should be well understood, so that they can be used to drive the MBT test generation.
2. **Organization of the test team:** MBT is a vector for testing industrialization, to improve effectiveness and productivity. This means that the roles (for example between the test analyst who designs the test model, and the test automation engineer who implements the adaptation layer) are reinforced.
3. **Team qualification - test team professionalism:** The qualification of the test team is an important pre-requisite. The test analysts and the test automation engineers and testers should be professional, and have had appropriate training in MBT techniques, processes and tools.
4. **The MBT tool chain:** This professional efficient testing team should use an integrated tool chain, including a MBT test generator integrated with a test management environment and a test execution automation tool.

2.2.3 Automated Test Design with MBT in Standardization

A number of MBT tool vendors and major industrial users have developed at ETSI a first binding standard ES 202 951 [20] to unify terminology and define a common set of concepts required to be supported by MBT tools, modelling styles or modelling notations in order to be suitable for automated test design.

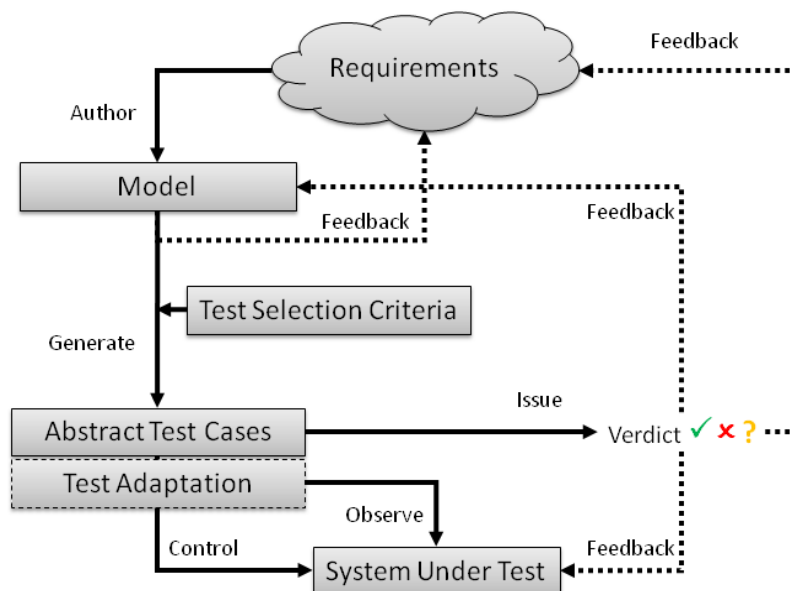



Figure 4. Model-Based Test Development [20]

2.3 MODELING FOR AUTOMATED TEST GENERATION

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 13 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

A variety of modeling styles, modeling notations, and tools can be used to model or express risk, functionality, as well as security aspects for test generation.

2.3.1 Modelling of Risk

2.3.1.1 CORAS risk analysis language

Name of the tool	The Coras Tool
Tool title phrase	CORAS Tool
Developed by	SINTEF Norway
Maturity	Open source tool
Supported modelling notations (if applicable)	CORAS risk analysis language
System requirements	Java run-time environment
Available	coras.sourceforge.net
Miscellaneous	

This tool enables specification and analysis of risk models specified based on the concepts of the CORAS method. It allows specification and creation of weighted associations between risks, threats, unwanted incidents, and assets.

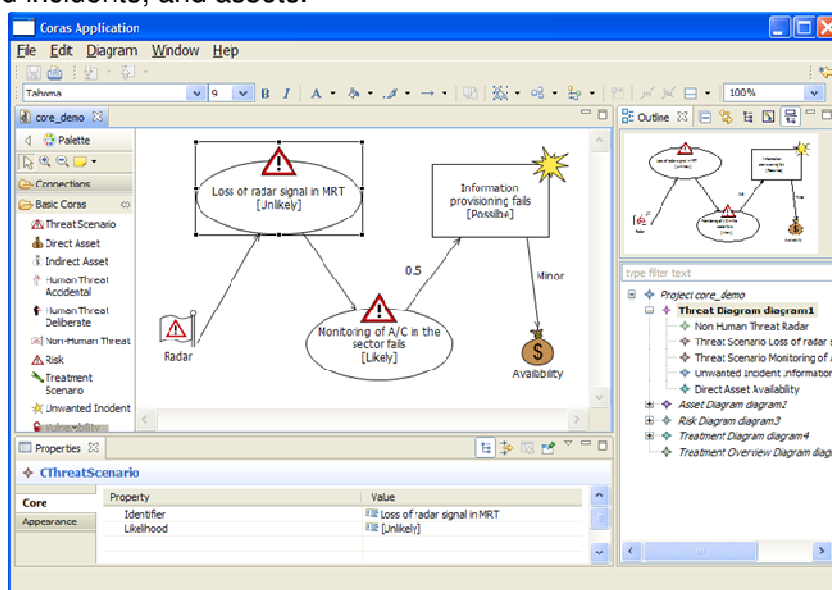


Figure 5. CORAS Tool Snapshot

2.3.2 Modelling of Functionality

The most widely deployed way of modelling functionality today especially in commercial MBT tools is UML. However individual MBT tools support usually different parts of UML or even profiles, i.e., they allow modelling with different sets of UML diagrams as well as different action languages.

2.3.2.1 QML (Conformiq Modelling Language).

This proprietary UML profile includes UML class diagrams and a subset of the standard UML statecharts, which includes support of hierarchy. In addition, it features a TTCN-like [58] system interface specification and type system. Its action language is Java based but extended with MBT specific concepts, e.g., for requirement annotation and specification of data constraints.

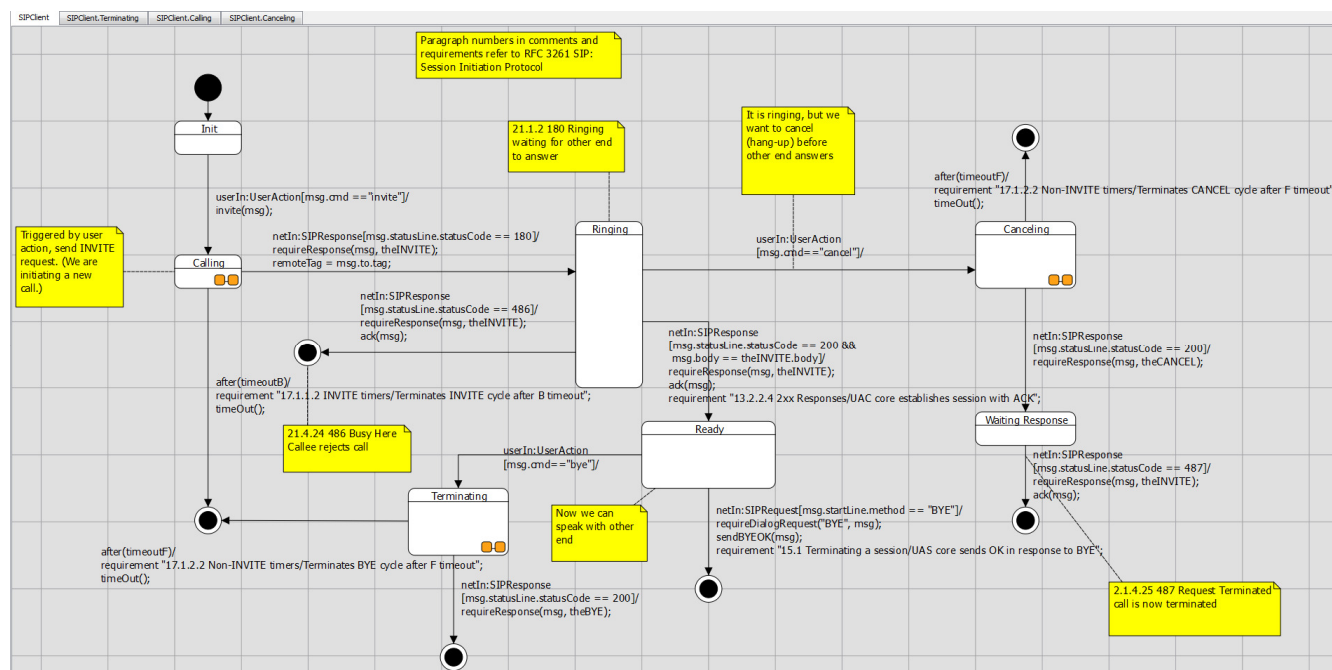



Figure 6 UML state chart with QML action language

The Conformiq tool chain allows integration with UML modelling tools such as:

Name of the tool	Conformiq Modeller
Tool title phrase	Statechart editor
Developed by	Conformiq Inc
Maturity	Free commercial tool
Supported modelling notations (if applicable)	Subset of standardized UML statecharts (open action language)
System requirements	Linux or Windows
Available	www.conformiq.com
Miscellaneous	Lightweight tool intended specifically for modelling with QML

Name of the tool	Rational Rhapsody
Tool title phrase	
Developed by	IBM
Maturity	Commercial product
Supported modelling notations (if applicable)	UML
System requirements	LinuxWindows
Available	http://www-01.ibm.com/software/awdtools/rhapsody/
Miscellaneous	General UML tool with Java integration

Name of the tool	Rational Systems Architect
Tool title phrase	
Developed by	IBM
Maturity	Commercial product

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 15 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

Supported modelling notations (if applicable)	UML
System requirements	Linux or windows
Available	http://www-01.ibm.com/software/awdtools/swarchitect/websphere/
Miscellaneous	General UML tool

Name of the tool	Enterprise Architect
Tool title phrase	
Developed by	Sparx systems
Maturity	Commercial product
Supported modelling notations (if applicable)	UML
System requirements	windows
Available	http://www.sparxsystems.com.au/
Miscellaneous	General UML tool

2.3.2.2 Smartesting's Modelling Language

The Smartesting CertifyIt MBT tool supports class and object diagrams as well as statecharts annotated with UML object constraint language. In addition, BPMN2 can be used expressing workflows.

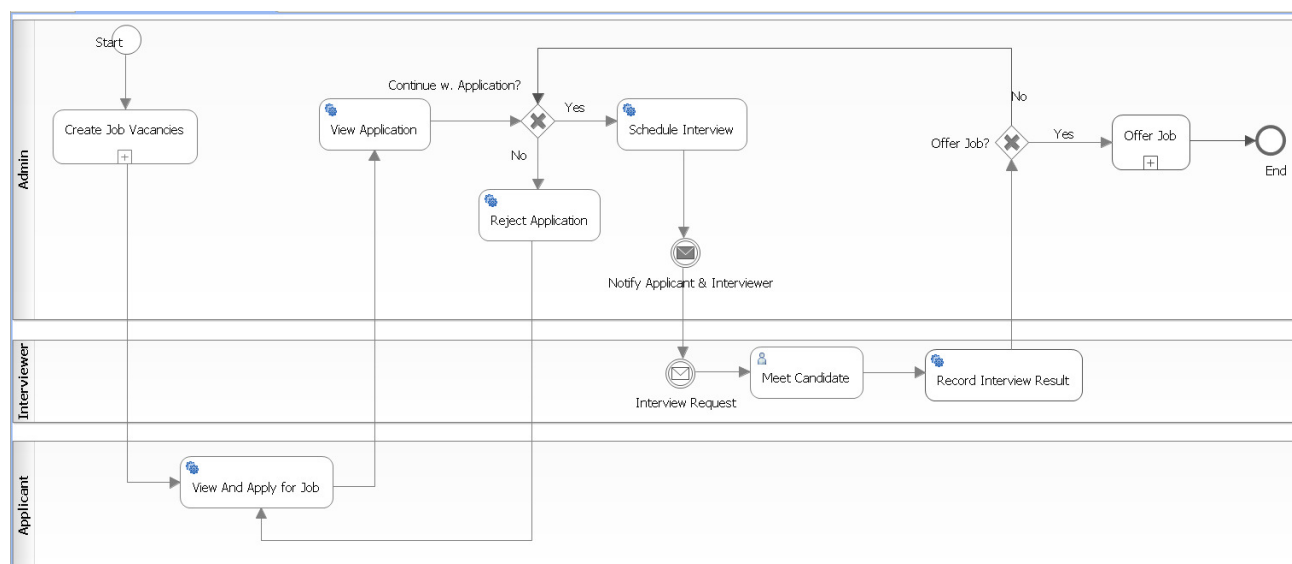


Figure 7. BPMN Example

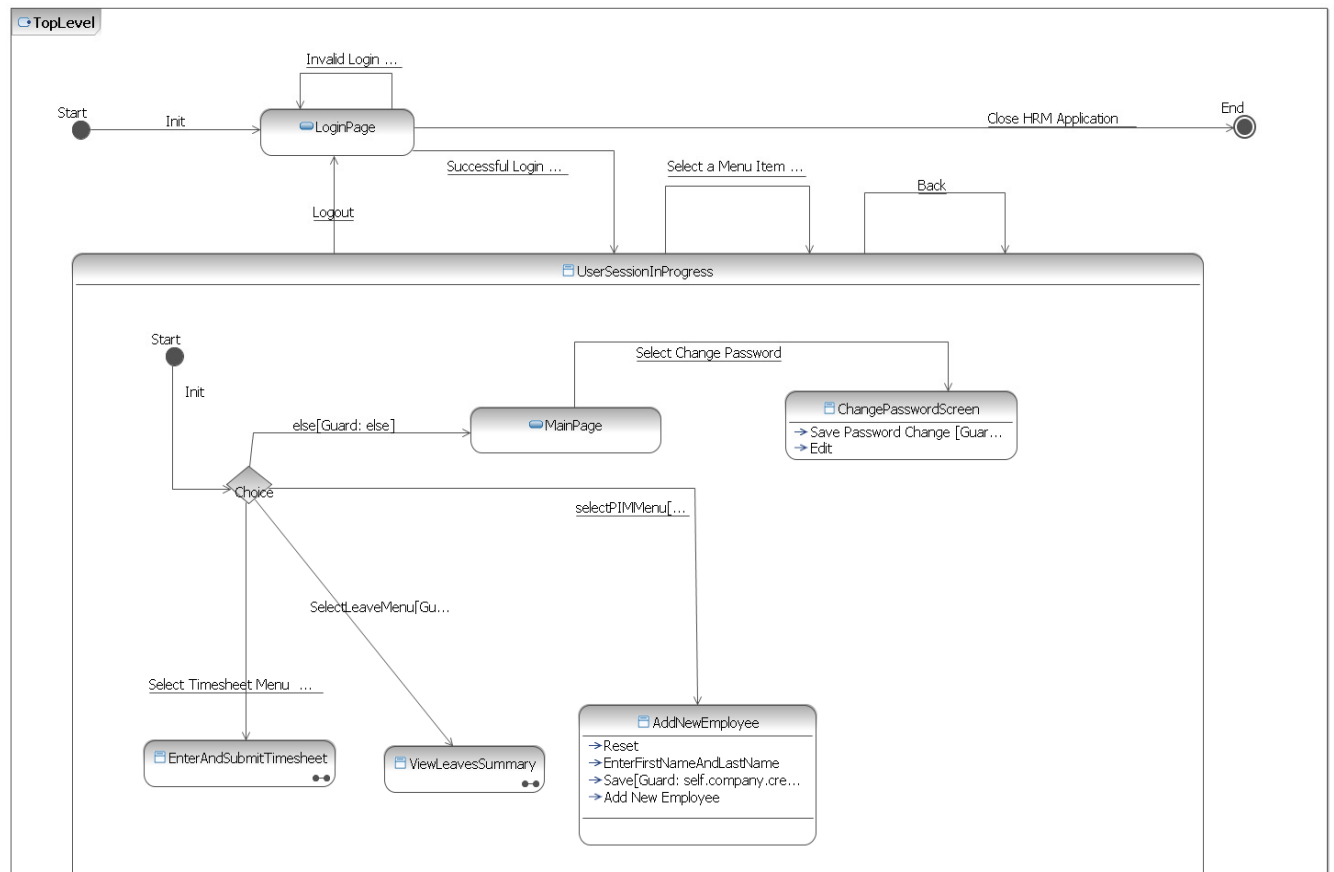


Figure 8. Example UML statechart diagram

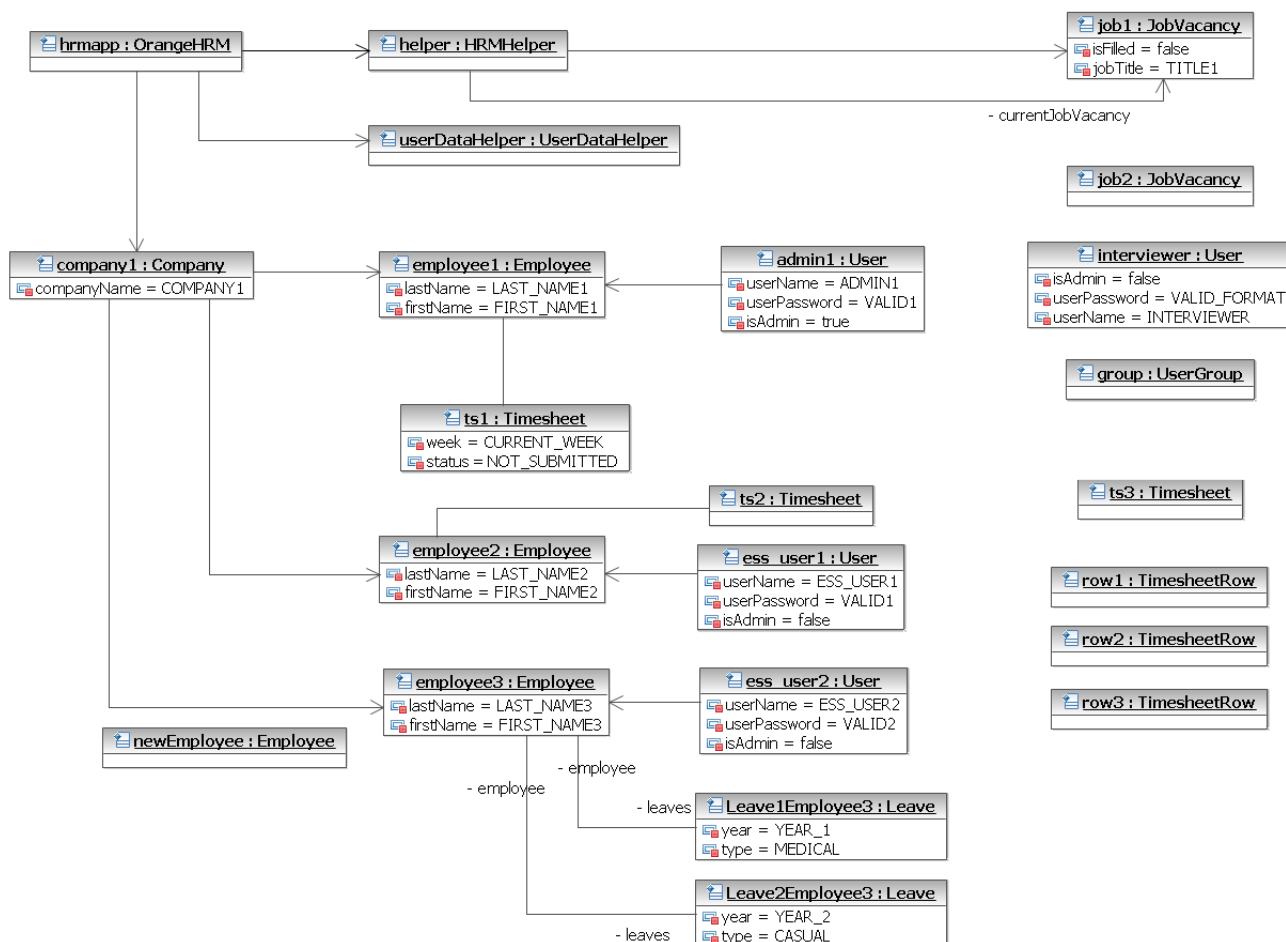



Figure 9. Example Object diagram

The Smartesting tool chain allows integration with UML modelling tools such as:

Name of the tool	Papyrus
Tool title phrase	
Developed by	CEA LIST
Maturity	Open source, under Eclipse Public License
Supported modelling notations (if applicable)	UML2, DI
System requirements	
Available	http://www.papyrusuml.org
Miscellaneous	

Name of the tool	Eclipse BPMN and UML modeller
Tool title phrase	
Developed by	Eclipse foundation
Maturity	Open source
Supported modelling	UML & BPMN

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 18 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

notations (if applicable)	
System requirements	Linux Windows
Available	
Miscellaneous	

Name of the tool	Rational Systems Architect
Tool title phrase	
Developed by	IBM
Maturity	
Supported modelling notations (if applicable)	UML
System requirements	Linux or windows
Available	http://www-01.ibm.com/software/awdtools/swarchitect/websphere/
Miscellaneous	General UML tool

2.3.2.3 UTP

Name of the tool	Fokus!MBT
Tool title phrase	
Developed by	Fraunhofer FOKUS
Maturity	Stable prototype
Supported modelling notations (if applicable)	UTP (no graphical notations supported)
System requirements	Eclipse UML2, version 3.0.0
Available	Currently not
Miscellaneous	Implementations also available for Papyrus and RSA

“UML Testing Profile is a language for designing, visualizing, specifying, analyzing, constructing, and documenting the artifacts of test systems. It is a test modeling language that can be used with all major object and component technologies and applied to testing systems in various application domains. The UML Testing Profile can be used stand alone for the handling of test artifacts or in an integrated manner with UML for a handling of system and test artifacts together.” [11]

The UML Testing Profile extends UML with test specific concepts like test components, verdicts, defaults, etc. These concepts are grouped into concepts for test architecture, test data, test behavior, and time. Being a profile, the UML testing profile seamlessly integrates with UML: it is based on the UML metamodel and reuses UML syntax.

The UML Testing Profile is based on the UML 2.0 specification. The UML Testing Profile is defined by using the metamodeling approach of UML. It has been architected with the following design principles in mind:

- UML integration: as a real UML profile, the UML Testing Profile is defined on the basis of the metamodel provided in the UML superstructure volume and follows the principles of UML profiles as defined in the UML infrastructure volume of UML 2.0.
- Reuse and minimality: wherever possible, the UML Testing Profile makes direct use of the UML concepts and extends them and adds new concepts only where needed. Only those

concepts are extended/added to UML, which have been demonstrated in the software, hardware, and protocol testing area to be of central relevance to the definition of test artifacts and are not part of UML.

Currently, the UTP is under revision by a Revision Task Force at the OMG. The revision process will be finished with the OMG Technical Meeting in June 2011.

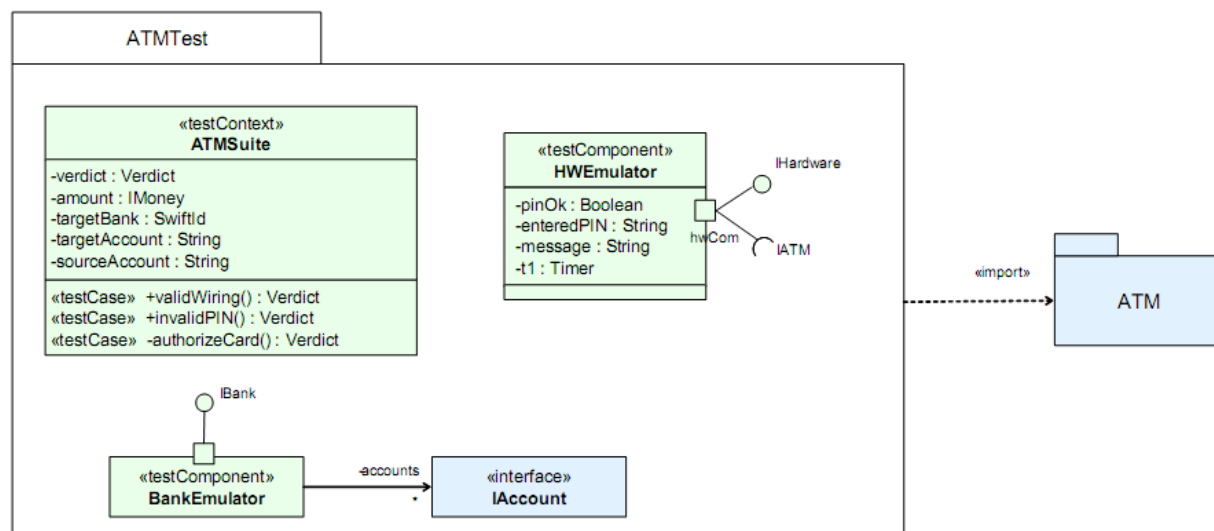


Figure 10. UTP example of a test component definition

2.3.2.4 doATOMSFramework with DSL

Name of the tool	doATOMSFramework
Tool title phrase	
Developed by	Dorner Consulting
Maturity	Commercial tool
Supported modelling notations (if applicable)	UML (Sequence Diagrams, Composite Structure Diagrams), SysML Internal Blockdefinition diagram (IBD)
System requirements	Windows XP SP2 (32 or 64 Bit) or later, sowie .NET Framework 3.5
Available	monika.enns@dorner-consulting.com
Miscellaneous	

The doAtoms Framework is based on two proprietary modelling languages. The Automotive Domain Modelling Language (ADML) allows to specify system models in and the eXtensible Application Markup Language (XAML) allows or the design of test case models. The following picture is an excerpt of the ADML and XAML model infrastructure. It gives the reader a first impression about the usage of the ADML profile with its dependencies to the implemented Domain Test Activities in XAML which are mostly instances of either UML Testing Profile metaclasses or ADML types.

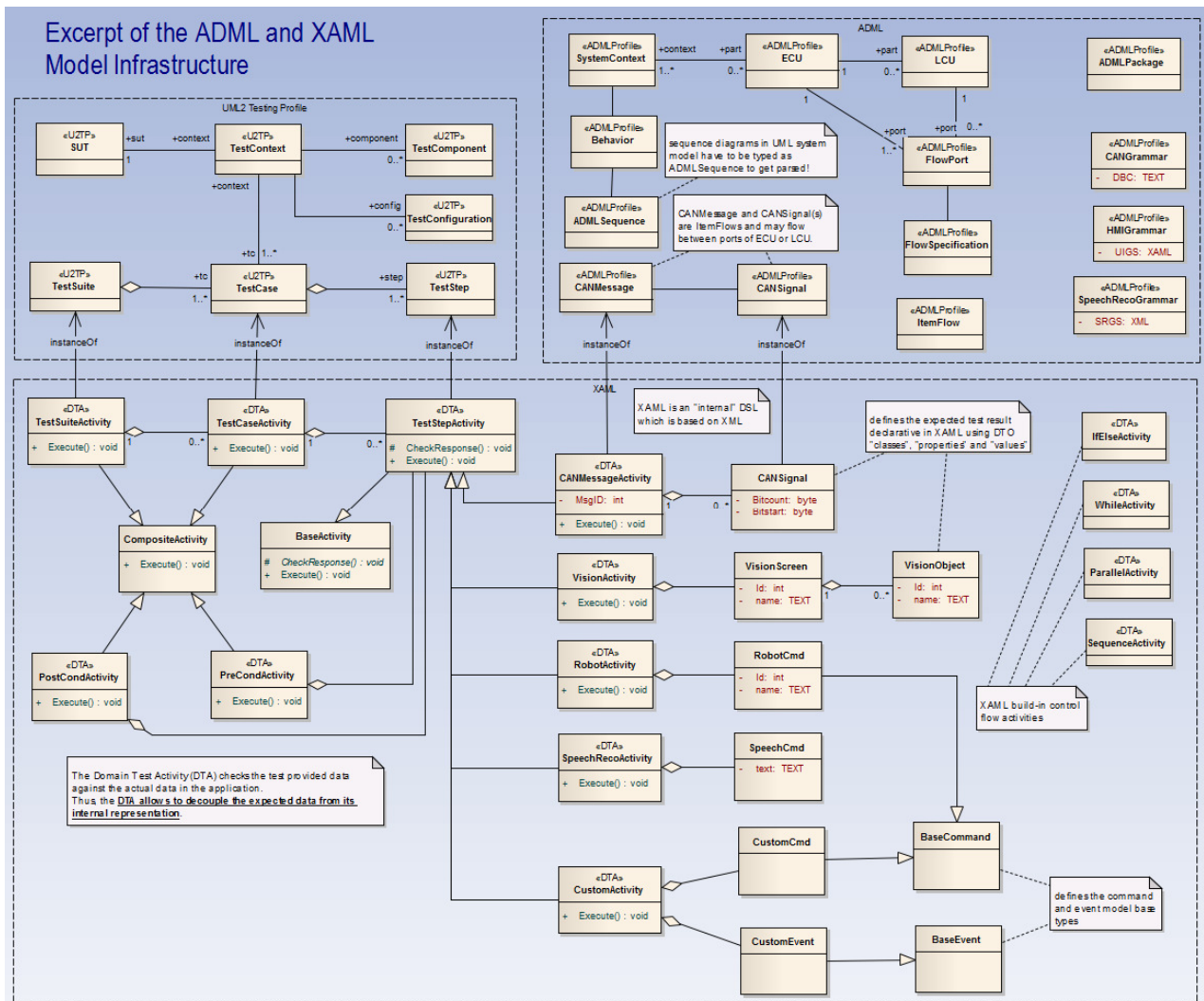


Figure 11. ADML and XAML Model infrastructure

2.3.2.4.1 System Models

A system model is defined in UML/ADML notation and specifies the behavior and structure of the system (formal requirements). On the other hand it can be used as an input to generate a test case model. This is done by means of a model to model transformation. The doATOMSFramework has a builtin M2M-transformation service based on a UML domain profile that is used to derive rules for transforming the system domain into the test domain.

In order to generate test cases out of the system model, the system model must comprise at least a system context diagram, one or more use cases with at least one interaction sequence diagram assigned, and an optional class diagram, which should define CAN message/signal structures to be used later for test data generation. The system context is designed as a UML composite structure diagram or a SysML Internal Blockdefinition diagram (IBD). It defines the ECU infrastructure with interaction points e.g. flow ports and connectors. Each flow port must have a flowSpecification where the IN and OUT signals and messages are defined as flowProperties.

