Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3675

Secure Deployment of Business Process Services via Public Networks

Stefan Renner

Course of Study:	Softwaretechnik
Examiner:	Prof. Dr. Frank Leymann
Supervisor:	Dipl.-Inf. Sebastian Wagner, Dipl.-Ing. (BA) Martin Smolny (IBM)
Commenced:	March 24, 2014
Completed:	September 23, 2014
CR-Classification:	H.4.1, K.6.5

Abstract

Modern Business Process Management Systems enable customers to model processes and to design user interfaces without charging expensive IT specialists. To enable them to integrate such a system with their public web applications in the same manner, the upcoming security issues need to be discussed. This thesis provides a collection of identified attack patterns which are related to this scenario. These patterns are enriched with appropriate prevention recommendations. Additional, an abstract analysis method is described to identify a web application firewall configuration to secure the scenario. This analysis method gets exemplary used to identify a web application firewall configuration for a scenario with IBM Business Process Manager v.8.5.0.1 Standard.

Contents

List of Figures

List of Listings

1 Introduction

This chapter provides a short summary on the context of this thesis. The problem statement and the outline are given here. In addition, the fundamental motivation behind this work is explained.

1.1 Motivation

In the past, the typical customer of a Business Process Management System was a larger company with its own IT specialists. The Business Process Management Systems provided the core-functionality an the respective companies enhanced the solution with developing their own user interfaces.

Today, even customers without a dedicated IT development department are able to build working overall solutions. This is enabled by the extended functionality of modern Business Process Management Systems. Those systems are sometimes deployed to public networks for interaction with external stakeholders. The enhanced range of functionality of these systems increases the target for attacks.

For example, a company that produces household appliances could handle warranty cases by the use of a public accessible Business Process Management System. A customer who bought a faulty device could transmit the defect description via such a system. If additional interaction is needed, the customer can get involved to the process equally to other users of the system. The handling of warranty cases as it is described here could attract fraudsters. The fraud obtaining of guarantee is a possible threat for the fictive company, so security is important.

IT security is complex and sometimes even IT specialists have problems with this topic. It is contrary to the idea of enabling customers to build such solutions, when they are not able to secure them on their own. This problem needs to be worked on to satisfy the needs of the customers.

1.2 Problem Statement

The aim of this thesis is to create a guideline for customers as well as software architects of modern Business Process Management Systems. This guideline describes the components of modern Business Process Management Systems and their security vulnerabilities when interfaces are available via public networks.

The needed examination is done in general as well as focused on IBM Business Process Manager v.8.5.0.1 Standard, which is an established product on the market. The scope is on the architecture and the external interfaces of a Business Process Management System.
As result, recommendations are given on how the identified security risks can be minimized.

These recommendations are dedicated to customers as well as software architects to enable the secure deployment of Business Process Services via public networks.

1.3 Outline

This document describes how modern Business Process Management Systems are composed, how they can be attacked when interfaces are public accessible, and how this scenario can be secured. These points are discussed in theory and for the specific product IBM Business Process Manager.

First, the fundamentals and general components of a Business Process Management are explained in theory. Especially, how the involvement of humans to the processes evolved.
Afterwards, the IBM Business Process Manager is described and mapped to the preceding general explanation.

Chapter 4 contains the theoretical discussion on security. It consists of two major parts: First, security in general and the plausible use cases for deploying interfaces to a public network are discussed. In the second part, common known attack patterns are translated to the previous described use cases. Each translated pattern provides a general description, an example, and dedicated prevention recommendations.

Chapter 5 addresses the analysis of a Business Process Management System to mine the needed information to secure a scenario as it is described in chapter 4. This is exemplary done with the IBM Business Process Manager v.8.5.0.1.
This analysis is complemented with two small tools, which are developed for this purpose. These tools are described in chapter 6.

Finally, a summary of this thesis and of the achieved results is given.

2 Fundamentals and State of the Art

This chapter is an introduction to Business Process Management software. This is a non-vendor specific point of view, that combines information gathered from standards and specialized literature.

2.1 Basic Concepts of Business Process Management

Here, the basic concepts of "Business Process Management" are discussed. The core terms and concepts are covered, but for a more detailed view some specialist literature should be consulted (I.g. *Production Workflow - Concepts and Techniques* by Frank Leymann and Dieter Roller [LR00]).

2.1.1 Terminology

The sequence of actions that is performed during the processing of a loan request is a common example for explaining business processes. Such a sequence of actions gets executed according to a specified pattern. This pattern is used multiple times to get equal things done. Such a pattern is called "Process Model". Every sequence of actions that follows this process model is called "process"[LR00].
Such processes do not need to be executed by a computer. The corresponding process models are the organizational structure of the real world. For example, a process model tells a clerk what to do with a piece of paper that contains a specific information.

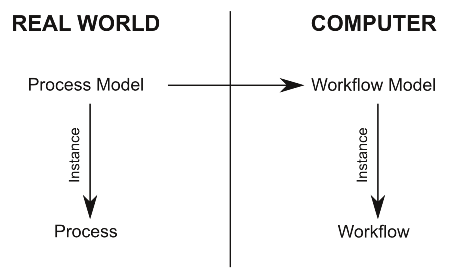


Figure 2.1: Processes and Workflows [LR00, p.7]

Addressing the technological progress, many companies started to execute parts or complete processes by the use of computer systems. The parts of a process model, which are mentioned to be executed by a computer, are called "Workflow Model". Consequently, an instance of a workflow model is called "Workflow"[LR00].

This explanation is equal to the formal definition of the term "Workflow" in the paper *The Workflow Reference Model* (WfRM)[HH94] that is published by the Workflow Management Coalition (WfMC)[1]

Definition - Workflow
The computerised facilitation or automation of a business process, in whole or part.

This leads to the term "Workflow Management System". Such a system provides the automation of processes in whole or part as mentioned in the definition of the term "Workflow". A formal definition can be found in the same paper:

Definition - Workflow Management System
A system that completely defines, manages and executes “Workflows” through the execution of software whose order of execution is driven by a computer representation of the workflow logic.

In reality and even in specialized literature the terms "Process" and "Workflow" are interchanged frequently and used imprecisely[LR00, p.8]. Normally, the context provides enough information to determine what is meant. This is justified by the fact, that it is seldom that a relevant difference exists. According to common speak, further the term "Process" is used.

2.1.2 Main Environments of a Workflow Management System

Workflow Management Systems are denoted as "Middleware". This type of software does not provide a final solution to a customer. Middleware is an enabler for building a solution.
To build such a solution by using a Workflow Management System, a modeling-environment is needed. The defined process models are instantiated afterwards in the execution-environment[HH94].

According to this, the characteristics shown in figure 2.2 are obvious. The mentioned environments are called "Buildtime" and "Runtime".

[1]http://www.wfmc.org

- **Buildtime** This environment provides the needed functionality to design process models. Many solutions provide also features to debug, analyze, and optimize the modeled processes.
- **Runtime** This component instantiates processes, navigates, and controls them. It invokes the defined operations according to the process models.

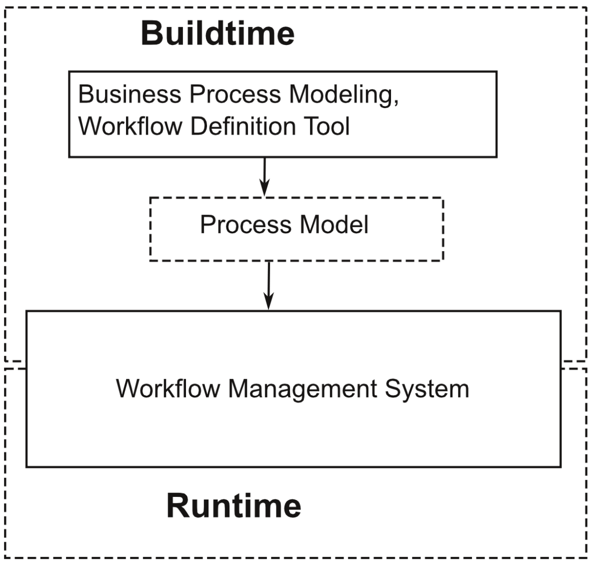

Buildtime
Business Process Modeling, Workflow Definition Tool
Process Model
Workflow Management System
Runtime

Figure 2.2: Runtime and Buildtime [LR00, p.62]

2.1.3 Users of a Workflow Management System

A Workflow Management System has to deal with a large amount of different users. These users have different assignments according to their role in the environment. According to *Production Workflow*[LR00, p.63-64] the users of a Workflow Management System are summed up in the following categories:

- **End Users** This is the biggest user group of a Workflow Management System. These users are supposed to execute their assigned work. These users have just a minimal administrative empowerment. For example, claiming work that is assigned to an ill colleague. This interactions are taking place via web interfaces.
- **Process Administrators** These users are responsible for the correct execution of the processes. A Process Administrator is typically assigned to a defined set of process models and/or instances.

- **Operation Administrators** They ensure that the Workflow Management System itself is running properly.
- **System Administrators** Operating the Workflow Management System is just one of their charges. They are responsible for the total system of a company. These guys own the highest privileges and they are called when Process Administrators or Operation Administrators are not able to solve a problem.
- **Customer Support People** When external users interact with the Workflow Management System, then they try to check the state of their flight reservation or start a loan request. But not everyone is familiar with a provided interface, so the companies provide support.
- **External Users** These users are not employed by the company that runs the Workflow Management System. The opposite is an Internal User.

2.1.4 Process Definition

To design a process model a syntax is needed that inherits the needed expressiveness. This syntax determines the possibilities of the process modelers. To achieve compatibility between the products of different vendors, these syntaxes are standardized. Here, two of these standards are discussed.

- **WS-BPEL** The "Business Process Execution Language"[2] is a system-centric language to define processes. BPEL is standardized by the OASIS[3] ("Organization for the Advancement of Structured Information Standards").
  There is no graphical representation of the processes in the standard. BPEL mainly focuses on the orchestration of webservices and is well integrated to the WS-stack [4].
- **BPMN** The "Business Process Modeling and Notation"[5] standard is about a human-centric approach to define processes. First, BPMN was only a graphical notation of the processes, but v.2.0 introduced a XML-based notation of the process models.
  Figure 3.3 shows an example of a BPMN process model.

[2] http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
[3] https://www.oasis-open.org/
[4] http://www.w3.org/standards/webofservices/
[5] http://www.omg.org/spec/BPMN/2.0/

2.2 The Workflow Reference Model

In 1995, the Workflow Management Coalition[6] (WfMC) published the Workflow Reference Model (WfRM)[HH94]. Before this publication Workflow Management software was already available, but the different vendors were still searching for the best way to automate business processes.
The Workflow Reference Model is important, because it represents a synchronization of the researching vendors on how to do it the best way.
The Workflow Reference Model does not reflect the exact functionality of all available Workflow Management products, neither in 1995 nor today. Rather, it is a common accepted compendium, even for the actual available solutions.

The following sections outline the key components of the Workflow Reference Model, but to get a full indoctrination the publications of the Workflow Management Coalition should be consulted.

2.2.1 Metamodel

To define processes a metamodel is needed. The Workflow Management Reference Model contains a basic metamodel to provide a fundamental principle for this.
The metamodel describes some basic types and their correlation to each other. These types can used to represent basic processes.

A general overview is provided by figure 2.3. Here, the explanation of the metamodel is restricted to a general description of some terms and concepts. To get a deeper understanding the appropriate publications of the Workflow Management Coalition should be consulted.

- **Workflow Type Definition** This is the "root" of a process definition. It holds the "Workflow Relevant Data" and consists of "Activities". For example, a property of this type is the name of the process model.
- **Activity** Do not mistake an "Activity" with the terms "Task" or "Workitem", which are used more often in this thesis. During the execution of a process multiple logical steps are reached. Such a step is called "Activity". For example, if a graph is used as base-structure of a process model, an "Activity" would be a node. Instead, a "Workitem" is something that has to be done. This could be the content of an "Activity".
- **Workflow Relevant Data** Data, generated during the execution of an activity, may has to be passed to a ongoing activity. To do this, types have to be defined and a structure

[6]http://www.wfmc.org/

has to be determined on how to store and access this data. To leak this data is a potential target for an attacker.

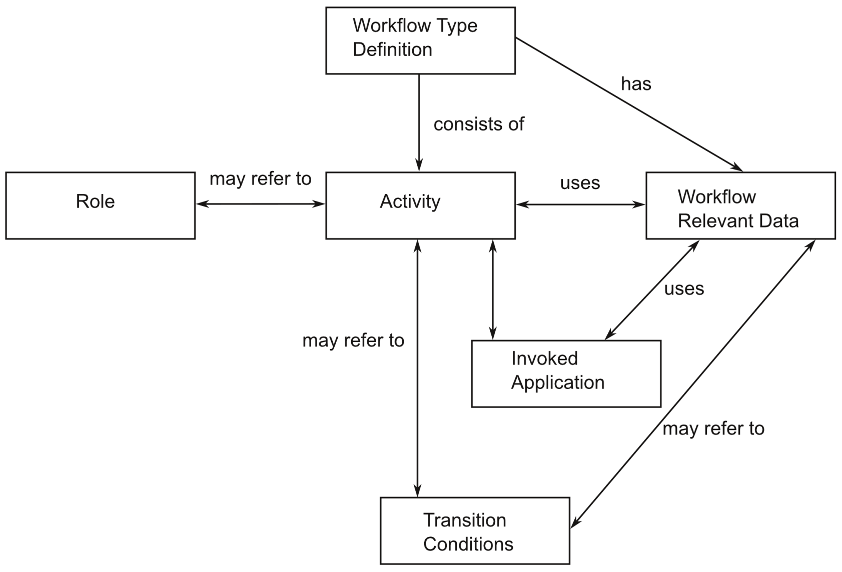


Figure 2.3: Metamodel published in the Workflow Reference Model [HH94]

2.2.2 Architecture

The second component that is discussed here, is an abstract architecture. This is the most important component of the Workflow Reference Model, because it identifies the general components of a Workflow Management System.
For understanding the architecture that is plotted in figure 2.4, some additional terms have to be explained.

- **Workflow Enactment Service** A Workflow Enactment Service is a dedicated instance of a Workflow Management System. It runs the particular processes and it is an interaction container as illustrated in figure 2.4. It contains at least one "Workflow Engine", but it could also contain multiple engines, or even different specialized engines.
- **Workflow Engine** This is the component that explicitly navigates the processes. A "Workflow Engine" can be specialized or more common.

It is important to mention, that no setup-related restrictions are specified here. The main reason why this architecture is important are the five stated interfaces of a Workflow Enactment Service.

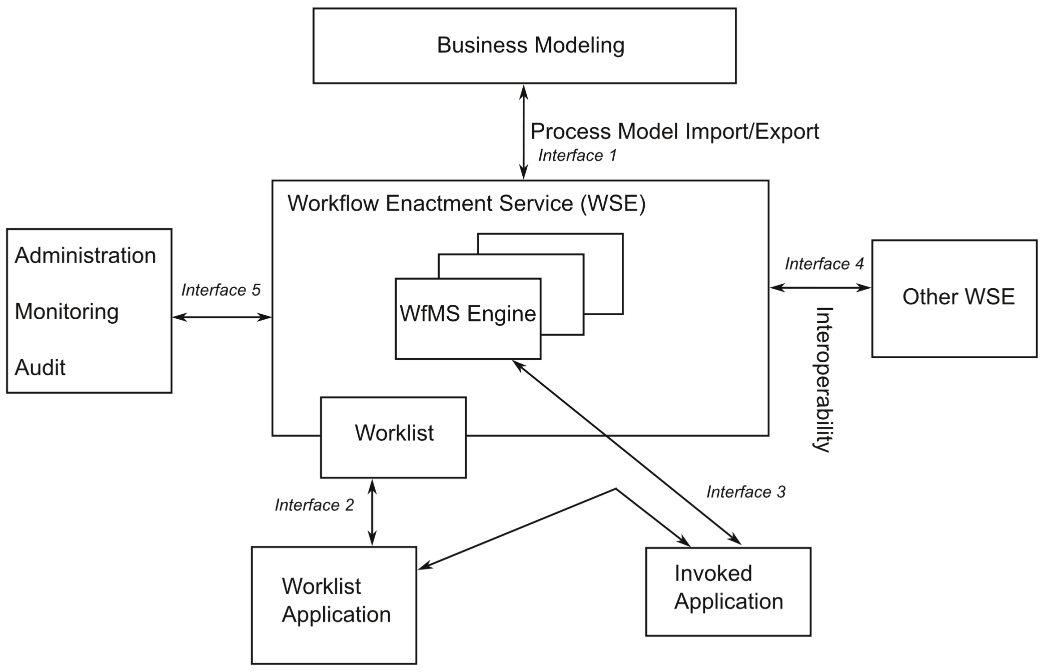


Figure 2.4: The Workflow Reference Architecture [HH94]

The following descriptions map to the annotations in figure 2.4

- **Interface 1** This interface enables other systems or modeling tools to import/export process definitions and metadata.
- **Interface 2** This interface enables the end users to interact with the system. This interface is described in section 2.3.
  An application caches workitems out of one or more Workflow Management Systems and processes them with by the involvement of humans. The process models have to define conditions on how to choose the appropriate team or user that is to be charged.
- **Interface 3** This interface enables to host implementations outside of the Workflow Management System. This is the most complex interface, because of the dozens of different techniques on how to interact with other systems.
- **Interface 4** This interface enables the engine to execute a sub-process by the call of a service that is provided by a different Workflow Enactment Service.
- **Interface 5** This interface provides the administrative functionality (Monitoring, Audit, etc).

The combined global interface of a Workflow Enactment Service is called **WAPI** (Workflow Application Programming Interface). This term summarizes all the mentioned interface methods to one technical interface. This is valid, because there are a couple of functions that are shared between the single interfaces and there is no need of a technical separation.

2.3 Involving Humans to a Workflow Management System

The best case with automating business processes would be the automation of everything. The amount of consumed time and money would be minimal. For example, labor costs would be decreased, because a big part of the employees would become needless.
This is just theoretical. A lot of scenarios need to involve humans, because they are able to perform actions which can not be executed by a computer.

Let's have a look at two examples, which are showing why the involvement of humans is necessary:

- During a the execution of a hiring process different steps are executed. Steps like checking the police clearance conduct or whether the candidate has an employment permit can be automated.
  But a hiring process contains also aspects to assure that a candidate has the necessary softskills or language competences. For getting these skills rated a specialized employee has to talk with the candidate. Additional, softskills can not be measured and displayed on a diagram. The knowledge of the human nature, which is necessary to achieve a good decision, is an absolute reason to involve humans to this process.
- The first example describes the collection and analysis of information, but this example it is all about providing information.
  The scenario is about processing a loan application in a bank. Normally, a customer is not familiar with the actual credit offerings of the bank. It would be possible to provide the necessary information in an automated manner (generated emails, websites, or even brochures), but this is no high-quality service. To achieve a good understanding of the provided information, a financial consultant should talk to the customer.

2.3.1 Simplified Approach and Basics

When humans interact with computers user interfaces are needed. The most important feature of these user interfaces is the ability to work with "Worklists" and "Workitems". A worklist is a set of workitems. Each workitem is an atomic action that has to be executed. A workitem can be assigned to a specific person or to a team. In the case of the assignment to a team, the workitem would appear on the worklists of each team-member.

By using the mentioned user interface, which is also a so called "Worklist Client", the user is able to view his worklists and the contained workitems. Now, the user probably decides to execute a workitem. He claims the workitem and the system removes the workitem from the worklists of the others.

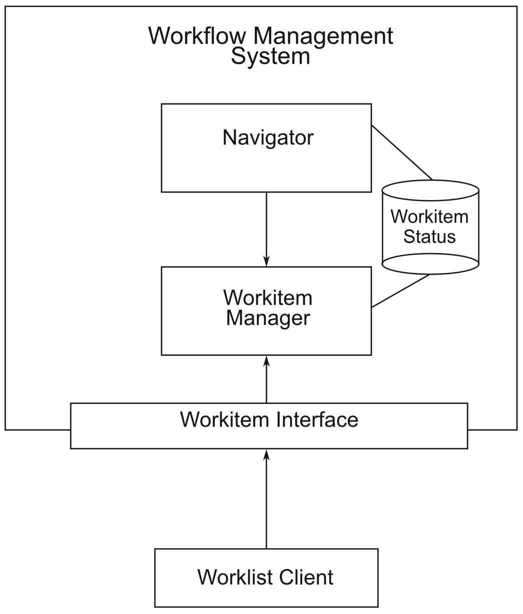


Figure 2.5: Processing of Workitems [Ley12]

Figure 2.5 shows the involved components of this scenario. The *Navigator* steps through the process and if the invocation of one or more humans is necessary, it passes activity A and the staff query q to the *Workitemmanager*. The *Workitemmanager* generates workitems based on this input and stores them to the database. The *Worklist Client* communicates via the *Workitem Interface* with the *Workitemmanager* to display the workitems.

While figure 2.5 provides a component-focused embodiment, figure 2.6 plots the sequential concept of the solution.

The first crux in this scenario is that the determination of the suitable agents is a query (Staff Query q) to the database.
The second crux is that the interaction between the client and the system is also a query that requests the assigned workitems.