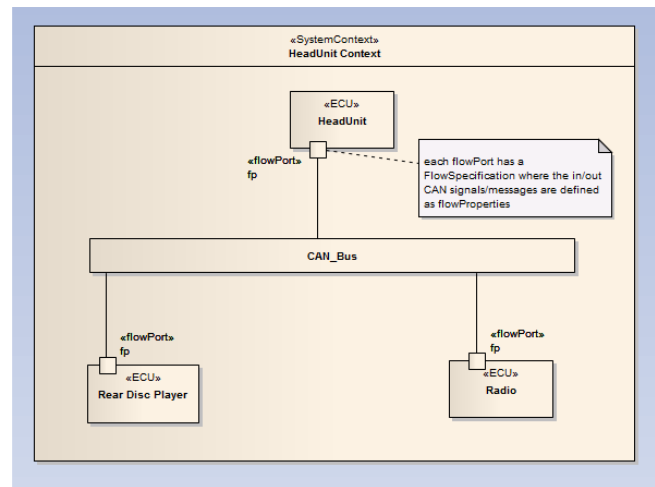


Figure 12. System model example

The behavioural system model is defined by a set of use cases with one or more assigned interaction sequences. Logical parameters (e.g. ARRequest, ARResponse, OPMODE, etc.) are mapped to the physical CAN signals automatically by a parser generator. While parsing the system model a semantic and syntax model check will be done. These model checks can assist the system model designer to verify the model against it's informal specifications.

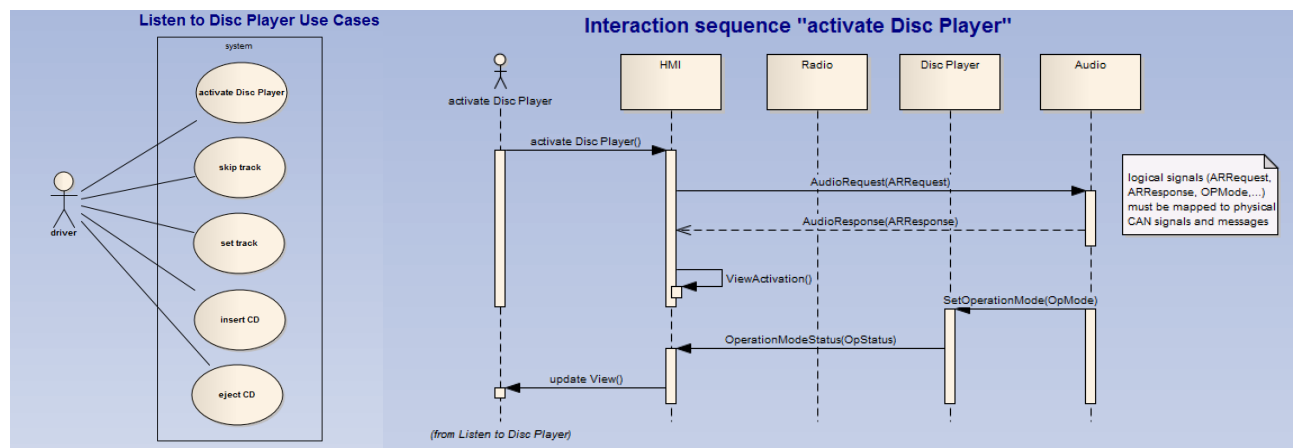



Figure 13. Behavioural system model example

2.3.2.4.2 Test Case Models

A test case model can be designed manually, based on existing informal specification. In this case the doATOMSFramework has a graphical Test Case Designer which can be extended by customers for individual needs. The Test Case designer uses an activity library where each activity represents a Test Step. Those Test Step activities can be composed in the designer via drag & drop technics to arbitrary complex test cases or test suites and saved into a test case model repository.

	Review of security testing tools Deliverable ID: D1_1	Page : 22 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

The Test Case Designer consist of the test case design IDE, the test step activity library and the test step parameter definition window. Either test cases can be generated out of the UML system model or they can be designed manually by easily drag and drop test step activities out of the test step activity library onto the Test Case Designer user interface. When test case design is finished, the test case can be executed immediately by pressing the “execute WF” button. All designed test cases can be saved in a test case library by using drag and drop technique. Once a test case is saved into the library, it can be linked with other testcase in the lib and even executed directly. A log recorder records all important information, warnings and errors during test case design and execution in a log database. All log information can be viewed online with LogDBViewer and exported as XML data for later offline analysis. CAN and LIN databases can be loaded in the appropriate DBViewer and then be used for parameterization the CANMessage test step activities or as test data in the test case generation process.

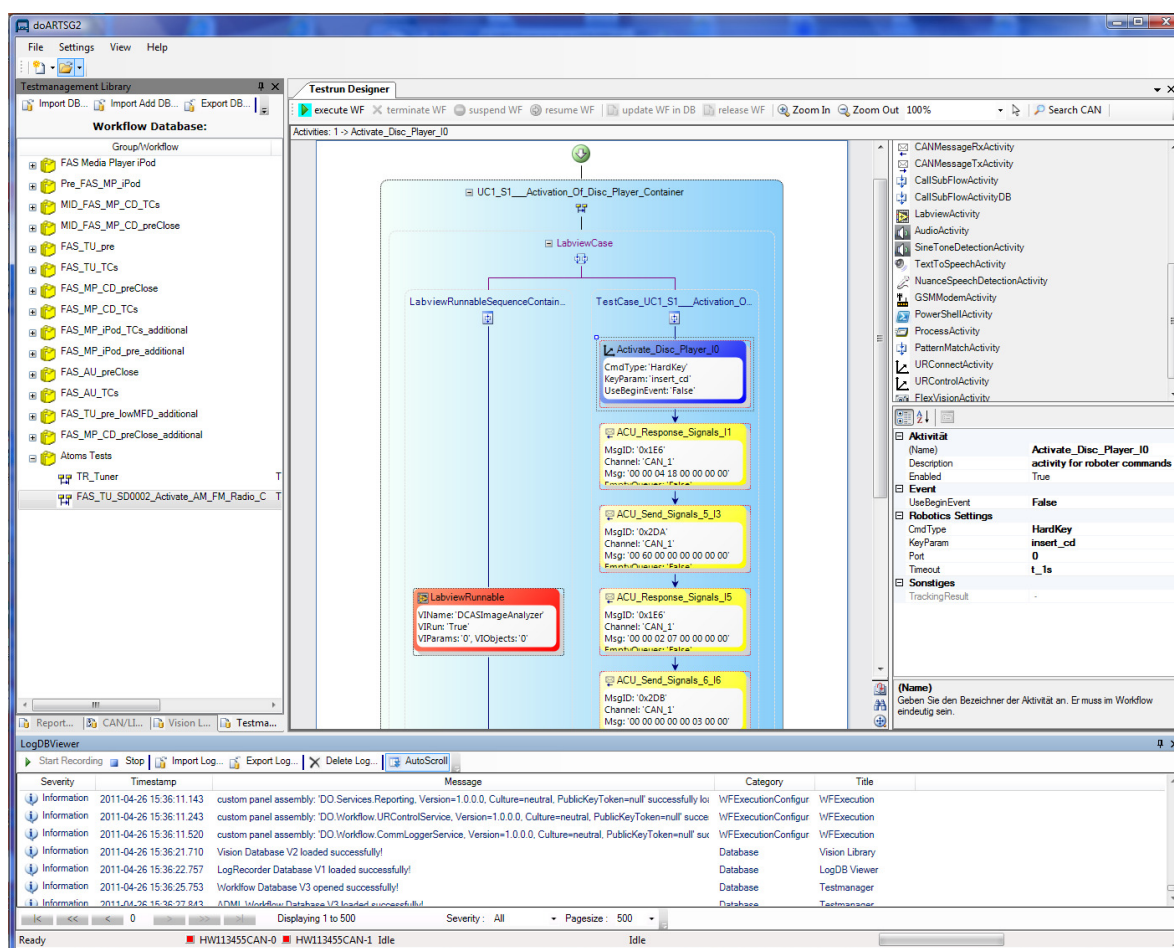



Figure 14. Test Case Designer

2.3.2.5 Codenomicon Modelling Language

Name of the tool	Codenomicon DEFENSICS
Tool title phrase	
Developed by	Codenomicon

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 23 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

Maturity	commercial tool
Supported modelling notations (if applicable)	Java & EBNF
System requirements	Windows, Linux
Available	http://www.codenomicon.com
Miscellaneous	

Codonomicon Defensics performs fuzzing, which means that software testing tools have to be capable of first creating valid message structures and message sequences, and then altering these to form nearly-valid messages that systematically anomalize some parts of the information exchange to test the target system for robustness. The Codenomicon modelling language is described below.

2.3.2.5.1 Extended Backus-Naur form with Java extensions

The core modeling notation is a proprietary extended BNF variant that allows for message exchange descriptions. The technique is called Extended BNF, or EBNF. It is described in [7].

An example of Codenomicon proprietary Extended BNF (EBNF):

```
TLS ClientHello:
client-hello = Handshake: {
  msg_type: { HandshakeType.client_hello },
  body: { @sid-len @cs-len @cm-len (
    !hs:client-version protocol-version
    !hs:client-random Random
    .session_id_length: !sid-len:target uint8
    .session_id: !sid-len:source (!hs:client-session-id SessionID)
    .cipher_suites_length: !cs-len:target uint16
    !cs-len:source !hs:client-cipher-suite !cipher-suite:cipher-type cipher-suites
    .compression_methods_length: !cm-len:target uint8
    !cm-len:source compression-methods
    # RFC4366: TLS Extensions
    .extensions: ()
  ) }
}
```


The Extended BNF is used to model the syntax of messages in both binary and textual protocols. Message sequence-level behavior is also modeled using BNF.

The way Extended BNF works on the message sequence level is that it uses rules to append the model with callbacks to Java code. Rules are used to:

- Perform I/O operations like connect, send, and receive, on message sequence level.
- Calculate fields like lengths, checksums, sequence numbers, inside protocol messages.

2.3.2.5.2 User sequences

To provide the end user with lots of ready-made test cases - that is, anomalized messages and message sequences - the users of the Codenomicon test tools may also specify the used message sequences and/or message content themselves. For this purpose a proprietary XML based sequence file is used. An example sequence file for Session Initiation Protocol (SIP) based interfaces may look like the following:

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 24 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

```

<sequences>
  <!-- SIP dialog definitions -->
  <sequence name="used-dialog" setting="user-sequence-file">
    <description name="PUBLISH request">A sequence sending a PUBLISH request
      and expecting a success response.</description>

    <send name="publish-request" type="sip-message" description="PUBLISH request">
      <content file="sip-publish.txx" format="text"/>
      ...
      <store attribute="call-id">...</store>
      ...
    </send>

    <recv name="publish-response-ok" type="sip-message" description="OK response">
      ...
      <match attribute="call-id">!sip:callId $publish-request:call-id</match>
      ...
    </recv>
  </sequence>
</sequences>

```

The actual content of the send message "publish-request" is specified in a separate file as raw data. The sequence file specifies that SIP header "call-id" must match the corresponding header line in the received message. Note that the real sequence specifications contains a lot more details than included here as the SIP protocol is rather complex.

2.3.2.5.3 Model Editing Capability in Defensics

As seen in a tool screenshot below, the users can launch editors to edit both the sequence models and the message structures. The models are edited as text.

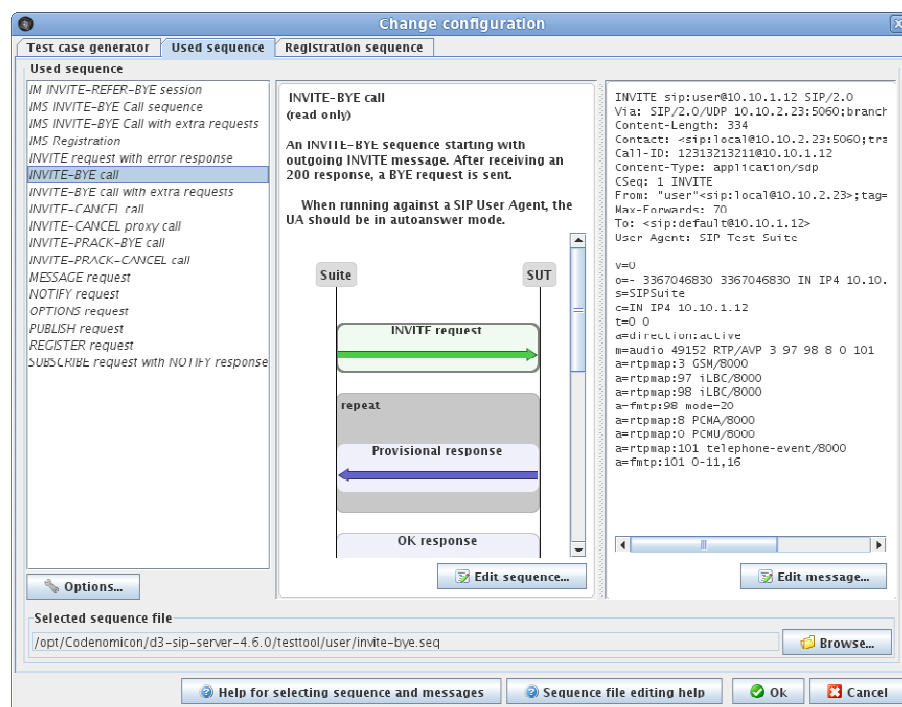



Figure 15. Model-editing in Defensics

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 25 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

2.3.3 Modelling of Security Aspects

2.3.3.1 UMLsec

Name of the tool	UMLsec tool
Tool title phrase	
Developed by	UMLsec tool group, esp. Jan Jürjens of Technical University of Dortmund
Maturity	alpha prototype
Supported modelling notations (if applicable)	UMLsec
System requirements	Windows, Linux
Available	http://inky.cs.tu-dortmund.de/main2/jj/umlsectool
Miscellaneous	

UMLsec is a profile of the Unified Modeling Language (UML) which attempts to model security aspects with formal semantics. Thus, it has the potential to make security testing techniques applicable at the design stage. The goal of this approach is to encapsulate security expertise/best practice into a UML extension or profile, so system designers can use them to construct secure systems without being themselves experts in this field. In UMLsec the standard extension mechanism of UML is used: Stereotypes, tags and values - i.e. labels and key/value pairs, with well-defined semantics - are packaged into a so-called UML profile.

UMLsec is described in [1]. Its aim is to enrich a system's UML model in such a way, that the correctness of design and implementation can formally be verified. The profile is the result of experience in several security-critical industry projects. It was successfully applied in the "analysis of CEPS, a candidate for a globally interoperable electronic purse standard" and with a "major German bank" in the sense that it discovered critical flaws.

The first two UMLsec properties important for the testing perspective is the capability of dealing with differentiated attacker types. This allows designers to model defaults attackers as well as system user attackers or even smart card issuer attackers.

The second is the variety of security-relevant information which is covered by the provided stereotypes. They fall into three categories: assumptions on the physical level, requirements on the logical level and policies the system should adhere to. These can be added to any of the following UML diagrams: activity diagrams, class diagrams, sequence diagrams, state charts, deployment diagrams, packages. Figure 16 depicts a sample UMLsec deployment diagram, which makes use of prominent UMLsec stereotypes.

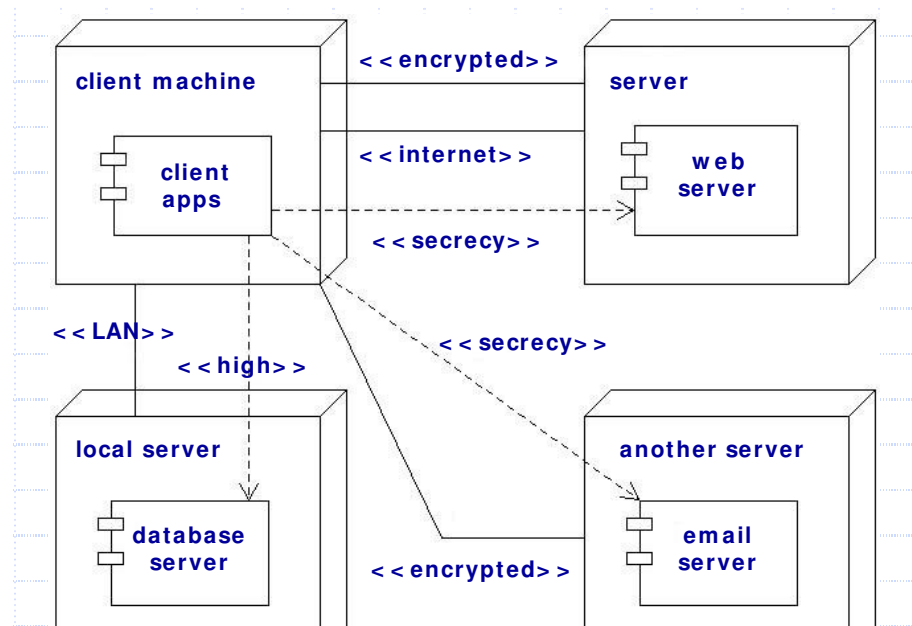


Figure 16. Sample UMLsec Deployment Diagram¹


A part of the stereotypes introduced by UMLsec are listed in Figure 17. The column Base Class refers to the UML element that can be enriched with the stereotype indicated in the first column. Key/value pairs that the stereotype entails are listed in the column Tags, moreover, constraints and a description are provided in the respective columns. For example `<<critical>>` can be assigned to “objects or subsystem instances containing data that is critical in some way” [97], the way in which a subsystem is critical can be further detailed with the tags `{secrecy}`, `{integrity}`, `{authenticity}`, `{fresh}`, and `{high}` indicating the required property.

Stereotype	Base Class	Tags	Constraints	Description
Internet	link			Internet connection
encrypted	link			encrypted connection
LAN	link			LAN connection
secure links	subsystem		dependency security matched by links	enforces secure communication links
secrecy	dependency			assumes secrecy
secure	subsystem		« call », « send » respect	structural interaction
dependency			data security	data security
critical	object	secret		critical object
no down-flow	subsystem		prevents down-flow	information flow
data	subsystem		provides secrecy	basic datasec
security				requirements
fair exchange	package	start,stop	after start eventually reach stop	enforce fair exchange

Figure 17. UMLsec stereotypes (excerpt) [4]

Some work [3] [5] is presented towards tools that automatically verify design constraints. However, only a subset of those has been implemented, i.e. there is no fully-fledged verification suite available. This reflects the academic acceptance of UMLsec, there have been continuous publications

¹ Source: http://www4.in.tum.de/~juerjens/papers/umlsec_tools_short.pdf

	Review of security testing tools Deliverable ID: D1_1	Page : 27 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

since the presentation of the profile in 2002, the authors are, however, clustered around the author of the original profile.

2.3.3.2 *secureUML*

SecureUML [6] is another approach to attack security issues at design time. Its focus is narrower than UMLsec's: it is limited to enriching UML with access control information. As such it is rather geared towards the rapid generation of secure systems, than enabling formal verification or testing.

From the access control point of view, secureUML offers an extension of traditional role-based access control systems (RBAC). Where those are normally limited to static assignment of permissions to roles and roles to users, secureUML allows for the definition of access constraints. These constraints consider the state of a system in time to evaluate access rights.

The meta-model presented by the secureUML-authors is depicted in Figure 18. The upper left part covers a classical RBAC-system: permissions are assigned to role and roles to users. Via the link ProtectedObject these permissions are attached to elements of the system's UML model. Using the class AuthorizationConstraint, the permissions associated with a model element can be further detailed, especially with state information.

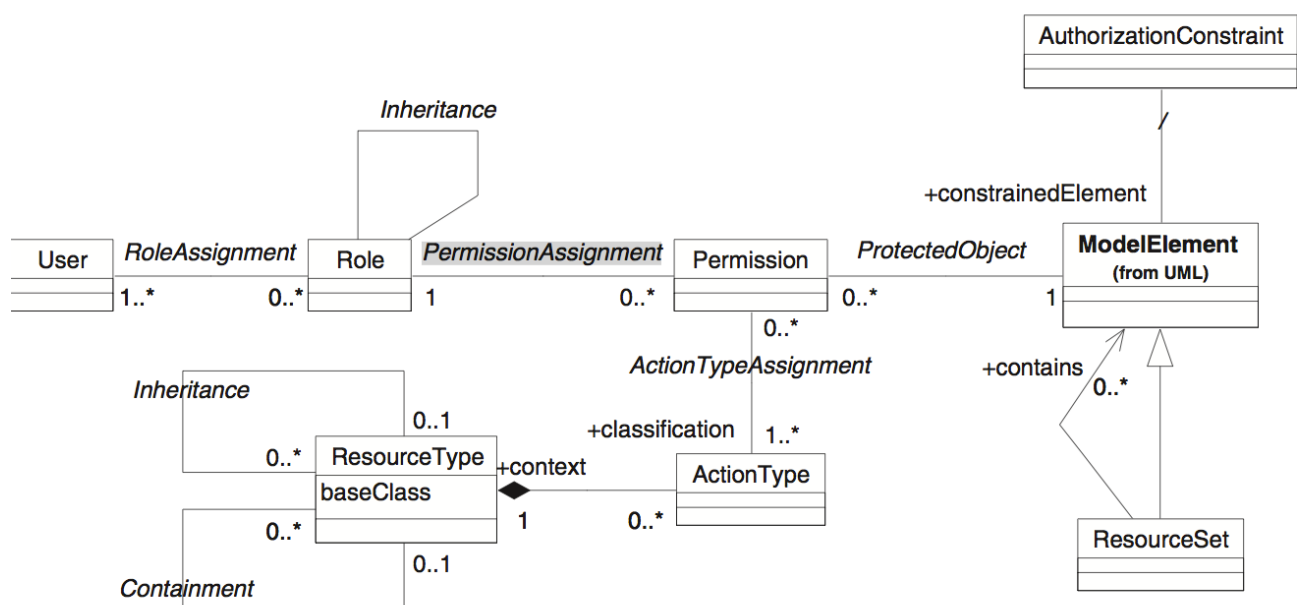



Figure 18. secureUML meta model

2.3.4 Fokus!MBT

Fokus!MBT is a flexible and extensible tool chain, which facilitates the development of model-based testing scenarios for heterogeneous application domains. It is based on a service-oriented communication infrastructure of loosely coupled services, interoperating with each other in a distributed environment. Fokus!MBT defines a proprietary testing meta model – called TestingMM. A test model is the main artifact in a model-based testing approach. It is a formal representation of test-specific information. A test model is used to increase automation, to avoid ambiguities and to facilitate test case derivation. The TestingMM represents an integrated data model used for data

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 28 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

and information exchange among the services of the Fokus!MBT tool chain. This allows adapted services to interoperate with each other as well as it improves the communication flow among involved stakeholders.

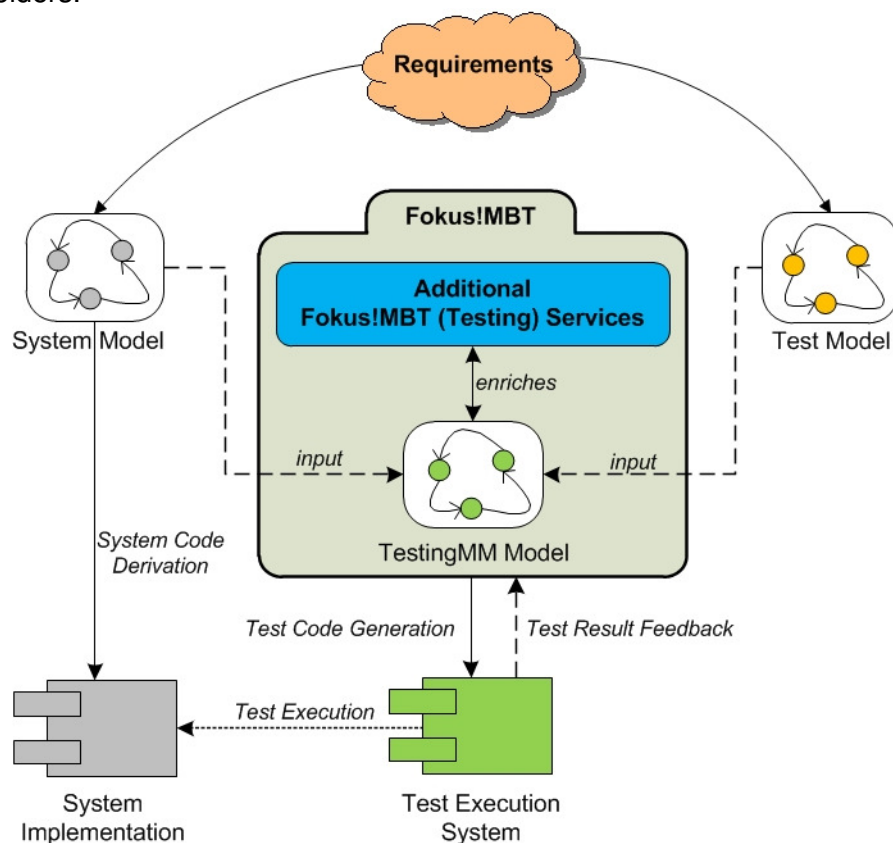


Figure 19. The FOKUS!MBT modelling and test generation approach

2.3.4.1 High-level Architecture

Each software development process varies in requirements, used techniques, involved stakeholders and its target environment. Testing is an essential part of the software development processes of any domain. A domain-specific test process needs to be tailored to its corresponding system requirements.

Fokus!MBT is a set of compound tools which support the model-based paradigm for testing purposes. It establishes a tooling landscape for the specification, development and documentation of tailored model-based testing scenarios. Current Fokus!MBT methodologies are based on, but not limited to well-known and established standards like UML, SysML and the UML Testing Profile. That allows Fokus!MBT to be applied to a wide range of heterogeneous system development processes. Integrating the ModelBus Service Infrastructure gives it greater flexibility and extensibility. The ModelBus' orchestration capabilities make it possible to automate various workflows. Overall time to market is reduced by significantly decreasing the number of error-prone and resource-consuming manual tasks.

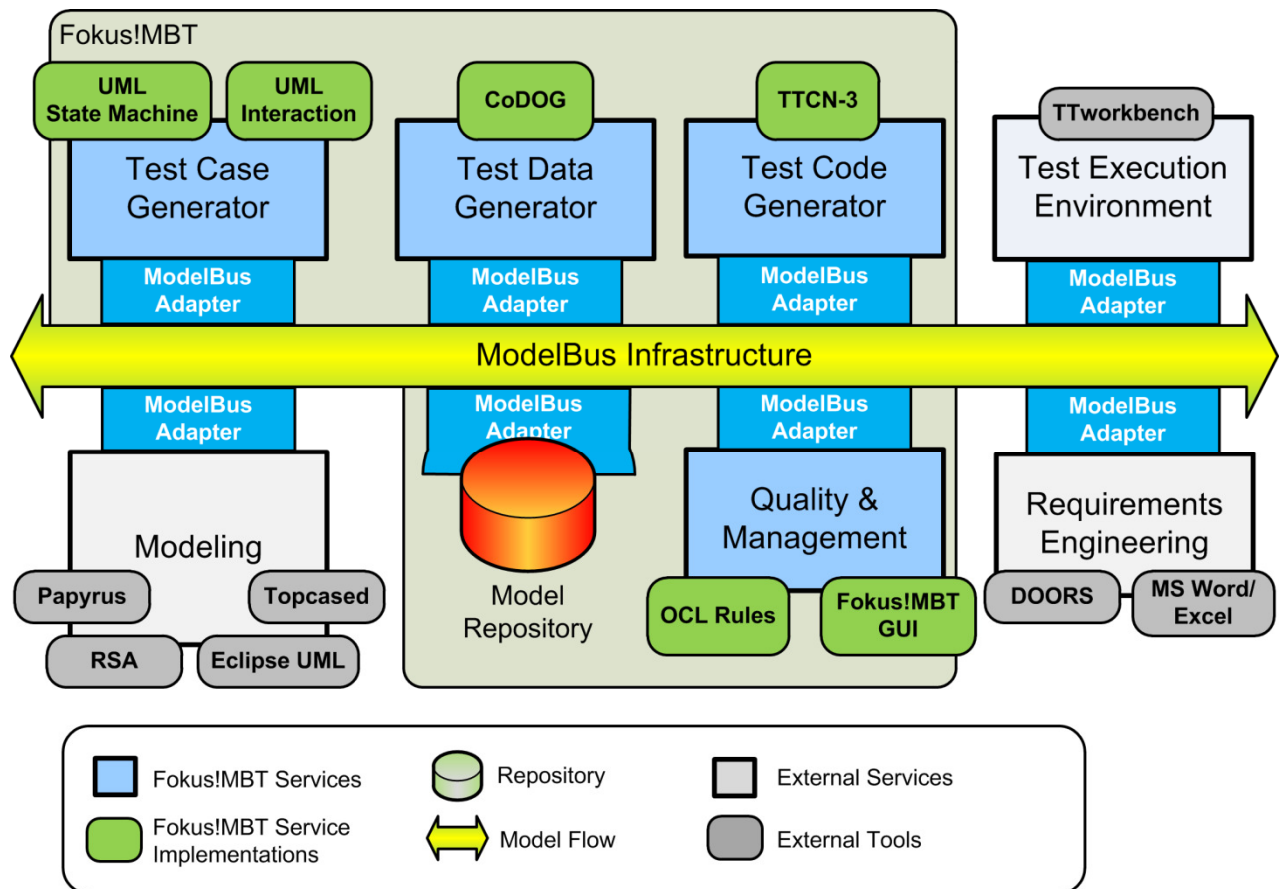



Figure 20. The FOKUS!MBT infrastructure

2.3.4.2 Fokus!MBT Testing Metamodel

A test metamodel integrates relevant concepts to describe tests or test processes. Beside structural aspects and the behavioral usage description of the SUT (system under test), it comprises aspects for test data, test execution and test results. Unfortunately, there is no common agreement about what concepts are considered to be test relevant. We (hence the specification of the TestingMM) have identified the following complementary aspects: test targets/objectives, test architectures, test behavior, test data, test directives, test execution, test results and test strategies.

The most important target during the specification of the integrated testing metamodel was to identify widely accepted testing concepts, put them into model artifacts and integrate those artifacts with each other coherently in a structured way, so it would be possible to describe tests precisely and concisely. This specification phase of TestigMM started by analyzing the MOF-based meta-model of the UML Testing Profile (UTP), which constitutes the fundament of our test metamodel. The UTP defines a firm foundation of elementary test concepts, already expressed in a semi-formal manner. **Error! Reference source not found.** shows all the concepts the TestingMM currently provides as well as which languages they were taken from. Additionally, it illustrates the package structure of TestingMM, since each row header result in a separate package in the meta-model.

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 30 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

- **UML.** UML contributes elementary object-oriented concepts like packages, classes, properties and interfaces. It is also used to provide instances of the TestingMM with a package-oriented structure, like any MOF or UML model itself.
- **UTP.** As already mentioned, UTP represents the core for TestingMM, since it defines the minimal required artifacts for the creation of a test model.
- **SysML.** The concise and intuitive possibility of expressing requirements inside UML models have been extracted from SysML. UML itself is not capable of modeling requirements natively.
- **TPTP.** The metamodel of the Test & Performance Tools Platform, a top level project of the Eclipse community, is comprises similar but significantly different concepts like the TestingMM. Among others, the concepts for expressing the results of a test case execution in a model-based manner have been taken from the TPTP metamodel.
- **New concepts.** The TestingMM was enriched with concepts, which have not been addressed by other known languages. This is also the reason why we decided to create a new proprietary metamodel instead of reusing TPTP for example. Each model covers only some of the concepts we indentified to be test relevant.


However, not all required test relevant aspects could be covered by already existing languages. Therefore, these aspects had to be developed and included from scratch by ensuring the syntactical and semantical integrity of the TestingMM. Thus, TestingMM can be seen as a conceptual merge of the participating languages and newly introduced concepts. In course of this merge we changed the weak-type manner to a strong-type manner, since pointers to external sources have been removed consequently in order to keep all information stored within in one model instance solely.

Foundation	Test Architecture	Test Behavior	Test Data	Test Time	Test Purpose	Test Generation	Test Execution	Test Deployment	Test Environment
Classifier	SUT	Test Case	Data Pool	Timer	Test Purpose	Generation Directive	Execution Trace	Deployment Configuration	Setup Directive
Interface	Test Component	Message	Data Partition	Timer Events	Requirement	Coverage Goal	Verdict	Machine	Teardown Directive
Port	Test Context	Events	Partition Rules	Timezone	Requirements Coverage	Stop Criteria	Verdict Trace	Artifact	Type Mapping
Type	Test Control	Validation Action	Partition Instances		Requirement Verdict		Execution History	Manifestation	Interface Mapping
Signal	Arbiter	Default	Wildcards					Location	
Operation		Logging	Coding Schemas						
Connection									

Concepts related to

UML	UTP	Newly Introduced	SysML	TPTP
-----	-----	------------------	-------	------

Figure 21. Conceptual Overview of TestingMM

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 31 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

2.3.4.3 Fokus!MBT Test Services

Out of the box, Fokus!MBT provides services for test case, test data and test code generation. The test case generator is called Eventeer and relies on conceptually limited UML state machines. As an example, state machines prepared for Eventeer are not allowed to contain orthogonal composite states. Test case generation is driven by so called test directives. Test directives are instructions for the derivation engine how to create test cases from the state machine. Currently, Eventeer provides two very fundamental directives, that are state and transition coverage, which can be configured with an arbitrary percentage. This percentage acts at the same time as stop criterion (if no more disjunctive criteria are defined). For example, it is possible to configure the test case generation with a test directive that tries to achieve 80% of transition coverage. Besides the structural coverage directives for the state machine under consideration, Eventeer provides two generation methods to drive the pathfinding: random and shortest. The latter one is an implementation of Dijkstra's A* algorithm.


Eventeer copes with the definition of guard conditions and variable expressions as introduced for extended finite state machines. As constraint and effect language, a proprietary Java-like notation is used, which is called Fokus!MBT Constraint and Effect Language (in short: FoCIL, spoken: fos-sil). FoCIL allows the definition of guard constraints, as well as the textual definition of behavior to be executed when the transition is fired. FoCIL was developed due to the lack of textual expression languages in the realm of the UML family that fit the needs to express both constraints and effects.

Eventeer is seamlessly integrated with Fokus!MBT CoDOG, a Constraint-based test data and oracle generator. CoDOG allows the generation of data instances, described by constraints which are applied on the type of the instances. If, for example, a particular transition is only allowed to fire if an attribute of a triggering signal satisfies a certain constraint, CoDOG is able to generate data instances that satisfies the constraint (if possible). Therefore, all applied constraints on a basic type of the received signal (or in general, classifier) is gathered in translated into the CoDOG constraint satisfactory metamodel (CSM). The CSM is augmented with all necessary information which may further restrict the allowed domain of the test data instance. This includes both actual attribute values of the context of the executed state machine, and the constraints of the guard condition, if the expression in the guard references the received signal. The complete CSM is passed into the CoDOG solving engine, which creates at least one solution variable (or more, or none, if the problem description was not solvable).

In combination, Eventeer and CoDOG allow the generation of fully executable test cases, automatically derived from a tailored UML state machine. Those test cases are represented as TestingMM interactions, a simplified implementation of the UML interactions. For the execution of the test cases, Fokus!MBT integrates a TTCN-3 code generator, which derives TTCN-3 test scripts out of TestingMM instances, which might be executed in a particular TTCN-3 execution environment (e.g. TTworkbench from Testing Technologies IST).

3. EXTEND TEST COVERAGE USING SECURITY-ORIENTED TEST PURPOSES

The Smartesting approach for testing security properties is discussed in D1.WP2 [59]. In this section we explain the principle and tool that Conformiq adopts for testing security properties and the methodology the ETSI has used to test security features of the IPV6 protocol.

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 32 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

3.1 CONFORMIQ APPROACH FOR TESTING SECURITY PROPERTIES

Name of the tool	Conformiq Designer
Tool title phrase	Automated Test Design
Developed by	Conformiq Inc
Maturity	Commercial Product
Supported modelling notations (if applicable)	QML (= UML Class Diagrams + UML StateCharts + proprietary (TTCN oriented) type system and system interface specification + proprietary Java-based UML action language) and proprietary tabular Use Case specification format
System requirements	Either standalone Windows or Linux executable or Eclipse Plugin
Available	www.conformiq.com
Miscellaneous	MBT Functional Test Case Generator


Conformiq Designer offers the ability to specify “use cases” separately from the modeled behavior – sometimes also referred to as test purposes. These use cases represent partial or full sequences of message exchanges with restrictions on data based on the specified system interface.

A use case in Conformiq describes essentially a high level, usually partial I/O sequence that a system under test (i.e., the black box) is expected to reproduce. For each message in such a sequence the message type and the port (as specified in the system interface specification of the model capturing the system operation) and expected time stamp have to be specified. By default any message contents are accepted for a message but can be refined by further constraining the message field values to specific values. Secondly, one or more so called “gaps” (represented by the asterisk symbol) can be inserted into any point at these sequence to express that any message can arrive or be sent on any port before the next message in the sequence occurs in a generated test. Besides the reuse of the system interface specification, use case specification is completely independent of the specification of functional behavior, i.e., it is possible to specify use case or (partial) message sequences that do not comply or violate to specified system operation.

Once specified use cases can be selected in Conformiq Designer as coverage criteria for models specifying system operation, i.e., marked to be specifying sequences that must be realizable via the modeled system behavior as well as cases that shall not be realizable. In case use cases are marked as targets Conformiq Designer will enforce that at least one but possibly more tests (depending on other selected coverage criteria) are produced in the test generation process. Use cases can also be used to encode security oriented test purposes or test patterns by expressing the security property to be verified in one or more use cases.

3.2 ETSI APPROACH TO SECURITY TESTBEDS SPECIFIC TO IPV6 SECURITY TESTING

The present clause describes the structure and the implementation of an automatic testbed for testing security procedures within the scope of the Internet Protocol version 6 (IPv6), namely IP Authentication Header (AH) defined in RFC 4302 [22], IP Encapsulating Security Payload (ESP) defined in RFC4303 [23] and the Internet Key Exchange (IKEv2) Protocol defined in RFC4306 [24]. The named procedures are working under the framework of RFC4301 [21] which defines the "Security Architecture for the Internet Protocol".

	<p style="text-align: center;">Review of security testing tools</p> <p style="text-align: center;">Deliverable ID: D1_1</p>	Page : 33 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

The work was performed under the umbrella of a Specialized Task Force (STF) of ETSI's Technical Committee MTS (Methods for Testing and Specification) and was co-funded by EC/EFTA.

Where the STF was originally tasked to produce conformance tests which usually concentrate on the communication on a given interface (messages formats, message sequences, etc.) the outcome of the project developed a stronger relation to security testing while testing IPV6 properties. Especially for testing the IKEv2 functionality all the concepts of that technology (public and private keys, "the seven secrets", etc) had to be modeled in TTCN-3 on top of the management of the message exchanges. Another task was naturally the handling of the encrypted IPV6 packets to allow their observation and interpretation in the testbed environment. All the above added many characteristics to the conformance tests that are usually addressed in functional testing.

3.2.1 Organization of the work

3.2.1.1 *Requirements Catalogue*

As a first step the relevant RFC documents were analyzed and a set of about 700 requirements were derived. This work was done with the resulting requirements being named, labeled (mandatory, optional, conditional) and categorized by device for which they apply. This requirement catalogue was published in ETSI TS 102 558 [25].

3.2.1.2 *Test Suite Structure and Test Purposes*

In the next step Test Purposes (TP) were written for IPV6 nodes according to the Requirements of the Requirements Catalogue in TS 102 558 [25]. Test purposes have been written for behaviours requested with "MUST" or "SHOULD", optional behaviour described with "MAY" or similar wording indicating an option was not turned into test purposes.


Furthermore, as one TP could cover several requirements, the overall number of TPs totaled at 103 covering all the mandatory requirements with the following split:

- AH 12 TPs
- ESP 15 TPs
- IKEv2 76 TPs

TPs were organized in the Test Suite Structure (TSS). For AH and ESP TPs no further sub-grouping was done but for the IKEv2 TPs the following sub-structuring was applied:

- Group 1 Exchange Message Structures
- Group 2 IKE Header and Payload Formats
- Group 3 IKE Informational Exchanges
- Group 4 IKE Protocol

All TPs were written manually using TPLan, a formal notation explicitly developed by ETSI for the definition of TPs in a standardized format allowing global consistency within the TP document through the use of pre-defined key words for conditions and actions. Following an example:

	Review of security testing tools Deliverable ID: D1_1	Page : 34 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

Test Purpose			
Identifier:	TP_SEC_6407_01		
Summary:	Test of generating CREATE_CHILD_SA request		
References:	RQ_002_6407, RQ_002_6035, RQ_002_6084, RQ_002_6085, RQ_002_6086, RQ_002_6128, RQ_002_6129, RQ_002_6232, RQ_002_6233, RQ_002_6236, RQ_002_6240, RQ_002_6250, RQ_002_6263, RQ_002_6344		
IUT Role:	Host	Test Case:	TC_SEC_6407_01
<pre> with { IUT having completed IKE_SA_INIT exchange and IUT having completed IKE_AUTH exchange } ensure that { when { IUT is requested to send CREATE_CHILD_SA_request } then { IUT sends CREATE_CHILD_SA_request containing (IKE_Header containing IKE_SA_Initiators_SPI set to IKE_SA_Initiators_SPI sent or received in the IKE_SA_INIT_request and containing IKE_SA_Responders_SPI set to IKE_SA_Responders_SPI sent or received in the IKE_SA_INIT_response and containing Major_Version set to 2 and containing Exchange_Type set to 36 CREATE_CHILD_SA and containing Flags set to 00010000'B' and containing Message_ID set to previous sent Message_ID plus 1) and containing (Encrypted_payload containing (Security_Association_payload containing at least 1 proposal containing at least 1 transform) and containing (Nonce_payload containing Nonce_Data of at least 128 bits and 'at least half the prf key length') and containing Traffic_Selector_payload_initiator 'Next Payload field of previous payload is set to 44' and containing Traffic_Selector_payload_responder 'Next Payload field of previous payload is set to 45')) } } </pre>			

Figure 22. Test Purpose in TPLan

For further information on TPlan see ETSI ES 202 553.

The TPs were published in ETSI TS 102 593 [26].

Using this form of test purpose description allows for the mere expression of message exchanges with encryption information only being mentioned verbally by adding the fact that a message or a message field is encrypted, see above keyword: “encrypted payload”. The way of encryption (encryption algorithm, keys) is to be determined dynamically during the test execution, as it varies widely between the key factors used during the negotiation processes. This is an aspect that shows the limits of the traditional approach to test purpose production when an additional test dimension, i.e. encryption information is added to the description of message exchanges.

3.2.1.3 Abstract Test Suite

The TPs were used as the base for the Abstract Test Suite (ATS) written in TTCN-3 as published in ETSI TS 102 594 [27].

An Abstract Test Method (ATM) as shown in figure 1 with a set of rules was defined at the start of

the ATS work:

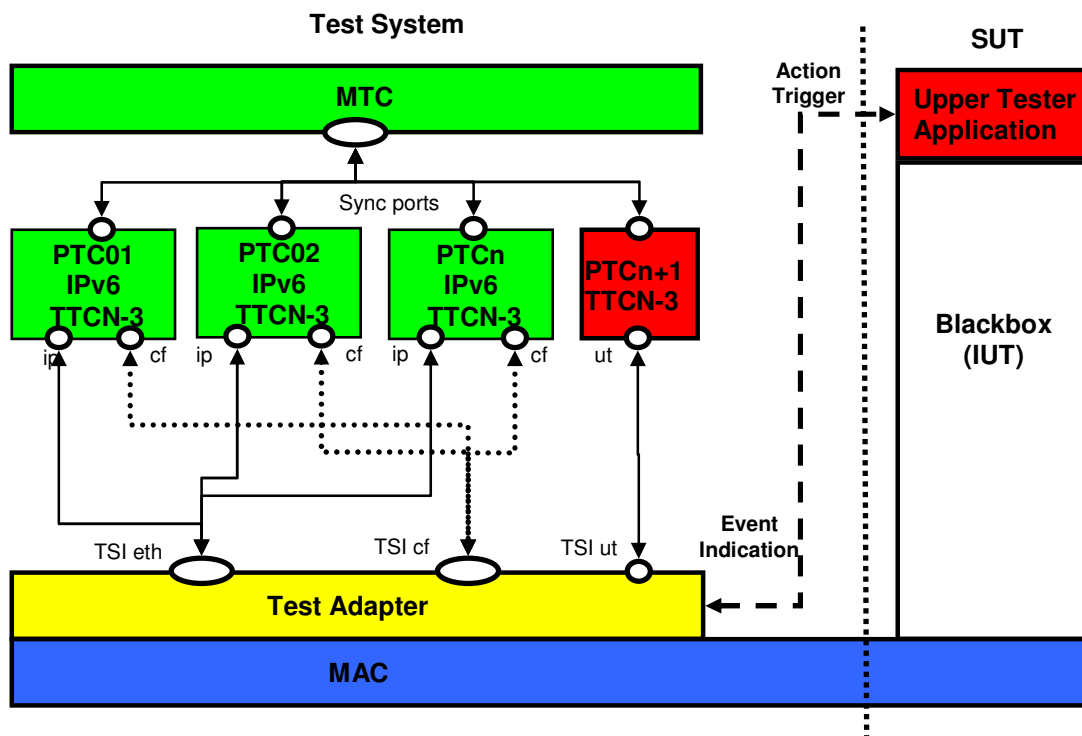



Figure 23. IPv6 Test Method

Rules for the Main Test Component (MTC) and the Parallel Test Components (PTC):

- MTC, PTC01, PTC02 to PTCn run MAC-independent TTCN-3 code.
- Each PTC has 1 cf port and 1 ip port.
- Any IPv6 message (unicast, multicast, all-nodes etc.) is sent via the ip port.
- Configuration messages are sent via the cf port in order to configure the test adapter.
- 1 and the same TSI cf port is mapped to all cf ports.
- 1 and the same TSI eth port is mapped to all ip ports.
- TTCN-3 uses ut port to control the Upper Tester Application.
- The Upper Tester Application allows to configure the IUT, trigger IUT actions and observe IUT events.
- MTC, PTC01, PTC02 to PTCn and its Test Adapter with MAC form the Lower Tester.
- MTC, PTCn + 1 and its Test Adapter with Upper Tester Application form the Upper Tester.
- Upper Tester implementation is not relevant for the prototype.

Two configurations were developed to test the two different modes for secured packet exchange, tunnel mode and transport mode. They are described in figures 2 and 3.

Tunnel mode:

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 36 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

The endpoints of communication are HS02 and NUT. Tunnel Start is RT01, Tunnel End is NUT. In the case where security parameters are negotiated with IKEv2, it is RT01 which negotiates the IKE security association.

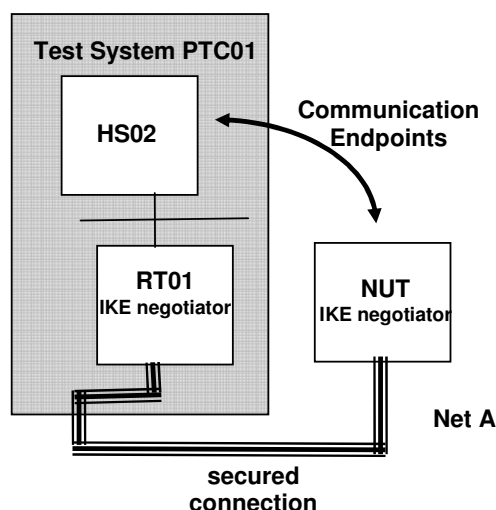


Figure 24. Tunnel Mode

Transport mode:

The endpoints of communication are HS02 and NUT. In the case where security parameters are negotiated with IKEv2, it is HS02 which negotiates the IKE security association. RT01 forwards all communication from and to HS02.

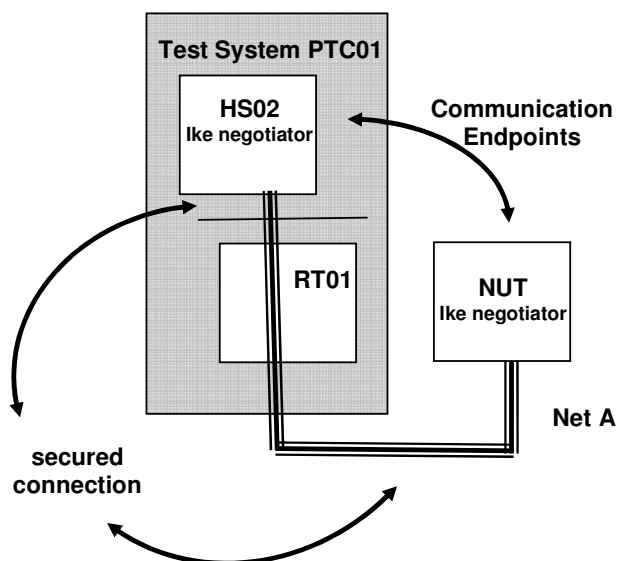



Figure 25. Transport Mode

The encryption algorithms for the IPv6 frames were not covered directly in the TTCN-3 code. Only the information about the negotiated factors were passed from the TTCN-3 code to the test adapter

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 37 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

where based on this input decryption and encryption took place. As a result, the TTCN-3 code did only deal with plain text messages and was unaware of the decoded messages. The advantages of “hiding” the en- and decryption into the test management eased the interpretation of the test results during test execution for the test operator for whom it would naturally be impossible to understand the content encrypted frames. From the total of 103 TPs a set of 13 were not implemented in the ATS due to the chosen ATM or other restrictions.

3.2.1.4 Test Environment

After the implementation of the ATS in TTCN-3 had finished, the integration into the test bed was performed. This included the production of a codec to code/decode the exchanged IPv6 frames taking into account the encryption algorithms in use.

To complement the test system the Test Adapter (TA) which connects the abstract test configuration of the ATS to the physical reality of the test bed was implemented. A set of TA requirements was defined to allow effective testing:

- TA shall receive the `AtsIpv6_TestConfiguration_TypesAndValues.CfMessage` from every participating PTC.
- TA shall set a MAC filter in order to capture IPv6 messages only.
- When receiving an IPv6 message from TRI, the TA shall:
 - assemble a MAC message containing the IPv6 message and the appropriate Mac Addresses; and
 - send the MAC message to the appropriate network device (MAC interface).
- When receiving a MAC message a network device, the TA shall:
 - extract the IPv6 message; and
 - queue the IPv6 message to the appropriate PTC.

The following figure 4 shows the overall architecture of the test bed:

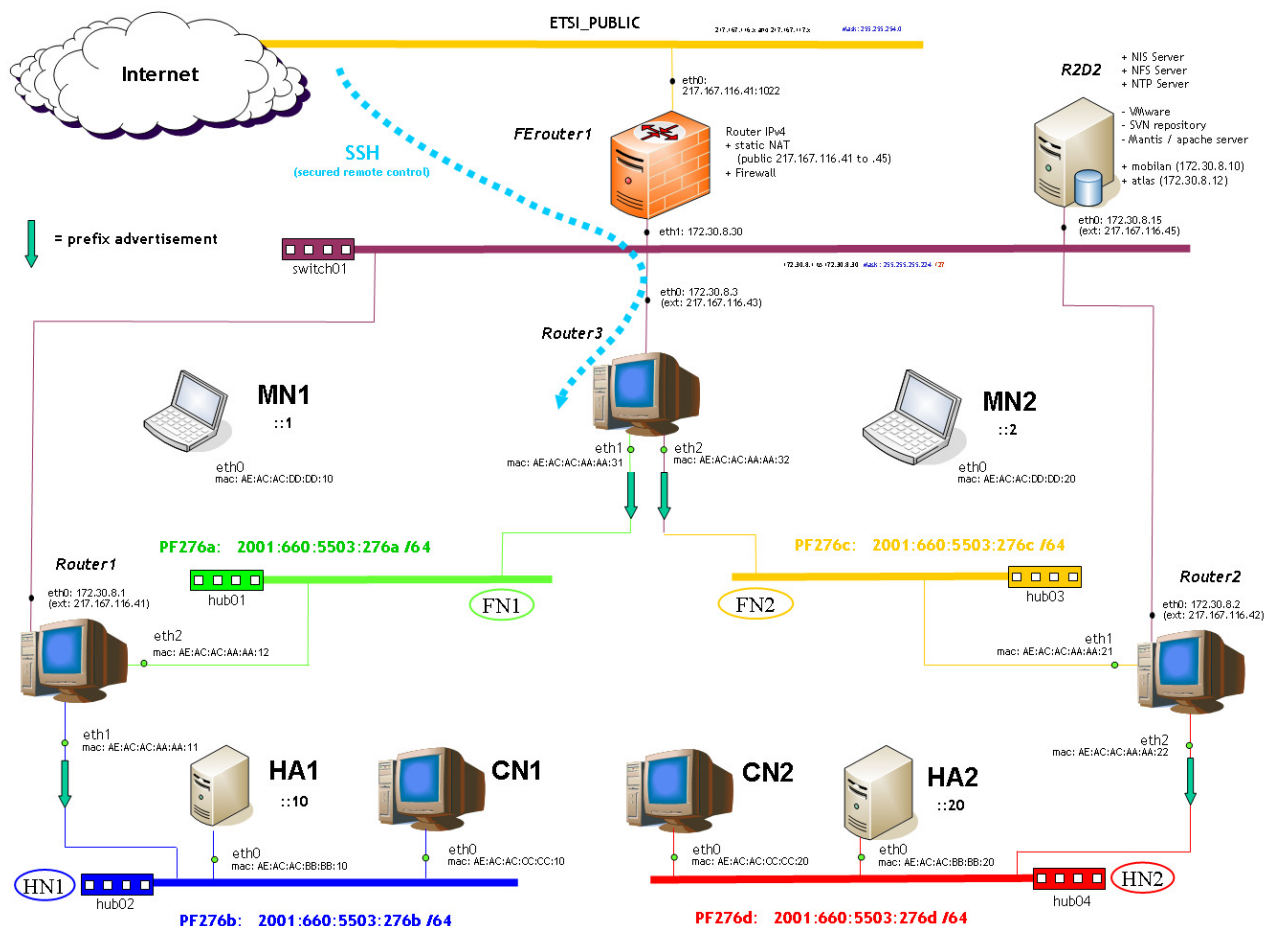


Figure 26. Testbed architecture


With this architecture it was possible to validate all the abstract test cases developed. This was done against a set of IPv6 implementations that were freely available through the internet.

3.2.2 Summary

The project developing the IPv6 testbed used traditional, i.e. mainly manual methods for producing the test specifications (Requirements, TP, ATS). This proved to be a time consuming task which may have been shortened by the use of models and automated procedures. However, at the time of the project and due to the relative complexity of the protocols under test (especially IKEv2), the manual method was the only option at hand.

As the encryption and decryption functionalities were transparent to the TTCN-3 code interpretation and validation of the test results remained independent of the encryption algorithm in use and allowed timely analysis of the test traces.

The TTCN-3 editor used was the Testing Technologies TTWorkbench Professional software that was then also used for the execution of the tests and for reporting the test results.

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 39 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

4. RANDOM, BLOCK-BASED AND MODEL-BASED FUZZING

Fuzzing is relevant to Diamonds as it is the key technology used to detect unknown zero-day vulnerabilities in security testing. Fuzzing enables software testers, developers and auditors to easily find defects that can be triggered by malformed inputs via external interfaces. This means that fuzzing is able to cover the most exposed and critical attack surfaces in a system relatively well, and identify many common errors and potential vulnerabilities quickly and cost-effectively. [60]

Fuzzing tools can be divided to four simple categories:

- Random Fuzzing
- Block-based Fuzzing
- Specification-based Fuzzing (Model-based)
- Template-based Fuzzing (Model-based)

In this chapter we will show an example of some of these.

Random Fuzzing


In his talk in the CanSecWest 2010 Dr. Charlie Miller [57] used random fuzzer which was just '5 lines of python'. The algorithm is to pick a number between 1 and (file length / 10) and randomly corrupt that many bytes in random locations. Charlie Miller comments on his slides that the approach is "so retarded" it shouldn't find any bugs at all. According to the talk, "Miller's fuzzer quickly uncovered 20 vulnerabilities across a range of applications as well vulnerabilities in Apple's Mac OS X 10.6, aka Snow Leopard, and its Safari browser. He also found the flaws in Microsoft's PowerPoint presentation maker; in Adobe's popular PDF viewer, Reader; and in OpenOffice.org, the open-source productivity suite." [29] [30]

Block-based Fuzzing

Peach is an open source fuzzing framework used to build fuzzing tools.

Name of the tool	Peach Fuzzer
Tool title phrase	Peach Fuzzer uses block-based models for fuzzing
Developed by	Michael Eddington
Maturity	Open source
Supported modelling notations (if applicable)	Proprietary PIT files, which include ASN.1, XML and BNF support
System requirements	Cross-platform
Available	http://peachfuzzer.com/
Miscellaneous	

Model-based fuzzing (both specification and template based approaches)

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 40 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

Name of the tool	Defensics 3.13
Tool title phrase	Codenomicon DEFENSICS™ test platform provides preemptive security and robustness for Internet, wireless and digital media systems
Developed by	Codenomicon
Maturity	In commercial use since 2001
Supported modelling notations (if applicable)	<ul style="list-style-type: none"> • RAW ASCII • BNF • ABNF • ASN.1 • XML/Schema • PCAP • PDML • EBNF • XML Sequences
System requirements	Cross-platform
Available	http://www.codenomicon.com/defensics/
Miscellaneous	Supports 200+ protocols

Since 2001, Codenomicon DEFENSICS™ test platform has been applying fuzzing techniques to provide preemptive security testing for network equipment manufacturers, operators, consumer electronics companies, enterprises and governmental organizations. Codenomicon DEFENSICS™ 3 bundles our accumulated experience from the past decade and takes security testing to a completely new level. The latest release of the test generation engine is 3.13 (March 2011).

The first figure shows the specification-based approach, where the model is built from protocol interface specifications. The model is pre-built by Codenomicon, but the user can edit both message sequences and the actual messages sent and received by the tool.