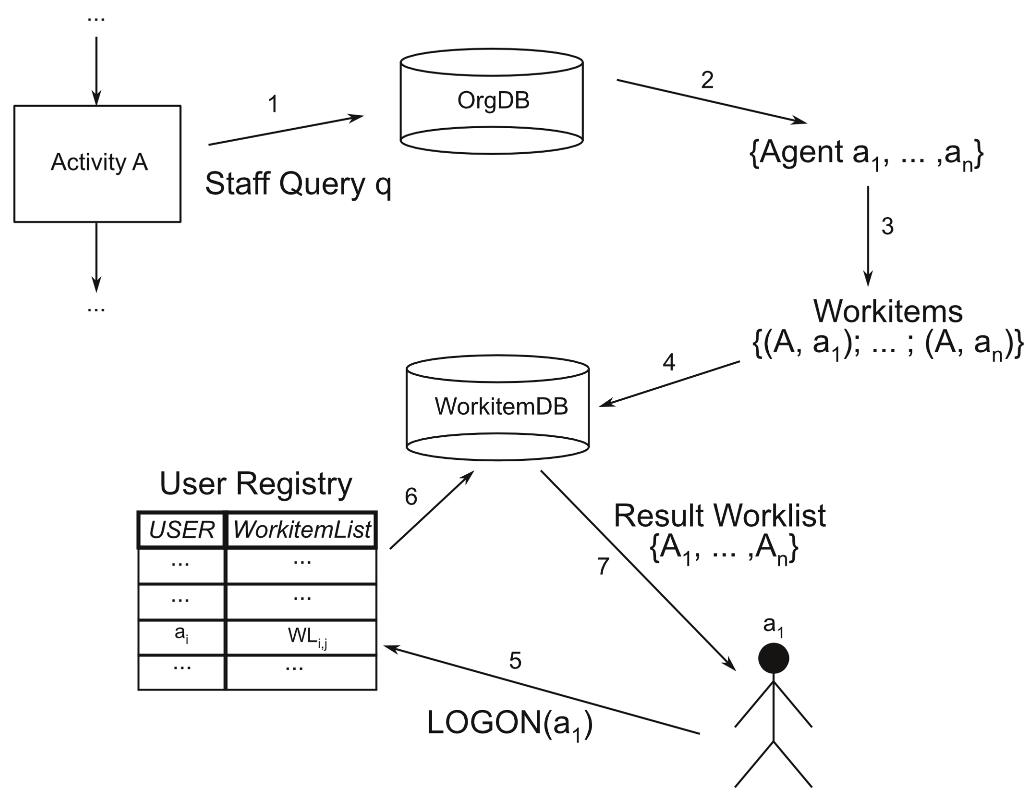


Figure 2.6: Sequential processing of Workitems [Ley12]

2.3.2 Refined Approach of actual Solutions

BPEL emerged as the most accepted language in industry. BPEL is about orchestrating webservices, so BPEL had to be extended to enable the involvement of humans.
In June 2007 two drafts were submitted to the OASIS for standardization. These drafts are called "WS-Human Task" and "BPEL4People". Even if they are not "real" OASIS standards by now, they are de-facto standards.

WS-Human Task contains an architectural approach on how "Human Tasks" can be processed. This new design is more complex than the approach that is presented in 2.3.1. This has to be seen as the next version of the earlier approaches and not as a complete new one.
As the name of the specification implies, the term "Workitem" is obsolete. Instead the term "Task" is introduced. Here, a task is something atomic that has to be performed by a human[KKL$^+$05, p.13].

The most evident change is the outsourcing of the task-processing from the Workflow Management System as shown in figure 2.7.

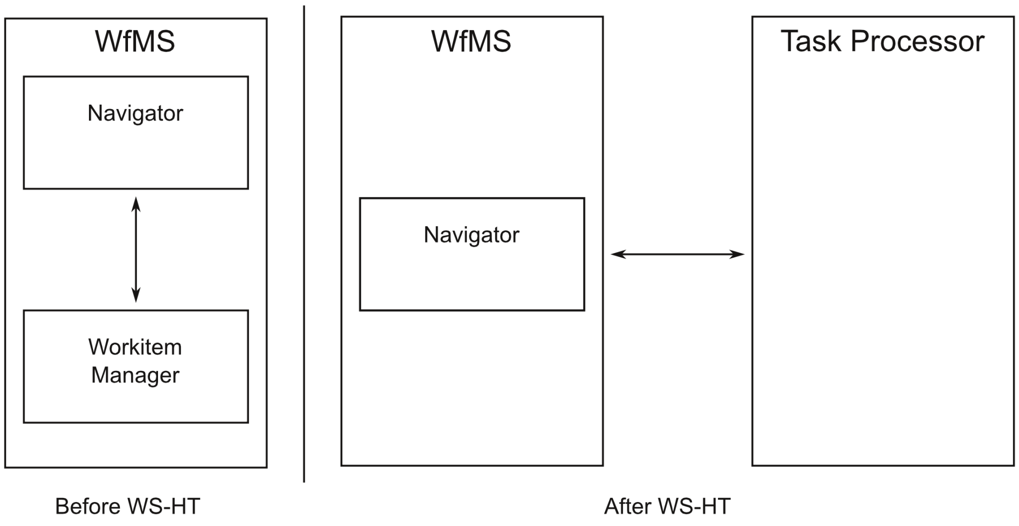


Figure 2.7: Design Change with WS-Human Task [CKM+10, p.10]

This design is more flexible and more handsome when it comes to multiple engines which run together. The "Task Processor" is accessible via common technologies and therefore it can be shared. The other way round, it can combine task lists to less or only one single list for one user.

Figure 2.8 is a detailed view on the architectural proposal that comes with WS-Human Task. The main components and their intended conduct are described in the here:

- **Task Processor** This is similar to the workitem manager that is described in 2.3.1. The behavior of a task processor has parallels to the behavior of a Workflow Management System. A task definition can be deployed to the system by a modeler and instances can be created. In short, this component hosts webservices, which can be called. These services include the special feature that humans are involved. However, this might lead to problems with timeouts or other issues which are related to the human nature. A modeler who calls such "Human Services" has to compensate this fact.
- **Task Parent** If a task definition is available on a task processor, instances can be created by calling the appropriate interface methods. The requester is named "Task Parent".
- **Task Client** The definition of the user interface, that is needed to perform the task, is contained in the task definition. Additional, a visualization of the task list and other essential functionality has to be provided. The client provides requests the user interface

definitions when a task is claimed.
Beware: It is possible that the user interface definition contains requests to other systems. With the exclusion of the client from the task processor, the client is enabled to interact with multiple task processors. Therefore, combined task lists can be provided.

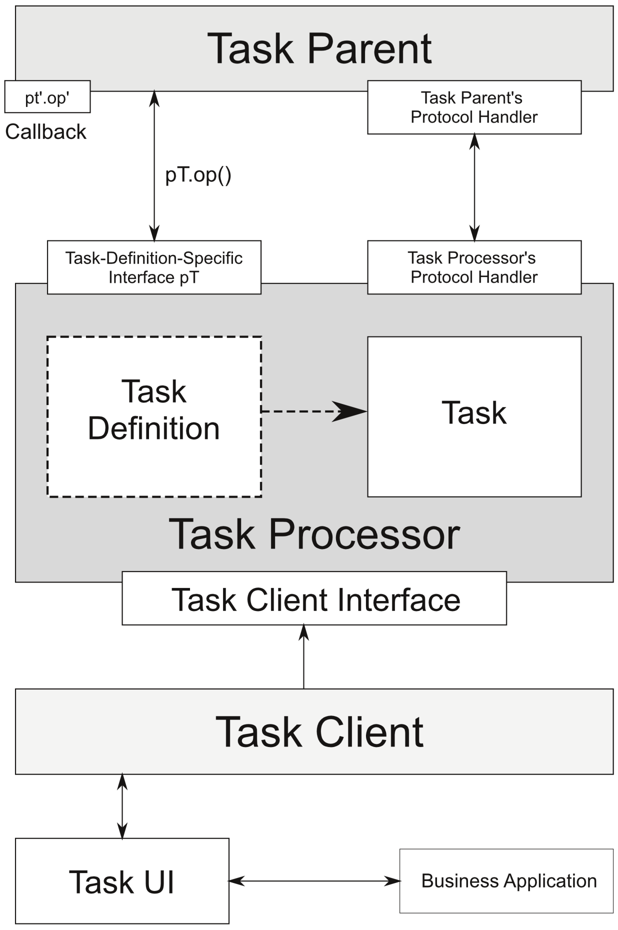


Figure 2.8: Component Interaction with WS-Human Task [CKM+10, p.13]

3 IBM Business Process Manager

This thesis examines IBM Business Process Manager v.8.5.0.1 Standard. The examination itself and the context of this examination is described in chapter 5. This chapter provides an overview to the product and the correlating concepts.

Chapter 2 is a theoretical introduction to Business Process Management software, while this chapter is focused on the specific product.

3.1 Configuration

There are three different configurations of IBM Business Process Manager [UP14, p.7]. These configurations envelop different features, licensing, and integration capabilities.

- **Express** This is the smallest configuration of IBM Business Process Manager. It is designed for small Business Process Management projects with just a few users. The limit of this configuration is that it can not be clustered. This configuration uses IBM Process Designer for modeling the process definitions.
- **Standard** Normally, this is the configuration a customer orders when he decides to go into production. Finally, this is the same as *Express*, but there is no restriction on clustering.
- **Advanced** This configuration includes the standard configuration and extends the portfolio with IBM Business Space Manager, IBM Business Process Choreographer and a common event infrastructure.

This enumeration references the portfolio that has evolved after the acquisition of Lombardi by IBM[1]. This set does not represent the complete IBM portfolio in Business Process Management software. IBM developed its own Business Process Management products long before the acquisition of Lombardi. This (latest) product is based on BPEL and is known as WebSphere Process Server (WPS)[2]. This technology can also be licensed with the *Advanced* configuration.

[1] http://www-03.ibm.com/press/us/en/pressrelease/28890.wss
[2] http://www-03.ibm.com/software/products/de/wps#WebSphereProcessServer70

3.2 Architecture

To understand how IBM Business Manager works, the general components need to be explained. These components are [AAG+13]:

- **Process Server** A process definition is published to a *Process Server* to put it into operation. In the context of IBM Business Process Manager process definitions are called "Business Process Definition"(BPD).
  For test-issues, a *Process Server* is also contained in a *Process Center* (so called: Playback Server).
  The component *Process Server* is completely located in the runtime-environment, which is described in 2.1.2.
- **Process Center** This component inherits the repository for the BPDs and provides tools to manage their entire lifecycle. The *Process Center* belongs to the runtime and the buildtime-environment, which are described in 2.1.2. But, this is not equal to the component "Workflow Management System" that is shown in figure 2.2. A Process Center is a part of such a system.
- **IBM Process Designer** The *IBM Process Designer* is used to model BPDs and correlating resources. It provides additional features to support the process modelers. For example, the BPDs can be debugged, analyzed and optimized by using the "Playback Server", that is contained in the Process Center.
  The *IBM Process Designer* is the equivalent to the "Workflow Definition Tool" that is shown in figure 2.2.
- **IBM Integration Designer** The *IBM Integration Designer* used by IT developers. It is mentioned here for completeness.

  IBM Business Process Manager is a huge software product. So, there are components, which are not mentioned here. More information about the IBM Business Process Manager is contained in the product related publications[3], especially in Neil Kolban's book about the IBM Business Process Manager [Kol14].

3.2.1 Process Server

A Process Server is a single runtime-environment. It inherits a web interface for administration purposes, an integrated portal-solution for task execution (IBM Process Portal), IBM Business Space, IBM BPM Performance Admin Console, and the IBM Business Process Choreographer Explorer. In addition, each Process Server is enabled to support performance data warehousing,

[3] http://www.redbooks.ibm.com/

which is needed for reporting and optimization.
There a two different configurations [AAG+13, p.22]:

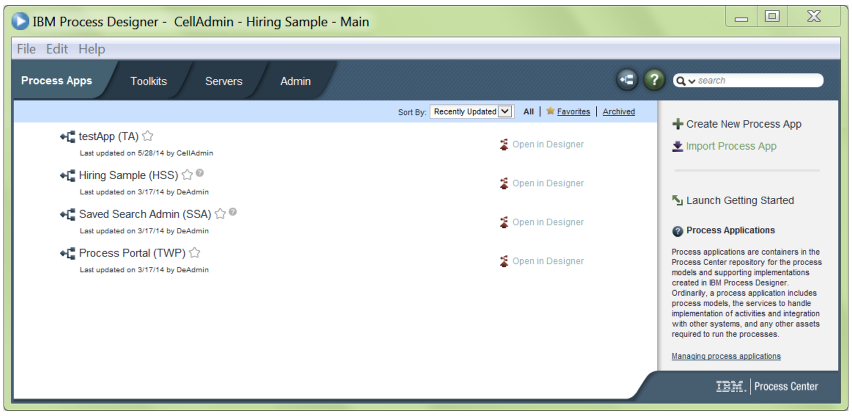
- **Online/Connected** This does not mean that the Process Server is connected to a special network. It means, that the Process Server is managed by a Process Center.
- **Offline** Contrary to the implicit meaning of the word "Offline", such a Process Server can be connected to one or more networks. This configuration is characterized by the fact that a Process Server is not connected to a Process Center.

3.2.2 Process Center

A Process Center is a central component of a IBM Business Process Manager system. It is a repository for Business Process Definitions and related resources.
But a Process Center is more than just a simple repository. It is also the link between runtime and buildtime-environment.

3.3 Process Modeling with IBM BPM Standard

A process modeler can use the IBM Process Designer to connect to a Process Center and to create, edit, debug, or optimize BPDs and all types of corresponding resources. This includes, among others, "Human Services", Ajax-Services[4], and "Coaches".



testApp (TA)
Last updated on 5/28/14 by CellAdmin
Open in Designer

Hiring Sample (HSS)
Last updated on 3/17/14 by DeAdmin
Open in Designer

Saved Search Admin (SSA)
Last updated on 3/17/14 by DeAdmin
Open in Designer

Process Portal (TWP)
Last updated on 3/17/14 by DeAdmin
Open in Designer

Process Applications

Process applications are containers in the Process Center repository for the process models and supporting implementations created in IBM Process Designer. Ordinarily, a process application includes process models, the services to handle implementation of activities and integration with other systems, and any other assets required to run the processes.

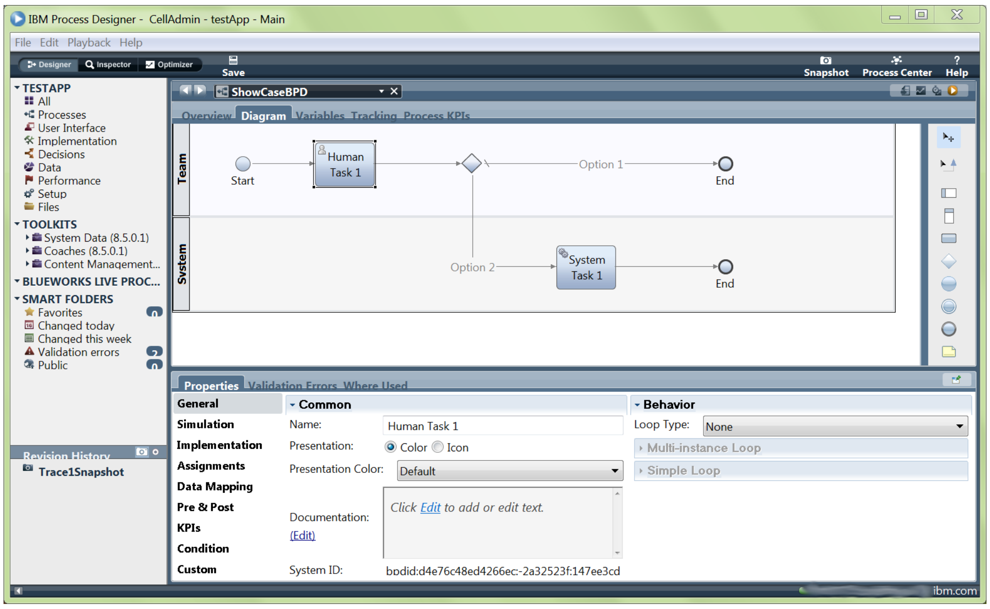
Managing process applications

Figure 3.1: IBM Process Designer connected to a Process Center

[4]http://ajaxpatterns.org/

A BPD can not be instantiated when dependent resources are not available. This is why all depended definitions are bundled. These bundles are called "Process Applications". Process Applications are containers, which are used for deploying functionality to Process Servers.

When the IBM Process Designer is used to connect to a Process Center, the Process Applications on this Process Center are listed and can be loaded for ongoing development, inspection, or optimization. This is shown in figure 3.1. Toolkits, which can be used in the Process Applications, can be managed as well as the connected Process Servers. This is explained in 3.5.



Name: Human Task 1

Presentation: [✓ Color] Icon

Presentation Color: Default

Documentation: (Edit) *Click Edit to add or edit text.*

System ID: bpdid:d4e76c48ed4266ec:-2a32523f:147ee3cd

Loop Type: None

Figure 3.2: Process Application opened in IBM Process Designer

IBM Process Designer uses the Business Process Management and Notation v.2.0 (BPMN) syntax for defining processes. BPMN is also described in 2.1.4.

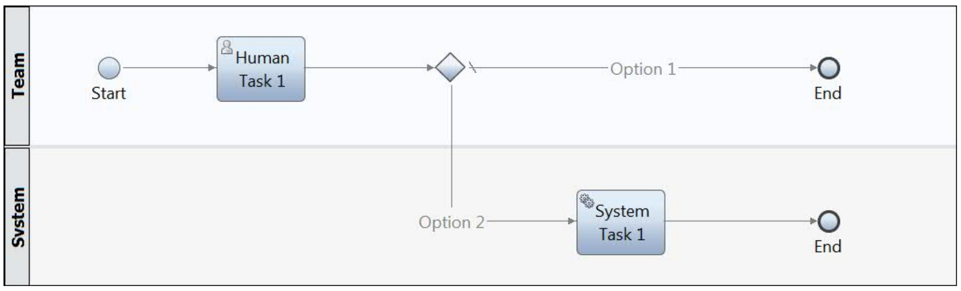
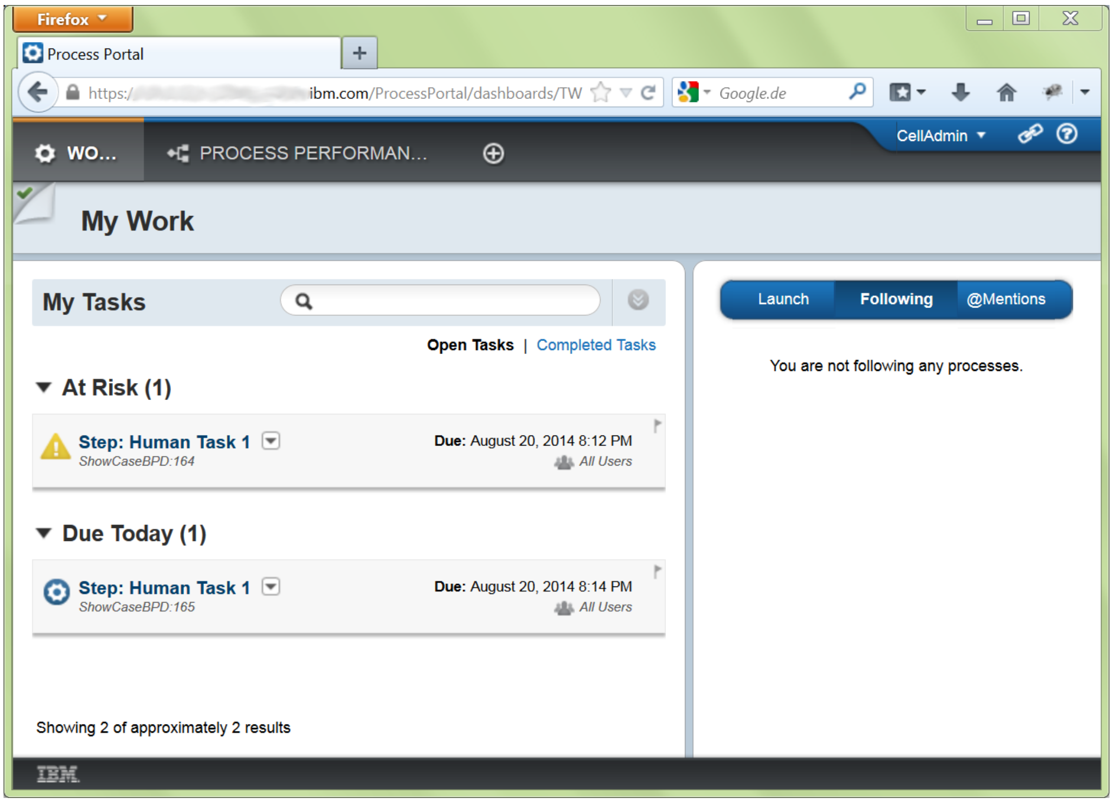


Figure 3.3: A Process modeled with BPMN

One of the key-concepts of BPMN are the so called "BPMN-Lanes". When an activity comes to execution, the lane determines WHO has to execute the corresponding task.
For example, figure 3.3 shows an activity that is called "Human Task 1". To determine who has to execute the task, the lane "Team" is processed. Each member of "Team" is a potential executor of the task. A BPD can contain more lanes than shown in figure 3.3.

3.4 Human Services

When a Human Task has to be executed, the engine calls a "Human Service" and the corresponding process sleeps until the service returns a result. This is equal to the setup that is explained in 2.3.2.



My Work

My Tasks

Open Tasks | Completed Tasks

At Risk (1)

Step: Human Task 1
ShowCaseBPD:164
Due: August 20, 2014 8:12 PM
All Users

Due Today (1)

Step: Human Task 1
ShowCaseBPD:165
Due: August 20, 2014 8:14 PM
All Users

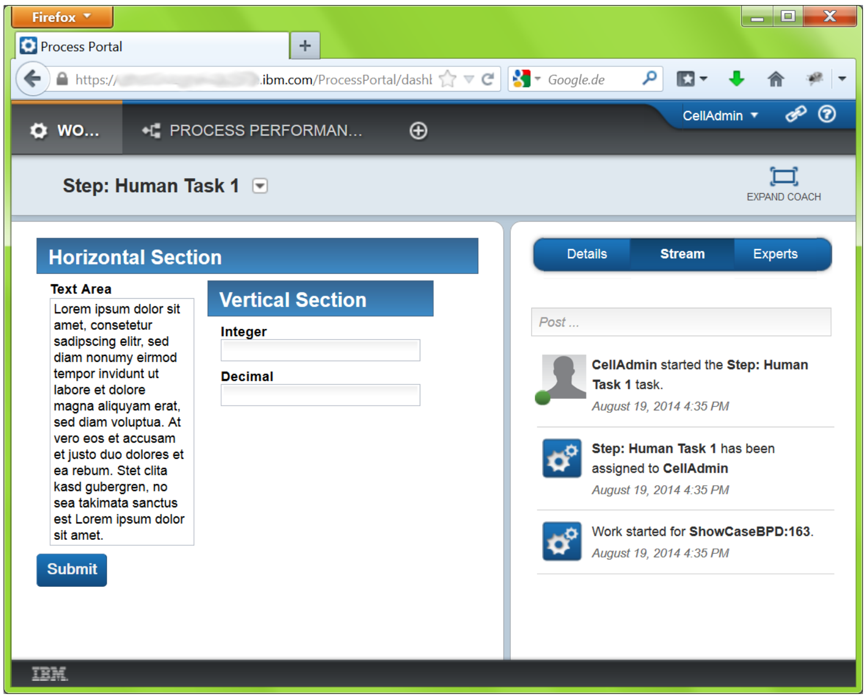
Showing 2 of approximately 2 results

Launch | Following | @Mentions

You are not following any processes.

Figure 3.4: Task List presented in IBM Process Portal

When a Human Service is called, an entry is added to the task lists of the team members of the corresponding BPMN-Lane. These users retrieve their personal task list by accessing the portal that is integrated to IBM Business Process Manager and that bases on IBM WebSphere Portal[5].



Step: Human Task 1

Horizontal Section

Text Area

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Vertical Section

Integer

Decimal

Submit

Details | Stream | Experts

Post ...

CellAdmin started the **Step: Human Task 1** task.
August 19, 2014 4:35 PM

Step: Human Task 1 has been assigned to **CellAdmin**
August 19, 2014 4:35 PM

Work started for **ShowCaseBPD:163**.
August 19, 2014 4:35 PM

Figure 3.5: A Coach is displayed in IBM Process Portal

When a user claims a task, he has to execute it afterwards. To do this, a user interface is needed. These user interfaces are called "Coaches" in this context. A sample "Coach" is shown in figure 3.7.

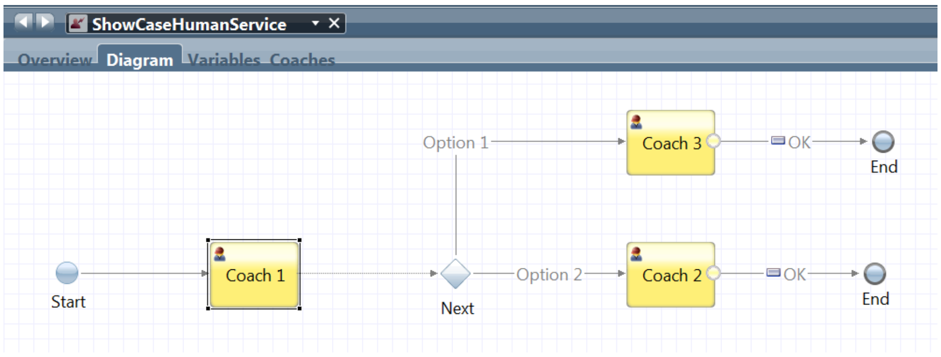
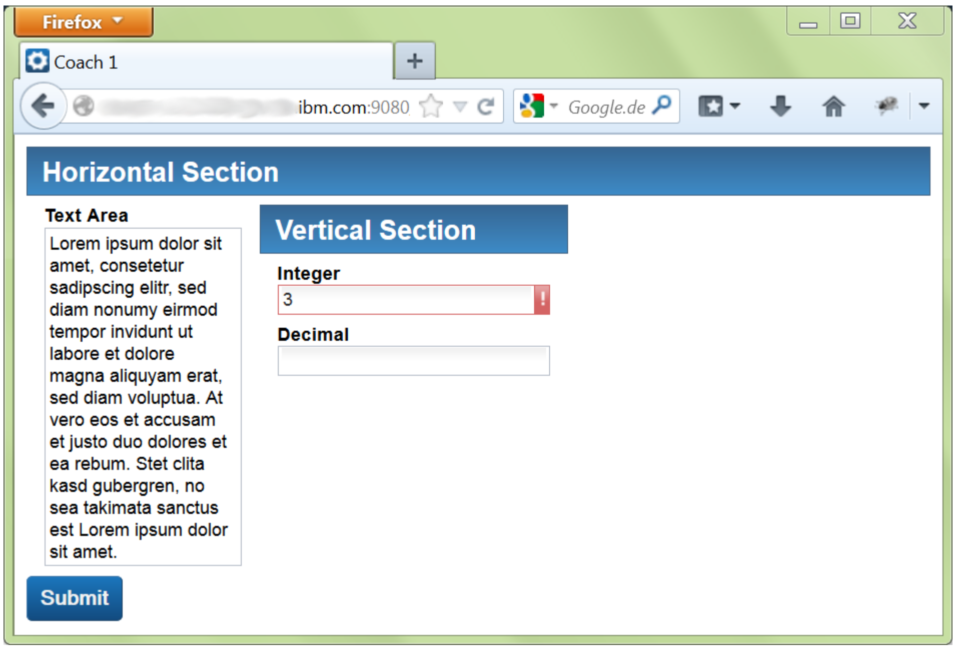


Figure 3.6: Modeling a Human Service that contains multiple Coaches

[5]http://www-03.ibm.com/software/products/en/websphere-portal-family

Additional, the Human Service may contains multiple Coaches in a sequence. The definition of this sequence is done as shown in figure 3.6.



Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Figure 3.7: Single Coach in a Browser

Coaches are able to provide functionality. Each Coach is a combination of single "Coach Views". For example, a text-field is a Coach View. Each Coach View can provide functionality like input-validation or auto-completion. Some functionality is implemented in scripts which are executed on the client-side, but other functionality has to be enabled via the transparent usage of Ajax-services. IBM Business Process Manager contains a set of standard Ajax-services which are usable in Coaches. Additional Ajax-services can be modeled with the IBM Process Designer.

For example, the result of an input-validation is shown in figure 3.7. The integer-field contains a value that does not fit the requirements.

3.5 Common System Setup

As described in 3.2.2, a Process Center is a repository for Process Application. A setup that is used by a large company does not run just one Process Center as shown in figure 3.8. A realistic topology is plotted in figure 3.9. No serious company puts the modeled Process Applications into production without testing and optimizing them.

To do this, multiple runtime-environments are needed. Different types of possible runtime-environments exist. Figure 3.8 shows these different environments, which are all connected to a Process Center.

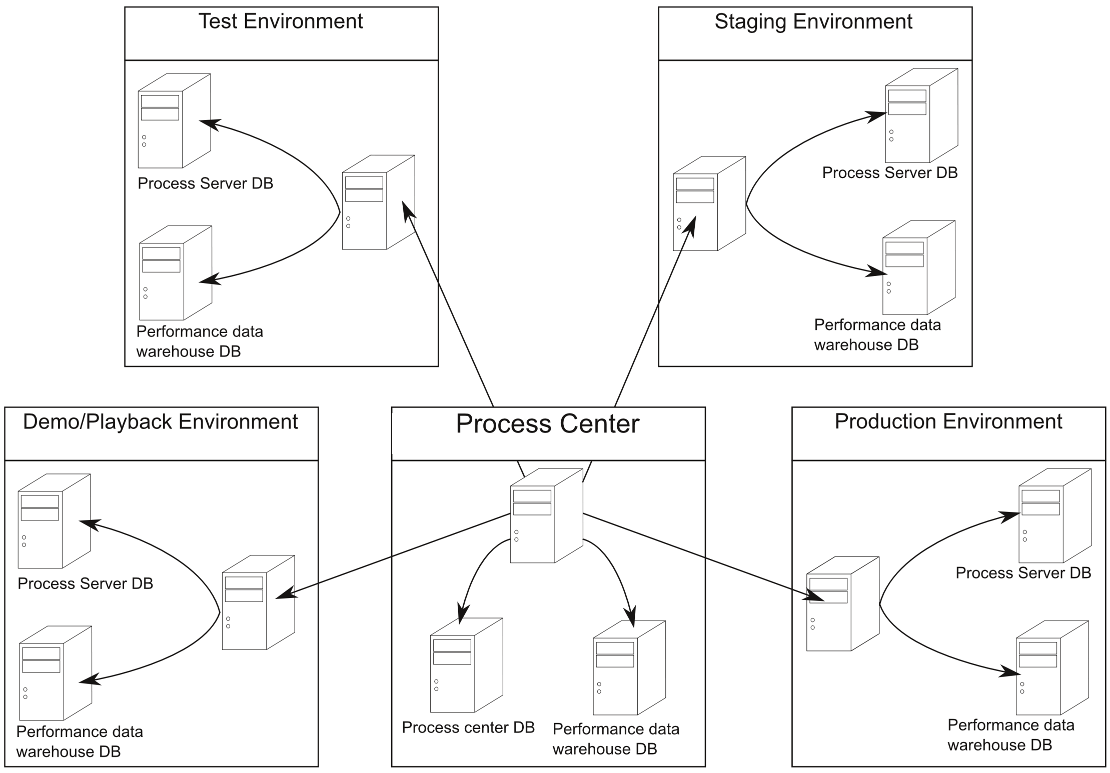

Test Environment
Process Server DB
Performance data warehouse DB
Staging Environment
Process Server DB
Performance data warehouse DB
Demo/Playback Environment
Process Server DB
Performance data warehouse DB
Process Center
Process center DB
Performance data warehouse DB
Production Environment
Process Server DB
Performance data warehouse DB

Figure 3.8: Different Runtime Environments with IBM Business Process Manager [AAG+13, p.27]

Not all mentioned types of runtime-environments are used in every company that uses IBM Business Process Manager. Here, the different types are explained separately:

- **Demo/Playback Environment** During modeling, debugging, or optimizing a Process Application it is an advantage to have a Playback-environment. Alternatively, the Process Server that is contained within a Process Center can be used for this type of work (Playback Server).
- **Test Environment** To ensure that a Process Application works correctly, testing is essential. After a modeler finished a piece of work, the result is deployed in a test-environment and tests are performed. This step addresses the functional tests and formal quality checks.

- **Staging Environment** After functional testing some issues are still not tested properly. A Process Application can interact with a lot of different systems and other processes. This means, the integration with these other systems and processes has to be tested.
- **Production Environment** After a Process Application passes all tests, it can be put into production. Of course, issues like restructuring the company need to be handled, so a Process Application will be replaced repetitive by a modified, optimized and updated one.

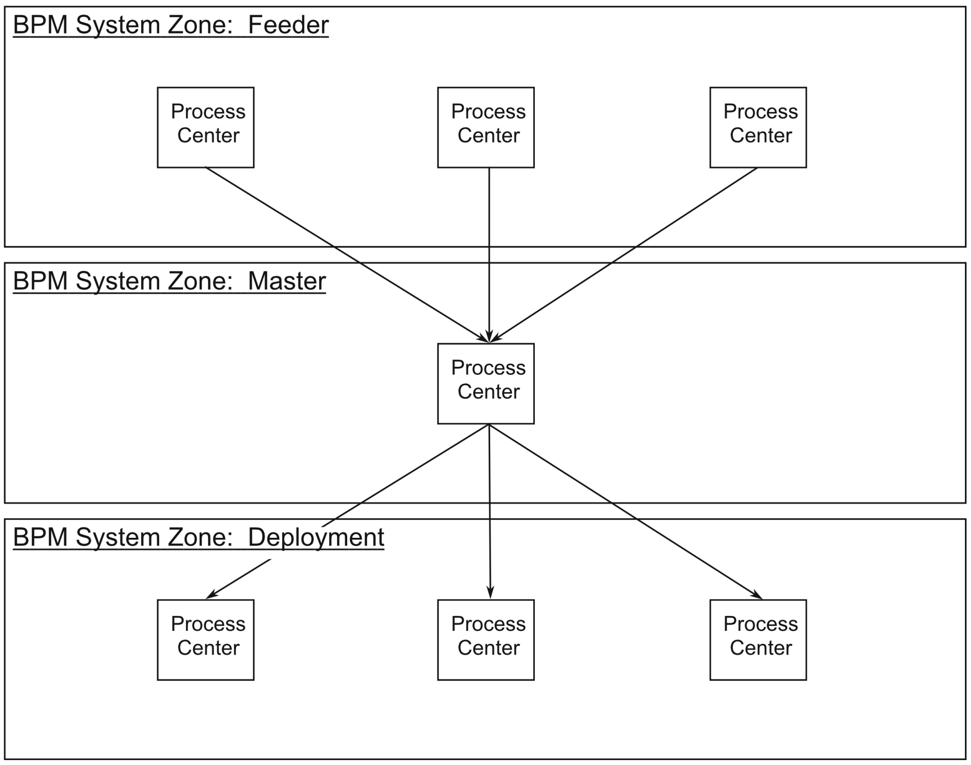


Figure 3.9: Usage of multiple Process Centers [AAG+13, p.34]

Figure 3.9 shows a topology of Process Centers that could be used in a company that uses automated Business Process Management.
The Process Centers in the first "BPM System Zone" are called "Feeders". They are connected to Playback-, Test- and Staging-environments. "Master" is connected to not a single Runtime-environment. It is the central repository and has no further destiny. The Process Centers in the "Deployment" zone are connected to Production-environments.

Runtime-environments do not necessarily consist out of one single Process Server. Each Runtime-environment can consist of multiple Process Servers. Finally, a single Process Server has maybe not enough system resources to compute all instances. To solve this, the same

Process Application is deployed on multiple Process Servers.
From the perspective of an external requester, these Process Servers have to perform as one. This is where "Clusters" come to business.

3.5.1 Clustering

IBM Process Servers are applications which are deployed on an "Application Server". IBM provides the WebSphere Application Server[6] that is used here. Such technologies provide the needed functionality to build "Clusters".

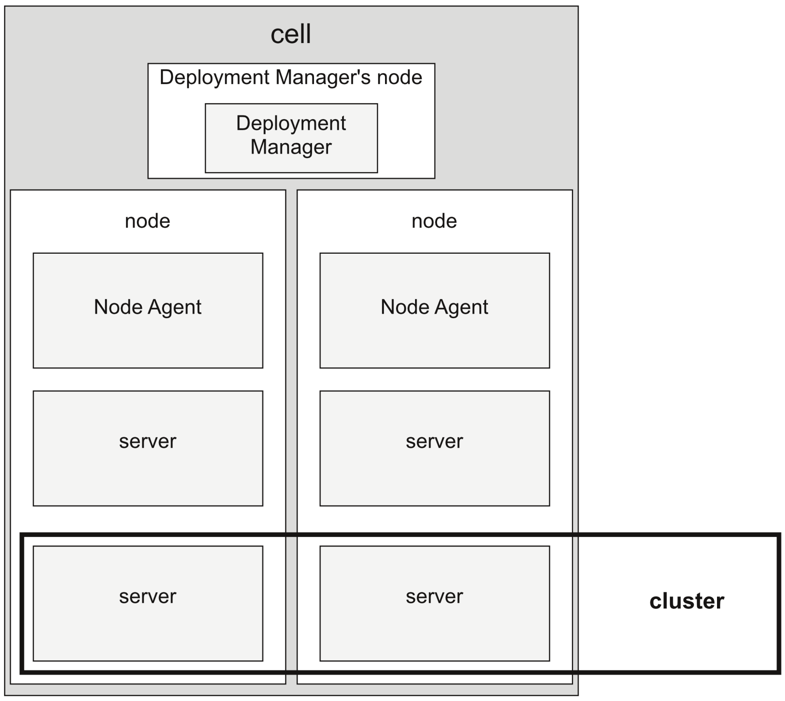


Figure 3.10: Cluster Topology with IBM Terminology [ABB+13]

Figure 3.10 explains a cluster as a bundle of servers. When a resource of is requested from the cluster, this request needs to be routed to a dedicated server.

The "Nodes" are representing hardware computers. Each node owns a "Node Agent". This is a software that manages the hosted servers and everything else that is happening on the node. Multiple nodes are summarized as a "Cell" or "Deployment Cell", that is managed by a "Deployment Manager". With the Deployment Manager servers and the hosted applications

[6]http://www-03.ibm.com/software/products/de/appserv-was

can be managed.
When two servers are configured as a cluster, then the underlying infrastructure keeps them in sync. Clustering enables the system to scale up/down the available resources. This is needed to serve an increasing amount of incoming requests or to execute an increasing amount of processes.

Additional, a technology is needed to route a request to a dedicated server within the cluster. This type of technology is called "Load Balancer".
The discussed setup enables also additional functionality. For example, if a node crashes, then the other nodes are still available and the system can perform.

A Load Balancer can be built by the use of the IBM HTTP Server(IHS)[7]. Of course, this works only if HTTP is the protocol that is used to communicate with the systems behind. A possible setup is plotted in figure 3.11.

The IBM HTTP Server bases on the Apache HTTP Server[8]. Equal to the Apache HTTP Server, the IBM HTTP Server is able to load a "Plugin". This plugin provides the logic of the dedicated instance. The static part of HTTP Server provides the implementation of the HTTP protocol and the possibility do extend the provided functionality by using so called "Modules" (for example: mod_ssl[9] to enable SSL/TLS[10] based encryption)

Together with the IBM HTTP Server, a plugin is licensed that can be used to route HTTP-requests to different ongoing servers or server groups. Additional, this plugin provides the possibility to define a property to identify correlating requests. This is needed, because correlating requests have to be routed to the same server.
The IBM HTTP Server should be available in a "Demilitarized Zone"[11] (DMZ).

But this is not the only possibility to build a Load Balancer. An alternative method is to use the IBM WebSphere Proxy Server as a "Reverse Proxy"[12]. The use of the IBM Websphere Proxy Server has pros and cons, equal to the use of the IBM HTTP Server.
It is related to specific design goals what is the best option. It is not part of this thesis to discuss this question. For interested readers, the article "*Proxy server versus the HTTP plug-in: Choosing the best WebSphere Application Server workload management option*"[13] provides additional information about this topic.

[7]http://www-03.ibm.com/software/products/en/http-servers
[8]https://httpd.apache.org/
[9]http://httpd.apache.org/docs/current/mod/mod_ssl.html
[10]http://en.wikipedia.org/wiki/Transport_Layer_Security
[11]http://www.linuxjournal.com/article/4415
[12]http://httpd.apache.org/docs/2.2/mod/mod_proxy.html#forwardreverse
[13]http://www.ibm.com/developerworks/websphere/techjournal/1010_pape/1010_pape.html

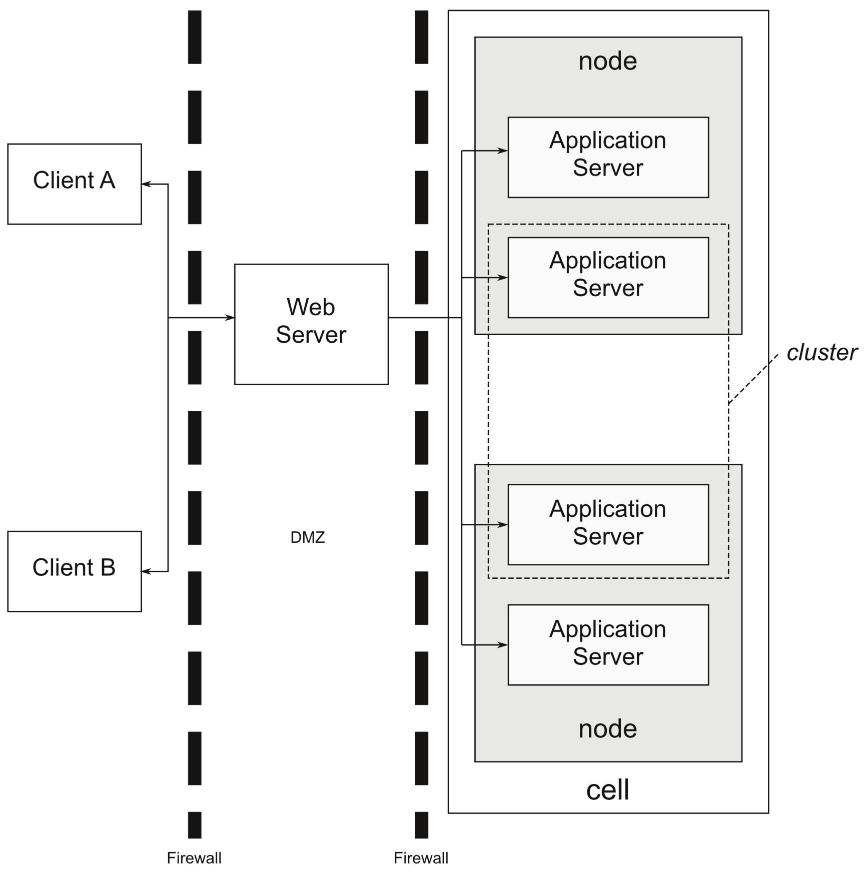


Figure 3.11: Webserver used for Load Balancing [ABB+13, p.69]

4 Security with Business Process Management Systems

This chapter addresses security as asked in the problem statement in 1.2. First, a short overview on the IBM Security Framework and the IBM Security Blueprint is given to explain what drives IT security.
Afterwards, plausible use cases of public available interfaces of Business Process Management Systems are defined. The collection of attack patterns that is contained in this chapter addresses potential vulnerabilities, which are related to these use cases.