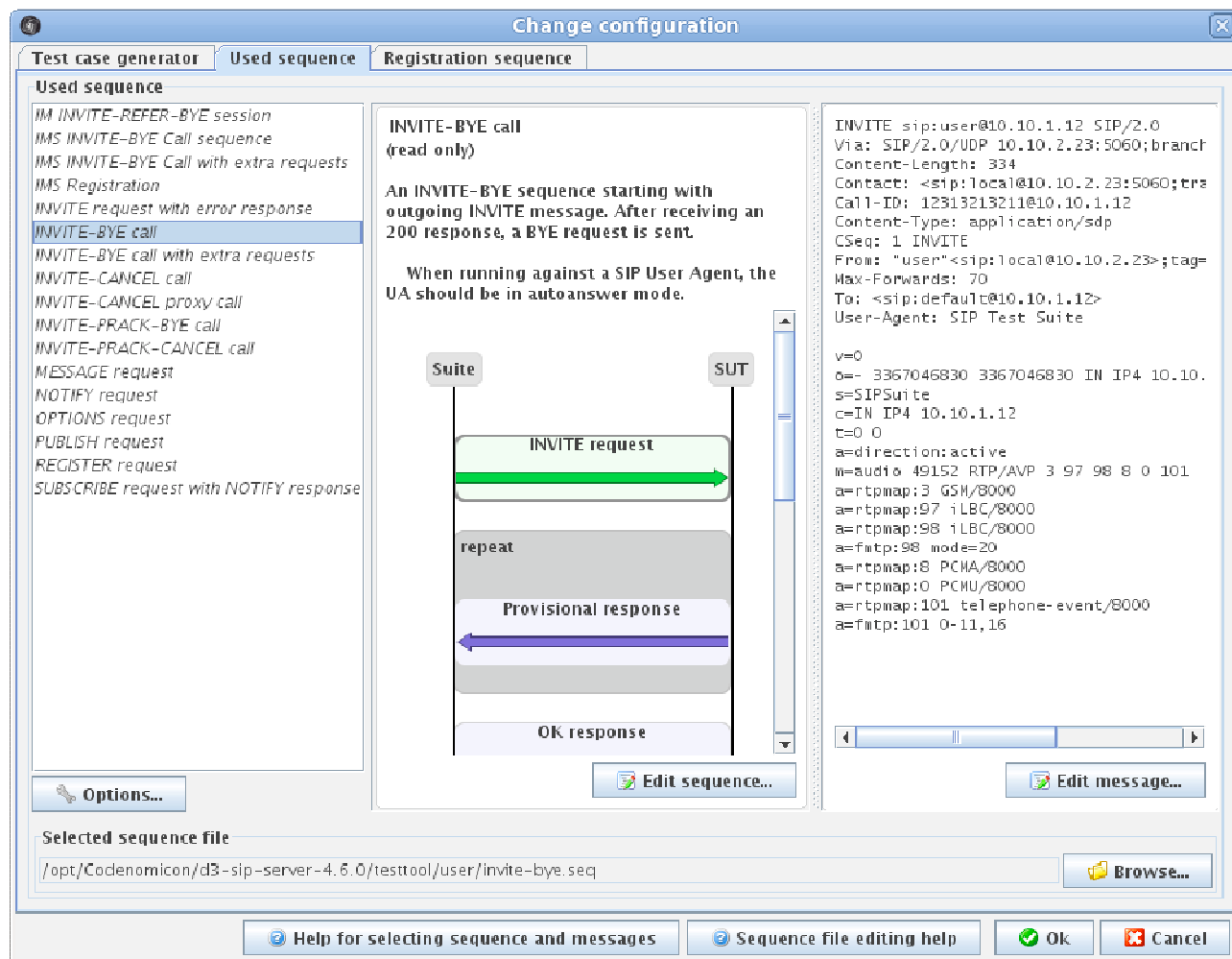


Figure 27. Specification-based approach

After model parameters are chosen, the test generator creates the test cases:

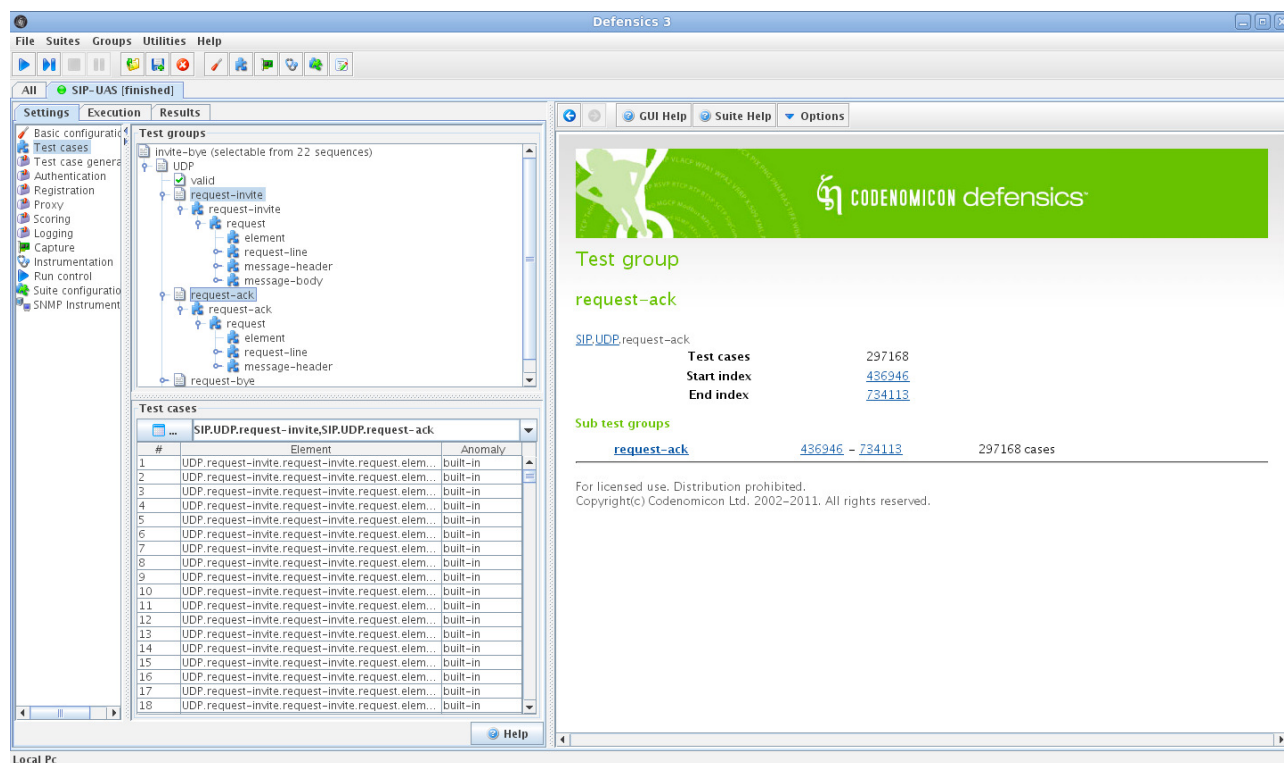



Figure 28. Test-case generation

In specification-based fuzzing, the tool user only needs to configure the required information about test target, as shown in the third figure:

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 43 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

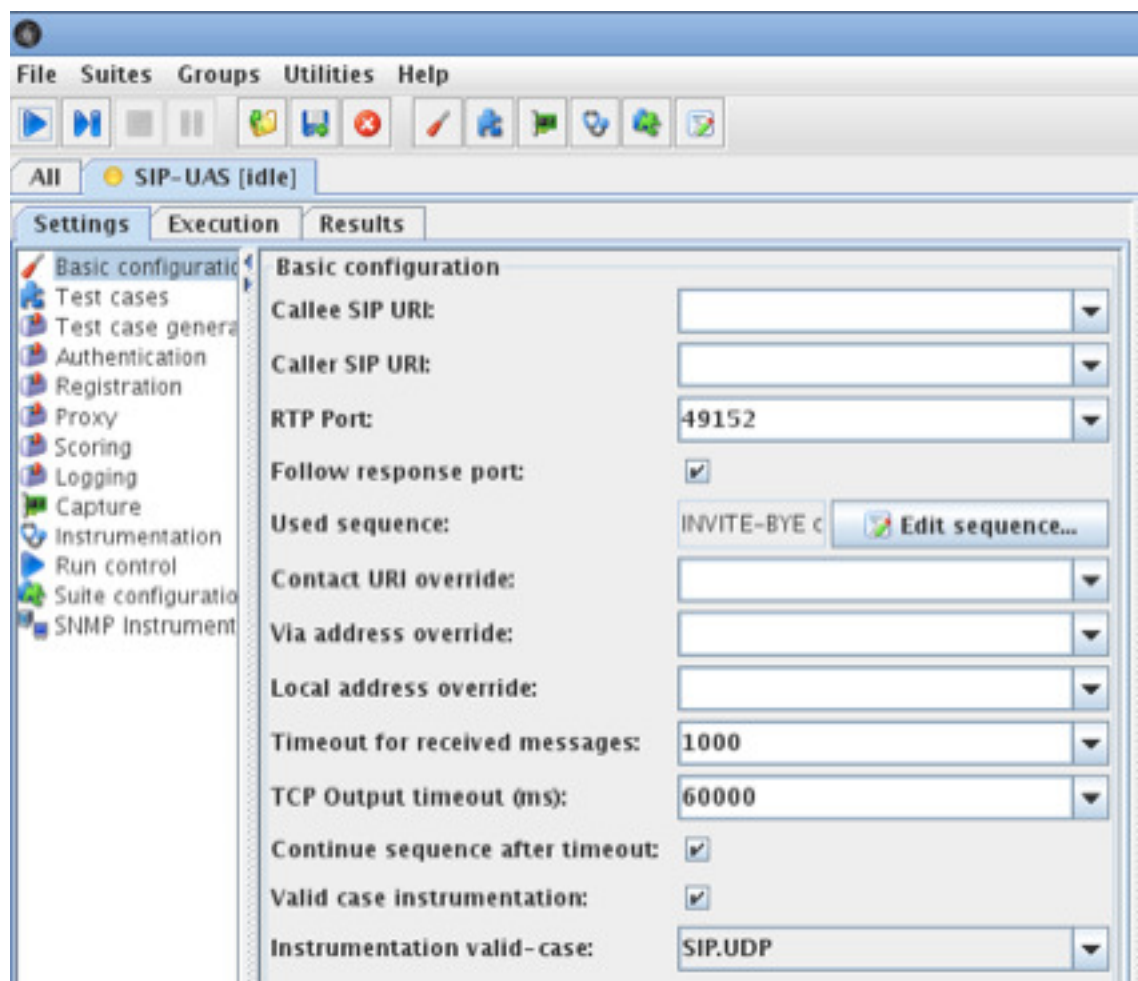


Figure 29. Test target information

The third figure shows the steps of using Defensics to create a protocol model from captured network traffic (template based fuzzing).

Step 1: Load PCAP file with network traffic and select the protocol you want to test:

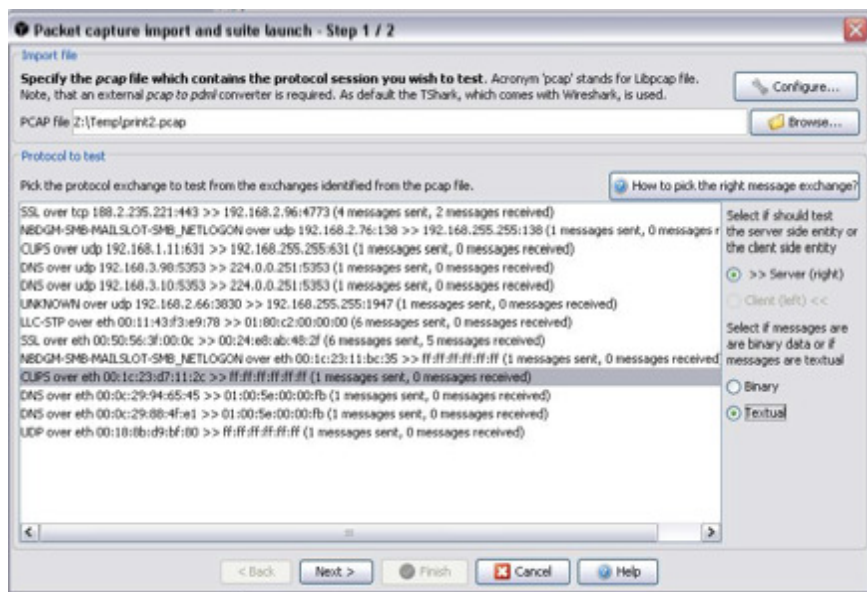


Figure 30. Step 1. Load PCAP file

Step 2: Protocol model and thousands of tests are automatically created, and the user only needs to select what elements from the protocol he wants to test:

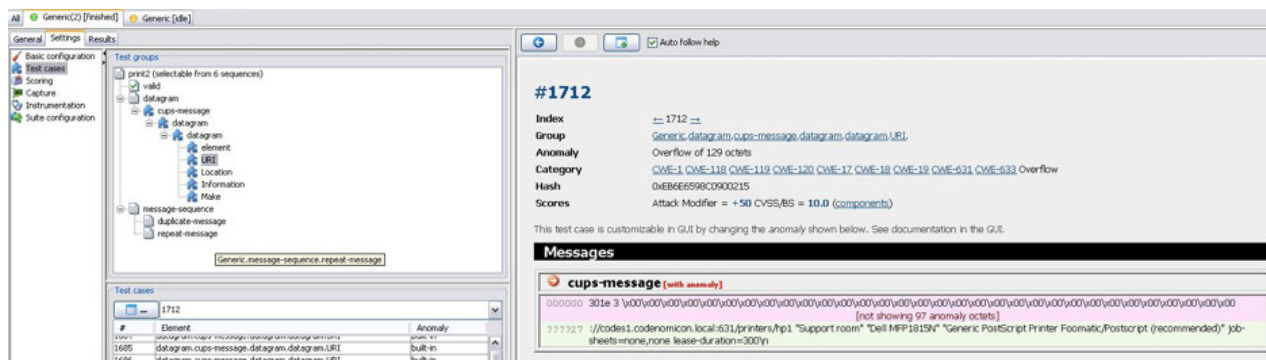



Figure 31. Step 2. Select protocol elements

Test execution and reporting work the same way as with specification-based fuzzers.

5. NETWORK SCANNING

A vulnerability scanner is a computer program designed to assess computers, computer systems, networks or applications for weaknesses. There are a number of types of vulnerability scanners available today, distinguished from one another by a focus on particular targets. While functionality varies between different types of vulnerability scanners, they share a common, core purpose of enumerating the vulnerabilities present in one or more targets. Vulnerability scanners are a core technology component of vulnerability management. [31]

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 45 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

Vulnerability scanners can be divided into several categories: port scanners, network enumerators, network vulnerability scanners, web application security scanners.

5.1 PORT SCANNERS

Name of the tool	Nmap
Tool title phrase	Nmap (Network Mapper)
Developed by	Gordon Lyon
Maturity	Stable, active
Supported modelling notations (if applicable)	XML output
System requirements	Cross-platform
Available	http://nmap.org
Miscellaneous	


Nmap is the most commonly used port scanning software, as it is state-of-the art in most aspects of security scanning. It can be used to discover hosts and services on a computer network, detect version numbers of services, as well as operating systems.

Name of the tool	Nessus
Tool title phrase	Nessus Vulnerability Scanner
Developed by	Tenable Network Security
Maturity	Commercial product
Supported modelling notations (if applicable)	XML output
System requirements	Cross-platform
Available	http://www.nessus.org
Miscellaneous	

Nessus is a comprehensive vulnerability scanning program. It is free of charge for personal use in a non-enterprise environment. Its goal is to detect potential vulnerabilities on the tested systems. For example:

- Vulnerabilities that allow a remote cracker to control or access sensitive data on a system.
- Misconfiguration (e.g. open mail relay, missing patches, etc).
- Default passwords, a few common passwords, and blank/absent passwords on some system accounts. Nessus can also call Hydra (an external tool) to launch a dictionary attack.
- Denials of service against the TCP/IP stack by using mangled packets

Name of the tool	OpenVAS
Tool title phrase	Open Vulnerability Assessment System
Developed by	Greenbone Networks
Maturity	Stable, active
Supported modelling notations (if applicable)	

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 46 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

System requirements	Cross-platform
Available	http://www.openvas.org/
Miscellaneous	

OpenVAS is a fork of the Nessus scanner from the last GPL-licensed version, Nessus 2. It is a framework of several services and tools offering a comprehensive and powerful vulnerability scanning and vulnerability management solution.

The actual security scanner is accompanied with a daily updated feed of Network Vulnerability Tests (NVTs), over 20,000 in total (as of January 2011).

6. MONITORING TOOLS FOR DETECTING VULNERABILITIES

6.1 INTRUSION DETECTION SYSTEMS

Intrusion detection systems (IDS) can be divided into roughly two categories, those two categories being network based and host based intrusion detection systems (NIDS and HIDS respectively). IDSs also include intrusion prevention systems (IPS), which are subset of the IDS. IPSs are IDSs with mechanisms built into them that can be used to respond to an intrusion. Intrusion detection systems report suspicious activity to the analyst in the form of alerts. The analyst's task is then to decide which alerts to act upon. In IPS systems some responsive activity is typically included when the action activation parameters are fulfilled. Some IDSs can be configured to keep various logs for forensic purposes, and some include extensive traffic analysis capabilities in addition to the more typical intrusion detection functions. IDSs are not used to limit the functionality or connections in the network in a way of firewalls. They are usually used only to alert users to attacks. IDSs with prevention functionality do overlap with application layer firewalls, and can be seen as one.


The main sources apart from the writers experience used for this section are: "The Tao of Network Security Monitoring" by Richard Bejtlich [32], and "Security Engineering", 2nd edition, by Ross Anderson [33].

6.1.1 Network Based Intrusion Detection Systems

Network based intrusion detection systems are used to monitor network traffic and alert on suspicious activity that is not consistent with network policy. Typically one network node is tapped from which the NIDS then gains its input. What network node should actually be tapped for the NIDS depends on the network structure in use. However, IDS systems in general function best in environments with limited amounts of noise. In very noisy environments the systems typically produce large amounts of alerts including a number of false positives depending on the system in use.

Network policy is typically defined to the NIDS using either a scripting language or a set of signatures known to be malignant. Both approaches are used in systems available today, and arguments to favour either one exist.

Scalability issues that plague NIDS installations in high volume network environments is discussed in the Section 6.1.3.

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 47 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

6.1.2 Host Based Intrusion Detection Systems

As the name implies, host based intrusion detection systems monitor the internal state of the host system they reside in. They monitor the resource usage and behaviour of various system components and software, and attempt to detect any malicious or suspicious activity by doing this. As NIDS is monitoring the success of enforcing network usage policies, HIDS monitors the users or systems adherence to the usage policies of the system. HIDS do not suffer from the same scalability issues as NIDS due to the differences in their operating environment. However, they do incur some overhead, the level of which depends on the strictness of the used detection methods and the context.

Differentiating between HIDS and anti-virus systems is not a straight-forward task. The two can include a lot of overlapping mechanisms and functionality.

6.1.3 Scalability

A single NIDS instance is classically used to monitor the network through a single network tap. This places significant demands on the ability of the hardware to cope with increasing amounts of traffic. Scalability of the systems is therefore an important priority, which some single-machine instance systems lack. The problem has been approached using several approaches, including:

- Increasing computing power of single-instance systems using of-the-shelf components,
- Increasing computing power of single-instance systems using specialized hardware components designed for the explicit purpose of high-bandwidth network monitoring.
- Building NIDS clusters to distribute the monitoring load over several back end machines running instances of the NIDS and controlled by front end systems.


All the approaches have their challenges, and the approach recommended to any network is dependent on the specific network in use and its context. With the networks with most high levels of traffic the most likely solution able to cope with realistic costs is the cluster approach.

6.1.4 Challenges

NIDS and HIDS have very different operational environments, but their main goal remains the same: To maximize the amount of true positives, preferably all, while minimizing the amount false positives and especially false negatives. This need drives the work on these systems. The challenges to this are listed in the following subchapters.

6.1.4.1 *NIDS*

- High volume networks demand flexibility and scalability.
- High false positive rates can drown out the true positives. This is especially true in very noisy environments.
- Evasion techniques are proliferating. Some techniques used in intentional attacks are aimed to evade NIDS, leading to a race between hackers and NIDS developers.

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 48 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

- Low number of actual incidents can lead to problems with the human analysts monitoring the NIDS. Experience is needed when operating them and weeding out the false positives.
- Network security policy must be well defined to the NIDS, typically using either scripts or signatures.
- Deployment of IPv6 networks will cause new attacks to emerge, it will take a while until the IPv6 detection capabilities are on par with the IPv4 capabilities.

6.1.4.2 HIDS

- Protection of the HIDS itself is critical. Typically attempts are made to subvert the HIDS as soon as the attacker gains access to the host system.
- Protection of HIDS logs is important. Writing logs to a location where they cannot be modified by the system through one-way communication pathway is one option. After compromising a system, any log files that could be used for forensics are likely to be targeted by attackers.
- HIDS generate overhead must be low enough not to hinder operations of the host system.
- Security policy must be explicitly stated and defined in an accurate way to the HIDS, this is a challenge in a dynamic and rapidly changing environment.

6.1.5 Examples of Current Intrusion Detection Systems

Few available IDS systems of different types are presented in this section. Presenting all available tools in this category is not feasible, but attempt has been made to adequately present current tools from different areas. More tools are also presented in section 7.2 dealing with intrusion prevention systems (IPS).

6.1.5.1 Snort

Snort is one of the most commonly deployed NIDS/IPS. Snort is open source software with commercial support and update through Sourcefire Inc. Snort is signature based and Sourcefire provides updates to the database containing them. The paying customers get the new signature updates first, and the rest of the users a defined period of time later. Snort can also be deployed in other modes besides IDS, those being sniffer and packet logger.

Name of the tool	Snort
Tool title phrase	Network Based Intrusion Detection System
Developed by	Sourcefire, Inc. Originally created by Martin Roesch
Maturity	Mature Open Source Software with GNU general public license.
Supported modelling notations (if applicable)	Snort signature format. Uses signature data base to detect suspicious activity.
System requirements	Depends on the volume of the monitored network. Runs on commodity hardware.
Available	http://www.snort.org
Miscellaneous	Widely used to monitor various types of network. Commercial support available from Sourcefire, Inc.

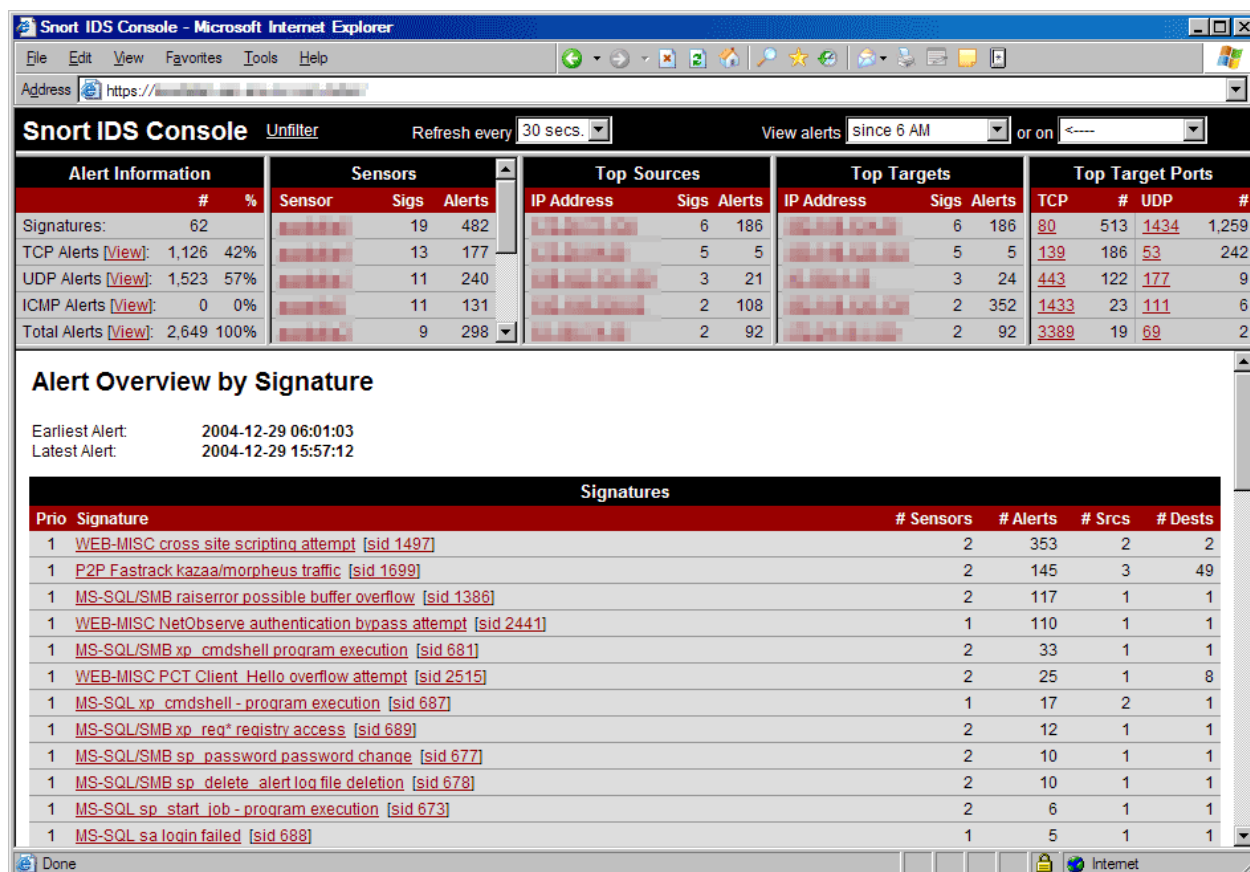



Figure 32. SNORT IPS Console

6.1.5.2 BRO

Name of the tool	Bro IDPS
Tool title phrase	Network Based Intrusion Detection and Prevention System
Developed by	Lawrence Berkeley National Laboratories, International Computer Science Institute, Originally created by Vern Paxson at LBL
Maturity	Mature Open Source Software with BSD license.
Supported modelling notations (if applicable)	Bro Scripting Language, domain specific scripting language for stating network policy to the system.
System requirements	Depends on the volume of the monitored network. Runs on commodity hardware
Available	http://www.bro-ids.org
Miscellaneous	Being used to monitor University of California, Berkeley high volume network using Bro cluster. No commercial support available.

Bro is an open-source, Unix-based Network Intrusion Detection System (NIDS) that passively monitors network traffic and looks for suspicious activity. Bro detects intrusions by first parsing network traffic to extract its application-level semantics and then executing event-oriented analyzers that compare the activity with patterns deemed troublesome. Its analysis includes detection of specific attacks (including those defined by signatures, but also those defined in terms of events) and un-

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 50 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

sual activities (e.g., certain hosts connecting to certain services, or patterns of failed connection attempts).

Bro uses a specialized policy language that allows a site to tailor Bro's operation, both as site policies evolve and as new attacks are discovered. If Bro detects something of interest, it can be instructed to either generate a log entry, alert the operator in real-time, execute an operating system command (e.g., to terminate a connection or block a malicious host on-the-fly). In addition, Bro's detailed log files can be particularly useful for forensics.

Bro targets high-speed (Gbps), high-volume intrusion detection. By judiciously leveraging packet-filtering techniques, Bro is able to achieve the necessary performance while running on commercially available PC hardware, and thus can serve as a cost-effective means of monitoring a site's Internet connection.

6.1.5.3 Suricata NIDS

Suricata is a fairly recent NIDS that had a beta version release during 2009. It uses the same ruleset for detection as Snort maintained by Sourcefire Inc. It also supports multithreading which is not currently available in all of the systems. It is extensible but documentation on their web site is limited.

How Suricata differs in use with Snort should be tested before commenting. The detection seems to be fairly similar, since it is based on Snort signatures but with some additional plugins by the Suricata team.


Name of the tool	Suricata
Tool title phrase	Network Based Intrusion Detection System
Developed by	The Open Information Security Foundation
Maturity	Stable release available, relatively new
Supported modelling notations (if applicable)	Snort signature format. Uses signature data base to detect suspicious activity
System requirements	Runs on commodity hardware
Available	http://www.openinfosecfoundation.org/
Miscellaneous	

6.1.5.4 StoneGate IPS

StoneGate IPS is a member of the proprietary StoneGate- family of products by Stonesoft, a company based in Finland. Information available concerning the internals of the system is restricted due to its commercial nature. StoneGate is a NIDS, monitoring web traffic.

All the StoneGate products can be centrally managed using a management center provided by Stonesoft. StoneGate IPS is available as either software or dedicated appliance.

Name of the tool	StoneGate IPS
Tool title phrase	Network Based Intrusion Detection System
Developed by	Stonesoft Oy
Maturity	Mature and proprietary tool
Supported modelling	-

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 51 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

notations (if applicable)	
System requirements	Intel- based platform with a limited number of actively certified platforms
Available	http://www.stonesoft.com/en/products/ips/
Miscellaneous	

6.1.5.5 *Samhain HIDS*

Samhain is an open source HIDS capable of centralized monitoring and management (client/server configuration). It can also be used as a standalone installation. Its main tasks are focused on file and host integrity monitoring. Samhain also includes stealth features, it can attempt to hide its existence through various ways, for example a hidden linux kernel module is available.


Samhain comes with a selection of different modules, from which the users can select what they need. Information for creating additional modules is also provided in the documentation. Samhain can also be compiled to include support for Prelude or Nagios integration. For more information on Prelude visit: <http://www.prelude-technologies.com/en/welcome/index.html> and for Nagios: <http://www.nagios.org/>

Name of the tool	Samhain HIDS
Tool title phrase	Host Based Intrusion Detection System
Developed by	Samhain Labs
Maturity	GNU General Public License
Supported modelling notations (if applicable)	NA
System requirements	Any POSIX compliant platform, commodity hardware.
Available	http://www.la-samhna.de/samhain/
Miscellaneous	Central monitoring and stealth attributes.

6.2 NETWORK MONITORING TOOLS

The term network monitoring describes the use of a system that constantly monitors a computer network for slow or failing components and that notifies the network administrator (via email, pager or other alarms) in case of outages. It is a subset of the functions involved in network management. Monitoring also can refer to passive analysis of network traffic at either the packet or flow level. SLAC maintains a comprehensive list of products at <http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html>.

In security testing, monitoring tools play a vital part. They can be used to understand the effects of testing, starting from the low level of study of the packet payloads of different test cases and the traffic test subjects generate as a result, to measuring the wide-scale effects of testing on, e.g., latency or availability of different nodes as a result of testing. Also, the system under test needs to be monitored to observe the attack surface in the system. A network monitoring tool will complement scanning technologies by revealing client side implementations, as more and more attacks target client implementations such as browsers and components that reach out to Internet for e.g. network time synchronization and network name services. Secondly, network and system monitoring tools are used during the test to analyze complex failures when security tests such as Fuzzing

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 52 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

is being performed. This is a critical capability for security testing in order to be able to fix the found anomalous behaviour.

6.2.1 Wireshark

Name of the tool	Wireshark
Tool title phrase	Packet Analyzer
Developed by	The Wireshark team
Maturity	Stable release 1.6.0 – Free and open source
Supported modelling notations (if applicable)	Not applicable
System requirements	Unix-like operating systems including Linux, Mac OS X, BSD, and Solaris, and on Microsoft Windows
Available	http://www.wireshark.org/download.html
Miscellaneous	GNU General Public License

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. It allows capturing and interactively browsing the traffic running on a computer network. Originally named Ethereal, in May 2006 the project was renamed Wireshark due to trademark issues. In the domain of security testing, Wireshark is widely used due to its support for a large number of network protocols, verifying the contents and validity of different fields etc.

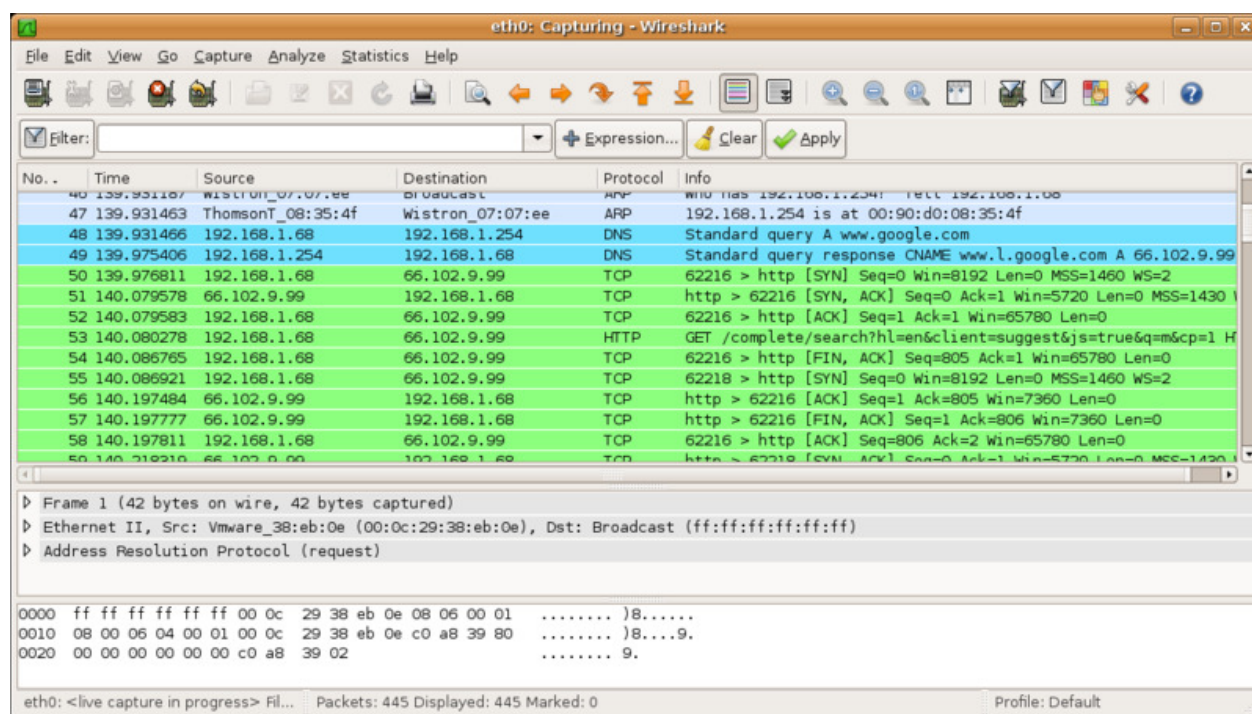



Figure 33. Wireshark

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 53 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public


6.2.2 OpenNMS

Name of the tool	OpenNMS
Tool title phrase	OpenNMS enterprise grade network monitoring and management platform
Developed by	Tarus Balog, OpenNMS Group
Maturity	Active, stable
Supported modelling notations (if applicable)	
System requirements	Cross-platform, written in Java
Available	http://www.opennms.org/
Miscellaneous	GPL license

OpenNMS is an example of a enterprise grade network monitoring and management system, which can be used as a part of a security testing framework to measure wide-scale effects of testing on the network.