4.1 IBM Security Framework and IBM Security Blueprint

IBM provides the "IBM Security Framework" to cover all security related topics from a business perspective. This framework bases on the "IBM Security Blueprint", that is a technical view. It is about the standards, technologies, and infrastructure designs that should be used.

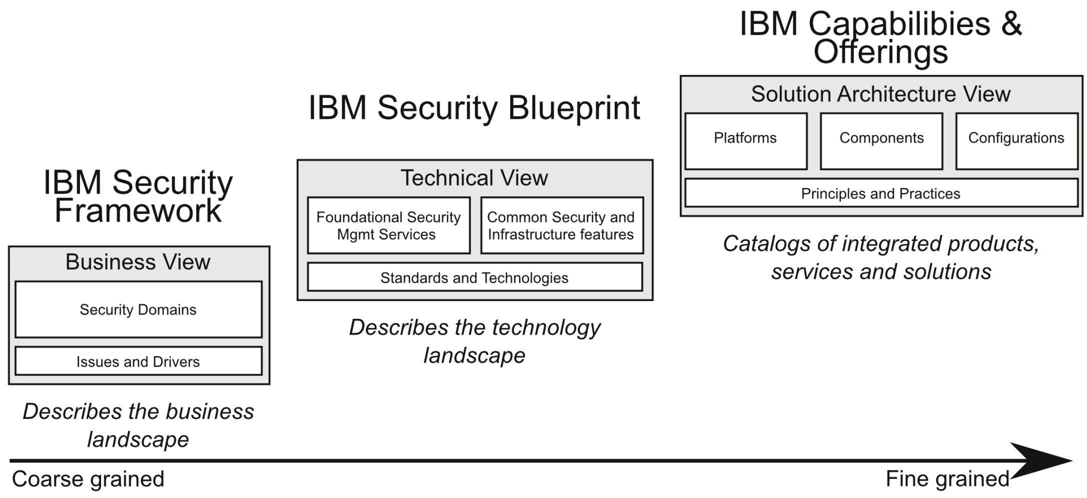

IBM Security Framework
Business View
Security Domains
Issues and Drivers
Describes the business landscape
IBM Security Blueprint
Technical View
Foundational Security Mgmt Services
Common Security and Infrastructure features
Standards and Technologies
Describes the technology landscape
IBM Capabilibies & Offerings
Solution Architecture View
Platforms
Components
Configurations
Principles and Practices
Catalogs of integrated products, services and solutions
Coarse grained
Fine grained

Figure 4.1: IBM Security Offerings [BAB+14, p.22]

By using the schema of convergence, from an abstract view to a fine-grained view as shown in figure 4.1, the possibility of overlooking a security threat is minimized.

4.1.1 What concerns Security?

According to the IBM publication "*Using the IBM Security Framework and IBM Security Blueprint to Realize Business-Driven Security*" two main categories of security drivers[BAB$^+$14, p.2] exist:

- **Business drivers** These drivers connect business with IT. Assumed that a department of a company has needs, which belong to IT. It is a security issue to fulfill these needs. This addresses risk management as well as legal compliance etc.
- **IT drivers** These drivers address the management of modern IT systems. For example, if the complexity of a system overshoots, this is a potential security issue. So, big systems have do be documented exactly. The capabilities to audit or to rapidly fix zero-day vulnerabilities are also important security drivers of this category.

4.1.2 IBM Security Framework - A Business Perspective

IBM provides the so called IBM Security Framework for ensuring that each security aspect is addressed. The framework is plotted in figure 4.2. The overall principles *Governance, Risk, and Compliance* are a central mnemonic to ensure that no important case is forgotten.
To get a more detailed explanation some specialized literature should be consulted. For a basic understanding the most important concepts of the framework are explained in the here:

- **Advanced Security and Threat Research** Addressing the increasing amount of evolving threats, it is necessary to stay on track. This is a huge research area, because many people are working all over the world on discovering these threats. The complexity of these threats is also increasing constantly (Involvement of social networks, geo-location, etc). To meet this challenge, it is necessary to build service capabilities in education, penetration testing, and computer forensics etc. In parallel, solutions which are able to eliminate the discovered threats have to be used.
- **People** It is important to be not sloppy with identity management. Across multiple domains and systems of a company, only one user management system should exist. The whole system should be designed to support Single Sign On[1] (SSO). This enables monitoring of the activities of the users.

[1] http://www-01.ibm.com/support/knowledgecenter/ssw_ibm_i_61/rzamz/rzamzconcepts.htm

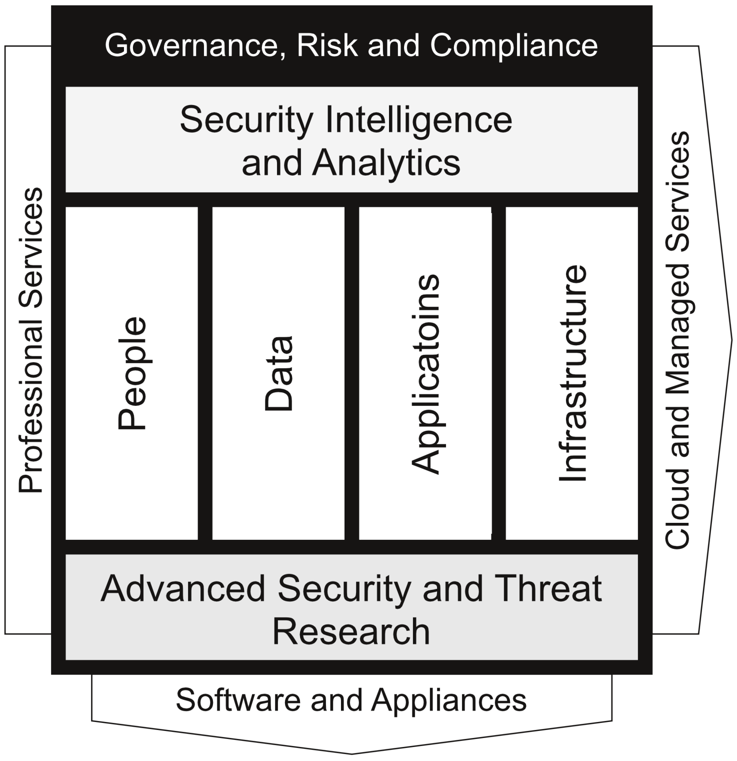

Governance, Risk and Compliance
Security Intelligence and Analytics
People
Data
Applicatoins
Infrastructure
Advanced Security and Threat Research
Professional Services
Cloud and Managed Services
Software and Appliances

Figure 4.2: The IBM Security Framework [BAB+14, p.9]

- **Data** Data that is stored on media, which can be stolen or crash, is not secure. This media should be backuped and encrypted.
  The structure of the data matters too. It is as bad as the loss of the data if nobody can find the appropriate information. Well structured data saves money and prohibits an administrative chaos.
- **Applications** Especially web applications are to mention here. They are complex and available for attackers. So, web applications are often exposed to attacks. They contain weaknesses and by exploiting these weaknesses some data can be leaked or an illegal user account can be created etc.
  To prevent such attacks, an enterprise-wide policy compliance enforcement is important. To be compliant with industry standards and to reach good test coverage during the development is also important to avoid weaknesses.
- **Infrastructure** This domain is getting more and more complex by the time. Due to this complexity, there is often a lack of skill in maintenance. Many administrators do not own the skills which are needed to secure such a modern infrastructure.
- **Security Intelligence and Analytics** It is very difficult to automate threat detection. This is often because of the loss of a centralized control system. Additional, it is hard to

detect new attack patterns in security related data. Due to the complexity and the hard quest, there are a lot of false positives with automated attack detection.

4.1.3 IBM Security Blueprint

While the IBM Security Framework addresses the business perspective on security, the "IBM Security Blueprint" refines the different domains. The IBM Security Blueprint is a product-neutral level to discuss technical approaches for securing systems. So, this is about standards and known best practices. The third, most fine grained level of detail is the product specific level. These dependencies are plotted in figure 4.1.

The IBM Security Blueprint is not discussed in detail here. So, a short description, based on figure 4.3, is given. For a deep dive some specialized literature has to be consulted. The white

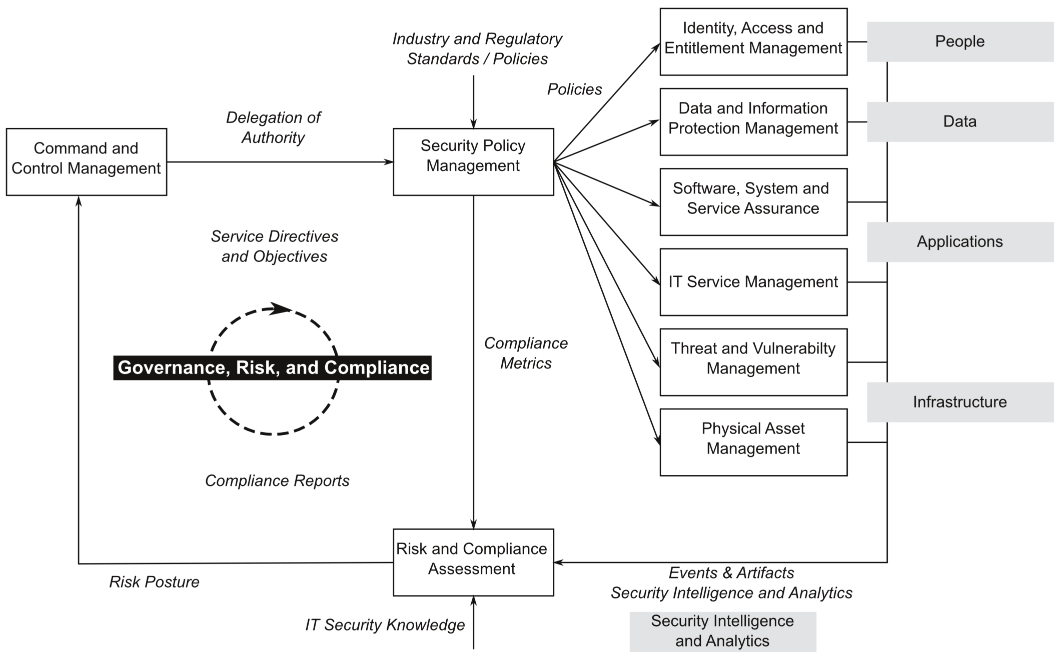

Industry and Regulatory Standards / Policies
Policies
Identity, Access and Entitlement Management
People
Data and Information Protection Management
Data
Delegation of Authority
Command and Control Management
Security Policy Management
Software, System and Service Assurance
Applications
Service Directives and Objectives
IT Service Management
Compliance Metrics
Governance, Risk, and Compliance
Threat and Vulnerabilty Management
Infrastructure
Physical Asset Management
Compliance Reports
Risk and Compliance Assessment
Risk Posture
Events & Artifacts
Security Intelligence and Analytics
IT Security Knowledge
Security Intelligence and Analytics

Figure 4.3: The IBM Security Blueprint Loop [BAB+14, p.33]

boxes in figure 4.3 are the foundational components of the IBM Security Blueprint. The grey boxes are the different domains of the IBM security Framework.
The cycle, which is modeled on the left, is the key to understand the concept. The constantly updated security policies are developed based on multiple factors: The result of security intelligence, security analytics, standard development etc.

Covering the mentioned capabilities within a company and to manage them appropriate is the key to security.

4.2 Scenario Description

Business Process Management software is not a determined application. Every Business Process Management System behaves different, because the modeled process definitions determine the final behavior. This means, it is kind of an enabler for a customer to build its own specialized solution.
The process modelers can introduce weaknesses by mistake. This is one reason why companies put specialists in charge for modeling processes and for the maintenance of the system.

It is not possible to describe all potential weaknesses of a large IT landscape within this thesis. There are too much of them and they are too complex.

The problem statement (1.2) asks for security considerations with public available interfaces of Business Process Management Systems. It is nothing new to access such systems from the outside of a company, but this is not what is implicated by publicity.
Public access means, for example, that there are no non-disclosure agreements and there is also the possibility of vandalism. This means that everyone from the outside is technically enabled to access parts of the provided functionality.

Without the framework, provided by service-level agreements (SLA), billing-, and security policies etc, every scenario can be neglected where a communication peer behaves as a server and not as a client. So, just the interaction of the task processor with external users remains as a plausible scenario.
Administrative access is also valid, but this would be contrary to every attempt to secure a system. Administrative access should always be restricted to the internal network of a company or at most be enabled via VPN.

Here, a task is a piece of work to be executed by a human being, as it is explained in 2.3.2. For each task a user interface comes to action. So, some information about the appearance of a user interface that belongs to a specific task has to be stated. This is done via the possibilities provided by the common web technologies.
A user interface can be loaded into a more holistic website that provides an authentication mechanism or other additional functionality. For example, IBM WebSphere Portal[2] is a product family that provides such functionality.

[2] http://www-03.ibm.com/software/products/en/websphere-portal-family

There are two different ways how a human being from the outside of the company can interact with a task. The first is fulfilling a role, which means that the user needs to be authenticated. The second way is that he is not authenticated. This might be the most public manner to access the system. There important characteristics, which differentiate these possibilities:

- **Authenticated** An authenticated external user can fulfill a role as defined in the WS-Human Task specification [CKM+10, p.19]. To be able to authenticate to the system, previously an account needs to be registered. With the existence of an account, the user can be a result of a staff query. So, an external user can get involved to a process as every other user.
- **Not authenticated** This comes with a bundle of problems. Assumed that a human being navigates to a website that contains a frame which is dedicated for displaying task user interfaces. Here, there is a decision problem: Shall this caller of the website receive a user interface or not? If yes, which one? Yet, it is not possible to even claim a task without a userid, because claiming a task means that a dedicated userid is set as executor of the task.

The use case "Unauthenticated Interaction" should not get rejected completely, because a task could be created for every call of the mentioned fictive website. A pre-requirement for such a task might be the presence of data. This data could be generated by the time the website is called, but this would be exactly the be behavior of a dynamic website as every other. So, there is no need of a Business Process Management System behind.
In combination with the use of Business Process Management functionality only two scenarios are valid for unauthenticated access:

- **Creating a Process** It is possible to create a task, that is not correlated to a process. This task gets claimed and someone executes it. After it is finished the inserted data has to be analyzed. Based on this data a decision is met whether a process is created or not.
- **Creating follow-on Tasks** This is exactly the same as *Creating a Process*, but the action performed after completion. With this scenario no processes are instantiated, but ongoing tasks.

This setup does not have many use-cases, because of the contradiction of getting involved with Business Process Management and anonymity. This could be solved by mapping a technical userid to the session identifiers, but this results in authenticated interaction.

Following these considerations, the discussed scenario is as shown in figure 4.4.
To reduce the complexity only a few components of the fictive Business Process Management System are plotted in figure 4.4. First, the three zones which are determined by the Transport Level Firewalls, have to be explained:

- **PUBLIC** This is the public internet that everyone uses. The users which are located here are further-on called "external users".

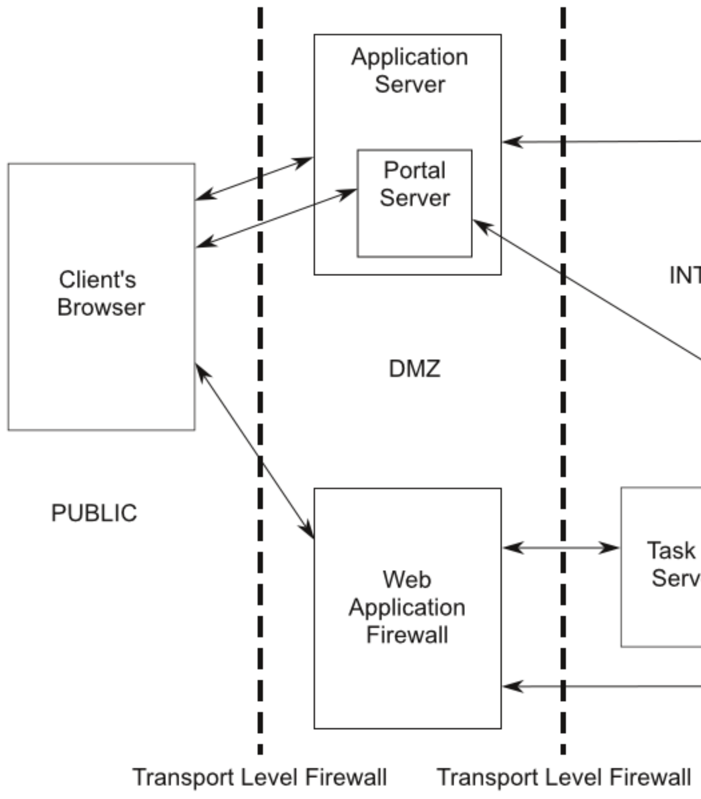


Figure 4.4: The Topology of the discussed Scenario

- **DMZ** This firewall zone is part of the network of a company, but accessible from the public and the intern firewall zone. This is called Demilitarized Zone[3] and is restricted by Transport Level Firewalls.
- **INTERN** This is the internal network of a company. This zone is not accessible from the public area. The users which are further-on called "internal users" are located here.

Figure 4.4 shows the components which are involved when a task is executed by an external user. The external user authenticates himself to the system via a portal system. This can be used also to list and claim tasks etc. To execute tasks, the external user has to interact with the task user interfaces. These are rendered by the Task UI Server according to the interface definitions in the task definitions.
An internal user is allowed to access the all resources of the system, but an external user has to cross the DMZ. This is enabled by the use of a web application firewall as a node in the DMZ that forwards the network traffic to the internal zone. The concept "Web Application Firewall"

[3]http://www.linuxjournal.com/article/4415

is described in 4.2.1.
It is important, that the web application firewall filters network traffic that belongs to an already authenticated and authorized communication. These mechanisms are equal for internal and external users, so the web application firewall is an additional barrier for external users.

This scenario is discussed in the follow to determine related security issues.

4.2.1 Web Application Firewalls

This subsection is a short introduction on "Web Application Firewalls" (WAF). The scenario, that is described in 4.2, bases on this technical concept. So, this concept needs to be explained.

There is no standardized definition of this term, but the Open Web Application Security Project[4] (OWASP) describes the term "Web Application Firewall" this way:

A web application firewall (WAF) is an appliance, server plugin, or filter that applies a set of rules to an HTTP conversation.[5]

This description is adopted as a definition for the context of this thesis.

4.3 Attack Patterns related to the Scenario

This thesis addresses security considerations, which come up with the scenario described in 4.2. Other security issues are important too, but not part of this scope. The scope is on the usage of Business Process Management software and the fact that users from outside of the company can interact with the system by executing tasks.

The problem with security is to get a complete collection of possible weaknesses and all thinkable attack techniques. Because it is not possible to list techniques, which will come up in the future, the idea is to translate already known attack patterns onto this scenario.
Business Process Management products are using web technologies for interaction. So, attack patterns, which are known from this domain, are the ones to translate.

Such patterns are enumerated by multiple communities, groups, or institutions. Here, the the "Common Attack Pattern Enumeration and Classification (CAPEC)"[6] is used. The CAPEC is a

[4]https://www.owasp.org/
[5]https://www.owasp.org/index.php/Web_Application_Firewall
[6]https://capec.mitre.org

community for classifying and documenting attack patterns in the IT world. It is maintained by the "Homeland Security Systems Engineering and Development Institute" of the MITRE Corporation[7]. The Homeland Security Systems Engineering and Development Institute is co-sponsored by the "Office of Cybersecurity and Communications"[8] at the US Department of Homeland Security.

The Open Web Application Security Project (OWASP)[9] is also an often referenced attack pattern enumeration. Especially, because of the "OWASP TOP 10". The OWASP TOP 10 is a yearly released summary of the most critical IT security risks. But the OWASP project focuses to much on the dedicated technical approach to be a good source for common attack patterns. So, the CAPEC is referenced here.