It contains the following features, which are representative of the functionality done by products in this area:

- Service polling - determining service availability and latency, including distributed measurement of availability and latency, and reporting on the results
- Data collection - collecting, storing and reporting on data collected from nodes via protocols including SNMP, JMX, HTTP, Windows Management Instrumentation, JDBC[1], and NSClient
- Thresholding - evaluating polled latency data or collected performance data against configurable thresholds, creating events when these are exceeded or rearmed
- Event management - receiving events, both internal and external, including via SNMP traps
- Alarms and automations - reducing events according to a reduction key and scripting automated actions centered around alarms
- Notifications - sending notices regarding noteworthy events via e-mail, XMPP, or other means

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 54 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

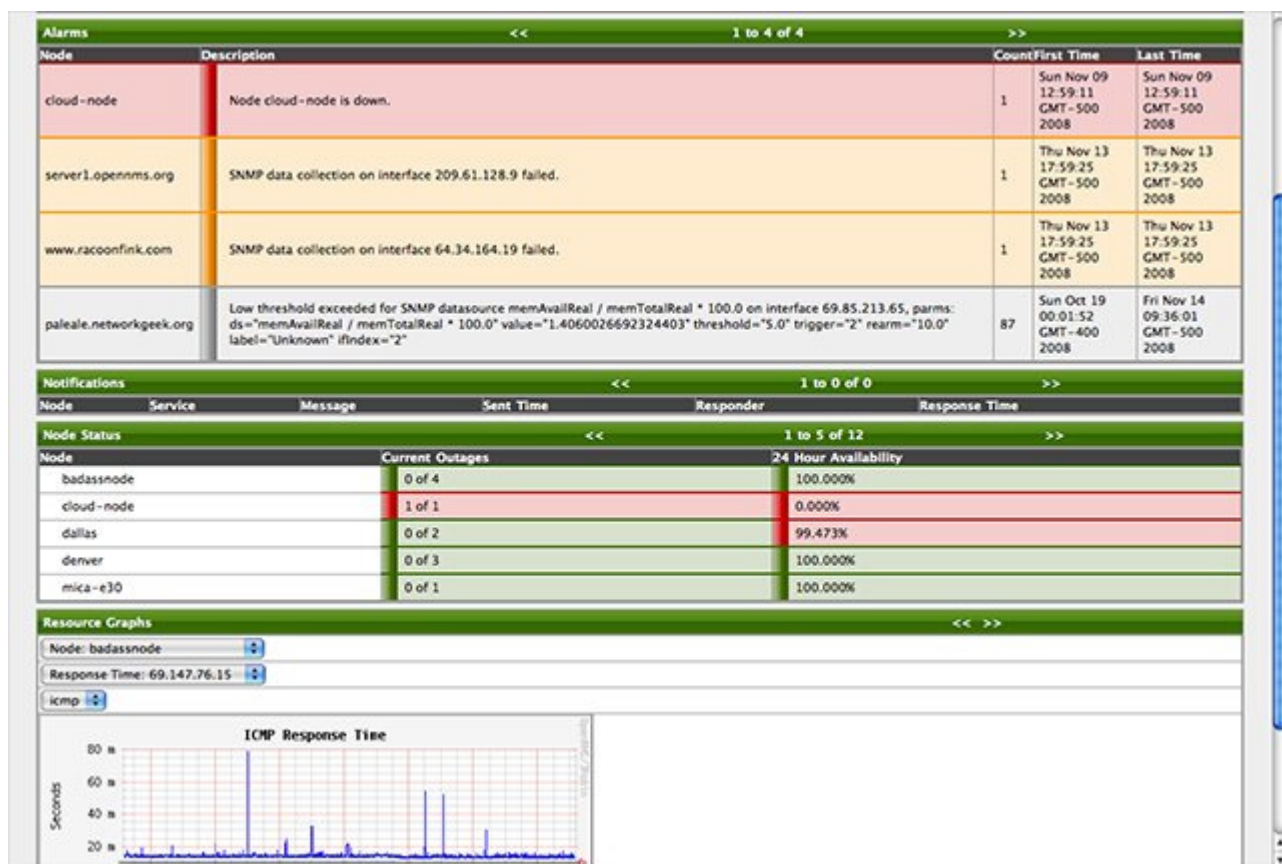



Figure 34. OpenNMS

6.2.3 OmniPeek

Name of the tool	OmniPeek
Tool title phrase	OmniPeek Network Analyzer
Developed by	WildPackets
Maturity	Commercial product
Supported modelling notations (if applicable)	
System requirements	Windows XP or later, P4 2.4 GHZ, 4 GB RAM
Available	http://www.wildpackets.com/products/
Miscellaneous	

OmniPeek gives network engineers real-time visibility and Expert Analysis into every part of the network from a single interface, including Ethernet, Gigabit, 10 Gigabit, 802.11a/b/g/n wireless, VoIP, and Video to remote offices. Using OmniPeek's intuitive user interface and "top-down" approach to visualizing network conditions, network engineers—even junior staff—can quickly analyze, drill down and fix performance bottlenecks across multiple network segments, maximizing uptime and user satisfaction.

6.2.4 Clarified Analyzer

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 55 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

Name of the tool	Clarified Analyzer
Tool title phrase	Clarified Analyzer is the tool of choice for collaborative analysis and visualization of complex networks.
Developed by	Clarified Networks Oy
Maturity	Commercial product
Supported modelling notations (if applicable)	
System requirements	Cross-platform
Available	https://www.clarifiednetworks.com/Clarified%20Analyzer
Miscellaneous	


Clarified Analyzer is a commercial network analysis tool that provides intuitive visualizations into raw network data, and has a focus on analyses that support security testing. Typical Users of Clarified Analyzer include testlab managers for distributed testlabs, cybercrime fighters, critical infrastructure providers, network auditors, security & PCI compliance auditors and VOIP solution integrators.



6.2.5 Tcpxtract

Name of the tool	tcpxtract
Tool title phrase	tcpxtract is a carver for network data
Developed by	Nick Harbour
Maturity	Open source, last version in 2005
Supported modelling notations (if applicable)	
System requirements	UNIX
Available	http://tcpxtract.sourceforge.net/
Miscellaneous	

tcpxtract is a tool for extracting files from network traffic based on file signatures. Extracting files based on file type headers and footers (sometimes called "carving") is an age old data recovery technique. Tools like Foremost employ this technique to recover files from arbitrary data streams. Tcpxtract uses this technique specifically for the application of intercepting files transmitted across a network. Other tools that fill a similar need are driftnet and EtherPEG. driftnet and EtherPEG are

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 56 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

tools for monitoring and extracting graphic files on a network and is commonly used by network administrators to police the internet activity of their users. The major limitations of driftnet and EtherPEG is that they only support three filetypes with no easy way of adding more. The search technique they use is also not scalable and does not search across packet boundaries. tcpextract features the following:

- Supports 26 popular file formats out-of-the-box. New formats can be added by simply editing its config file.
- With a quick conversion, you can use your old Foremost config file with tcpextract.
- Custom written search algorithm is fast and very scalable.
- Search algorithm searches across packet boundaries for total coverage and forensic quality.
- Uses libpcap, a popular, portable and stable library for network data capture.
- Can be used against a live network or a tcpdump formatted capture file.

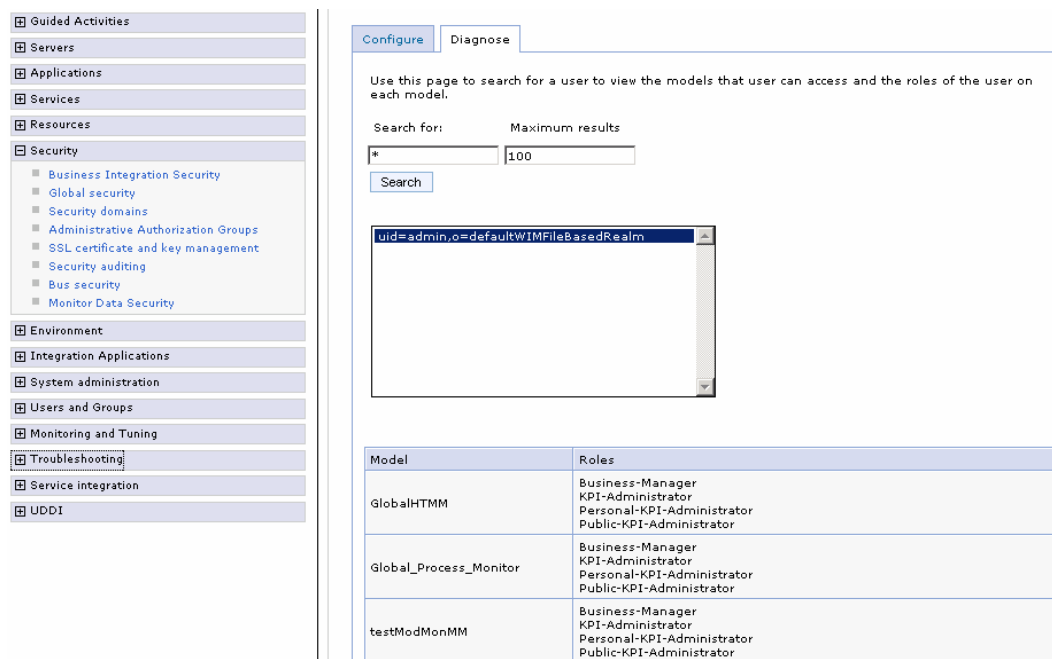
What makes tools, such as tcpextract especially relevant for the project is that they make it possible to automatically extract samples to be used for model-inference based fuzzers from live networks.

6.3 BUSINESS ACTIVITY MONITORING

Business Activity Monitoring (BAM) broke onto the scene three or four years ago, stimulated by the growing interest in Business Process Management (BPM), which made it possible to understand more clearly the relationship between real-time IT operations and business activities. Business activity monitoring (BAM) [8] is Gartner's term defining how we can provide real-time access to critical business performance indicators to improve the speed and effectiveness of business operations. Unlike traditional real-time monitoring, BAM draws its information from multiple application systems and other internal and external (inter-enterprise) sources, enabling a broader and richer view of business activities. As such, BAM will be a natural extension of the investments that enterprises are making in application integration.

6.3.1 IBM Business Monitor

Name of the tool	IBM Business Monitor V7.5
Tool title phrase	Cross-process, cross-system business activity monitoring (BAM) software
Developed by	IBM
Maturity	Commercial tool
Supported modelling notations (if applicable)	WebSphere ILOG Rules
System requirements	Operating systems AIX, HP, Linux, Mobile OS, Solaris, Windows, z/OS.
Available	http://www-01.ibm.com/software/integration/business-monitor/
Miscellaneous	



Model	Roles
GlobalHTMM	Business-Manager KPI-Administrator Personal-KPI-Administrator Public-KPI-Administrator
Global_Process_Monitor	Business-Manager KPI-Administrator Personal-KPI-Administrator Public-KPI-Administrator
testModMonMM	Business-Manager KPI-Administrator Personal-KPI-Administrator Public-KPI-Administrator

Figure 35. IBM Business Monitor configuration

IBM Business Monitor (formerly IBM WebSphere Business Monitor) provides end-to-end business process and activity monitoring along with dashboards representing insight that can be used in process optimization.

- Provides a high-performance business activity monitoring solution for processes and applications running in disparate environments which may or may not be implemented using any BPM technology.
- Built-in tools and runtime support for integrated Business Activity Monitoring of IBM Business Process Manager
- Fine-grained security to enable or prevent anyone to see a wide range of information depth or detail
- Enhanced business user customization of data filtering and dashboard controls & reports.
- Enable views of KPIs, metrics, and alerts through Web interfaces, mobile devices, and corporate portals.

6.3.2 Oracle Business Activity Monitoring

Name of the tool	Oracle Business Activity Monitoring
Tool title phrase	Monitoring business processes and services
Developed by	Oracle
Maturity	Commercial
Supported modelling notations (if applicable)	
System requirements	Operating systems AIX, HP, Linux, Mobile OS, Solaris, Windows, z/OS.
Available	http://www.oracle.com/technetwork/middleware/bam/downloads/index.html
Miscellaneous	Oracle BAM is available for stand-alone or integrated installation in this integrated installation for the Oracle SOA Suite. Directions to install Oracle BAM stand-alone or Oracle BAM with the other components of the SOA Suite are available in the SOA Suite Install Guide documentation.



Figure 36. Oracle BAM

Oracle Business Activity Monitoring (Oracle BAM) is a complete solution for building interactive, real-time dashboards and proactive alerts for monitoring business processes and services. Oracle BAM gives business executives and operation managers the information they need to make better business decisions and take corrective action if the business environment changes.


6.4 DATABASE ACTIVITY MONITORING

Database activity monitoring (DAM) [9] is a database security technology for monitoring and analyzing database activity that operates independently of the database management system (DBMS) and does not rely on any form of native (DBMS-resident) auditing or native logs such as trace or transaction logs [10]. DAM is typically performed continuously and in real-time.

Database monitors and log monitors are relevant for Diamonds as they can be used to detect failures and anomalies in database access in order to detect zero-day attacks such as SQL injection. Also load-based DDoS attacks can be observed with these tools.

6.4.1 IBM InfoSphere Guardium

Name of the tool	IBM InforSphere Guardium
Tool title phrase	Real-Time Database Activity Monitoring
Developed by	IBM
Maturity	Commercial tool
Supported modelling notations (if applicable)	A user interface to design monitoring and security rules is provided within the tool

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 59 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

System requirements	InfoSphere Guardium's cross-platform solution supports all major DBMS platforms and protocols running on all major operating systems (Windows, UNIX, Linux, z/OS), as well as Microsoft SharePoint and FTP environments
Available	http://www-01.ibm.com/software/data/guardium/database-activity-monitor/
Miscellaneous	The IBM InfoSphere Guardium solution continuously monitors database transactions through lightweight software probes installed on the database servers. The probes monitor all database transactions, including those of privileged users, at the operating system kernel level without relying on database audit logs.

IBM InfoSphere Guardium provides a simple and robust solution for assuring the privacy and integrity of trusted information in many kinds of data centers (SAP, PeopleSoft, Cognos, Siebel, etc.) and reducing costs by automating the entire compliance auditing process in heterogeneous environments.

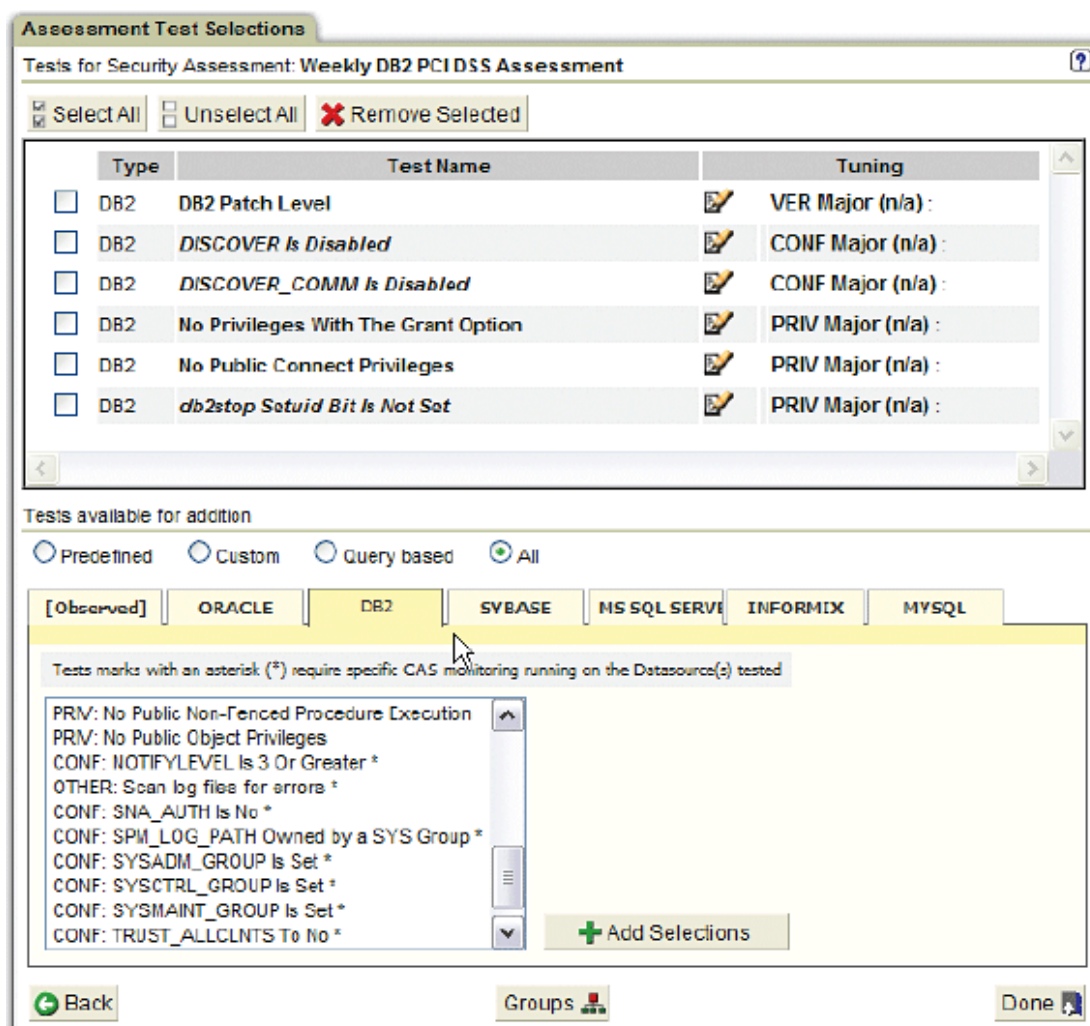



Figure 37. IBM InfoSphere Guardium user interface


It deploys centralized and standardized controls for real-time database security and monitoring, fine-grained database auditing, automated compliance reporting, data-level access control, database vulnerability management and auto-discovery of sensitive data. It allows to:

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 60 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

- Prevent Cyberattacks with Database Activity Monitoring: Proactively identify unauthorized or suspicious activities by continuously tracking all database actions -- without impacting performance or modifying databases.
- Monitor Privileged Users: Detect or block malicious or unapproved activity by DB administrators, developers and outsourced personnel without relying on native logs, triggers or other DBMS-resident mechanisms.
- Monitor Enterprise Application Users for Fraud: Identify end-user fraud with application-layer monitoring for multi-tiered environments (SAP, PeopleSoft, Cognos, etc.).
- Audit and Validate Compliance: Simplify SOX, PCI-DSS, and Data Privacy processes with pre-configured reports and automated oversight workflows (electronic sign-offs, escalations, etc.).
- Enforce Database Change Control: Ensure data governance by preventing unauthorized changes to critical database values or structures.
- Prevent Database Leaks: Detect and block leakage in the data center.

6.4.2 dbWatch

Name of the tool	bdWatch
Tool title phrase	Database monitoring tool for managing critical database environments
Developed by	dbWatch Software
Maturity	Commercial tool
Supported modelling notations (if applicable)	No reference
System requirements	Windows or Linux Server (VMWare virtual server supported), 1 GB RAM, 1 GB HD Space
Available	http://www.dbwatchsoftware.com/Download-Evaluation
Miscellaneous	

	Review of security testing tools Deliverable ID: D1_1	Page : 61 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

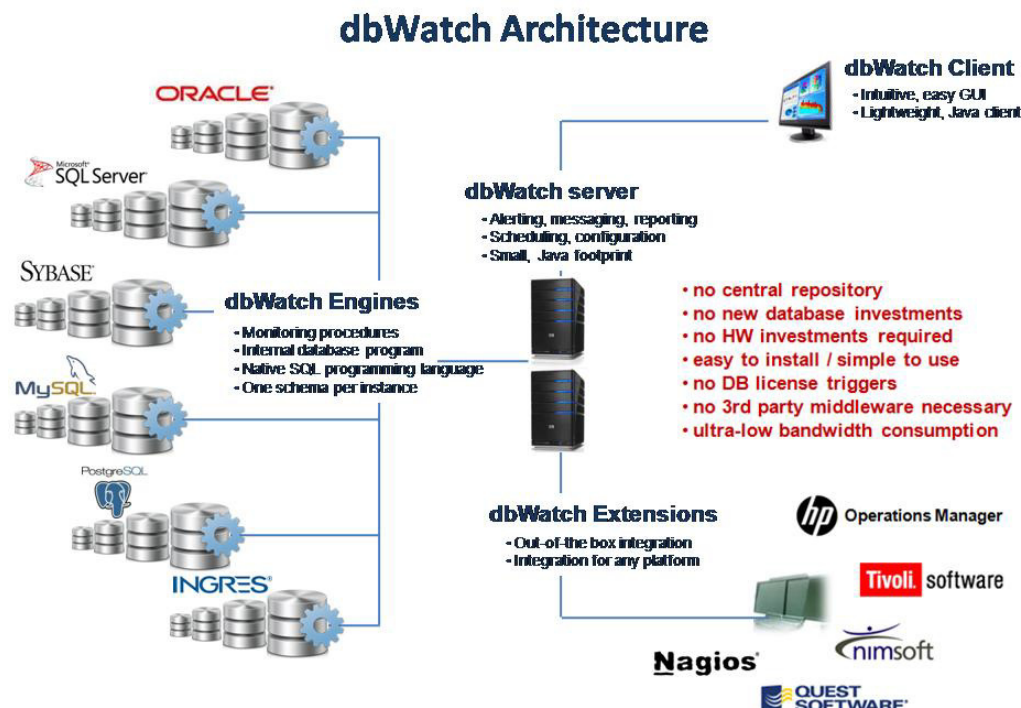


Figure 38. dbWatch Architecture


dbWatch is a cross platform database monitoring tool for managing critical database environments. Designed to be lightweight and easy to use, dbWatch does not sacrifice functionality or capabilities. dbWatch's architecture is designed to support very large database environments. dbWatch supports IT departments with multiple database technologies because it works with across most platforms. With dbWatch, users including professional database administrators (DBA), IT operations managers, and IT Security Officers can create their own monitoring procedures customized to their organization's needs, allowing to monitor their business processes, no matter how complex. dbWatch is an interesting tool for monitoring any of the following:

- large database environments,
- heterogeneous database environments, and
- complex databases

In addition to broad and in-depth monitoring capabilities, dbWatch also has excellent reporting and SQL administration tools.

6.4.3 DB Audit 4.2.29

Name of the tool	DB Audit 4.2.29
Tool title phrase	Database Auditing, Compliance & Security Solutions
Developed by	Soft Tree Technologies
Maturity	Commercial tool
Supported modelling notations (if applicable)	No reference
System requirements	Full compatibility with all host Operation Systems on which the supported databases can run, including but not limited to Windows NT, UNIX, Linux, VMS, OS/390, z/OS.
Available	http://www.softtreotech.com/idbaudit.htm

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 62 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

Miscellaneous

DB Audit Expert is a professional all-in-one database security and auditing solution for Oracle, Sybase, DB2, MySQL and Microsoft SQL Server. DB Audit Expert enables database and system administrators, security administrators, auditors and operators to track and analyze any database activity including database security, access and usage, data creation, change or deletion. What makes DB Audit really unique is its built-in support for multiple auditing methods giving you the flexibility to choose the best fit for your database security requirements.

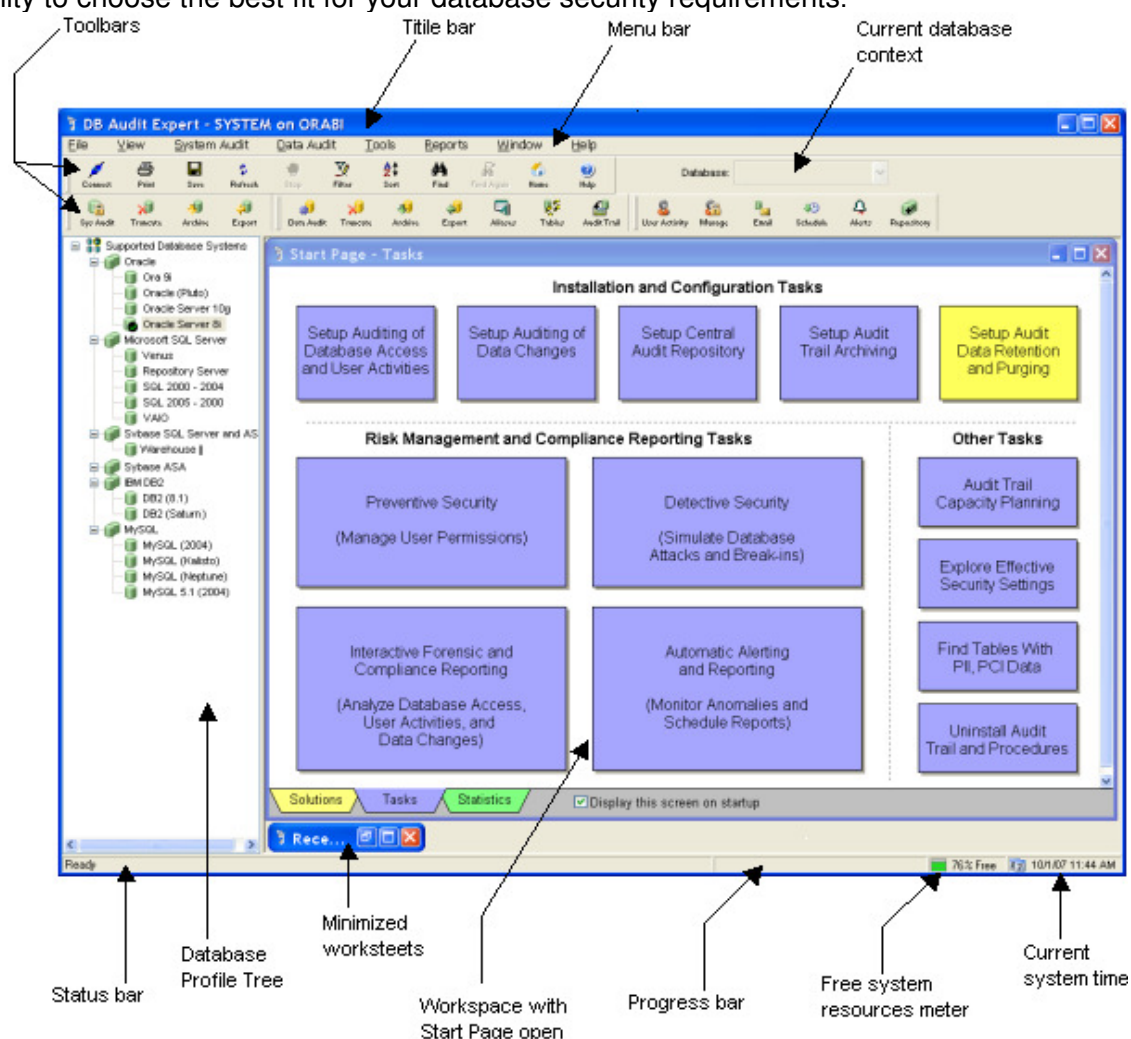



Figure 39. DB audit user interface

Key Benefits

- Improves system security and ensures system accountability. Captures both regular and "back-door" access to audited database systems.
- Features centralized security and auditing control of multiple database systems from a single location providing ease of management.
- Features unified auditing graphical interface that shortens the learning curve and is easy to use.
- Provides analytical reports that reduce large amounts of audit data to comprehensive summaries thus enabling to easily identify various database security violations.

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 63 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

- Provides analytical reports that help to identify which processes and users are hogging system resources.
- Provides audit trail details that are unavailable from native database audit utilities.
- Provides ability to generate real-time email alerts to key personnel when changes occur to sensitive data.
- Frees DBA from the need to create and manage finely-tuned database triggers for data-change auditing purposes.
- Supports flexibility auditing configurations, enabling security personnel to choose specific types of database operations and data changes that must be monitored and recorded in the audit trail.
- Provides transparent system-level and data-change auditing of any existing applications without requiring any changes to be made in those applications.


6.5 FIREWALLS, SPAM AND VIRUS DETECTION TOOL

Firewalls are often used to check information coming from the Internet or a network, and then either block it or allow it to pass through to your private network. Spam and virus detection tools are also very used to defend your private network against malicious software and attacks. These kinds of tools can be interesting in the context of DIAMONDS project as they deal with contextual access control and allows to detect several types of deny of service attacks.

6.5.1 Firewalls

A firewall is a device that guards the entrance to a private network and keeps out unauthorized or unwanted traffic. The access permission is based upon a set of rules and is frequently used to protect networks from unauthorized access while permitting legitimate communications to pass. The list of firewalls includes:

Tool	Company	URL
Comodo Internet Security Plus	Comodo Group	http://www.comodo.com/home/internet-security/security-software.php
Online Armor Premium	Tall Emu Pty Ltd	http://www.online-armor.com/
Kaspersky Internet Security	Kaspersky Lab	http://www.kaspersky.com/
Outpost Firewall Pro 7	Agnitum	http://www.agnitum.com
Norton Internet Security 2011	Symantec	http://www.symantec.com
BitDefender Internet Security	SoftWin	http://www.bitdefender.com
ZoneAlarm Pro Firewall	Check Point Software Technologies LTD	http://www.checkpoint.com/
Trend Micro Internet Security	Trend Micro Inc	http://www.trend Micro Inc.com
eScan Internet Security	MicroWorld Technologies Inc.	http://www.escan.com

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 64 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

McAfee Internet Security	McAfee, Inc.	http://www.mcafee.com
Norman Personal Firewall	Norman Personal Firewall	http://www.norman.com

6.5.1.1 Comodo Internet Security Plus

Name of the tool	Comodo Internet Security Plus
Tool title phrase	Comodo can scan, filter, block and stealth ports, making it difficult for non authorised software or users to access the sensitive information.
Developed by	Comodo Group
Maturity	Commercial tool
Supported modelling notations (if applicable)	
System requirements	Windows: XP, Vista, 7 Mac:
Available	http://www.comodo.com/home/internet-security/security-software.php
Miscellaneous	Classified as the best personal firewall in 2011 by toptenreviews.com

Comodo Internet Security Complete 2011 (Figure 40) promises protection of end user Windows-based PC against viruses and malware through Comodo's Auto Sandboxing and patent pending Default Deny Protection technology. Protects against viruses, trojans, adware, spyware and other malware threats, the program features a prevention-based technology that identifies safe, unsafe and questionable files. Provides real-time. Virus-Free Guarantee offers repair of infected PC with CISC installed that cannot be restored to working condition by Comodo tech support team.

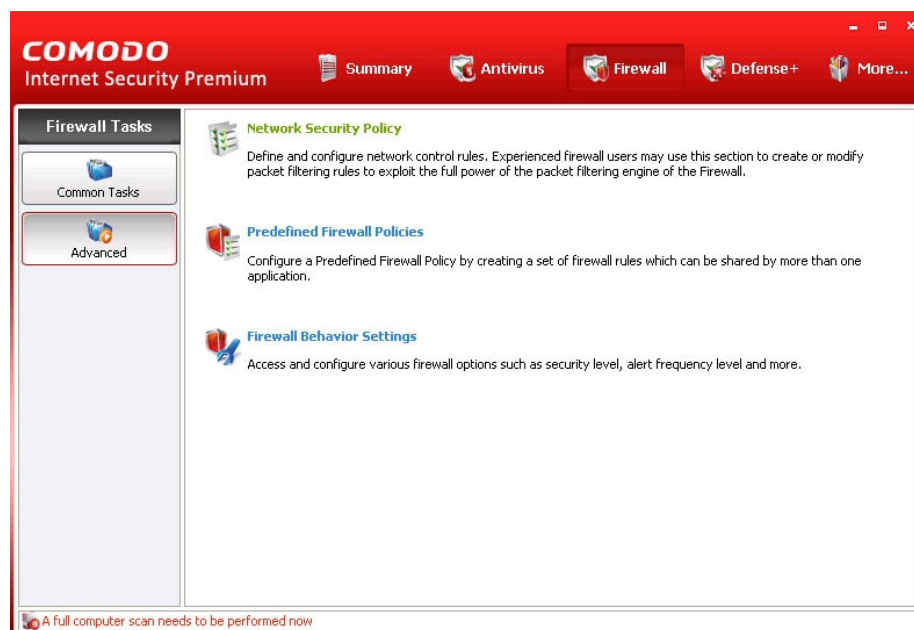



Figure 40. Comodo Firewall

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 65 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

6.5.2 Virus detection


According to about.com, “An "antivirus" is protective software designed to defend your computer against malicious software. Malicious software or "malware" includes: viruses, trojans, hijackers, dialers, and other code that vandalizes or steals your computer contents. In order to be an effective defense, your antivirus software needs to run in the background at all times, and should be kept updated so it recognizes new versions of malicious software.” The list of antivirus software includes:

Tool	Company	URL
BitDefender	SoftWin	http://www.bitdefender.com
Webroot	WebRoot	http://www.webroot.com
Norton	Symantec	http://symantec.com/norton/antivirus
AVG	AVG Technologies	http://www.avg.com
Avira	Avira GmbH	http://www.avira.com
Trend Micro	Trend Micro Inc	http://www.trend Micro Inc.com
Avast	AVAST Software a.s.	http://www.avast.com
F-Secure	F-Secure Corporation	http://www3.f-secure.com
BullGuard	BullGard	https://www.bullgard.com
eScan	MicroWorld Technologies Inc.	http://www.escan.com
McAfee	McAfee, Inc.	http://www.mcafee.com
ZoneAlarm	Check Point Software Technologies LTD	http://www.checkpoint.com/
Panda	Panda Security SL	http://www.pandasecurity.com/
Kaspersky	Kaspersky Lab	http://www.kaspersky.com/

6.5.2.1 BitDefender Antivirus

Name of the tool	BitDefender Antivirus Pro
Tool title phrase	BitDefender is designed to protect computers from viruses and spyware.
Developed by	SoftWin
Maturity	Commercial tool
Supported modelling notations (if applicable)	
System requirements	Windows: XP, Vista, 7 Mac: Mac OS X Tiger (10.4), Leopard (10.5) and Snow Leopard (10.6) SymbianOS
Available	www.bitdefender.com
Miscellaneous	It was launched in November 2001, and is currently in its thirteenth version. The 2011 version was launched in August 2010, and it includes several protection and performance enhancements. It has been selected as the best 2011 Antivirus by toptenreviews.com

BitDefender gives protection against viruses, spyware, hackers, spam and other e-threats without harming the performance of the computer. Features: Firewall; iPhone-friendly Parental Controls,

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 66 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

for monitoring and controlling what websites your kids visit and who they IM with; QuickScan, to detect viruses; IM Encryption; Video Tutorials; a customizable dashboard that lets you choose how much or how little you want to see; system maintenance tools; File Encryption, for protecting sensitive files; File Shredder (eliminates all traces of deleted files); Laptop Mode, to prolong battery life.


	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 67 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public



Figure 41. BitDefender Dashboard

This is the intermediate user profile view of BitDefender Antivirus Pro 2011. The dashboard (illustrated in Figure 41) can be customized to include the wanted tools. BitDefender Antivirus Pro features real-time protection from viruses and other online threats as shown in the Figure 42.

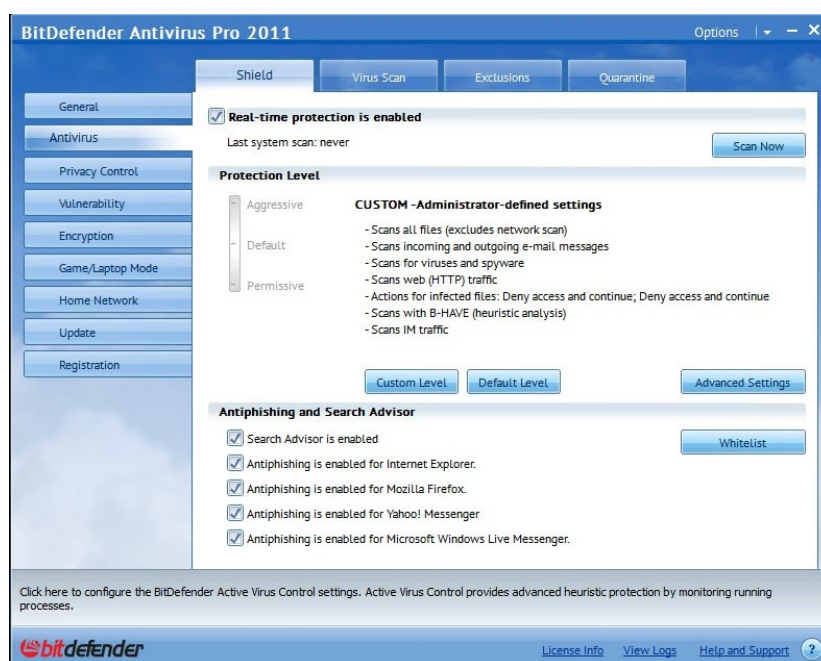



Figure 42. Antivirus Configuration

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 68 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

6.5.2.2 *Norton AntiVirus*

Name of the tool	Norton AntiVirus
Tool title phrase	Provides malware prevention and removal during a subscription period.
Developed by	Symantec
Maturity	Commercial tool
Supported modelling notations (if applicable)	
System requirements	Windows: XP, Vista, 7 Mac: Mac OS X
Available	http://symantec.com/norton/antivirus
Miscellaneous	Norton AntiVirus 2011 does a great job of balancing system resources, simplicity, and security. The protection level is second to none, and new features continue to set the standard for antivirus software.

Norton AntiVirus 2011 detects and eliminates viruses, spyware, and other threats lying in wait to infect your PC, so you can chat, email, and share files safely. It provides protection against hidden threats lurking in downloads, emails, and instant messages without slowing down your PC's performance.

Last year's newest features were only available to customers using the most popular web browsers, Norton now supports a large number of browsers (Chrome, Opera, Safari), email clients (Outlook and Outlook Express) IM clients (Yahoo, AOL and MSN Messenger), download managers (FileZilla) and even P2P sharing clients (Bittorrent, Limewire).

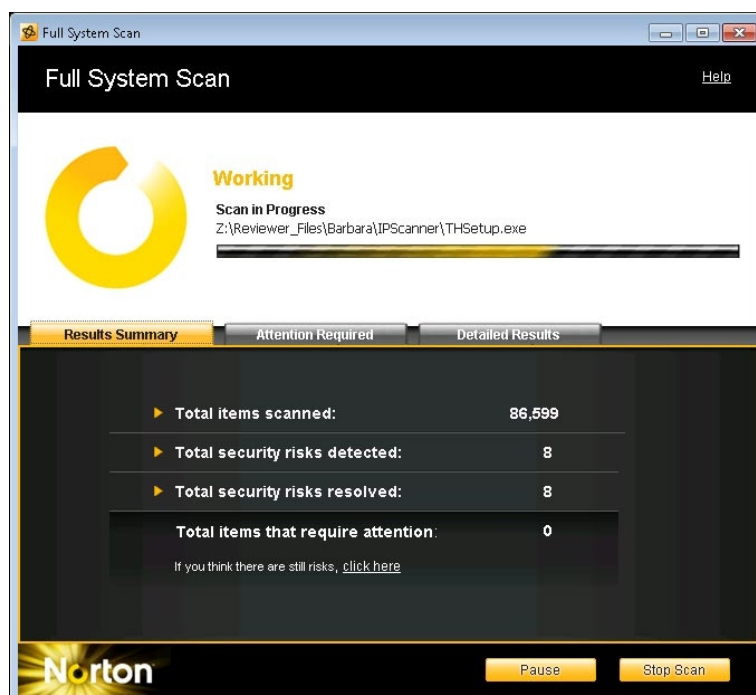



Figure 43. Norton AntiVirus Scan

Figure 43 shows the Norton AntiVirus scan in progress. As many other antivirus, it is possible to run a manual quick scan, (longer) quick scan or a custom scan of designated files and folders. Since some years, Norton introduced the Norton Insight Network (illustrated in Figure 44) which is a dynamic database of information from Norton and community users on the relative threat and

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 69 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

reputation of files. This information can help you know the reputation of files before you download them.

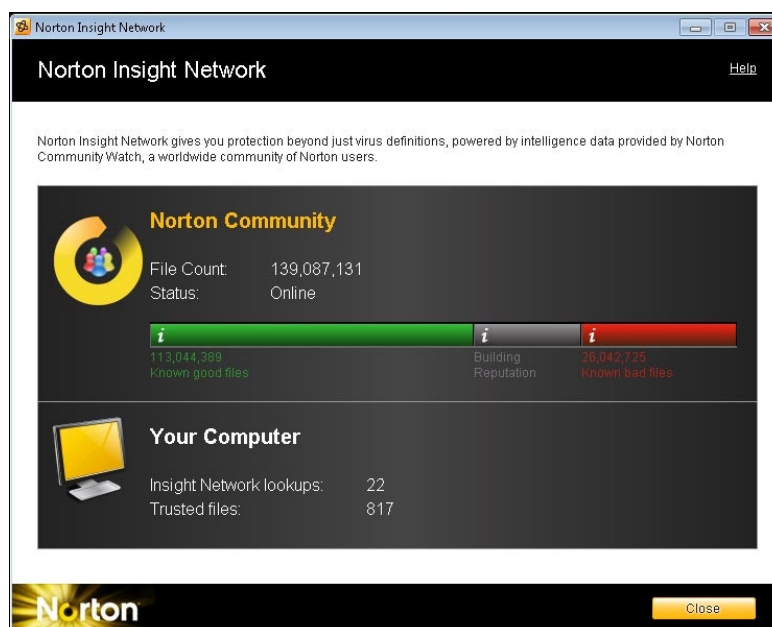


Figure 44. Norton Insight Network

6.5.2.3 Kaspersky AntiVirus

Name of the tool	Kaspersky AntiVirus
Tool title phrase	Kaspersky Anti-Virus is a comprehensive program, protecting users from a number of threats.
Developed by	Kaspersky Lab
Maturity	Commercial tool
Supported modelling notations (if applicable)	
System requirements	Windows: XP, Vista, 7 Mac: Mac OS X Linux
Available	http://symantec.com/norton/antivirus
Miscellaneous	Kaspersky is often among the first to detect new viruses.

Kaspersky Anti-Virus 2011 (illustrated in the Figure 45) defends against both known and emerging viruses, spyware and malware with streamlined technologies that won't slow down the computer. Kaspersky Anti-Virus keeps the digital identity and passwords safe and secure when the computer is with low charge.


	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 70 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public



Figure 45. Kaspersky Antivirus

6.5.2.4 ClamAV


Name of the tool	ClamAV
Tool title phrase	ClamAV is mainly installed on the server-side to check virus on emails or stored files.
Developed by	SourceFire Inc.
Maturity	Stable Version (0.97.1)
Supported modelling notations (if applicable)	
System requirements	Windows, OS X, Linux
Available	http://www.clamav.net
Miscellaneous	

ClamAV is an open source antivirus engine designed for detecting Trojans, viruses, malware and other malicious threats. It is very used for storage servers and mail gateway scanning.

6.5.3 Spam Detection and Filtering

The Cambridge dictionary defines anti-spam as produced and used to prevent people sending and receiving unwanted emails, especially advertisements. To prevent spamming, it uses different anti-spam techniques. Some of these techniques are embedded in products, services and software to ease the burden on users and administrators. Anti-spam techniques can be broken into four broad categories: those that require actions by individuals, those that can be automated by e-mail administrators, those that can be automated by e-mail senders and those employed by researchers and law enforcement officials. Among the different Anti-Spam software it can be listed:

Tool	Company	URL
SPAMfighter Pro	SPAMfighter	http://www.spamfighter.com/
Cloudmark Desktop	Cloudmark Inc	http://www.cloudmarkdesktop.com/

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 71 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

One Pro		
MailWasher Pro	FireTrust	http://mailwasher.net
ChoiceMail One	DigiPortal	http://www.digiportal.com/
iHateSpam	Sunbelt	http://www.sunbeltsoftware.com/Home-Home-Office/iHateSpam/
CleanMail Home	Byteplanet	http://antispam.byteplant.com/products/cleanmail/
Spam Bully	Axaware	http://www.spambully.com/
SpamEater Pro	High Mountain Software	http://www.spameaterpro.com/
Spam Buster	Contact Plus	http://www.contactplus.com/Spam-Buster.html

6.5.3.1 **SPAMfighter**

Name of the tool	SPAMfighter
Tool title phrase	The users can create their own whitelists. However, they are dependent on the spam filter software's community-based external blacklist.
Developed by	SPAMfighter
Maturity	Commercial tool
Supported modelling notations (if applicable)	
System requirements	Windows: XP, Vista, 7
Available	http://www.spamfighter.com/
Miscellaneous	

SPAMfighter Pro implements a community-based method of filtering spam. Over a million users belong to the spam blocker's online community. These users power and feed the spam blocker's online database. By accessing the spam filter software's integrated toolbar, users can report spam, phishing attacks and other unwanted email in real-time. Once several people have reported the same email sender or domain address as an unwanted sender, all future emails from the known spammer are blocked. SPAMfighter configurable settings (Illustrated in Figure 11) become active as soon as the spam blocker loads into the operating system.


	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 72 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public




Figure 46. SPAMfighter Pro Screenshot

6.5.3.2 SpamAssassin

Name of the tool	SpamAssassin
Tool title phrase	This software can run on the server or the client side: It can be integrated with the mail server to automatically filter all mail for a site. It can also be run by individual users on their own mailbox and integrates with several mail programs.
Developed by	Apache Foundation
Maturity	Stable (version 3.3.1)
Supported modelling notations (if applicable)	DNS-based blackhole lists and DNS-based whitelists, URI blacklists, Sender Policy Framework
System requirements	Windows, Linux, OS X
Available	http://spamassassin.apache.org/
Miscellaneous	It is also available as a Comprehensive Perl Archive Network module

SpamAssassin is released under the Apache License 2.0 used for e-mail spam filtering based on content-matching rules. It uses a variety of spam-detection techniques, which includes DNS-based and checksum-based spam detection, Bayesian filtering, external programs, blacklists and online databases.

This software is a Perl-based application which is usually used to filter all incoming mail for one or several users. It can be run as a standalone application or as a subprogram of another application or as a client.

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 73 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

7. DIAGNOSIS AND ROOT-CAUSE-ANALYSIS TOOLS

7.1 DIAGNOSIS TOOLS FOR SECURITY TESTING

Root cause analysis (RCA) is a structured evaluation method that identifies the root causes for an undesired outcome and the actions adequate to prevent recurrence. Root cause analysis helps determine what happened, how it happened, and why it happened which is one of the objective of DIAMONDS project in the context of a security failure detection. The practice of RCA is predicated on the belief that problems are best solved by attempting to address, correct or eliminate root causes, as opposed to merely addressing the immediately obvious symptoms. RCA is often considered to be an iterative process, and is frequently viewed as a tool of continuous improvement.

7.1.1 RCAT

Name of the tool	RCAT
Tool title phrase	Root Cause Analysis Tool
Developed by	NASA
Maturity	Commercial
Supported modelling notations (if applicable)	No reference
System requirements	Windows XP Operating System
Available	https://pbma.nasa.gov/root-cause-analysis-tool/
Miscellaneous	For information related to NASA's root cause analysis training and mishap investigation procedures you may also go to: https://secureworkgroups.grc.nasa.gov/mi


The NASA Root Cause Analysis Tool (RCAT) is designed to facilitate the analysis of anomalies, close calls, and accidents and the identification of appropriate corrective actions to prevent recurrence. The RCAT software provides a quick, easy, accurate, and repeatable method to perform and document root cause analysis, identify corrective actions, perform trending, and generate data usable in precursor analysis and probabilistic risk assessment.

After extensive review, NASA found that none of the commercially available tools and methods would support a comprehensive root cause analysis of all the unique problems and environments NASA faces on the Earth, in the ocean, in the air, in space, and on moons and planetary bodies. Existing tools were designed for a specific domain (e.g., aviation), a specific type of activity, a specific type of human error (e.g., errors of omission) or had a limited set of cause codes. The NASA RCAT, a paper-based tool with companion software (now available free to government Agencies and contractors), was designed to address the shortcomings identified in existing tools.

The NASA RCAT was designed with the whole system in mind, so that all potential types of activities and all potential causes of accidents, whether they be initiated by hardware, software, humans, the environment, weather, natural phenomenon, or external events, could be incorporated into the timeline, fault tree, and event and causal factor tree.

The RCAT aids users by providing a step-by-step guide, intuitive logic diagramming capability, standard terminology, standard definitions and standard symbols.

The RCAT software provides the analyst with a quick and easy method to perform the following:

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 74 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

1. Document case file properties,
2. Identify and document the undesired outcome,
3. Create and edit a detailed timeline,
4. Create and edit a short timeline,
5. Create and edit a fault tree,
6. Create and edit an event and causal factor tree,
7. Generate a report; and
8. Trend case file properties, causes, contributing factors, and other information.

7.1.2 XFRACAS

Name of the tool	XFRACAS
Tool title phrase	Web-Based Failure Reporting Software
Developed by	ReliaSoft
Maturity	Commercial
Supported modelling notations (if applicable)	No reference
System requirements	Web-based software system (deployed via Internet Explorer) that can be implemented with a SQL Server or ORACLE database
Available	http://www.reliasoft.com/xfracas/
Miscellaneous	

ReliaSoft's XFRACAS software tool is a Web-based enterprise-wide incident reporting / failure reporting, data analysis and corrective action software system. The software has been designed for the acquisition, management and analysis of product reliability, quality and safety data from multiple locations, along with team-based problem solving and related activities.

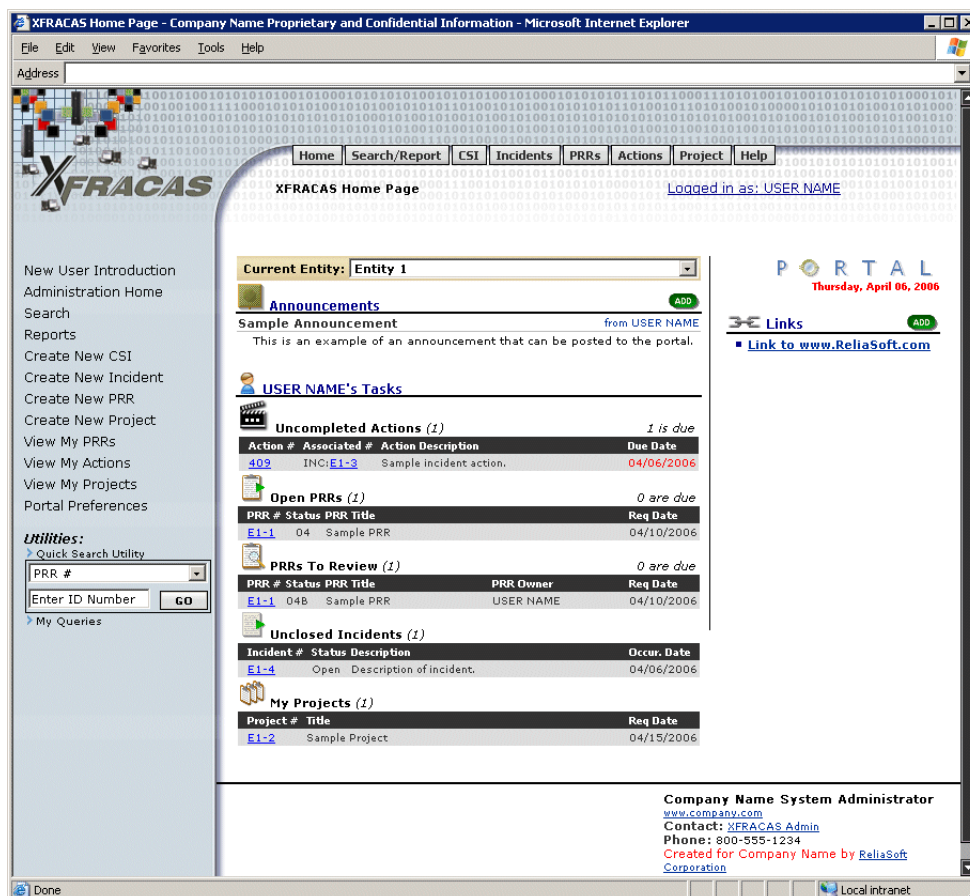



Figure 47. XFRACAS user interface

The XFRACAS software allows to capture the data required for important reliability, quality, safety and other analyses in order to resolve underlying problems and build a “knowledge base” of lessons learned that will be instrumental to future troubleshooting and development efforts. The XFRACAS system is configurable and flexible. The system’s Web-based user interface allows for easy access, collaboration and deployment throughout multiple sites, suppliers and dealers.

Some of the potential applications and benefits of performing FRACAS and related activities with ReliaSoft’s XFRACAS software include the ability to:

- Address data capture and management deficiencies to provide timely and accurate product reliability, quality and safety data.
- Streamline incident reporting and problem resolution activities.
- Provide a closed-loop system for managing corrective actions.
- Contribute to design improvements, faster product release, better service and enhanced customer satisfaction.
- Generate financial rewards through better product designs, enhanced control of product warranties and more efficient customer support.

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 76 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public


7.2 INTRUSION PREVENTION SYSTEMS

Intrusion prevention systems (IPS) are considered as extensions of intrusion detection systems because they both monitor network traffic and/or system activities for malicious activity. The main differences are, unlike intrusion detection systems, intrusion prevention systems are placed in-line and are able to actively prevent/block intrusions that are detected. The list of IPS is too large and includes:

Tool	Company	URL
Attack Mitigator	Top Layer Networks	http://www.toplayer.com/content/products/index.jsp
BBX	DeepNines	http://www.deepnines.com/bbx.php
Bro	Vern Paxson	http://bro-ids.org/
Cisco IPS	Cisco Systems	http://www.cisco.com/en/US/products/hw/vpndevc/index.html
Cyclops	e-Cop.net	http://www.e-cop.net/
DefensePro	Radware, Ltd.	http://www.radware.com/content/products/dp/default.asp
Dragon	Enterasys Networks, Inc.	http://www.enterasys.com/products/ids/
eTrust Intrusion Detection	Computer Associates	http://www3.ca.com/solutions/Product.aspx?ID=163
Juniper Networks IDP	Juniper Networks	https://www.juniper.net/products/intrusion/
IntruShield	Network Associates	http://www.mcafee.com/us/enterprise/products/network_intrusion_prevention/index.html
iPolicy	iPolicy Networks	http://www.ipolicynetworks.com/products/ipf.html
Proventia	Internet Security Systems	http://www.iss.net/products/product_sections/Intrusion_Prevention.html
SecureNet	Intrusion	http://www.intrusion.com/
Sentivist	Check Point Software Technologies	http://www.nfr.com/solutions/sentivist-ips.php
Snort	Sourcefire	http://www.snort.org/
Sourcefire	Sourcefire	http://www.sourcefire.com/products/is.html
StoneGate	StoneSoft Corporation	http://www.stonesoft.com/en/products_and_solutions/products/ips/
Strata Guard	StillSecure	http://www.stillsecure.com/strataguard/index.php
Symantec Network Security	Symantec Corporation	http://www.symantec.com/enterprise/products/index.jsp
UnityOne	TippingPoint Technologies	http://www.tippingpoint.com/products_ips.html

7.2.1 Cisco intrusion prevention system

Name of the tool	Cisco intrusion prevention system
Tool title phrase	Network-based intrusion prevention system
Developed by	Cisco
Maturity	Commercial
Supported modelling notations (if applicable)	Cisco Intrusion Prevention System Signatures
System requirements	
Available	http://www.cisco.com/en/US/products/sw/secursw/ps2113/index.html
Miscellaneous	

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 77 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

This tool deploys network-based intrusion prevention system that identifies, classifies, and stops known and unknown threats with the Cisco Intrusion Prevention System (IPS). As an essential part of the Cisco Secure Borderless Network, Cisco IPS is one of the most widely deployed intrusion prevention systems, providing protection against more than 30,000 known threats and timely signature updates and Cisco Global Correlation to dynamically recognize, evaluate, and stop emerging Internet threats.

Cisco IPS includes industry-leading research and the expertise of Cisco Security Intelligence Operations. Cisco IPS protects against increasingly sophisticated attacks, including: firefected attacks, worms, botnets, malware, and application abuse.



Figure 48. Cisco IPS 4270 Sensor


Cisco IPS provides intrusion prevention that:

- Stops outbreaks at the network level, before they reach the desktop
- Prevents losses from disruptions, theft, or defacement
- Collaborates with other network components, for end-to-end, networkwide intrusion prevention
- Supports a wide range of deployment options, with near-real-time updates for the most recent threats
- Decreases legal liability, protects brand reputation, and safeguards intellectual property

8. TOOL INTEGRATION PLATFORMS

Tool integration is a challenge linked to the optimization and automation of development and test processes. This area of research and development has acquired much attention in the couple of last years. It is now seen as one way to further improve the efficiency of development and test processes in a more and more complex world of system and software construction.

Looking at the tool integration problem, you have to look at the characteristics of the individual tools used today. Typically, tools work on their own data structures, which are well-suited to the task which needs to be performed with or by the tool. So the tool can only process data which is relevant for the tool. Tools can save and load their internal data to a file which may have a proprietary format. In such cases it is very difficult to make use of the tool specific data in a different context than the respective tools. So the question is how to transfer the data between the tools. Tool integration is not limited to the question of data exchange. The var-

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 78 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

ious aspects of tool integration have been discussed in literature for long time and a couple of tool integration solutions have been developed.


Basically, based on work of Thomas and Nejme [18] and Wassermann [19] the aspects of tool integration can be summarized as follows:

- **Data Integration:** Data Integration is probably the most obvious aspect of tool integration. The interesting point here is to share data between different tools. This is very crucial for software development tasks, because certain data sets are needed at different locations throughout the complete development process. If data cannot be shared between tools, they need to be replicated by manual work. For example, if a requirements engineering tool stores requirements data in a proprietary format only, the requirements data can hardly be used in other tools, e.g. used for testing. Manual work for having those requirements also in the testing tools may be needed. But it gets even worse, if requirements changes over time as this imply major efforts in updating the data used in the several different tools in a consistent way.
- **Control Integration:** Control Integration is about the availability of certain functionalities provided by a tool in the context of another tool. Control Integration may help in avoiding the implementation or deployment of similar or same functionality in various different tools or locations. In most cases it is sufficient to have a specific functionality (e.g. spell checker) available only once in a given environment instead of implementing a spell checker several times in all tools which needs this kind of functionality. Control Integration is not easy to archive since tools might need a modular architecture which reflects the service and consumer paradigm. So tools must provide their functionality via dedicated interfaces.
- **Presentation Integration:** The objective of presentation integration is to give a user a homogeneous user experience, which means to provide a common look and feel. So tools do share the same UI elements and there are in that way hard to separate. The benefit of presentation integration is the reduction of the learning and training phase for new tools. So users know already how the UI of the tool is working. One prominent example is the save menu, which is present in almost every tool which comes with a graphical user interface. This menu entry is very often at the same place and does have the same name or icon. There are a couple of frameworks and toolkits which contribute to achieve presentation integration (e.g. Swing, or SWT).
- **Process Integration:** This aspect of tool integration is focussing on how tools may interact in order to support a development process. So it is important to identify certain process steps and to figure out which tools can be used for which part of the process and how they have to interact. This includes also the input and output required to complete certain steps. In particular events usually play an important role for the process integration.

There are in general two different architectural approaches for integrating tools. This could be either done in a tool coalition approaches or in a tool federation approach. Where tool coalition is based on point-to-point connection between tools and tool federations are based on a central integration platform. Depending on the context of the tool integration both approach have benefits and drawbacks. The major difference is that tool coalitions can be used easier in small and ad-hoc environments, where tool federations better fit to larger environments. Some tool federation platforms are presented in the next sections.

8.1 MODELBUS

Name of the tool	Modelbus
Tool title phrase	Model-driven tool integration framework
Developed by	Fraunhofer FOKUS
License	Open Source
Supported modelling	EMF-based

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 79 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

frameworks	
System requirements	
Available	www.modelbus.org
Miscellaneous	

ModelBus is a model-driven tool integration framework which allows you to build a seamlessly integrated tool environment for system engineering and test processes. It comprises a tool integration platform (run-time environment) as well as the tool integration development environment.