The CAPEC uses an indexing system to identify different assets: For example, CAPEC-66 references the common known attack pattern "SQL Injection". This index is used in the following to reference specific assets in the CAPEC collection.
The CAPEC community releases frequently updated versions of their pattern collection. This thesis references the version 2.6, that was released at 23.July 2014 and contains 463 entries. Some attack patterns will not be mentioned here. These patterns are well known and documented in literature as well as in specialized enumerations like OWASP or the CAPEC. It is important to understand, that these attacks can be enabled when they are combined with a weakness that is related to the scenario that is discussed here. Attack patterns are seldom usable alone, so they have to be combined. But a pattern that is not related to this scenario is not mentioned here.

The CAPEC is grouped in two different ways: By the abstract mechanism (CAPEC-1000) and by the domain (CAPEC-3000) of the patterns. The second grouping is too coarse-grained for this purpose. So, the first one is used here.

The following structure of the document maps the top-level categories of the CAPEC grouping that bases on the abstract attack mechanism (CAPEC-1000). The top-level categories of this grouping which are not discussed here are listed in 4.3.9 with an explanation why they are not discussed here.
Every discussed category contains one or multiple attack patterns. These sub-patterns are derived during the creation of this thesis by interpreting the related CAPEC sub-patterns of the corresponding CAPEC-category.

[7] http://www.mitre.org
[8] http://www.dhs.gov/office-cybersecurity-and-communications/
[9] https://www.owasp.org

4.3.1 Gather Information (CAPEC-118)

Gathering information addresses ways of collecting facts and assumptions about the target system. This is not a traditional attack itself, but for executing more dangerous attacks many information is needed. The more information is reachable about the target system, the more likely it is to detect weaknesses.

In the traditional domain of web technologies, it would be interesting which version of a wordpress[10] is used, for example. Every kind of information about the target system can be useful for an attacker, especially when multiple facts can be combined.

In the scenario of this thesis gathering information is a point.

Process Discovery

Description

Normally, the internal processes of a company are confidential. But when an attacker interacts with the Business Process Management System, he could come to conclusions about the process definitions.
This can become more revealing for the attacker when he is able to combine his knowledge with additional information. Such additional information can be gathered from any kind of source. These sources exist in multiple ways, especially because multiple external users interact with the system. The result is, that the information that is located outside of the company increases constantly.

Example

An attacker interacts with a Business Process Management system of a bank. He has to execute some tasks which belong to a process that handles a loan application of the attacker. It can be assumed that a threshold exists which is used for involving a clerk. The goal of the attacker is to determine an approximation of this threshold.
The threshold can be isolated when the attacker is able to derive whether a clerk was involved or not. Receiving mail or a call is also ongoing interaction that can be used for deriving information. The target threshold can be approximated by placing multiple loan applications with different values.

[10] http://wordpress.org/

Such excessive interaction gets probably detected by the system or the employees. To reduce the needed interaction, additional information sources have to be consulted. This can be financial documents of other customers of the bank or the information that are gathered with additional accounts.

Prevention

To minimize the amount of information that is derivable about the process definitions, two recommendations should be followed:
First of all, a process should not involve multiple external users. This could enable an attacker, who works with two faked accounts, to examine what happens between two tasks. So, only one external user should be allowed per process.
Furthermore, the external users should execute as few tasks as possible. It is strongly recommended to restrict the interaction with an external user to just one task.

In general, the process modelers should be educated about this attack pattern to take it into account.

Exploit Information Leakage

Description

This is more an accident than an attack, but it has to be mentioned, because it is possible that an attacker receives confidential information. In a traditional configuration of a Business Process Management System every user has signed a non-disclosure agreement.
With the usage of a Business Process Management System that involves external users, this is changed. The external users are potential attackers, competitors, or auditors. No external user has signed a non-disclosure agreement, so none of them is allowed to receive confidential data.

This pattern addresses the possibility that this restriction fails and external users receive confidential information. This can happen through human or technical failure.

Example

A good example for such leakage by accident is a scenario where the process definition was just updated. If an internal user was not informed about this change, he may enters confidential information to a user interface that gets forwarded to an external user.

It is nearly the same with technical failure. If a service returns data, which is familiar with external users, a process modeler used it maybe in a process definition. Assumed this service gets suddenly updated to return confidential data, this data could be presented to external users.

Prevention

To prevent the leakage of information, the internal users should be informed about the process definitions. They have to know whether the data of a task will be accessed by an external user or not. This could be done by education or by tagging the appropriate task user interfaces in an unambiguous way.
To prevent leakage, by accidentally changing the behavior of a involved software component, an accurate documentation is needed. Based on this documentation the developers, administrators, and modelers are able to check whether a piece of software is used with external users.

Account Mining

Description

The target of this attack pattern is not the Business Process Management System itself, but the portal which is needed for interaction. If this portal is built with vulnerabilities, then an attacker probably is enabled to retrieve information about accounts of other users.
This is critical in multiple ways. Information about the accounts of other users is always confidential, at least because of privacy protection. But some information about the accounts of the internal users could be retrieved too and this can be used for ongoing attacks.

Example

Portals provide log-in functionality. This is normally done by entering a username and password combination. If the inserted data is not valid, a message is shown to the user. If this message looks like this: "The password is incorrect", then this is a weakness. When an attacker gets a message like this, he knows a valid username. This is a common technique to mine valid usernames, because this can be automated with minimum effort. An attacker who owns valid usernames of foreign accounts is a security threat.
In particular, when these usernames are valid for the entire intranet of a company.

Prevention

First of all, the used statements should be checked whether they suggest more information than intended. This additional information is the target of this pattern.

To ensure that no such data can be mined, it is recommended to isolate the public available Business Process Management System from the other systems of the company. For the automation of the internal processes another installation should be used.

4.3.2 Manipulate System Users (CAPEC-527)

This category addresses attack patterns, which use psychological techniques to influence the behavior of someone else. Such techniques inherit no need for a face-to-face communication. They are well suitable for the use with digital communication technologies.

Exploiting a Communication Channel

Description

When an external user interacts with the Business Process Management System, it is likely that he not just receives information, but also inserts some information. If the process is defined in a way that an internal user has to process this input, then a new communication channel is created. This communication channel can be rudimentary, but it is also possible to model a process in way, that a communication channel is created that enables an intense exchange of messages.

Such a communication channel can be used for techniques that are described in CAPEC-416 ("Target Influence via Social Engineering").

Example

If an external user is a customer of a insurance company and interacts with the Business Process Management System to state an accident report, this is a potential weakness. Such reports need a lot textual input and they are normally processed by humans. The attacker can use psychological tricks to arouse feelings of guilt or compassion in the clerk that processes the report. Such feelings are able to influence decisions.

Prevention

To prevent the usage as a communication platform for manipulation, multiple actions can be taken. First of all, employees should be educated on how to recognize and avert manipulation attempts. A periodical instruction on these attack patterns is very important, independent of the existence of the communication channel that is discussed in this pattern.
Furthermore, the process modelers are recommended to minimize the possible communication channels between the external and the internal users. Every message that can be exchanged is an enabler for this attack pattern.
At least, the communication that takes place should be monitored for quality assurance, equal to communication via email or telephone. When the internal users know about the monitoring, they work more impartial and by this, they are less vulnerable for manipulation.

4.3.3 Deplete Resources (CAPEC-119)

Depleting Resources is a category of attack patterns that is very dangerous. The most prominent member of this category is the "Distributed Denial of Service" (DDoS) [LRST00] attack pattern. This is known by many people, because it causes well visible effects and it is also used by a lot of internet activists to state political protest. The general approach is simple: Bind as many resources as you can to cause an overload or provoke another technical problem which results in denial. Technically, this could be done in multiple ways and to defend a system against such an attack is complex.

Instance Flooding

Description

This attack pattern can be used by an attacker when external users are able to create process or task instances. Every Business Process Management System is limited on the amount of instances. Even if this limit is very high, the malicious creation of tons of instances will at least result in slowing down the system or a complete overload. This reduces the quality of service for other users in a recognizable manner.
This is especially dangerous when the creation of instances can be automated by the attacker.

Such an attack is a worst-case scenario for a company if an attacker is able to cause the cessation of work in the company.

Example

If the Business Process Management System is configured in the way that no account is needed as described in 4.2, it is easy to write a script that loops and creates instances.

When an account is needed, then it becomes more complicated, because it is likely to need multiple accounts to be able to create that much instances.
If an insurance company shall be attacked with this technique, this is a problem for the attacker, because every account needs a real insurance contract. But online shops, which allow to register accounts with only a nickname and a password, can be attacked very well, because the account creation can be automated.

Prevention

To prevent attacks according to this pattern, creating instances should never be possible for unauthenticated users, as it is described in 4.2.
Authenticated external users should also not be allowed to create instances. If it can not be avoided to enable external users to create instances, the Business Process Management System should be protected in addition.
An attacker needs to automate the creation of instances to reach a denial of the system. To prevent the automation, CAPTCHAs[11] have to be used. A CAPTCHA is a simple test to prove whether an action is triggered by a human or not. Furthermore, it is possible to limit the amount of instances that can be created by an external user.

Increase the Workload of other Users

Description

This is similar to "Instance Flooding", but with one important difference. The attacker aims not for on an overload of the system itself, but for creating a huge workload that has to be handled by other users (especially employees to the target company). Using this pattern causes an effect on the internal performance of the company.

Example

A trading platform like ebay[12] has to manage notifications about fraud. This could be done by using the scenario that is discussed here. Such notifications are likely to be processed by humans. If an attacker finds a possibility to automate the creation of such notifications, this will cause an immense workload for the internal users. It is probable that they are able to recognize the faked notifications, but this will still decrease the performance of the appropriate department.

[11] http://www.captcha.net/
[12] http://www.ebay.com/

Prevention

This attack pattern is similar to "Instance Flooding". So, all prevention techniques of "Instance Flooding" can come to action with this pattern too. The difference is the involvement of humans on server-side. These users of the system should be educated about this attack pattern to be able to recognize an attack.

Amplification

Description

This is a pattern to optimize the effort/benefit ratio of the attack patterns of this category. An attacker is probably able to create a bunch of instances in an automated manner, but it can be hard to reach the needed amount to cause an effect. If the attacker knows how the ongoing interaction with these instances has to look like to cause the creation of additional instances, an amplification effect can occur.

Example

If the denying processes are loan applications, then it is thinkable that the attacker has to execute a task after the application is stated. During the execution of this task the attacker is asked for additional information which is needed to process the application. If the process instance that handles the loan application is modeled to validate the input and to decide afterwards whether the real loan application process is started or if the application is rejected instantly, the this could be used for amplification. When a pure flooding attack is executed, the attacker does not execute any tasks that belong to the flooding instances. But with this special constellation, a denial of the system could be achieved much faster. The attacker has to extend the attack to trigger the real loan application handling process.

Prevention

This attack pattern is an extension of "Instance Flooding" and "Increase the Workload of other Users". So, the prevention techniques of these base-patterns prohibits also the amplified usage of them.
To prevent amplification, the creation of heavy processes should be approved by internal users.

4.3.4 Deceptive Interactions (CAPEC-156)

This category of attack patterns is characterized by spoofing. This is a huge category in the traditional web environment and the techniques for spoofing do not differ from this scenario. Attack patterns like "Content Spoofing (CAPEC-148)" can be used equally here.
So, the fact itself that the Business Process Management System is available from public networks is the only weakness that is discussed here.

Exploit Availability
Description The traditional usage of Business Process Management Systems is scoped to the internal network of a company. This is sometimes extended by opening the system for affiliates. But opening the system for public usage is a marginal change that creates a public identity. With the existence of this identity every attack pattern that bases on spoofing related content (CAPEC-148) or the identity itself (CAPEC-151) becomes probably possible.
Example An example for attacks that follow this pattern is a Phishing attempt (CAPEC-98). To interact with the Business Process Management System, a portal solution is needed. A Phishing attempt can be to send fraud emails to the users of this portal. This email contains a link to a faked website that looks similar to the original. If the user enters his log-in credentials to this fraud website, the attacker can use them further-on.
Prevention The key to prevent attacks according to this pattern is education. Both, internal and external users need to be taught on how to recognize such attacks. Additional, the company that runs the public available Business Process Management System should provide a tracking system to enable the reporting of attacks. Based on the reported incidents, warnings can be stated and public authorities can investigate.

4.3.5 Exploitation of Authentication (CAPEC- 225)

This category of attack patterns is characterized by interacting with the target system and to fool the authentication mechanism. Techniques to bypass this mechanisms or to register an illegal account are not in the scope of this thesis, because they are equal to the techniques which are used in common web environments.

It is possible to use the pubic available Business Process Management System as an enabler for one type of attack that belongs to this category. The target of such an attack can be the system itself or another system in the target company. This inherits a lot of potential risk, which is the reason for a more detailed discussion here.

CSRF via the Business Process Management System

Description

A Cross-Site Request Forgery (CAPEC-62) is a technique that is described in many publications about security. The idea is to exploit the trust that a server has in an authenticated user. When a harmless user interacts with the target system, the attacker tricks this user to perform malicious actions. If this user owns some extra privileges, he is a valuable victim to an attacker.

To execute an attack like this, a possibility to trick the victim is needed. Nowadays, most of the computer workers are educated about these dangers. Every security briefing includes the instruction not to click on suspicious links and to delete curious mails.
A Business Process Management System, that is used as discussed here, can be used to trick other users of the system and to perform a CSRF attack. This is likely to work, because the users are probably not educated with this threat.

Example

This example is about how to terminate another process that is active in the Business Process Management System. For this example, it is assumed that the identifier of the process is known to the attacker.
Additional it is assumed that the REST-API of the Business Process Management System provides the functionality to terminate a process by executing a HTTP-GET request onto this URL:

http://<host>:<port>/rest/process/<PROCESS_ID>?action=terminate

Now, the attacker tricks an internal user of the system to perform this request. He could probably exploit a fault in the implementation that generates the task user interfaces. If the text, that is presented, is not escaped properly, then this is a weakness. This text will be interpreted by the browser as if it is valid HTML.
So the attacker inserts this string to a text area of a task user interface:

<img src="http://<host>:<port>/rest/process/ <PROCESS_ID>
?action=terminate" >

(Of course, the host, port, and PROCESS_ID placeholders have to be filled with the appropriate data.)

If an internal user executes a task that works with this piece of data, the HTTP-GET request is executed when the task user interface is loaded in the user's browser.
That shows how this type of attack generally works. But such a fault in the implementation is rare. Today, more complex approaches are used:
An attacker could postulate a link to a website that is maintained by the attacker and that contains the request in a hidden inline-frame. If the attacker is able to convince an internal user about the trustworthiness of the link and the internal user follows the link, then the attack succeeds.

There is technically no difference if this attack is performed via email or a chatroom, but the non-technical difference is immense: Today, computer workers are sensitized to reject such attacks, because they know a suspicious link should not be clicked. But when the link is received via a Business Process Management System, it is probably not even indicated whether this link sources from an external user or an internal user.

Prevention

The prevention of CSRF attacks is a technical deep dive to the technology stack of a Business Process Management System. The CSRF is part of this pattern collection, because it is enabled via the discussed scenario. Here, an overview on common prevention techniques is provided, but for a better understanding additional literature has to be consulted.

There are three approaches which represent the most common techniques do defend against Cross-Site Request Forgery attacks: Validating the HTTP-Referrer header, validating a secret token, and validating custom headers attached to XMLHttpRequests [BJM08].

Using the HTTP-Referrer header is the most simple variant. If this option is enabled, respectively this functionality is implemented in the dedicated browser, the URL of the site that executes the request is added to the HTTP header. With this, the web server is able to evaluate whether the request comes from the site itself or not[BJM08].
This approach does not provide the possibility to secure a web application completely. For example, it is possible to redirect a user to access a resource via the FTP protocol[13]. Then, a Cross-Sire Request Forgery can be executed. No HTTP-Referrer header will be set at all[BJM08] in this scenario.

[13] http://www.w3.org/Protocols/rfc959/

The second possibility is to work with a secret token. There are different techniques on how to work with a secret token for defending Cross-Site Request Forgery Attacks[BJM08].

Session Identifier: The session identifier of a user's session is also used as secret validation token. This technique is not very comfortable for integrating the web application with other systems. This provides less security than the other variants of using a secret token.

Session-Independent Nonce: This approach uses a random security token that depends not on the established session. The token is set as a cookie when the website gets accessed first. Afterwards, it is constantly validated by the server. This prevents from a "passive" Cross-Site Request Forgery, but the attacker is able to set a valid token to a malicious request. This is possible by requesting the website and extracting the valid token. The server holds no mapping of the tokens to the session identifiers. So, the attack succeeds.

Session-Dependent Nonce: This is a refinement to the previous mentioned Session-Independent Nonce, which introduces the server-side mapping of the tokens to the session identifiers. This is the most clean and secure technique pro prevent CSRF, but it has one big disadvantage: By the usage of a server-side mapping of the tokens to the session identifiers, the server-side computing load increases heavily.

HMAC of Session Identifier: The security is increased when the token is bound to a session identifier as described. To eliminate the server-side mapping, that produces a lot of computing load, asymmetric cryptography could be used. "Keyed-Hash Message Authentication Codes" (HMAC)[14] are predestined for this. As long as each server holds the HMAC key of the website, the token can be verified for every request.

Finally, a technique which is based on the idea to use custom HTTP-headers is discussed here. By the use of XMLHTTPRequest[15] custom header properties can be set. Afterwards, these custom header properties can be validated by the server.

Browsers allow custom headers just for requests against their originating site. So, a first security gain is reached by just using these technology.

To exploit this technology completely, frameworks like the Google Web Toolkit[16] (GWT) have to be used.

[14] http://tools.ietf.org/html/rfc2104
[15] http://www.w3.org/TR/XMLHttpRequest/
[16] http://www.gwtproject.org/

4.3.6 Exploitation of Authorization (CAPEC-232)

Characteristic properties of this category are the correct usage of the authentication mechanisms, the unwarranted trust in common authorization algorithms, and that different resources are not separated fine-grained enough due to authorization.

The scope of this thesis is also on separating the external users of the Business Process Management System from the internal users. This is done by redirecting the requests from the external users to a web application firewall as described in 4.2. This firewall needs to be configured.
As an example, the HTTP-traffic is filtered based on the URL-path here. The listing 4.1 shows an example configuration that is interpreted as a white-list.

```
<UriGroup Name="default_host_URIs">
  <Uri Name="/login.jsp"/>
  <Uri Name="/rest/service/*"/>
  <Uri Name="/portal/*"/>
</UriGroup>
```

Listing 4.1: Sample Section of a Web Application Firewall Configuration

Listing 4.1 shows a possible configuration of a web application firewall. The defined rules filter the HTTP-traffic based on the URL-paths. There are two different types of rules: Static and dynamic rules. The first rule of listing 4.1 is a static one. The URL-path is determined exactly. The second and the third rules are dynamic. The * is a placeholder for any kind of string, even additional path-segments are allowed.

The pattern "Exploit the fixated Firewall Configuration" references this example filtering concept. This pattern exploits the dynamic configuration of the firewall that inherits the danger of accidental approval of restricted resources. This is an example that bases on the URL-paths as filtered property.

Exploit the fixated Firewall Configuration
Description This attack pattern exploits the fact that the configuration of the firewall is fixated. At the point in time when the system gets installed, the web application firewall gets configured. But some of the process definitions, Ajax-services, or task user interfaces are modeled after this point in time. So, it is possible that the system administrators configure the web application firewall in a way to deal with this fact.

So, dynamic rules have to be used, as exemplary shown in listing 4.1. It is probable that the configuration becomes too coarse-grained with the use of dynamic rules. As a result, the external users are probably able to request resources which are not meant to be accessible for them.
Example To adopt the example that was introduced with listing 4.1, the second rule of this listing is discussed here. This rule enables the access to a REST-API. Each single service is addressed by a service-ID that is the third segment of the URL-path. An example for a service URL could look like this: *http://<host>:<port>/rest/service/1.347837* If the administrators configured the web application firewall as it is shown in listing 4.1, every service that is introduced later-on is accessible too. An attacker is now able to use a tool like soapUI[17] to call these services with legal user credentials.
Prevention Web application firewalls should not be configured by the use of such dynamic rules. This freezes the available functionality and by this some additional security goals are achieved. Without dynamic rules, every resource that shall be available has to be approved by someone who owns write-permission for the web application firewall configuration. Additional to this, precise documentation is an inevitable prerequisite to prevent attacks according to this pattern. A careless managed configuration can result in unnecessarily available resources.

Man in the Middle
Description

[17] http://www.soapui.org/

This attack pattern is a direct adoption of the traditional "Man in the Middle" attack that is described in CAPEC-94. If a Business Process Management System used in the way is is discussed here, a Man in the Middle attack is more likely to work than with a traditional setup. This is, because if a Business Process Management System is only accessible from the intranet of a company, an attacker has to place a malicious software within this intranet. But with the possibility to access it via a public network, the attacker can place the malicious software outside of the intranet.
So, this attack pattern is not enabled by the use of a Business Process Management System as such. It is enabled by the way this system can be accessed.

Example

A good example for a Man in the Middle attack is a scenario, where the attacker was able to manipulate the configuration of the victim's private router. By adding additional rules to the routing table[18], the victim gets redirected to a server that is maintained by the attacker when the Business Process Management System is accessed. By using a tool like sslsniff[19] the attacker is able to decrypt the transmitted data. After the data is manipulated, it is encrypted again and forwarded to the correct server. The answers can be manipulated in the same way.
The SSL/TLS standards[20] provide techniques to detect such attacks, but not every user takes care of the warnings.

Prevention

The SSL/TLS[21] standard provides the needed functionality to authenticate a server to a client and thereby validate whether someone executes a Man in the middle attack.
The problem with this prevention technique are the users. If the users continue the interaction with the system, even when warnings are stated, an attacker can read or manipulate the communication in future. So, the education of the users with these warnings is important.
If nothing helps, SSL/TLS-based client authentication can be used. This increases the maintenance effort, but in enables the server to detect Man-in-the-middle attacks.

[18] http://linux-ip.net/html/routing-tables.html
[19] http://www.thoughtcrime.org/software/sslsniff/
[20] http://tools.ietf.org/html/rfc5246
[21] http://tools.ietf.org/html/rfc5246

4.3.7 Abuse of Functionality(CAPEC-210)

This category of attack patterns is characterized by the exploitation of the provided functionality to perform malicious actions. These patterns are all about using functionality in a different context as intended.

Exploit the Misuse of Concepts

Description

This is an abstract pattern that addresses the vulnerabilities which occur when concepts are misused. It is important that the use of a Business Process Management System in the way it is discussed here is not common. If users use functionality that is valid without the possibility to access the system from the outside, this does not ensure that this is also valid with the use in the scenario that is discussed here.

Example

The BPEL-extension "BPEL4PEOPLE"[AAD$^+$07] provides the possibility to use a concept that is called "Four Eyes Principle". A task modeler can use this concept to instruct the task processor to submit data to a second user for approval. This happens before the service, that is called by the engine, returns. For example, a CSRF attack could be executed by using this communication channel between the two different users. This is even more dangerous if the concept is used across borders which separate internal and external users.

Prevention

To prevent such security incidents, the process modelers have to be taught on the circumstances of the special scenario. Some concepts are not designed to be used with the special configuration that is discussed here.

Password Reset

Description

The pattern Account Mining in 4.3.1 describes how to mine for information about the accounts of other users. This pattern is about how such an account could be hijacked. The idea is to use the possibility to reset a forgotten password. If the attacker has access to the third-party system (email account, mobile phone, etc.) that is used for authentication, he is able to reset the password.

This is dangerous for a company that runs a public available Business Process Management System, because there is no guarantee for proper security maintenance of the third-party systems.

Example

Nowadays, an email account is the base of a digital identity. Assumed that the attacker is able to access the email account of another user of the public available Business Process Management System, then he is able to reset the password of this user. Of course, this works only if the needed authentication is performed via email accounts.

Prevention

To prevent account hijacking, as a consequence of insecure third-party systems, an additional barrier has to be installed. A common technique is to ask a security question. The user deposits the answer to that question while he registers the account. When an attacker attempts to reset the user's password as described in this pattern, he has to answer this question exactly as the proper user would do it.

Actually, a relatively new technology becomes more known, which addresses the prevention of attacks like this. This technology is called "Multifactor Authentication"[22]. To prevent an attack according to this pattern by the use of this technology, the system sends i.g. an SMS to the owner of the account. This SMS contains a random code that needs to be entered while resetting the password. It is unlikely that an attacker as also access to the target's mobile phone to steal the code.

Utilization as Tool

Description

One of the most important facts about Business Process Management Systems is that the provided functionality depends only on what is defined by the process modelers. If the provided functionality could be used as a tool for executing attacks, this is a security problem.

Example

[22] http://www.ibm.com/developerworks/aix/library/au-security_auth/index.html

If the accessible functionality can be used as an email sprayer, this is a security problem. Assumed an attacker is able to define a set of destination email addresses and to write some text, then the system can be used for sending spam mails. The domain of the company that runs the system could get black-listed after an attacker used it for sending spam mails.

Prevention

To prevent attacks according to this pattern, the use of the system without being authenticated should not be enabled.
Additional, the process modelers should be educated about the dangers of this pattern to be able to incorporate appropriate security considerations.

Exploit missing server-side Validation

Description

This is a complex pattern that can cause a lot of damage. The task user interfaces validate the inserted data on the fly. If every input field is validated to be OK, the task can be finished and the inserted data is transmitted to the server.
This attack exploits the fact that this final transmission can be faked by using appropriate tools. This means, an attacker could compose a set of data that contains values which would fail the validation and send it to the server. If the process modelers defined no server-side re-validation of the received data, this is a possibility to trick the system to accept invalid data.

Example

A good example is to assume that a task is finished by executing a HTTP-POST request on a REST-service. The request can be faked by using a tool like soapUI [23]. To be able to fake the request, it is important to know how the transmitted data needs to be structured. This could be determined by analyzing the network traffic with a tool like firebug [24].
If no server-side validation of the data is done, the attack succeeds.

Prevention

[23] http://www.soapui.org/
[24] http://getfirebug.com/

This attack pattern exploits a modeling fault that results of a wrong understanding of the system's behavior. The process modelers have to be briefed to re-validate every new information that was created by a human being. It does not matter whether an internal or an external user authored this new information.

4.3.8 Probabilistic Techniques (CAPEC-223)

This category inherits attack patterns which are characterized by reaching a goal through the try-and-error principle. This is a category of attack patterns that is potentially dangerous for any kind of system if the attacker is able to perform the attack in an automated manner.

Brute Force User Credentials
Description This attack pattern exploits the availability of the Business Process Management System. To access the system, a user as to log-in. So, an authentication mechanism is available for attackers. In theory, this authentication mechanism could be used instantly for a Brute Force attack as described in CAPEC-112. In practice, this can last for many years if the system is well secured. But this attack will succeed always. This can be dangerous for the customers in mainly two ways. The first possible security problem is a weak password policy that allows users to use weak passwords. The second security problem depends on a risky configuration. If the internal users are able use the system with their standard intranet account, an attacker is able to brute force user credentials of intranet accounts.
Example An attacker could use a tool like brutus [25] to automate the attack. If the attacker was able to identify a valid username, the needed amount of time is reduced heavily. Tools like brutus are able to perform attacks via multiple protocols like HTTP, SMTP, FTP, SMB etc. If the password policy of the system is known, brutus can be configured to not waste time with trying invalid passwords. So, the password policy should also be handled confidential.
Prevention

[25] http://sectools.org/tool/brutus/

To increase the difficulty of guessing a correct password, password policies should be used. These policies should define a minimal length and complexity for a password.

Additional, user directories are a separate topic for security research. Hence, they can be secured as well against brute force attacks. For example, an account could be blocked if a wrong password was entered three times.

4.3.9 Not discussed Categories

In this subsection the excluded CAPEC top-level categories (CAPEC-1000) are listed.

Injection (CAPEC-152)

Attack patterns of this category are characterized by the "injection" of data that is interpreted afterwards by the target. This is done to alter the behavior of the target in a way to enable the access to restricted information etc.

The translation of this pattern onto the discussed scenario would result in the propose to manipulate a process definition or a task definition. This is no relevant point for process modelers or administrators if the underlying technologies are secured properly.

Manipulate Timing and State (CAPEC-172)

Characteristic for this category of attack patterns is to exploit fault situations. A typical example for a pattern of this category is to interact with a software in a way to trigger a deadlock (CAPEC-25) or to utilize a race condition (CAPEC-26). It is easy to translate these problems into the Business Process Management domain, because a process can get stuck in a deadlock too. Such a possible deadlock needs to be handled, but this is equal with modeling processes in the normal usage of Business Process Management software. But here the scope is on describing security problems with public available interfaces as described in 4.2. These and similar modeling faults are standard faults which need to be avoided always.

Manipulate Data Structures (CAPEC-255)

This category of attack patterns is characterized by the exploitation of badly protected basic data structures. A typical pattern of this category is called "Integer Attack (CAPEC-128)". It exploits the fact that summarizing two integers can lead to a result with a negative prefix. If there is no appropriate protection, the value of a variable may contains an illegal value during

the execution of a program. If an attacker is able to manipulate a value, he can provoke an overflow.
Manipulated data structures are always dangerous and a lot of damage can be caused by them. But for executing such an attack the attacker needs to access the data structure somehow. These patterns address underlying technology and so they are not discussed here.

Manipulate Resources (CAPEC-262)

This category addresses the manipulation of resources. The large amount of CAPEC sub-patterns of this top-level pattern ensures nothing is forgotten. Resources that are referenced as a potential target for manipulation are network infrastructures(CAPEC-89), audit or log files (CAPEC-268), or environment variables (CAPEC-176). The manipulation of these resources is a valid pattern to provoke a broad variation of effects. But to be able to manipulate such resources, a vulnerability in the correlating technology has to be exploited.

Analyze Target (CAPEC-281)

The CAPEC top-level category "Analyze Target" is different to "Gather Information". Analyzing the target means to dissect the targets technology stack. A typical member of this category is "Cryptanalysis (CAPEC-97)" that addresses the search for weaknesses in the used algorithms. This can be very useful for an attacker, but this is not in the scope of this thesis.

Gain Physical Access (CAPEC-436)

This category addresses the burglary of system components or how to get over fences around a computing center. This is not in the scope of this thesis.

Malicious Code Execution (CAPEC-525)

This addresses everything that depends on the execution of software like viruses on a target computer. This is not part of the scope of this thesis.

Alter System Components (CAPEC-526)

This attack pattern addresses the change of hardware components to achieve an effect. This can be the destruction of one or multiple components or the systematic alteration (i.g. hardware keyloggers, etc). This is not in the scope of this thesis.

5 Examination of IBM Business Process Manager

This chapter explains the examination of IBM Businsess Process Manager v.8.5.0.1 Standard. The aim of this examination is to exemplary mine the an appropriate configuration of a web application firewall if the IBM Business Process Manager is used in the scenario that is described in 4.2. So, the HTTP-resources, which are requested when a task is executed by an external user, have to be identified.

The abstract analysis method is not fixated on this case. The general idea is to not rely on a specification, internal or external available documentation, or code/configuration reviews. Instead, the needed information is mined by copying the behavior of a doctor of medicine who auscultates the chest of a patient to find out what is going on inside the body.
Such an auscultation is simple: The stethoscope is placed, then the patient is ask to cough, and finally the doctor interprets what is hearable. By combining this information with the results of other auscultations a final result of the examination created.

This pattern can be adopted to trace network traffic. This adoption results in this abstract analysis protocol:

1. **Setup of a test-environmet** Equal to the doctor who needs a patient, this analysis method needs a test-environment. This environment can differ from a real system in every shape, but it has to produce the targeted traffic. It is all about correct simulation.
2. **Separation of the target traffic** The target traffic needs to be clean from other network traffic. This is important to get reliable results. This is similar to the correct positioning of the stethoscope on the patients chest. The needed separation of the target traffic can be achieved by the design of the test-environment, by the configuration of the software that triggers the target traffic, and by appropriate filtering of the traced data. This is stated as the second step of the analysis method, because the separation by design of the test-environment inherits the lowest fault tolerance.
3. **Triggering the target traffic** When the setup is ready for tracing the target traffic, this traffic needs to be triggered. This is similar to the cough of the patient in the example that is used here. It is important to be sure that all targeted traffic gets triggered for a trace. Otherwise this can falsify the result.
   It is hard to examine network traffic that can not be triggered explicitly. The difficulty is to ensure the completeness of the trace.

4. **Tracing the target traffic** This is similar to the concentrated listening of the doctor. An appropriate tool has to be used to trace the traffic. There are a bunch of sniffing tools on the market that can be used. The decision depends on the context of the analysis.
   It is important to ensure that no transmitted data is missed.
5. **Processing the single trace** It is possible that the traced data needs to be processed somehow before it can be processed with other traces. For example, the application of a filter could be needed.
   This maps to the recognition of a symptom by the doctor in the example that is used here.
6. **Processing with other traces** To derive ongoing conclusions, multiple traces can be processed together. It depends on the aim of the examination how this processing needs to be.
   In the example that is used here, this maps to the combination of the recognized symptom with other symptoms to derive a diagnosis.
7. **Validation of the processed results** This is the last step in both, the example with the medical examination and this analysis method. When the doctor derived a diagnosis from the identified symptoms, he executes a final test for evaluating this diagnosis. Similar to this, the test-environment can be reconfigured in a way to enable a test that evaluates the results of the previous step.

The following sections document an exemplary examination of IBM Business Process Manager to determine a exemplary web application firewall configuration.

5.1 Examination Approach

The aim of this exemplary examination is the analysis of the network traffic that has to pass the web application firewall as described in 4.2. So, the involved HTTP-resources have to be identified.

As described in section 3.6 IBM Business Process Manager provides the possibility to define task user interfaces which are called "Coaches". Each Coach consists of single so called "Coach Views". IBM delivers a set of so called "Stock Coach Views" with the IBM Business Process Manager.
Advanced users may define additional Coach Views or they may configure additional systems to communicate with the IBM Business Process Manager APIs to enable to involvement of humans. It is not possible to include these specific solutions into this exemplary examination, because it is not possible to determine them.
So, only Coaches which consist of Stock Coach Views are part of this examination. To analyze the traffic that is produced by these Stock Coach Views is an example that can be repeated

with other Coach Views.

The approach for identifying every involved HTTP-resource is to trace every possible network traffic that is produced by every single Stock Coach View.

This exemplary examination mines for a web application firewall configuration that filters the network traffic based on the URL-paths of the requested HTTP-resources. This adopts the example that is already made in 4.3.6, it enables the examination by reusing technologies which are already described in 3.10, and by just focusing on one HTTP-property the example is kept simple.

In a production system other configurations of web application firewalls are possible. Every property and even content of the body of a HTTP-request can be used for defining such configurations.

5.2 Test Environment

According to the abstract analysis method, a test-environment is needed. This environment has to enable the separation of the network traffic that is produced by the portal and the traffic that is produced by the Coaches. Additional, network traffic will be created to call the appropriate Human Services which contain the Coaches. This is explained in 3.4.

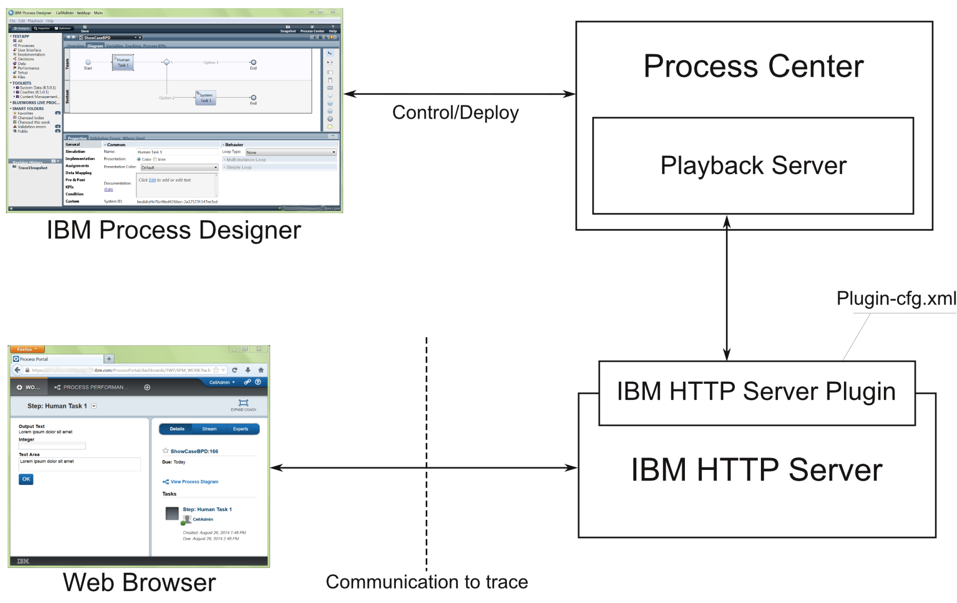


Figure 5.1: The Test-Environment

Figure 5.1 plots the used test-environment. It is not necessary to configure a dedicated instance of IBM Process Server, because the Playback Server is sufficient. To separate the the network traffic that is produced by the IBM Process Designer from the traffic that shall be traced, the IBM Process Designer accesses the Process Center directly and the remaining traffic is redirected via an IBM HTTP Server.

The IBM HTTP Server is already described in 3.5.1 in terms of load balancing. But, it can be reused as a simple web application firewall in this test-environment to validate the results of the examination. The IBM HTTP Server Plugin is configurable to forward or reject HTTP-requests based on the URL-path. This is exactly what is needed here.

It is not possible to separate the traffic belonging to the Coaches from the traffic that belongs to the portal by design of this test-environment. This is be done later-on.

5.3 Tracing

This section summarizes the tree core-steps of the abstract analysis method: Triggering the target traffic, tracing the network traffic, and filter out remaining data that does not belong to the trace.

Here, the tracing protocol is demonstrated for the case of the traffic produced by the Stock Coach View "Decimal". This is an editable field that deals with decimals.

First, a Human Service is modeled that contains exactly one Coach consisting of the Stock Coach View "Decimal". This is shown in figure 5.2. If the Stock Coach View is able to communicate with a REST-service to provide additional functionality, a dummy-service has to be modeled. The Stock Coach View "Decimal" does not provide such additional functionality.

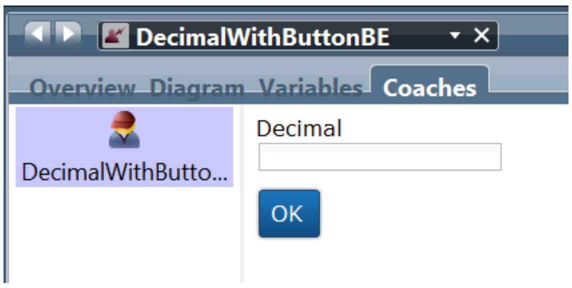


Figure 5.2: Modeled Coach with Stock Coach View "Decimal"

It is important to explain why figure 5.2 shows an additional button. This button is needed, because a "Decimal" Stock Coach View is not able to finish the Coach. So, an additional view

has to be used in the Coach. In the set of Stock Coach Views just "Button" is able to do this by default.
Of course, it is possible to script the behavior of the Coach to finish the Coach when a value is inserted, but this is not necessary. There is no difference in the produced network traffic when the button finishes the Coach instead of a JavaScript function. This was evaluated during the development of this analysis protocol.

To trace a Coach's network traffic a Human Service has to be executed to trigger the target traffic. This is done by requesting the URL that is shown in listing 5.1.

```
https://<host>:<port>/teamworks/executeServiceByName?
   processApp=<ProcessApplicationName>&
   serviceName=<ServiceName>
```

Listing 5.1: URL to call a Human Service

When the URL in listing 5.1 is browsed then the browser gets immediately redirected to the first Coach of the Human Service. In this scenario, the Coach is presented standalone and not contained in the IBM Process Portal.
To request this URL, an already authenticated session with the server is needed. Additional, to ensure every involved resource is requested, it is necessary to clean the browser's cache before a trace.

Here, the sniffing tool "Wireshark" [1] is used to trace the network traffic. Wireshark is chosen, because it enables filtering on the traced network traffic, to export the filtered data in a bunch of different formats, and to reconstruct higher OSI-levels from SSL/TLS-encrypted data if the private keys are known.

Based on these facts, the used tracing protocol is described here:

1. **Open a private browsing window** By the use of the private browsing function, which is provided by nearly all modern browsers, all caches are clean. It is also possible to clean the caches by hand before every trace, but using the private browsing windows is free of any risk to falsify the result by accident.
2. **Authenticate** Before the trace can be started, the new browsing session has to be authenticated to the system. This is done by requesting the URL of a REST-service that does not change the server's state. For example, the state of a not existing process can be requested.
   This provokes a redirection to a log-on screen and after logging on, the session is authenticated. Now, this browser instance can be used for tracing the target traffic.

[1] https://www.wireshark.org/

3. **Start the Trace** Now, the trace can to be started. This is done by starting a unfiltered "Live Capture" in Wireshark.
4. **Call the Human Service** By requesting the URL in listing 5.1 the Human Service, which contains the target Coach, gets executed. This is done by requesting the URL with the authenticated browsing session.
5. **Use the Coach** The Coach needs to be used by hand. It is important to achieve a complete exploitation of the Coach's functionality. For example, if auto-completion is provided by the target Coach View, it has to be assured that the REST-service is called.
6. **Finish the Coach** The Coach gets finished by clicking the button.
7. **Stop the Trace** The Wireshark capture has to be stopped now.
8. **Decrypt the Trace** It is important to keep the order of these steps. If this is not done, the decryption probably is not possible. This would happen if the SSL/TLS-handshake is not part of the traced data.
   Additional, the private key of the IBM HTTP Server has to be imported to Wireshark to decrypt the traced data.
9. **Filter the Trace** To get proper data for the evaluation, everything that does not belong to the intended trace has to be filtered out. For example, a mail-reception that happened in the background or the hidden download of the latest java-update. This is done by applying the Wireshark-filter in listing 5.2.
10. **Export the Result** Now, the data is exported to a CSV-formatted file. This file contains all necessary information that is needed later on and it is readable for the tool "URL Path Analyzer" that is described in 6.1.

```
http && (ip.src == <IP-address of the IBM HTTP Server> ||
   ip.dst == <IP-address of the IBM HTTP Server>)
```

Listing 5.2: Wireshark Filter to clean the Trace

5.4 Analysis

This section describes how the traced data is processed with other traces.
By using the tool "URL Path Analyzer", which is described in 6.1, set operations like "Merge", "Intercept", and "Complement" can be performed to process different traces with each other. This can be done to get a set of URL-paths that contains all requested resources or to determine the resources which are just appearing in some traces.
Using of the possibility to mark single URL-path segments as "dynamic" and the automated

restructuring of the data, ephemeral resources can be handled to not falsify the results of a set-operation.