ModelBus is based on SOA principles. It consists of a central bus-like communication infrastructure, a number of core services and a set of additional management tools. Depending on the usage scenario at hand, different development tools can be connected to the bus via tool adapters. Once a tool has been successfully plugged in, its functionality immediately becomes available to others as a service. Alternatively, it can make use of services already present on the ModelBus.

The particular strength of ModelBus is the automation of individual development steps by using the orchestration facility. It helps to automatically trigger and execute sets of actions needed for running a development process.

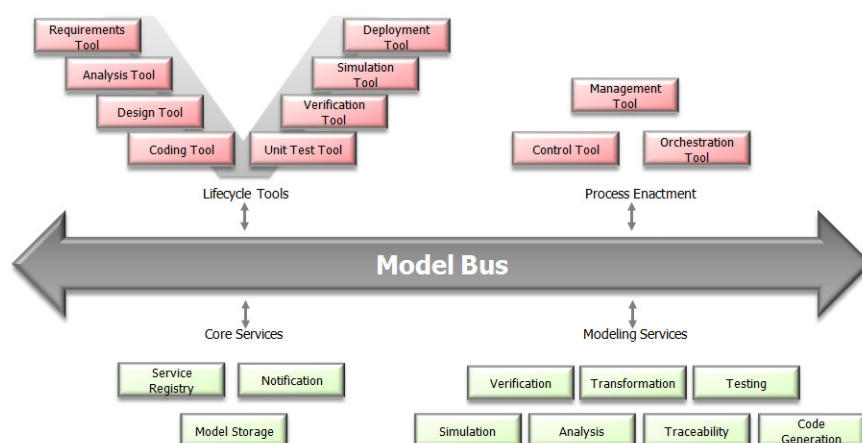



Figure 49 ModelBus Framework

The key point of ModelBus is the idea of freedom of data. So data either consumed or produced by tools are in general available within the ModelBus context. Which means that it can be used for any relevant purpose like traceability and impact analysis or report generation. This freedom of data is basically achieved by creating open and complete models of the data that is used within a tool. In addition, data access is controlled by a role based access control system.

ModelBus comes with an Eclipse Integration (ModelBus TeamProvider) which is the easy way to benefit from ModelBus infrastructure in all Eclipse-based tools. In addition to that, ModelBus offers a couple of tool adapters for Commercial off-the-shelf (COTS) tools including Enterprise Architect, Doors or Simulink.

ModelBus is a very flexible solution which allows the customisation and adaptation of ModelBus-based development environments. This is achieved by defining a clear but flexible architecture allowing the integration and instantiation of various existing and well-established technological assets. The architecture comprises a tool layer, a services layer and a repository layer and is supported by a work-flow engine. ModelBus can handle data based on metamodeling principles, which allows the full employment of MDE technologies. But ModelBus can also work on traditional development artefacts such as source code or documents or binaries. ModelBus can work seamlessly on all different kinds of artefacts simultaneously.

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 80 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

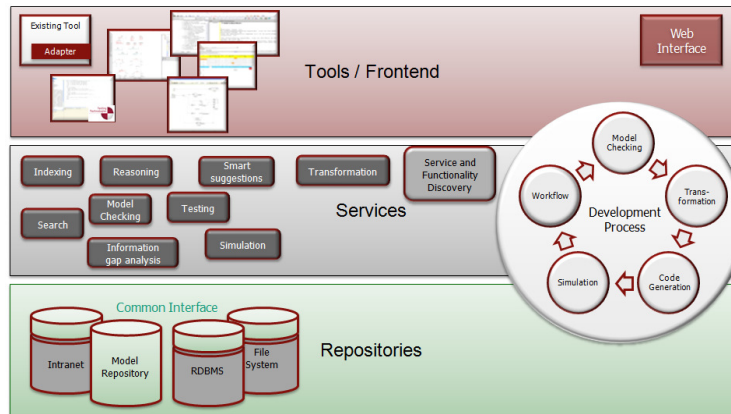


Figure 50 ModelBus architecture

ModelBus was initiated by European projects Modelware and Modelplex and it has a record of more the 6 years of development. ModelBus has been and is used in several different application scenarios including Verification and Validation tasks. ModelBus is free software and can be used even in commercial environments without license fees.

8.2 JAZZ

Name of the tool	JAZZ
Tool title phrase	IBM Rational Jazz technology platform
Developed by	IBM
License	Commercial
Supported modelling frameworks	-
System requirements	
Available	www.ibm.com/software/rational/jazz/
Miscellaneous	

Jazz is a tool integration approach of IBM. It basically allows the data integration and control integration. The data ownership concept of Jazz is such, that every tool holds its own data. A centralised and externalised data repository is not part of the architectural design. Only via specific interfaces, subsets of the tool data are made public. Jazz has a particular approach to presentation integration as it allows the rendering of data located in remote tools with the user interface of that tool. This works for specific subsets of data. The communication of Jazz Integration Architecture is based on REST-full web services.

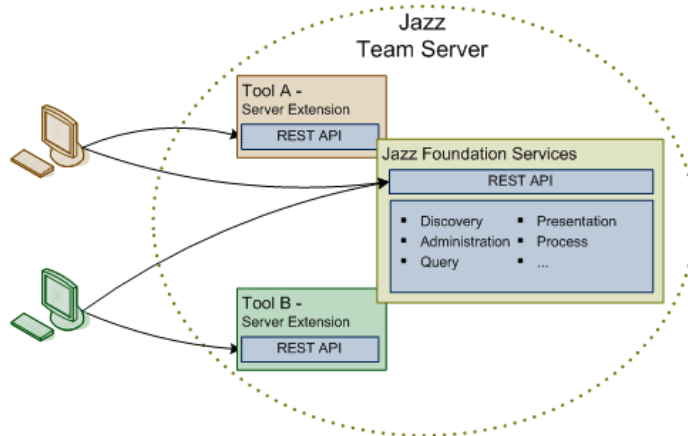


Figure 51 Example of JAZZ deployment

Jazz is a commercial product of IBM which is targeted on the area of tool integration. Jazz is based on the Open Services of Lifecycle Collaboration (OSLC) initiative, which is a forum for specifying interfaces between tools. OSLC divides the universe of tools into several categories (e.g. change management tools) and defines for each category a set of interfaces and data schemas, which shall be implemented and used by the respective tools in order to ensure interoperability among those tools. OSLC is following a different data integration approach than ModelBus. OSLC is assuming that each tool is responsible for managing its data and only references to fractions of the tool internal data are given to other tools. This has some implications on the availability of data. So the interfaces need either tool specific extensions or the standardized interface definition needs to be extended in a subsequent version of the OSLC specification.

Jazz is a young technology and but some Jazz-based products are already made available by IBM, including Rational Team Concert. The usage of Jazz in custom-made development environments is currently not easy to achieve, there is only a limited number of tools which implement the OSLC specification to a sufficient extend and OSLC is still subject to frequent evolution. Practical experience have been made in some experiments showing that the OSLC specification leaves enough room for vendor specific extension which are sometime hard to work with, when trying to integrate tools.

8.3 CONNECTED DATA OBJECTS – CDO

Name of the tool	CDO
Tool title phrase	Connected Data Objects
Developed by	Eclipse Project
License	Open Source
Supported modelling frameworks	EMF-based
System requirements	
Available	http://www.eclipse.org/cdo/
Miscellaneous	

CDO is an Eclipse project and hosted by the Eclipse Foundation. CDO is a data integration technology which is based on a database principle either with object-relational or object-oriented mappings. CDO is a common way to persist data which is based on the Eclipse Modeling Framework – EMF. It is based on classical client-server architecture. CDO supports basically an online mode, which allows seeing changes done by a team member immediately in the development environments of the other team members. Every update made by clients is communicated to the server and then back to the other clients.

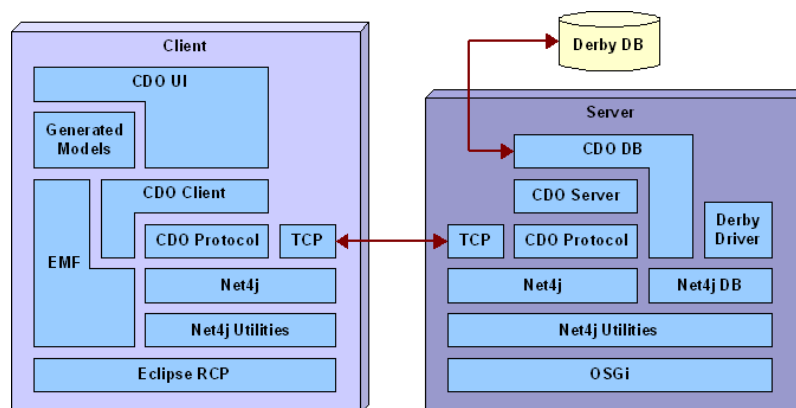


Figure 52 CDO architecture

CDO uses a specific protocol for communication between client and server, the net4j protocol which can be based on TCP. CDO is targeted towards the Eclipse IDE and is well integrated into that. But CDO is as such not a tool integration framework. CDO does not address the automation of development steps nor the integration of non-eclipse based tools and data. This needs to be achieved with the help of additional software. CDO works well for newly created meta models (data schemas) but has problems when working with legacy models and UML models. EMF models (and legacy models) need special preparation in order to be used in a CDO environment. In addition working with CDO in an offline mode is difficult as CDO requires always a connection to a server. These two issues make it sometimes difficult to use CDO in some development environments. However, CDO is a powerful and efficient framework for storing EMF-based models in Eclipse-IDE environments.

8.4 EMF STORE

Name of the tool	EMF Store
Tool title phrase	model repository for EMF
Developed by	Eclipse Project
License	Open Source
Supported modelling frameworks	EMF-based
System requirements	
Available	http://www.eclipse.org/emf-store/
Miscellaneous	

EMF Store became recently an Eclipse project hosted at the Eclipse Foundation. It is another model repository for EMF-models in parallel to the CDO one. EMF store follows a very similar approach to ModelBus as it supports both the online and the offline model for collaboration.

EMFStore emphasises the merge process in Eclipse IDE based environments. Furthermore, it claims to have special support for migration of models. This means that it in particular support the evolution of meta-models. Similar to CDO in EMFStore the meta models needs special preparation before it can be used with EMF-store.

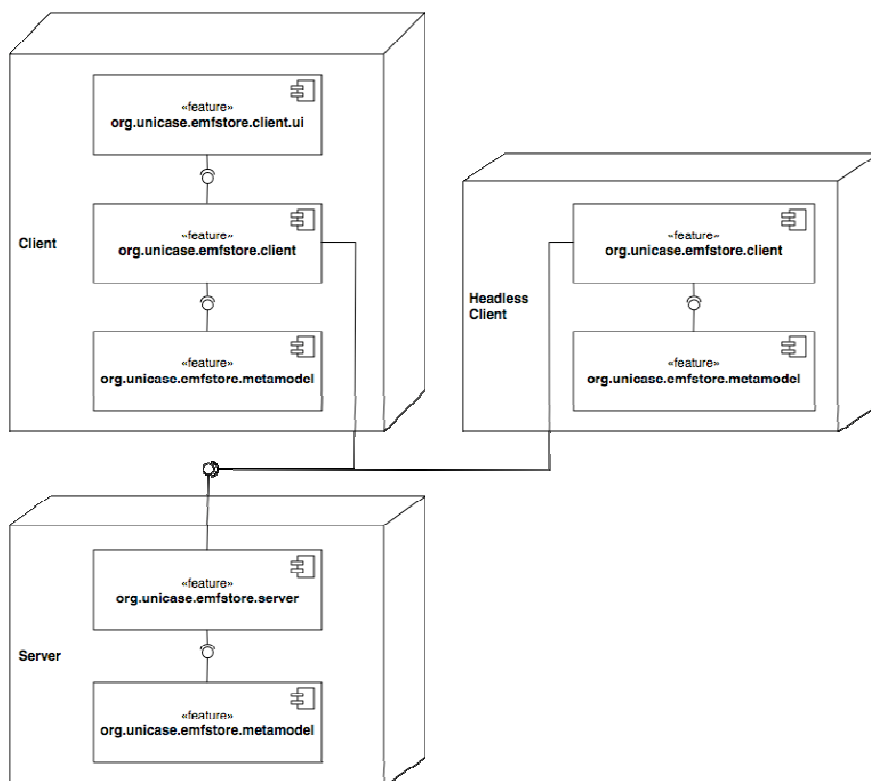



Figure 53 Example of EMFstore deployment

EMFStore is a model repository which can be used in a tool integration environment. It is concentrating on the data integration aspect. In particular, it provides its own Eclipse-based editors for merging data coming from different developers. EMFStore is not concerned with the other aspects of tool integration in particular the control and process integration aspects.

9. RISK ANALYSIS AND MODELING TOOLS

In this section, we give a short overview of risk analysis and modeling tools for IT security. These risk analysis and modeling tools differ in scope of support, underlying methodologies and modeling techniques and compliance to IT standards. All of the discovered tools support general risk analysis and management phases with the capabilities for manual risk identification, risk analysis, evaluation, classification of risks as well as generating various reports. The difference is the used methodologies and modeling techniques.

Checklists based tools use guided question catalogues, questionnaires or best practice documents. ISO 17799 Risk Analysis Toolkit [34] is an open source toolkit for risk analysis of security in enterprises or public organization based on a guided question catalogue, which has the possibility to generate security policies predicated on the given answers. The free available version has as application language Spanish. Several tools [ISO 27000 Toolkit [35], ISO27k Toolkit [36], MOF [37]] provide series of general IT security materials (e.g. process description, checklists, question-

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 84 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

naires, typically used tables, and more) and best practice documents. All these important elements are provided in electronic form for reuse proposes. Special security aspects such as e-Security are also supported by document based toolkits. For example, the e-Security Toolkit [38] is a valuable collection of items and documents to assist in ensuring that your e-security and e-commerce security is sound. It comprises a LAN/Network audit questionnaires, a firewall audit questionnaire, an e-security checklist, and security questionnaires covering virus management, network routers, data access, contingency, system access, dial-in, internet access and more.

Engineering based tools often use a top-down Information Security Management System (ISMS) approach for risk identification, analysis and management. Most of these tools offer knowledge databases for vulnerabilities and controls. Threat Analysis & Modeling tool (TAM) [Microsoft Threat Analysis & Modeling [39], SDL Threat Modeling Tool [40]] focused on finding threats against an already designed application by using libraries (design rules, attacks, safeguards, or countermeasures). The developer starts to model an overview of the system architecture in the tool and will be supported by suggestions and various reports. The knowledge database of the RiskWatch [41] tool are completely customizable by the user, including the ability to create new asset categories, threat categories, vulnerability categories, safeguards, question categories, and question sets. Based on these knowledge databases it conducts automated risk analysis and vulnerability assessments of information systems. For distributed risk management it is possible to use the real-ISMS [42] tool and Citrus ONE [43] to identify and manage IT Security risks. These tools are web-based and use server based databases to answers several ISO 27001 criteria automatically.

Other common methodology like CRAMM (CCTA Risk Analysis and Management Method) focuses also on non technical security aspects during the risk identification, analysis and management. Therefore, the database of the CRAMM tool covers all aspects of information security including technical, physical, personnel, documentation and procedural measures.


Another aspect is the compliance to one or more IT standards, like ISO/IEC 17799 [12] (CORAS, CRAMM, Callio SECURA [44], Proteus [45], RiskWatch), ISO/IEC 27000 series [13] (CRAMM, Callio SECURA, Proteus, RiskWatch), NIST SP 800 series [14] (CounterMeasures [46], RiskWatch), IT-Grundschutz [15] (GSTool [47]).

Another main approach is the evaluation towards a model-based risk assessment, to allow the combination of complementary risk assessment methods and the integration into the model-based development. This methodology is supported by the CORAS tool. This tool is a good starting point for the model-based risk assessment, but it does not cover the gap between system design model and the risk model.

Several scientific approaches have been published in order to close the gap between risk models, systems design and implementation models. Chivers [16] introduce in his thesis the Security Analyst Workbench (SAW) for the model-based Security Design Analysis Framework (SeDAn), which can reference the system design to its security environment. The basic function of the tool is threat path analysis based on the imported system design model in UML. SAW is able to determine paths of attacks between various types of attacker and assets of concern.

Eichler [17] propose another approach based on a textual domain specific modeling language, Eclipse Modeling Framework (EMF) [48] and Xtext [49]. Through the modular design of the EMF meta-model, this tool can be integrated well in existing tool chains.

Next to the risk analysis tools for IT Security exist several risk related tools that are not dedicated for IT Security. Some of these tools can be adapted towards IT Security, like MQ1 Risk Manage-

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 85 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

ment system [50] or Enterprise Risk Manager [51].

One of the main lack is, that many of these tools are separated from the system itself and more or less filled by informal description methods. This means within these tools the running system is never explicitly modeled, referenced or data from the system is used. Therefore a gap exists between the system design model itself and the model within the risk analysis tool. Only the SeDAn Framework support references between these models.

In the next sections we describe three tools in more detail.

9.1 MICROSOFT THREAT MODELING

Name of the tool	SDL Threat Modeling Tool
Tool title phrase	Threat Modeling for the Microsoft Security Development Lifecycle
Developed by	Microsoft
Maturity	Free
Supported modelling notations (if applicable)	STRIDE
System requirements	Windows Vista or higher, Visio 2007
Available	http://go.microsoft.com/?linkid=9706808
Miscellaneous	

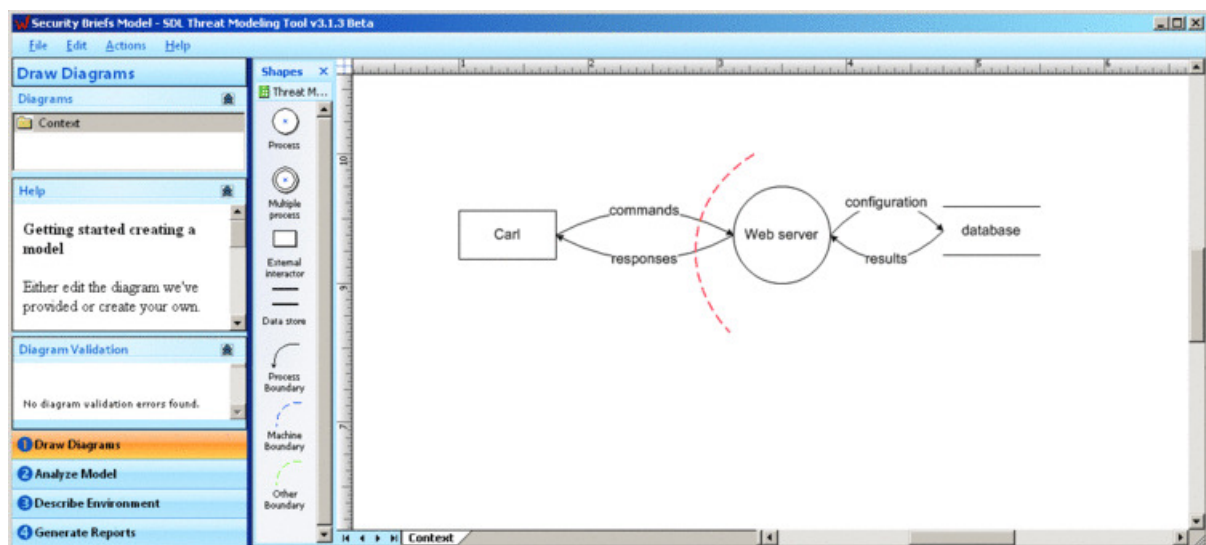



Figure 54. Screenshot of SDL Threat Modeling Tool [52]

The SDL Threat Modeling tool is one of the Security Development Lifecycle (SDL) tool chain provided by Microsoft to community. The tool is used in the design phase of the SDL and support software architects to identify and mitigate security risk issues.

One of the main differences is that the approach is centered on the software and not on assets or attackers. Therefore, the first step in the underlying process is to model an overview of your application architecture using data flow diagrams (see Figure 54) and pointing out the trust boundaries in the application landscape. After this step it offers a guided analysis of threats by providing tips to identify threats and by automatic generation of elements for the STRIDE method.

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 86 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

Using the STRIDE method one can define threats for every component and possible counter-measures. The reporting capabilities for various reports (security activities, testing reports, etc.) can be used for further communication.

Microsoft has tried to create a threat modeling tool that can be used by system designers, without being security experts.

9.2 THE CORAS TOOL

Name of the tool	The CORAS tool
Tool title phrase	
Developed by	SINTEF
Maturity	Open source
Supported modelling notations (if applicable)	CORAS risk analysis language
System requirements	OS Independent, Java
Available	http://sourceforge.net/projects/coras/
Miscellaneous	

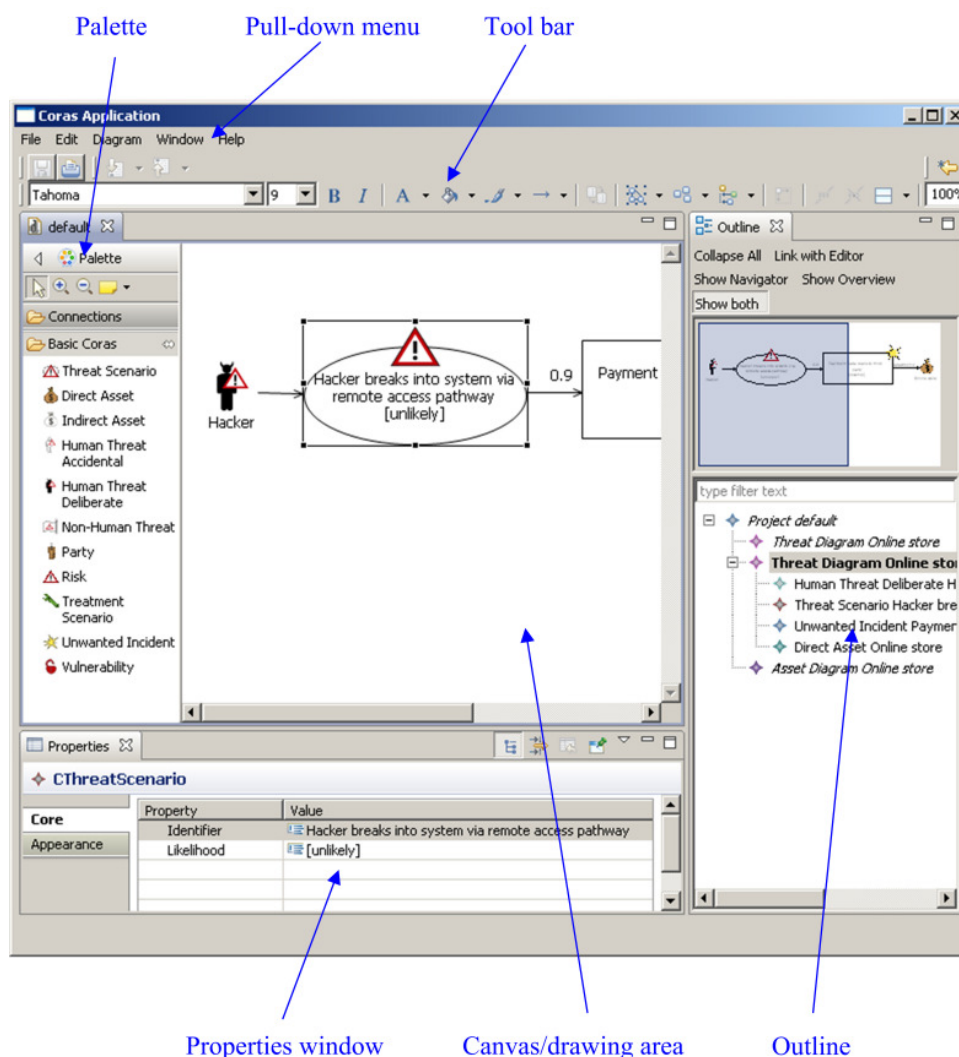



Figure 55. Screenshot of the CORAS tool

The CORAS tool is an open source diagram editor that supports the CORAS risk analysis language. The CORAS language is a graphical language whose constructs correspond to notions that are relevant during a risk analysis, e.g. threats, vulnerabilities, risks, unwanted incidents, threat scenarios and assets. The CORAS tool is intended to be used intensively during workshops where information is gathered through structured brainstorming. The tool is also intended to be used to document a risk analysis and to present the risk analysis results.

The CORAS tool is designed to support on-the-fly modelling using all five kinds of basic CORAS diagrams, thus facilitating the entire CORAS risk analysis process. A screenshot of the CORAS diagram editor is given in **Figure 55**. As indicated in the figure, the editor has six main parts:

- A pull-down menu that offers standard functions such as open, save, copy, cut, paste, undo and print.
- A tool-bar that offers easy access to the standard functions of the pull-down menu.
- A palette that contains the model elements and relations for drawing the diagrams.
- A drawing area or canvas for drawing the diagrams.

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 88 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

- A properties window that lists the properties of a selected element, and that can be used to edit the values of the properties.
- An outline that presents a project and its diagrams as a tree.

Except for the pull-down menu and the tool bar, all parts of the tool can be closed or hidden.

In the tool, a project is a collection of diagrams, and each diagram must belong to a project. A project must therefore be created before any diagrams are created.

The outline contains a tree representation of the project. The diagrams of the project are listed at the first level, and under each diagram all the diagram elements are listed. When a new element is created in the drawing area, it is automatically added to the tree under the correct diagram.

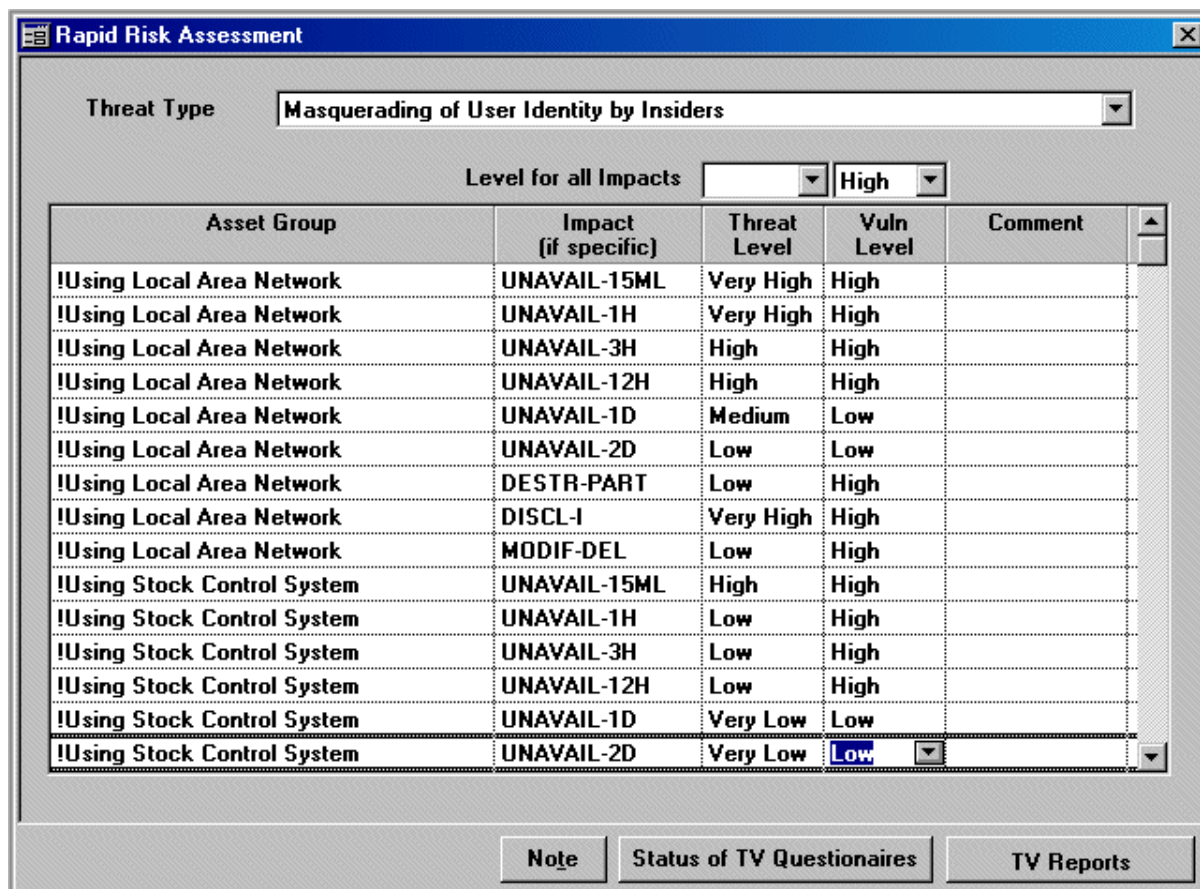
The drawing area is the part of the tool where the diagrams are made by inserting, editing, annotating and deleting elements. This is also where likelihoods and consequences are inserted to diagrams as part of the risk estimation, and it is also where risk levels are inserted as part of the risk evaluation.

9.3 CRAMM - CCTA RISK ANALYSIS AND MANAGEMENT METHOD AND TOOL

Name of the tool	CRAMM 5
Tool title phrase	Information Security Toolkit
Developed by	Siemens
Maturity	commercial
Supported modelling notations (if applicable)	CRAMM
System requirements	•Windows NT, 2000, XP
Available	http://www.cramm.com
Miscellaneous	

The First version of CRAMM (CCTA Risk Analysis and Management Method) was developed in 1985 by tasked the Central Computer and Telecommunications Agency on request of the UK Government. The resulted CRAMM methodology is divided into three stages:

- Asset identification and valuation
- Threat and vulnerability assessment
- Countermeasure selection and recommendation.




Asset Group	Impact (if specific)	Threat Level	Vuln Level	Comment
!Using Local Area Network	UNAVAIL-15ML	Very High	High	
!Using Local Area Network	UNAVAIL-1H	Very High	High	
!Using Local Area Network	UNAVAIL-3H	High	High	
!Using Local Area Network	UNAVAIL-12H	High	High	
!Using Local Area Network	UNAVAIL-1D	Medium	Low	
!Using Local Area Network	UNAVAIL-2D	Low	Low	
!Using Local Area Network	DESTR-PART	Low	High	
!Using Local Area Network	DISCL-I	Very High	High	
!Using Local Area Network	MODIF-DEL	Low	High	
!Using Stock Control System	UNAVAIL-15ML	High	High	
!Using Stock Control System	UNAVAIL-1H	Low	High	
!Using Stock Control System	UNAVAIL-3H	Low	High	
!Using Stock Control System	UNAVAIL-12H	Low	High	
!Using Stock Control System	UNAVAIL-1D	Very Low	Low	
!Using Stock Control System	UNAVAIL-2D	Very Low	Low	

Figure 56, Screenshot of CRAMM [53]

CRAMM's risk assessment tools can be used to answer single questions, to look at organizations, processes, applications and systems or to investigate complete infrastructures or organizations. Users have the option of a rapid risk assessment tool or a full, more rigorous, analysis. The following risk aspects of different risk can be answered:

- Determining if there is a requirement for specific controls, eg. strong authentication, encryption, power protection or hardware redundancy
- Identify the security functionality required for a new application
- Developing the security requirements for an outsourcing or managed service agreement
- Review the requirements for physical and environmental security at a new site
- Examine the implications of allowing users to connect to the Internet
- Demonstrate compliance with legislation such as the Data Protection Act
- Develop a security policy for a new system
- Audit the suitability and status of security controls on an existing system

The CRAMM tool provides a comfortable way to apply the CRAMM methodology, currently developed by Siemens. All three stages of the method are supported using a staged and disciplined approach embracing both technical (eg. IT hardware and software) and non-technical (e.g. physical and human) aspects of security. The tool comes in different versions: CRAMM expert, CRAMM express, BS 7799 Review, CRAMM NATO.