Finally, a prepared dataset can be exported to the normalized CSV-format for a validation run. Of course, the dynamic URL-path segments must be removed or filled with a dummy segment for a validation run.

5.5 Validation

The tool "URL Path Validator", which is described in 6.2, can be used to check if the HTTP-response codes behave as expected when HTTP-resources are requested.
This can be used to check, if the configuration of the web application firewall works as expected.

It has to be mentioned here, that the URL Path Validator requests just a list of URL-paths. There is no guarantee that there are no additional available resources. So, the administrator has to be careful while defining the web application firewall policies.

5.6 Examination Result

To be able to configure a web application firewall in the way that Coaches can be used via this firewall all involved resources need to be accessible. All traces have to be merged to a single dataset by using the URL Path Analyzer. This dataset contains all the resources that have to be accessible via the web application firewall.

The following subsections address the identified involved resources. They are categorized based on their semantics and whether they should be configured with a dynamic path segment or not as it is described in 4.3.6.

5.6.1 Static addressable non-functional Resources

This subsection addresses all resources which are requested with a HTTP-GET request and do not fulfill a logical function.
Of course, these resources are needed for functionality, but the request itself is not part of the logic. For example, the transmission of a JavaScript-file is needed, but no there is no logical semantic in it.

All URL-paths in listing 5.3 must be enabled for HTTP-GET operations in the web application firewall.

```
/portal/js/dojo/1.6.1/dojo/dojo.js
/teamworks/images/common/loading.gif
/teamworks/script/coachNG/com/ibm/bpm/coach/resources/iframe_history.html
/teamworks/script/coachNG/com/ibm/bpm/coach/resources/images/favicon.ico
/teamworks/script/coachNG/dojo/1.8.3/dijit/_editor/nls/FontChoice.js
/teamworks/script/coachNG/dojo/1.8.3/dijit/_editor/nls/LinkDialog.js
/teamworks/script/coachNG/dojo/1.8.3/dijit/_editor/nls/commands.js
/teamworks/script/coachNG/dojo/1.8.3/dijit/dijit_bpm.js
/teamworks/script/coachNG/dojo/1.8.3/dijit/form/Select.js
/teamworks/script/coachNG/dojo/1.8.3/dijit/form/nls/ComboBox.js
/teamworks/script/coachNG/dojo/1.8.3/dijit/form/nls/validate.js
/teamworks/script/coachNG/dojo/1.8.3/dijit/icons/images/editorIconsDisabled.png
/teamworks/script/coachNG/dojo/1.8.3/dijit/icons/images/editorIconsEnabled.png
/teamworks/script/coachNG/dojo/1.8.3/dijit/nls/common.js
/teamworks/script/coachNG/dojo/1.8.3/dijit/nls/dijit_bpm_ROOT.js
/teamworks/script/coachNG/dojo/1.8.3/dijit/nls/loading.js
/teamworks/script/coachNG/dojo/1.8.3/dijit/themes/claro/form/images/buttonArrows.png
/teamworks/script/coachNG/dojo/1.8.3/dijit/themes/claro/form/images/checkboxRadioButtonStates.png
/teamworks/script/coachNG/dojo/1.8.3/dijit/themes/claro/form/images/commonFormArrows.png
/teamworks/script/coachNG/dojo/1.8.3/dijit/themes/claro/form/images/error.png
/teamworks/script/coachNG/dojo/1.8.3/dijit/themes/claro/images/calendarArrows.png
/teamworks/script/coachNG/dojo/1.8.3/dijit/themes/claro/images/loadingAnimation.gif
/teamworks/script/coachNG/dojo/1.8.3/dijit/themes/claro/images/tooltip.png
/teamworks/script/coachNG/dojo/1.8.3/dojo/cldr/nls/currency.js
/teamworks/script/coachNG/dojo/1.8.3/dojo/cldr/nls/gregorian.js
/teamworks/script/coachNG/dojo/1.8.3/dojo/cldr/nls/hebrew.js
/teamworks/script/coachNG/dojo/1.8.3/dojo/cldr/nls/islamic.js
/teamworks/script/coachNG/dojo/1.8.3/dojo/cldr/nls/number.js
/teamworks/script/coachNG/dojo/1.8.3/dojo/dojo.js
/teamworks/script/coachNG/dojo/1.8.3/dojo/nls/dojo_ROOT.js
/teamworks/script/coachNG/dojo/1.8.3/dojo/resources/blank.gif
/teamworks/script/coachNG/dojo/1.8.3/dojox/dojox_bpm.js
/teamworks/script/coachNG/dojo/1.8.3/dojox/form/nls/CheckedMultiSelect.js
/teamworks/script/coachNG/dojo/1.8.3/dojox/grid/enhanced/nls/EnhancedGrid.js
/teamworks/script/coachNG/dojo/1.8.3/dojox/grid/enhanced/nls/Pagination.js
/teamworks/script/coachNG/dojo/1.8.3/dojox/grid/enhanced/resources/images/sprite_icons.png
/teamworks/script/coachNG/dojo/1.8.3/dojox/grid/resources/images/header_shadow.png
/teamworks/script/coachNG/dojo/1.8.3/dojox/grid/resources/images/row_back.png
```

```
/teamworks/script/coachNG/dojo/1.8.3/dojox/grid/resources/images/td_button_down.png
/teamworks/script/coachNG/dojo/1.8.3/dojox/nls/dojox_bpm_ROOT.js
/teamworks/script/coachNG/verspec/1.8.3/theme/coach_ng.css
/teamworks/webasset/2064.3ad61f02−4da6−4881−84de−455a3377602a/W/Error_icon_24x24.png
/teamworks/webasset/2064.3ad61f02−4da6−4881−84de−455a3377602a/W/checkbox_sprite.png
/teamworks/webasset/2064.3ad61f02−4da6−4881−84de−455a3377602a/W/coach_ng_controls.css
/teamworks/webasset/2064.3ad61f02−4da6−4881−84de−455a3377602a/W/coach_ng_sprite.png
/teamworks/webasset/2064.3ad61f02−4da6−4881−84de−455a3377602a/W/coach_ng_utilities.zip/com/ibm/bpm/coach/controls/utilities.js
/teamworks/webasset/2064.3ad61f02−4da6−4881−84de−455a3377602a/W/utilities.js
```

Listing 5.3: URL Paths of all identified static non-functional Resources

5.6.2 Static addressable functional Resources

This category contains resources which are always involved when Coaches are used. These resources are addressed by the URL-paths in listing 5.4 and listing 5.5.

```
/teamworks/executeServiceByName

/teamworks/process.lsw

/teamworks/fauxRedirect.lsw

/teamworks/jsonResBundleLoader

/teamworks/tm_process_finished.lsw
```

Listing 5.4: Static addressable functional HTTP-GET Resources

First, the URL-paths in listing 5.4 are discussed:

The first two URL-paths are needed when the execution of a Human Service begins. If the Human Service is called by the engine, then the browser is directed to */teamworks/process.lsw* by the portal. This is the "switch" to the Coach. Afterwards, the browser gets redirected to */teamworks/fauxRedirect.lsw*. This is the main URL-path and every browser ends on this URL-path during the execution of Coaches.

The URL-path */teamworks/jsonResBundleLoader* is needed to address a resource that is used for requesting data. The requested data is defined via a URL-query parameter. In all traces of this examination there was only one dataset requested. This is called "JavaScriptMessages" and the response contains strings which are needed for the displayed notifications (Exception messages, etc).

The last URL-path in listing 5.4 is used for redirecting the browser after the Human Service is finished (*/teamworks/tm_process_finished.lsw*). It depends to the overall logic what happens afterwards.

```
/rest/bpm/wle/v1/service/1.3402b466-ea01-464c-bc00-317677d58a67

/teamworks/ajaxCoach
```

Listing 5.5: Static addressable functional HTTP-POST Resources

The URL-paths in listing 5.5 belong to this category too, but there is one difference: They are accessed with HTTP-POST requests.

The first URL-path is used to transmit browser properties like the system-language to the server and to get appropriate data that depends on these parameters.
The URL-path */teamworks/ajaxCoach* is one of the most important to mention here. It is used to post the final data of a Coach to the server and by this, to finish the Coach.

5.6.3 Dynamic addressed functional Resources

This category contains the services which are called when it comes to server-side validation or auto-completion etc.
Services like these are addressed via their service-ID, that is the last segment of the URL-paths in listing 5.6. Due to the fact that new services are modeled and provisioned after the system-setup, these category is named as dynamic addressed. This means, the web application firewall has to be configured in a way, that all service-IDs which are used with the URL-path "/rest/bpm/wle/v1/service/<SERVICE-ID>" are enabled. This is a potential security problem (Attack Pattern "Exploit the fixated Firewall Configuration" in 4.3.6).
Services like these are typically requested via HTTP-POST requests.

The two URL-paths in listing 5.6 are examples and sourced by the tracing of the single Stock Coach Views that was done during this examination.

```
-- TestSelectionService
/rest/bpm/wle/v1/service/1.19fa95ef-9199-4174-b6c1-31b28f236389

-- TestAutoCompletionService
/rest/bpm/wle/v1/service/1.f21f53b4-b5c6-452f-bd13-270a0b83aaa8
```

Listing 5.6: URL-Paths of functional dynamic addressed Resources

5.6.4 Dynamic addressed non-functional Resources

The resources belonging to this category are called dynamic addressed because of the same reason because of the resources in 5.6.3 are called this way. The difference is that the resources of this category are not part of the logic. These are similar to 5.6.1. During the examination only one URL-path was found that belongs to this category. This URL-path is shown in listing 5.7.
The referenced resource was involved with the Stock Coach View "Image", that is used to display an image. There is no further functionality.

```
/teamworks/webasset/2064.f76ad095-36db-40cc-b6a0-81b89db796ab/W/testPic.png
```

Listing 5.7: URL-Paths of non-functional dynamic addressed Resources

5.7 Security Reflection of the Examination

In 5.6.3 and 5.6.4 some URL-paths are discussed whose existence begins after the installation of the system. These resources are created during the modeling and the deployment of the process applications.
For example, to access all existing services, which belong to the category "Dynamic addressed functional Resources" (5.6.3), a rule has to be added to the web application firewall configuration that enables every service-ID. The generation of service-IDs is not suggestible for the modelers. This is why a dynamic rule is needed. Yet, it is also difficult to identify the ID of a service.

If the process applications are getting frozen after the development, it is possible to mine for the generated service-IDs and to address them statically in the web application firewall configuration. Static addressing is strongly recommended to prohibit the provisioning of restricted functionality by accident.
It is the same for the resources which are addressed by the URL-paths which are discussed in 5.6.4.

This correlates with the pattern "Exploit the fixated Firewall Configuration" that is described in 4.3.6.

5.8 General Security Reflection

This section contains two points which have to be discussed with the use of IBM Business Process Manger in the examined scenario. These points are related to the attack patterns,

which are described in 4.3.
Every pattern of this collection is probably dangerous for such a system, but these points have to be highlighted here.

5.8.1 Task Reassignment

The engine calls a Human Service during the navigation of a process and waits until it returns. Assumed that an internal user finishes some Coaches of a Human Service and reassigns the remaining tasks to an external user.

The concept of the Human Services just orders the Coaches. The Human Service concept includes atomicity. This means, it is not designed to be secure on a reassignment like this. The external user would be able to manipulate the data that was inserted by the first user. This can be done by using tools for manual composing of HTTP-requests.

This correlates with the pattern "Exploit the Misuse of Concepts" that is described in 4.3.7.

5.8.2 User Directory

Business Process Management Systems join the involved user directories. This is needed to enable the collaboration of these users in the processes. This means, a complete separation between multiple user directories (i.g. internal and external users) is not possible with one Business Process Management System. IBM Business Process Manager does not provide a solution for this.
Hence, when IBM Business Process Manager is configured to use multiple user directories in the discussed scenario to separate internal and external users, the desired protection needs to be realized with additional mechanisms.

This correlates with the patterns "Account Mining" in 4.3.1 and "Brute Force User Credentials" in 4.3.8.

6 Tools

This chapter documents the tools "URL Path Analyzer" and "HTTP Setup Validator", which were implemented to facilitate the examination that is described in chapter 5. Both tools are written in Java-1.7. As UI-framework Java Swing[1] is used.
This diploma thesis is created in cooperation with IBM Germany Research and Development, so the tools are implemented to be compatible with the IBM Java Platform [2].

6.1 URL Path Analyzer

The "URL Path Analyzer" is designated to parse CSV-formatted Wireshark traces and to process them. The purpose is to enable the user to analyze the traced data focusing on the URL-paths, the HTTP request type, and the dynamic parts of the URL-paths.

6.1.1 Use Cases

This section summarizes the main use cases of the URL Path Analyzer. Here, they are ordered as they can be executed.

Parse Wireshark CSV

In section 5.3, the tracing protocol is described. If this protocol gets executed exactly, the produced file should have the same format as shown in listing 6.1.
The URL Path Analyzer is able to import these files and to parse them into a persistent tree structure. This casts the representation of the traced requests from a "per request"-view to a "which resources were involved"-perspective. A resource that was requested multiple times is represented as one entry. Each data structure, that was created by parsing such a file, is called "Dataset".

[1]http://www.ibm.com/developerworks/java/tutorials/j-intswing/j-intswing.html
[2]https://www.ibm.com/developerworks/java/jdk/

```
"No.","Time","Source","Destination","Protocol","Length","Info"\r\n
"01","00.000000000","<Source IP>","<Target IP>","HTTP","1312","GET /teamworks/executeServiceByName?processApp=TA&serviceName=CombinedViewsService
      HTTP/1.1 "\r\n
"02","00.000000001","<Target IP>","<Source IP>","HTTP","171","HTTP/1.1 303 See Other (text/html)"\r\n
"03","00.000000002","<Source IP>","<Target IP>","HTTP","1408","GET
      /teamworks/fauxRedirect.lsw?zWorkflowState=2&zTaskId=p1&zComponentName=CoachNG&zComponentId=3028.3b266857-ad72-4542-998e-441f89e19bab&zDbg=0
      HTTP/1.1 "\r\n
"04","00.000000003","<Target IP>","<Source IP>","HTTP","869","HTTP/1.1 200 OK (text/html)HTTP/1.1 200 OK (text/html)HTTP/1.1 200 OK (text/html)"\r\n
"05","00.000000004","<Source IP>","<Target IP>","HTTP","1440","GET /teamworks/script/coachNG/dojo/1.8.3/dojo/dojo.js HTTP/1.1 "\r\n
"06","00.000000005","<Target IP>","<Source IP>","HTTP","91","HTTP/1.1 200 OK (application/javascript)"\r\n
"07","00.000000006","<Source IP>","<Target IP>","HTTP","140","GET /teamworks/jsonResBundleLoader?bundle=JavaScriptMessages HTTP/1.1 "\r\n
"08","00.000000007","<Target IP>","<Source IP>","HTTP","91","HTTP/1.1 200 OK (application/json)"\r\n
"09","00.000000008","<Source IP>","<Target IP>","HTTP","268","POST /rest/bpm/wle/v1/service/1.19fa95ef-9199-4174-b6c1-31b28f236389 HTTP/1.1
      (application/x-www-form-urlencoded)"\r\n
"10","00.000000009","<Target IP>","<Source IP>","HTTP","91","HTTP/1.1 200 OK (application/json)"\r\n
"11","00.000000010","<Source IP>","<Target IP>","HTTP","268","POST /rest/bpm/wle/v1/service/1.19fa95ef-9199-4174-b6c1-31b28f236389 HTTP/1.1
      (application/x-www-form-urlencoded)"\r\n
"12","00.000000011","<Target IP>","<Source IP>","HTTP","91","HTTP/1.1 200 OK (application/json)"\r\n
"13","00.000000012","<Source IP>","<Target IP>","HTTP","236","POST /rest/bpm/wle/v1/service/1.3402b466-ea01-464c-bc00-317677d58a67 HTTP/1.1 \r\n
"14","00.000000013","<Target IP>","<Source IP>","HTTP","91","HTTP/1.1 200 OK (application/json)"\r\n
"15","00.000000014","<Source IP>","<Target IP>","HTTP","652","POST /teamworks/ajaxCoach HTTP/1.1 (application/json)"\r\n
"16","00.000000015","<Target IP>","<Source IP>","HTTP","203","HTTP/1.1 200 OK (application/json)"\r\n
"17","00.000000016","<Source IP>","<Target IP>","HTTP","76","GET /teamworks/tm_process_finished.lsw?applicationInstanceId=null&applicationId=1
      HTTP/1.1 "\r\n
"18","00.000000017","<Target IP>","<Source IP>","HTTP","1043","HTTP/1.1 200 OK (text/html)"\r\n
"19","00.000000018","<Source IP>","<Target IP>","HTTP","1248","GET /favicon.ico HTTP/1.1 "\r\n
"20","00.000000019","<Target IP>","<Source IP>","HTTP","379","HTTP/1.1 404 Not Found (text/html)"\r\n
```

Listing 6.1: Sample Wireshark CSV File

View a Dataset

The imported datasets are listed in the main window of the URL Path Analyzer, as shown in figure 6.1. This window provides also a set of actions that can be executed on the Datasets.

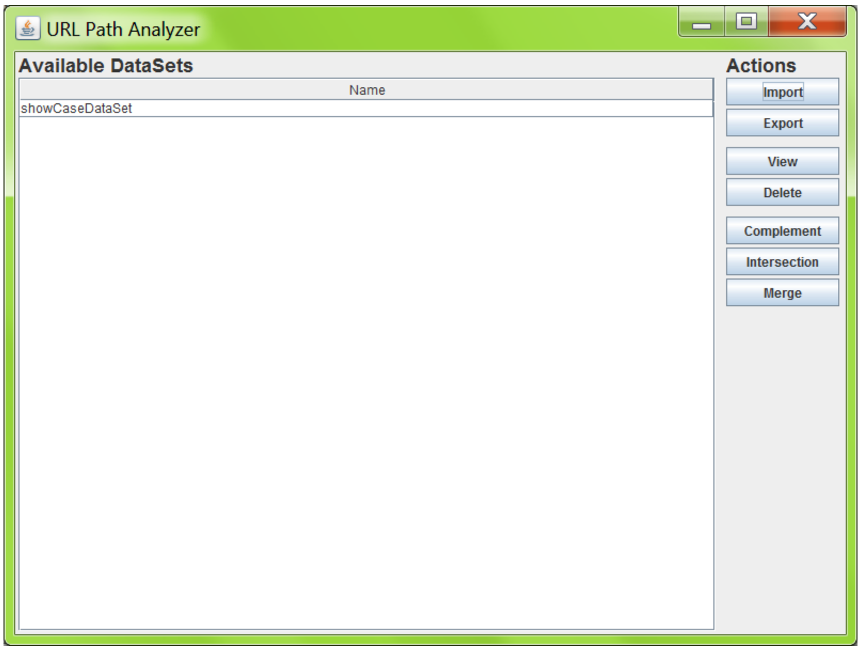


Figure 6.1: Main Window of URL Path Analyzer

By selecting a dataset and hitting the "View"-button, the Dataset-Viewer is executed, as shown in figure 6.3. This is a core component of the URL Path Analyzer. It is essential to understand how the data is presented.

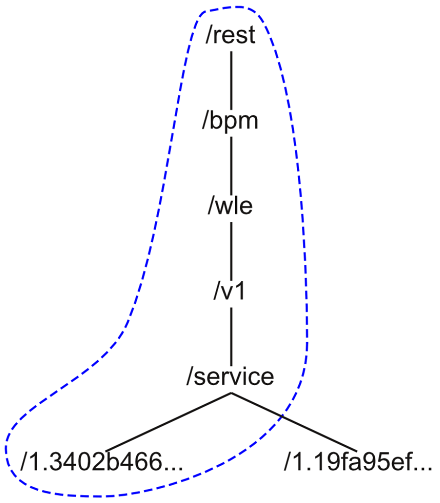
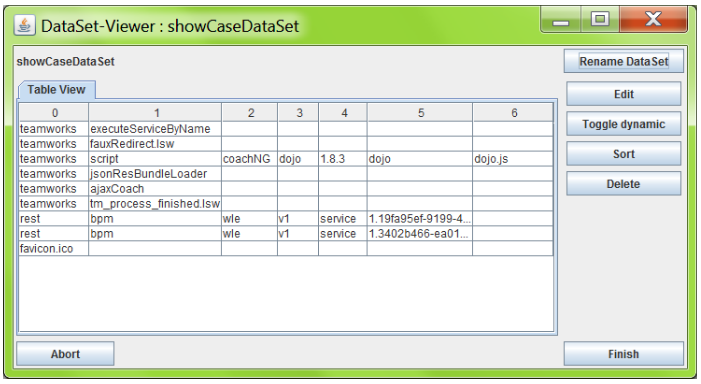


Figure 6.2: Tree Structure of a Dataset

Figure 6.2 shows how the data is structured in memory. Path structures are trees and these trees are reconstructed when a file is parsed. Every node of these trees represents a segment of a path. Additional, requests are represented by tagging the appropriate node.
To render a dataset in the Dataset-Viewer, every tagged node is represented as a row of a table. This means, a tagged node that is not a leaf node, is represented as a dedicated row too. So, every row of the Dataset-Viewer represents a resource that was requested one or multiple times.