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 90 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

9.4 MOTORBAC

Name	MotOrbac
Models supported	Or-BAC
System requirements	Java runtime environment
Developed by	Institut TELECOM (France)
Available	http://sourceforge.net/projects/motorbac/
Miscellaneous	GNU General Public License (GPL), Mozilla Public License 1.1 (MPL 1.1)

Or-BAC [54] stands for Organization Based Access Control language. It is an access and usage control model based on first logic order that allows an organization to express its security policy including contextual rules. For this purpose, Or-BAC defines two abstraction layers. The first one is called abstract layer and describes a rule as a *role* having the permission, prohibition or obligation to perform an *activity* on a *view* in a given *context*. A *view* is a set of objects to which the same security rules apply. A *role* is set of users with similar privileges and an *activity* considers a set of actions with similar properties. The second layer is the concrete one. It is derived from the abstract level and grants permission, prohibition or obligation to a *user* to perform an *action* on an *object*. Thus, according to the Or-BAC syntax, a typical security rule has the following form:

- *Obligation* (S, R, A, V, C): this rule means that within the system S , the role R is obliged to perform the activity A targeting the objects of view V in the context C .
- *Permission* (S, R, A, V, C): this rule means that within the system S , the role R is permitted to perform the activity A targeting the objects of view V in the context C .
- *Prohibition* (S, R, A, V, C): this rule means that within the system S , the role R is prohibited to perform the activity A targeting the objects of view V in the context C .

In Or-BAC, we use contexts to express different types of extra conditions or constraints that control activation of rules expressed in the access control policy:

- The temporal context that depends on the time at which the subject is requesting for an access to the system.
- The spatial context that depends on the subject location.
- The user-declared context that depends on the subject objective (or purpose).
- The prerequisite context that depends on characteristics that join the subject, the action and the object.
- The provisional context that depends on previous actions the subject has performed in the system.

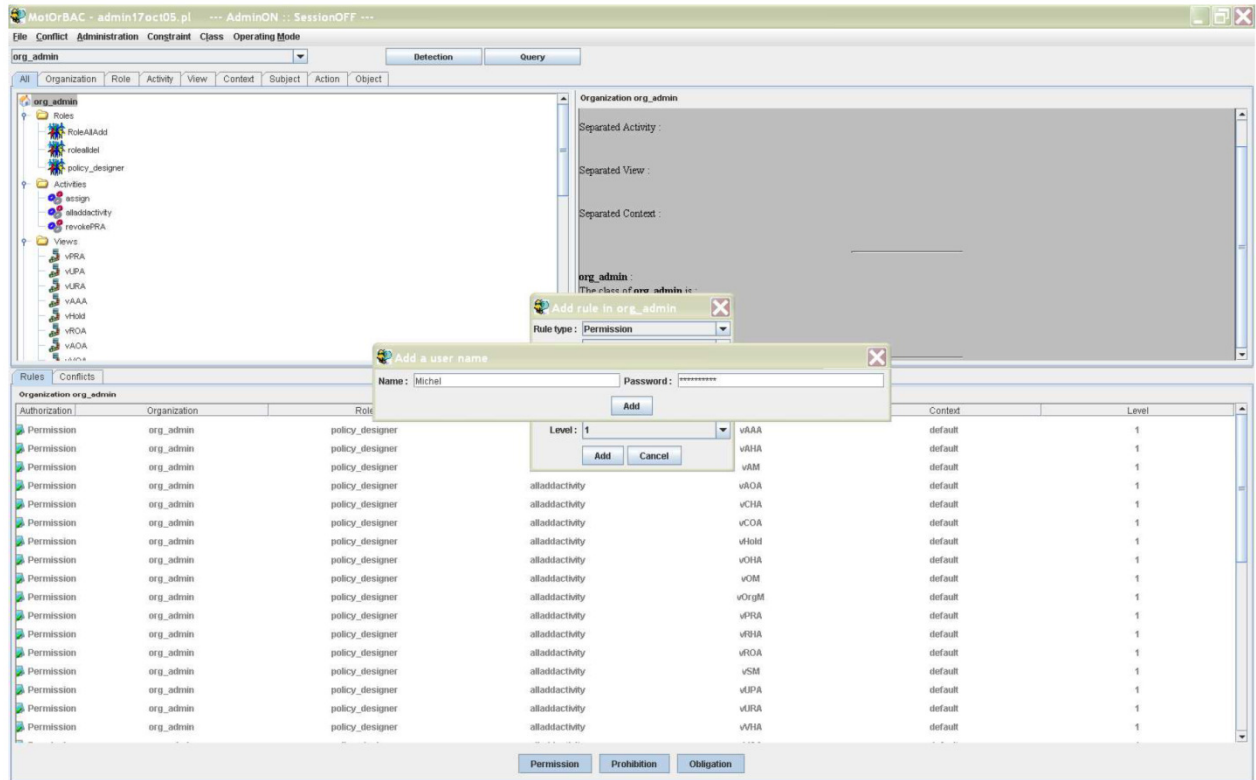


Figure 58. MotOrBAC user interface

9.5 GOAT

Name	GOAT
Models supported	VCG (Vulnerability Cause Graphs), SAG (Security Activity Graphs), VDC (Vulnerability Detection Conditions), SGIT (Security Goal Indicator Trees), VID (Vulnerability Inspection Diagram), TSM (TestGen Security Models) and many others
System requirements	Java runtime environment, MySQL
Developed by	Linköpings Universitet, Sweden
Available	http://www.ida.liu.se/divisions/adit/security/goat/
Miscellaneous	Free Software released as GPLv3

The GOAT modeling tool is one of the results of the [SHIELDS EU Project](#). It supports editing a variety of SHIELDS models, and connects to the SHIELDS SVRS for model upload and download. GOAT is a platform-independent Java program. It has been developed mostly on Windows but should work on other platforms as well. GOAT is distributed as an executable JAR file. In the following subsections, we will present VDC and TSM plugins in GOAT.

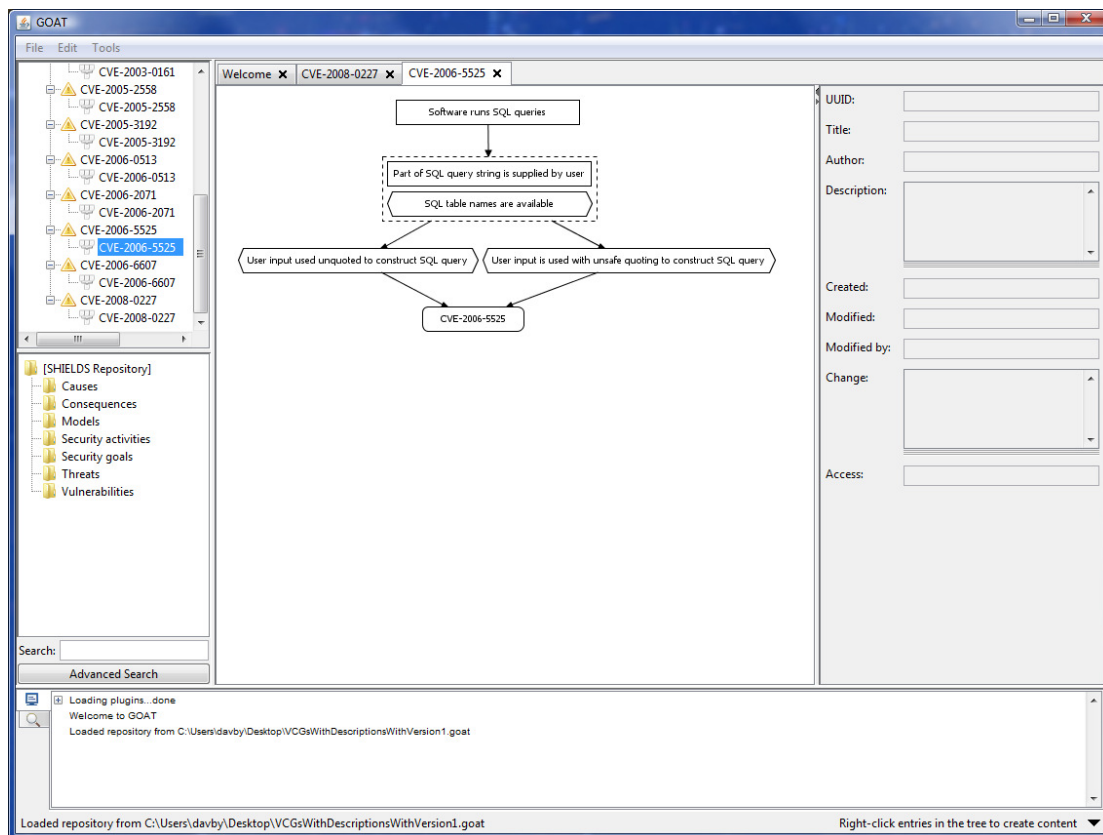


Figure 59. GOAT user interface

9.5.1 VDC editor plugin for GOAT

The VDC editor is a GOAT plug-in, which offers security experts the possibility to create vulnerability detection conditions (VDCs). These VDCs will be used to detect the presence of vulnerabilities by checking software execution traces using Montimage TIC testing tool².


The VDC editor user interface includes some features that allow simplifying the construction and composition of VDCs. The VDC editor has the following functionalities:

- The creation of new VDCs corresponding to vulnerability causes from scratch and their storage in an XML format.
- The visualisation of already conceived VDCs
- The editing (modification) of existing VDCs in order to create new ones.

The VDCs are stored in an XML file that constitutes one of the inputs for the Montimage TIC tool. This tool aims at detecting potential vulnerabilities in the execution traces of a code during runtime. A vulnerability is discovered if a VDC signature is detected on the execution trace.

To better understand the design of the VDC editor, we provide the syntax of a VDC. A VDC is a formalism used to determine the presence of vulnerabilities in a piece of software. It basically indicates that the execution of an action under certain conditions could be dangerous or risky for the program.

² TIC testing tool is one of Montimage tools. It is a dynamic code analysis tool that aims at detecting vulnerabilities by analysing the traces of the code while it is executing.

	<p style="text-align: center;">Review of security testing tools</p> <p style="text-align: center;">Deliverable ID: D1_1</p>	Page : 94 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

A VDC has a formal expression given by:

$$VDC ::= a/P(Var,Act) \mid a[/P(Var,Act)]; P'(Var,Act)$$

Where a denotes an action, $P(Var,Act)$ and $P'(Var,Act)$ represents any *predicate* on variables Var and actions Act .

- A vulnerability detection condition $a/P(Var,Act)$ means that action “ a ” occurs when specific conditions denoted by *predicate* $P(Var,Act)$ hold.
- A vulnerability detection condition $a[/P(Var,Act)]; P'(Var,Act)$ means that action “ a ” used under the optional conditions $P(Var,Act)$ is followed by a statement whose execution satisfies $P'(Var,Act)$. Naturally, if action a is not followed by an action, the *predicate* $P'(Var,Act)$ is assumed to be true.

Thus, we can say that a VDC is composed of at most 3 parts:

1. **Master condition:** The triggering condition called also *master action* (denoted a). When analysing the execution trace, if this condition is detected, we should verify if the state and *post conditions* of the VDC hold as well. If this is the case, then a vulnerability has been detected. The master condition is mandatory in a VDC.
2. **State condition:** A set of conditions related to the system state (denoted $P(Var,Act)$). The state condition describes the states of the specified variables at the occurrence of the *master action*. The state condition is mandatory in a VDC.
3. **Post condition:** A set of conditions related to the system future state (denoted $P'(Var,Act)$). If a *master action* is detected in the state condition context, then we should verify if the *post condition* holds in the execution that follows. If this is the case, a vulnerability has been detected. This *post condition* is not mandatory in a VDC.

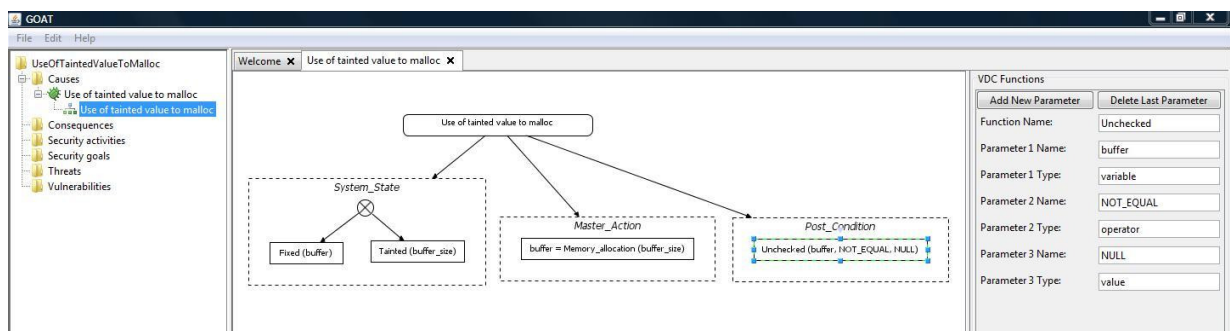


Figure 60. Vulnerability detection condition for “Use of tainted value to malloc” in GOAT


9.5.2 TSM editor plugin for GOAT

The **TEG Security Model (TSM)** editor is a GOAT plug-in that offers security experts the possibility to create TEG³ security models that will be used to detect the presence of vulnerabilities in communicating systems by stimulating them with a set of automatically generated TSM-based test cases.

The TSM editor has the following functionalities:

- The creation of new TSM corresponding to vulnerability causes.

³ TEG is one of Montimage testing tools. It allows to automatically generate and execute security test cases on Web-based applications.

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 95 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

- The visualisation of already created TSMs.
- The editing (modification) of existing TSMs in order to create new ones.

The TSMs are stored in an XML file that constitutes one of the inputs for the Montimage TEG tool. This tool aims at automatically generating test scenarios from a formal description of the system under test. This generation is made according to specific TSM-driven objectives called test purposes (e.g. a test objective can be for instance to check that the studied system can resist against denial of service or brute force attacks).

To better understand the design of the TSM editor, we provide the syntax of a TSM. TSMs permits to formally express security rules a system has to respect in order to prevent known security vulnerabilities. Each security rule describes a property that regulates the nature and the context of actions that can be performed within the studied system. A TSM provides a way to describe permissions, prohibitions and obligations related to the system actions within different contexts and potentially with time constraints.

A TSM model is composed of 3 parts:

- The Rule: It can be a permission (*P*), prohibition (*F* for forbidden) or obligation (*O*) rule.
- Rule context: It represents a set of conditions (on the system variables values and performed actions) that needs to be verified by the system under test in order to apply the rule.
- The property to verify: It represents a set of conditions (also on the system variables values and performed actions) that the system has to satisfy if the rule context holds.

In an obligation (respectively prohibition) security rule, if the rule context holds, then the property to verify MUST be *true* (respectively *false*).

Within the TSM model, a condition expresses a predicate related to the system status including the actions being performed, variables values and time constraints. It has two abstraction levels:

- On the abstract level, this condition is expressed in natural language and allows understanding the meaning of the conditions without any formal link with the studied system.
- In the instantiated level, this condition is linked to the real system. In our case, in order to automatically generate test cases, TEG tool relies on the functional model of the system under test specified as a finite state machine extended with variables and time. Thus, a condition within a TSM can be related to the system data (*variables*), system *input* or *output* messages, system *state*, system *tasks* or *transitions*. It can also be the logical combination of these simple conditions using classical logical operators (*AND*, *OR* and *NOT*) or chronological operators (like *FOLLOWED BY*). The instantiation of an abstract TSM allows generating test cases in order to verify if the security rule is respected or not.

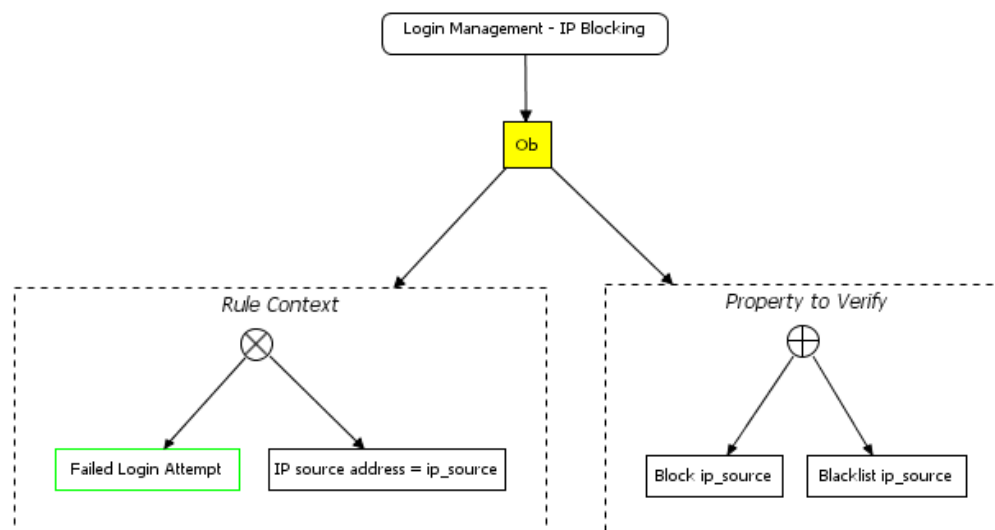


Figure 61. Graphical representation of IP blocking security rule

9.6 SEAMONSTER

Name	SeaMonster
Models supported	VCG, SAG, Attack tree, Misuse case
System requirements	Java runtime environment 1.5 or higher
Developed by	SINTEF ICT (SINTEF)
Available	http://sourceforge.net/projects/seamonsster
Miscellaneous	Open source, released under the LGPL license.

SeaMonster is a graphical security modelling tool based on a set of Eclipse frameworks. SeaMonster is continuously being developed by an open source community lead by SINTEF. The unique features of SeaMonster are that it supports notations and modelling techniques that security experts and analysers are already familiar with, and adds the functionality of linking them together based on a common information model. Modelling and linking different security aspects, such as causes, threats and countermeasures within the same tool, enables developers to use SeaMonster as a common platform for security modelling.

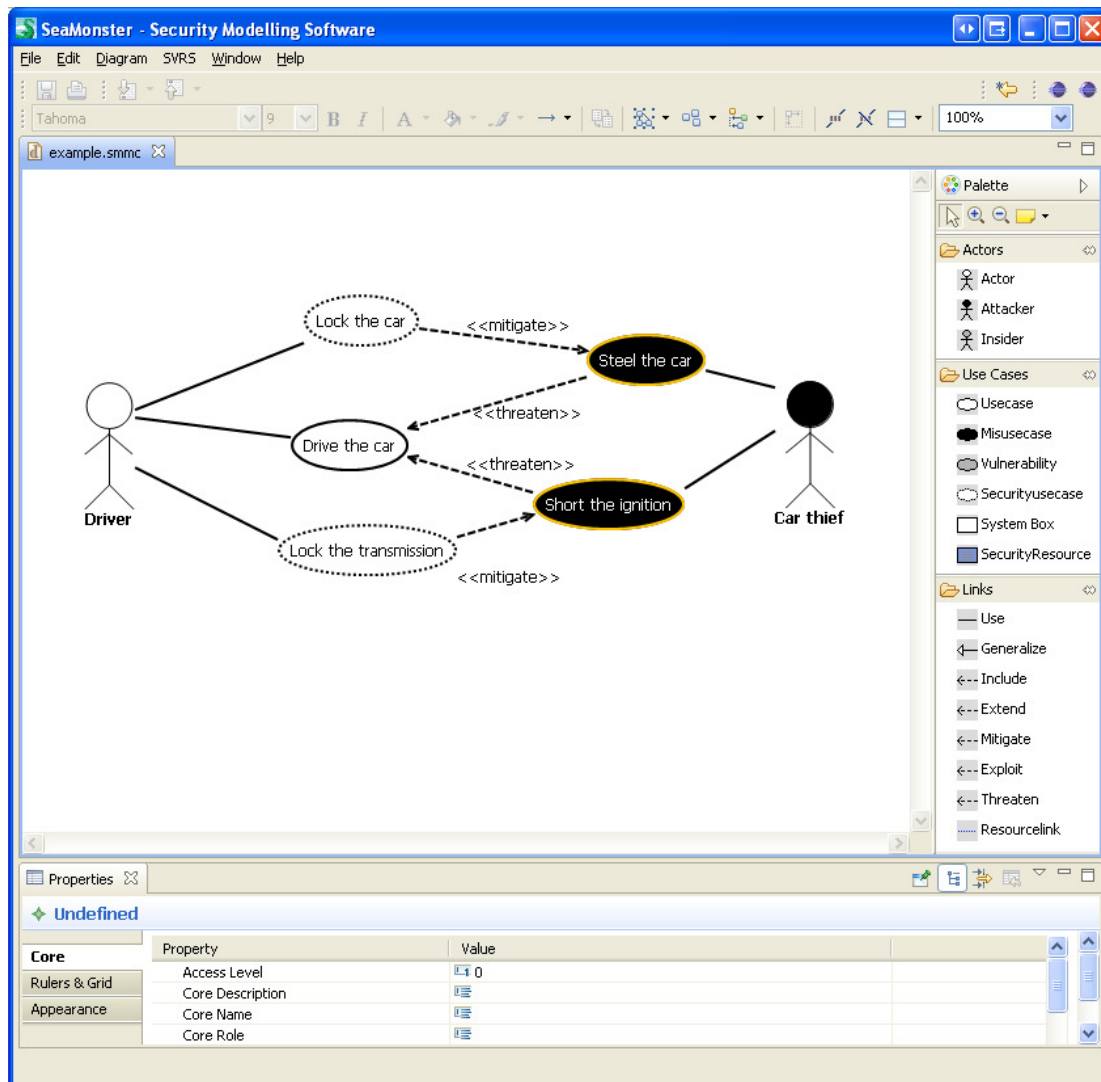




Figure 62, SeaMonster screen capture

10. REFERENCES

- [1] H. Götz, M. Nickolaus, T. Roßner, K. Salomon, "Model Based Testing - Modelling and generation of tests - basics, criteria for tool use, tools in the overview" (in German), iX Studie, 01/2009
- [2] Jürjens, J.: *Secure Systems Development with UML*, Springer, 2005
- [3] Jürjens, J.; Schreck, J. & Yu, Y.: *Automated Analysis of Permission-Based Security Using UMLsec; Fundamental Approaches to Software Engineering*, 11th International Conference (FASE), Springer, 2008, 4961, 292-295
- [4] Jürjens, J. Jézéquel, J.-M.; Hussmann, H. & Cook, S. (Eds.) *UMLsec: Extending UML for Secure Systems Development*, The Unified Modeling Language, Springer Berlin / Heidelberg, 2002, 2460, 1-9


	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 98 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

- [5] Montrieux, L.; Jürjens, J.; Haley, C. B.; Yu, Y.; Schobbens, P.-Y. & Toussaint, H.: *Tool support for code generation from a UMLsec property*; 25th IEEE/ACM International Conference on Automated Software Engineering (ASE), ACM, 2010, 357-358
- [6] Lodderstedt, T.; Basin, D. A. & Doser, J. Jézéquel, J.-M.; Hußmann, H. & Cook, S. (Eds.) *SecureUML: A UML-Based Modeling Language for Model-Driven Security*; The Unified Modeling Language, 5th International Conference, Springer, 2002, 2460, 426-441
- [7] Kaksonen, Rauli. A Functional Method for Assessing Protocol Implementation Security (Licentiate thesis). 2001. Espoo. Technical Research Centre of Finland, VTT Publications 447. 128 p. + app. 15 p
- [8] David W. McCoy. "Business Activity Monitoring: Calm Before the Storm". Gartner publication on April 2002. <http://www.gartner.com/resources/105500/105562/105562.pdf>
- [9] Aaron Newman. "Database Activity Monitoring: Intrusion Detection & Security Auditing". By Application Security Inc. CTO & Founder. http://www.appsecinc.com/presentations/DAM_wp82305.pdf 2011.
- [10] Jon Oltsik. "*Market Research Study: database security and compliance risks*". December 2009. <http://www.appsecinc.com/media/ESG-2010/ESG-Research-Summary-AppSecInc-Database-Security-December-2009.pdf>
- [11] Object Management Group (OMG). *UML 2.0 Testing Profile*, Final Adopted Specification. Version 1.0, July 2005. Available at: <http://www.omg.org/spec/UTP/1.0>
- [12] ISO/IEC. *Information technology - security techniques - code of practice for information security management*, 2nd edition. SS-ISO/IEC 17799, 2005.
- [13] ISO/IEC. *Information technology - security techniques - Information security management systems - overview and vocabulary*, First Edition. ISO/IEC 27000:2009, 2009
- [14] NIST. *Risk management guide for information technology systems*. NIST Special Publication 800-30, 2002
- [15] BSI. BSI-Standard 100-3: *Risk Analysis based on IT-Grundschutz*, Version 2.5, 2008
- [16] Howard Robert Chivers, Doctor Thesis: *Security Design Analysis*, University of York, 2006
- [17] Jörn Eichler: *Lightweight modeling and analysis of security concepts*. In Proceedings of the Third international conference on Engineering secure software and systems (ESSoS'11) Springer-Verlag, Berlin, 2011, p 128-141.
- [18] Thomas, I.; Nejme, B: *Definitions of tool integration for environments*, Software IEEE, 1992
- [19] Wassermann, A.: *Tool integration in software engineering environments*, Software Engineering Environment, Lecture Notes in Computer Science, 1990
- [20] ETSI ES 202 951 v 1.1.1: "Methods for Testing & Specification (MTS); Model-Based Testing (MBT); Requirements for Modelling Notations"
- [21] IETF RFC 4301: "Security Architecture for the Internet Protocol".
- [22] IETF RFC 4302: "IP Authentication Header".
- [23] ETF RFC 4303: "IP Encapsulating Security Payload (ESP)".
- [24] IETF RFC 4306: "Internet Key Exchange (IKEv2) Protocol".
- [25] ETSI TS 102 558: "Methods for Testing and Specification (MTS); Internet Protocol Testing

	<p style="text-align: center;">Review of security testing tools</p> <p style="text-align: center;">Deliverable ID: D1_1</p>	Page : 99 of 100
		Version: 1.1 Date : 27.6.2011
		Status : Final Confid : Public

(IPT): IPv6 Security; Requirements Catalogue".

- [26] ETSI TS 102 593: "Methods for Testing and Specification (MTS); Internet Protocol Testing (IPT); IPv6 Security; Conformance Test Suite Structure and Test Purposes (TSS&TP)".
- [27] ETSI TS 102 594: "Methods for Testing and Specification (MTS); Internet Protocol Testing (IPT); IPv6 Security; Conformance Abstract Test Suite (ATS) and partial Protocol Implementation eXtra Information for Testing (PIXIT) proforma".
- [28] ETSI ES 202 522: "Methods for Testing and Specification (MTS); TPLan: A notation for expressing Test Purposes".
- [29] ComputerWorld. 25 Mar 2010. Online:
http://www.computerworld.com/s/article/9174120/Pwn2Own_winner_tells_Apple_Microsoft_to_find_their_own_bugs
- [30] Dailydave emailist: <http://seclists.org/dailydave/2010/q3/10>
- [31] http://en.wikipedia.org/wiki/Vulnerability_scanner
- [32] Richard Bejtlich, "The Tao of Network Security Monitoring", Addison-Wesley; July 2004, ISBN 0321246772
- [33] Ross Anderson, "Security Engineering", Wiley Publishing Inc, 2nd edition, 2008
- [34] <http://sourceforge.net/projects/ratiso17799/>
- [35] <http://www.27000-toolkit.com/>
- [36] http://www.iso27001security.com/html/iso27k_toolkit.html
- [37] <http://www.microsoft.com/MOF>
- [38] <http://www.e-security-e-commerce-security.com/>
- [39] <http://www.microsoft.com/downloads/details.aspx?FamilyID=59888078-9DAF-4E96-B7D1-944703479451>
- [40] <http://www.microsoft.com/security/sdl/adopt/threatmodeling.aspx>
- [41] <http://riskwatch.com/>
- [42] <http://www.realismsoftware.com/>
- [43] <http://www.citicus.com/>
- [44] <http://www.callio.com/secura.php>
- [45] http://www.infogov.co.uk/proteus_enterprise/
- [46] <http://www.countermeasures.com/>
- [47] <http://www.bsi.bund.de/gstool>
- [48] <http://www.eclipse.org/modeling/emf/>
- [49] <http://www.eclipse.org/Xtext/>
- [50] <http://www.cebos.com/solutions/risk-management-software/>
- [51] <http://www.incom.com.au/Products/EnterpriseRiskManager.aspx>
- [52] <http://msdn.microsoft.com/en-us/magazine/dd347831.aspx>
- [53] <http://www.cramm.com/capabilities/risk.htm>

	<p align="center">Review of security testing tools</p> <p align="center">Deliverable ID: D1_1</p>	Page : 100 of 100
		Version: 1.1
		Date : 27.6.2011
		Status : Final Confid : Public

- [54] A. Abou El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel and G. Trouessin. Organization Based Access Control. *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, Lake Como, Italy, June 4-6, 2003.
- [55] <http://motorbac.sourceforge.net/>
- [56] <http://www.orbac.org/>
- [57] Charlie Miller. "Babysitting an army of monkeys: An analysis of fuzzing four products with 5 lines of Python". CanSecWest 2010. Vancouver, Canada. March 25, 2010.
- [58] ETSI ES 201 873-1: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language";
<http://webapp.etsi.org/workprogram/SimpleSearch/QueryForm.asp>
- [59] Diamonds consortium, "Deliverable D1.WP2: State of the Art in Risk Analysis Techniques For Security Testing".
- [60] Heikki Kortti. Buzz on Fuzzing. Codenomicon whitepaper. Dec 9th, 2007. Available online: <http://www.codenomicon.com/products/buzz-on-fuzzing.shtml>