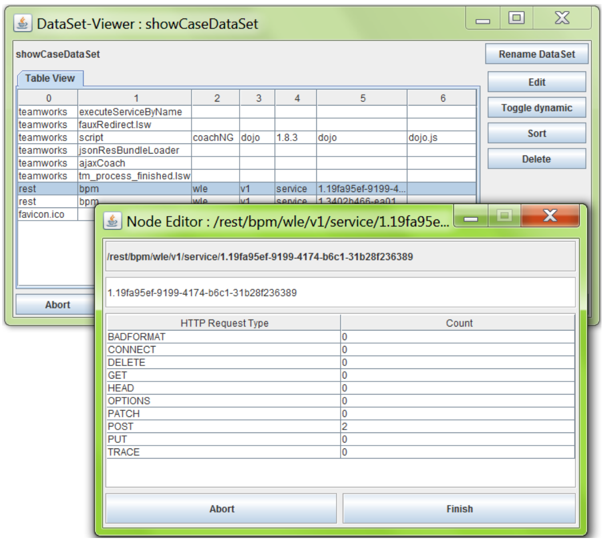


0	1	2	3	4	5	6
teamworks	executeServiceByName					
teamworks	fauxRedirect.lsw					
teamworks	script	coachNG	dojo	1.8.3	dojo	dojo.js
teamworks	jsonResBundleLoader					
teamworks	ajaxCoach					
teamworks	tm_process_finished.lsw					
rest	bpm	wle	v1	service	1.19fa95ef-9199-4...	
rest	bpm	wle	v1	service	1.3402b466-ea01...	
favicon.ico						

Figure 6.3: Dataset-Viewer presenting a Dataset

Each cell of the table, that is shown in the Dataset-Viewer, represents a node in the tree structure. Some nodes (i.g. "/rest") are represented by multiple cells, because multiple descendant nodes are tagged.

Every cell can be selected and the "Node-Viewer" can be used to inspect/edit the URL-path segment. Additional, the tags of the node can be viewed.
In figure 6.4, the Node-Viewer shows a REST-service that was used twice.


DataSet-Viewer : showCaseDataSet
showCaseDataSet
Rename DataSet
Table View
Edit
Toggle dynamic
Sort
Delete
Abort
Node Editor : /rest/bpm/wle/v1/service/1.19fa95e...
Abort
Finish

0	1	2	3	4	5	6
teamworks	executeServiceByName					
teamworks	fauxRedirect.lsw					
teamworks	script	coachNG	dojo	1.8.3	dojo	dojo.js
teamworks	jsonResBundleLoader					
teamworks	ajaxCoach					
teamworks	tm_process_finished.lsw					
rest	bpm	wle	v1	service	1.19fa95ef-9199-4...	
rest	bpm	wle	v1	service	1.3402b466-ea01	
favicon.ico						

/rest/bpm/wle/v1/service/1.19fa95ef-9199-4174-b6c1-31b28f236389

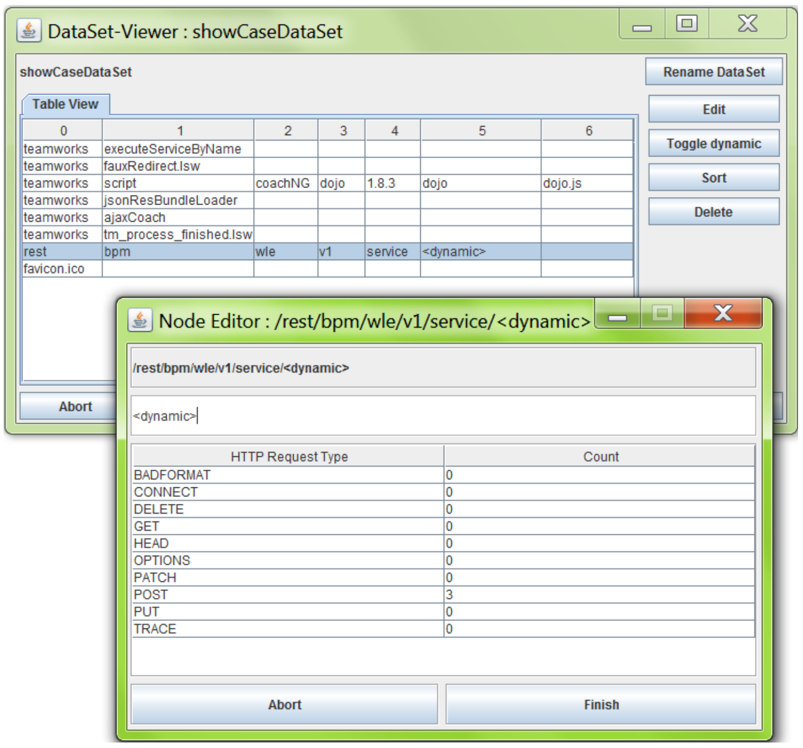
1.19fa95ef-9199-4174-b6c1-31b28f236389

HTTP Request Type	Count
BADFORMAT	0
CONNECT	0
DELETE	0
GET	0
HEAD	0
OPTIONS	0
PATCH	0
POST	2
PUT	0
TRACE	0

Figure 6.4: Node Editor

Handing of Dynamic Paths

Some path-segments are dynamic. This variability is already discussed in 4.3.6 and 5.6.3. The examples, that are used in this chapter, contain the URL-paths of two REST-services. These paths are dynamic in the last segment. This means, every REST-service is addressed by the use of the same URL-path up to the last segment. This segment identifies the service that shall be used. Such fugitive URL-path segments can be marked as "dynamic" by changing the string to "<dynamic>" or by using the button "Toggle dynamic". If multiple dynamic nodes are children of the same parent node, the URL Path Analyzer restructures the tree by merging the dynamic nodes.



0	1	2	3	4	5	6
teamworks	executeServiceByName					
teamworks	fauxRedirect.lsw					
teamworks	script	coachNG	dojo	1.8.3	dojo	dojo.js
teamworks	jsonResBundleLoader					
teamworks	ajaxCoach					
teamworks	tm_process_finished.lsw					
rest	bpm	wle	v1	service	<dynamic>	
favicon.ico						

/rest/bpm/wle/v1/service/<dynamic>

<dynamic>

HTTP Request Type	Count
BADFORMAT	0
CONNECT	0
DELETE	0
GET	0
HEAD	0
OPTIONS	0
PATCH	0
POST	3
PUT	0
TRACE	0

Figure 6.5: Node Editor with a dynamic Node

For example, the URL-paths of the two REST-services, which can be seen in the previous figures and listings, are marked as dynamic in the last segment. Figure 6.5 shows the merged node that is tagged now three times. This is the summation of the tags of the two separated nodes.

Set Operations

Figure 6.1 shows the main window of the URL Path Analyzer. Additional to other actions, three set operations are provided. These can be used to process the datasets with each other.
For example, all datasets could be merged to generate an overview of all involved URL-paths of a traced network traffic.

Additional to "Merge", the set operations "Intercept" and "Complement" are provided. These set operations are working also with dynamic nodes.

Export

The URL Path Analyzer provides two CSV-formattings, that can be used to export a dataset.

- **PATHCENTRIC** Each line consists of a URL-path in the first CSV-cell. This URL-path is unique per file. The remaining cells of the line contain HTTP-request types which represent the tags of this path. If a URL-path is tagged with four HTTP-GET requests, four cells are filled with "GET".
- **NORMALIZED** This formatting addresses the fact that a URL-path can be used with multiple HTTP request types. This is the format for the ongoing use with the HTTP Setup Validator. This CSV-formatting consists of test assets. Every line consists of two CSV-cells. The first cell contains a URL-path and the second cell contains a HTTP-request type.

Listing 6.2 matches both CSV-formattings, but with different semantics. It fulfills the pattern PATHCENTRIC if every URL-path is tagged once and it also fulfills the NORMALIZED pattern if every URL-path is tagged with one HTTP-request type.

```
"/teamworks/executeServiceByName","GET"
"/teamworks/fauxRedirect.lsw","GET"
"/teamworks/script/coachNG/dojo/1.8.3/dojo/dojo.js","GET"
"/teamworks/jsonResBundleLoader","GET"
"/teamworks/ajaxCoach","POST"
"/teamworks/tm_process_finished.lsw","GET"
"/rest/bpm/wle/v1/service/1.19fa95ef-9199-4174-b6c1-31b28f236389","POST"
"/favicon.ico","GET"
```

Listing 6.2: CSV File to be parsed by HTTP Setup Validator

The URL Path Analyzer is able to import both CSV-formattings by selecting "CSV" as import configuration. But, the NORMALIZED-formatting inherits the loss of the information how often a URL-path is tagged with the same HTTP request type. This is compensated by tagging the URL-paths once.

6.1.2 Technical Perspective

This section gives a small view on the technical internals of the URL Path Analyzer. To understand the URL Path Analyzer, it is important to know that every functionality is about managing the tree structure that is already described in figure 6.2.

This tree structure is enriched with additional values. Every node is valued by an indexer to indicate the amount of tagged nodes in the sub-tree of a node. Additional, the tree is traversed in pre-order[3] when the re-indexing is triggered. This traversal identifies only the tagged nodes with an integer that represents their position in the pre-order traversal. By the use of this index, the tagged nodes can be located the same way as in a binary search tree. This decreases the needed time to find a tagged node in the tree. This is important, because the model of the table in the Dataset-Viewer re-renders the tagged nodes every time the tree changes.

[3] http://www.brpreiss.com/books/opus4/html/page258.html

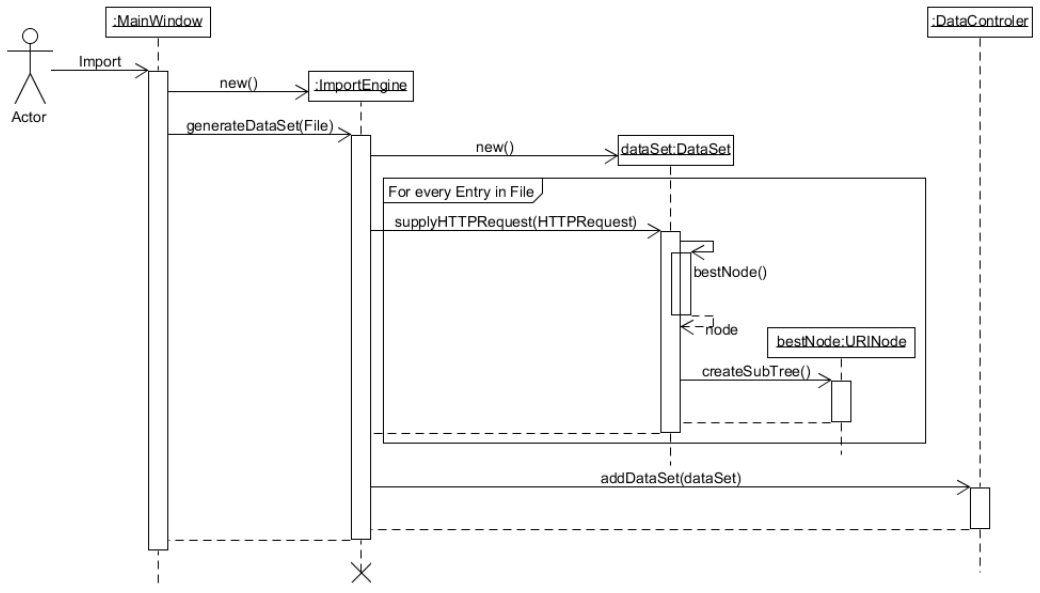


Figure 6.6: Sequence Diagram of Path Supplement

For a general understanding of the tree structure, figure 6.6 shows the tree construction during the supplement of URL-paths from an *ImportEngine*. This supplement is divided in two phases. First, the best node is searched. This means, the part of the new path that is already represented by the tree is traversed. This traversal leads either to a complete match with the new path or to a partial match. In the second case a new sub-tree is added to the best matching node.

6.2 HTTP Setup Validator

The HTTP Setup Validator is a tool to check the HTTP response codes of a set of HTTP-requests. This can be used to test the configuration of a web application firewall, as it is described in 4.2 and 5.2.
A test set is defined by a dataset that is prepared with the URL Path Analyzer. The HTTP Setup Validator supports SSL/TLS protected communication (every server certificate is accepted!). Additional, HTTP Basic Authentication[4] (username and password) can be used to authenticate a test-user. This is probably needed to receive meaningful results.

[4]https://www.ietf.org/rfc/rfc2617.txt

At least, a set of color codes can be defined that is used to underlay the different response codes. So, it is possible to evaluate the test results automatically.

6.2.1 Use Cases

This subsection lists the use cases of the HTTP Setup Validator and by this, instructions how to use the tool.

Define Color Codes

The URL Path Analyzer can be configured to use colors to underlay the response codes. By default, every HTTP-response code is underlaid with a strong red. The only exception is "200", which is is the HTTP response code for "OK". The color that is used when an exception occurred is white (no color). If some other colors shall be used, they can be defined in a configuration file. This file is passed to the tool as a command-line argument as shown in listing 6.3.

```
java -jar URLPathValidator.jar <ColorCodeFile>.csv
```

Listing 6.3: Command to start the HTTP Setup Validator

The configuration file is CSV-formatted and by this readable and editable for users. Each line consists of three entries. The first entry is an HTTP request type, the second entry is a response code, and the third entry is a hexadecimal color code. Listing 6.4 contains two configurations. The first rule ensures that every time 200 is the response code of correlating to HTTP-POST request, the line is underlaid with purple. The second line works equal for the response code 401 and the color cyan.

```
"POST","200","#FF00FF"
"POST","401","#00FFFF"
```

Listing 6.4: Sample Color Configuration File

Define the Target

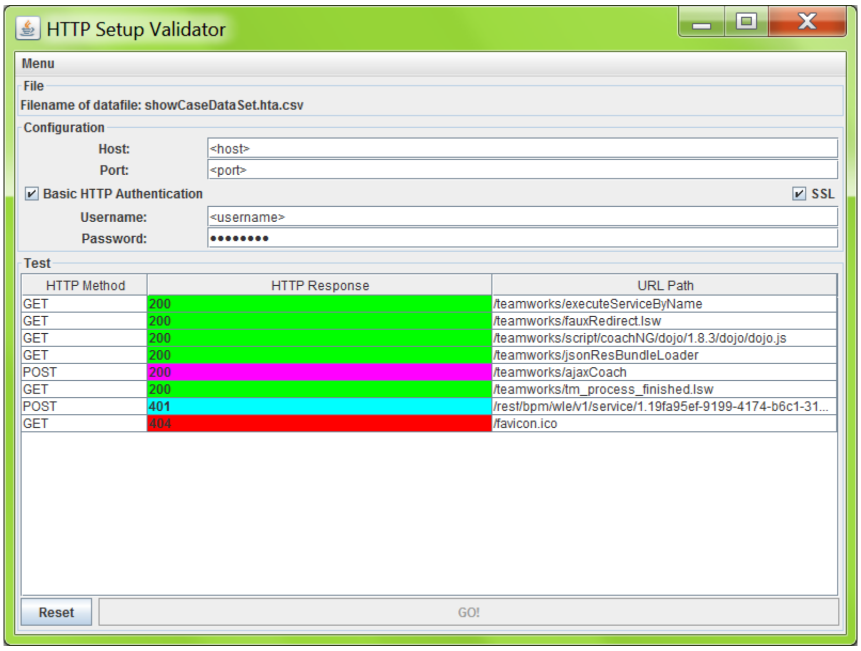
To execute a HTTP-request, a complete URL is needed. These URLs are constructed from the entered information. The HTTP Setup Validator is able to perform a simple test of a system configuration.

To be widely applicable, the HTTP Setup Validator accepts every SSL/TLS-certificate during the server-authentication. This enables an attacker to fake the server's identity and to trace the test run by faking the responses. This can hide a faulty configuration to the tester. So, the

HTTP Setup Validator should be used in a secured environment. This correlates with the attack pattern "Man in the Middle" that is described in 4.3.6.

Validation

The test is started by clicking on the "GO!" button. After the test is finished, the results are rendered according to the color configuration. Figure 6.7 shows how the rules of listing 6.4 are interpreted.



HTTP Setup Validator

Menu

File

Filename of datafile: showCaseDataSet.hta.csv

Configuration

Host: <host>

Port: <port>

[✓] Basic HTTP Authentication [✓] SSL

Username: <username>

Password: ••••••••

Test

HTTP Method	HTTP Response	URL Path
GET	200	/teamworks/executeServiceByName
GET	200	/teamworks/fauxRedirect.lsw
GET	200	/teamworks/script/coachNG/dojo/1.8.3/dojo/dojo.js
GET	200	/teamworks/jsonResBundleLoader
POST	200	/teamworks/ajaxCoach
GET	200	/teamworks/tm_process_finished.lsw
POST	401	/rest/bpm/wle/v1/service/1.19fa95ef-9199-4174-b6c1-31...
GET	404	/favicon.ico

Reset GO!

Figure 6.7: HTTP Setup Validator after a Validation

6.2.2 Technical Perspective

This section documents how the execution of a validation run works internally. This is important to document the general behavior of the HTTP Setup Validator. Figure 6.8 shows what happens during a validation run. It is to mention that this sequence diagram shows a reduced perspective. This reduction is needed for to improve legibility.

Generally, the HTTP Setup Validator works as described here, when a validation run is started: The event-listener of the "GO!" button creates an instance of the class *HTTPResponseAgent* and configures it. This object holds the test configuration and provides a *performTest()* method.

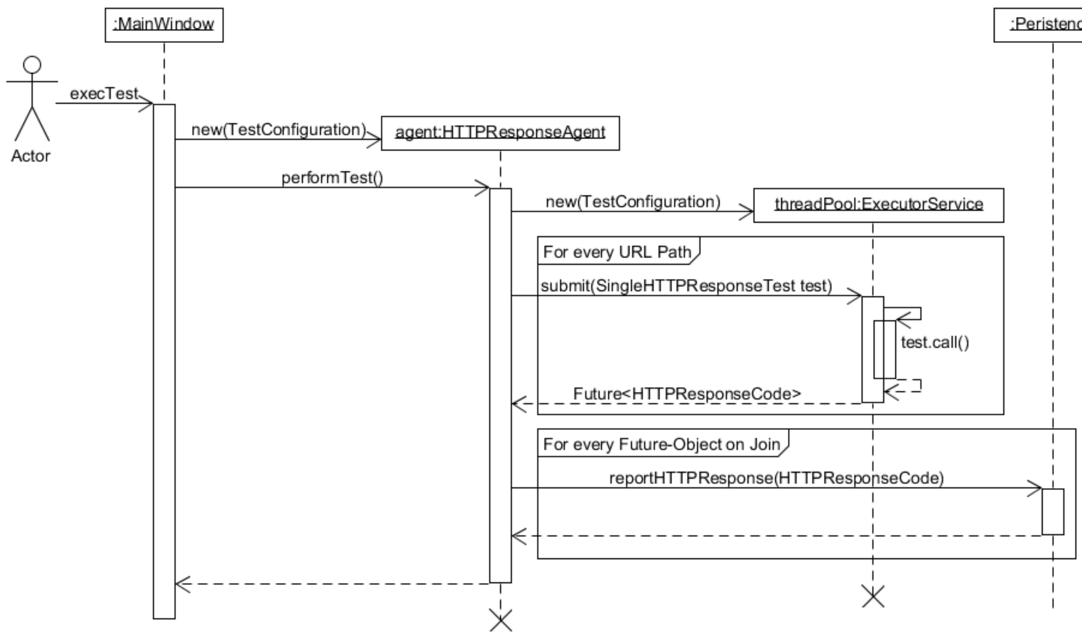


Figure 6.8: Sequence Diagram of a Validation

When this method is called every test-URL is used to create an instance of *SingleHTTPResponseTest*. This class implements the interface *Callable*. Now, these objects are submitted to an instance of *ExecutorService*, which works generally as a threadpool. After the single tests are finished, the received responses are reported to the persistence.

Finally, it is to mention that the HTTP-requests, which implicate a transmission of data in the client to server direction, have empty bodies. The HTTP Setup Validator does not generate test-content.

7 Summary and Outlook

Large companies run appropriate IT departments which are staffed with qualified IT specialists. These companies run large and well secured computer systems, which are combining a big bunch of technologies.
Modern Business Process Management Systems like IBM Business Process Manager address also smaller companies. These companies run just a reduced IT department to manage the existing systems. In the past, such customers charged external consultants to introduce automated Business Process Management to the company. This includes the development of appropriate user interfaces, process modeling, and system integration etc.
Modern products like IBM Business Process Manager enable those customers to do these things by themselves. This increases flexibility, independence, and it saves money.

The larger companies integrated their Business Process Management Systems often with their public available web interfaces. Smaller companies intend to do this too. This raises security questions, because these companies need to be enabled to secure such public available systems.
This thesis identifies a collection of attack patterns (4.3), which could be used to attack such a system if it is not secured properly. Each identified pattern is described in general, specified with an example, and enriched with prevention recommendations. This pattern collection can be used by software architects as well as customers as a security guideline for securing such configurations.

To secure the public available interfaces, the network traffic is redirected via transport level firewalls as well as web application firewalls. This is done to determine exactly what is available to the public networks and to filter the traffic.
Chapter 5 presents an example how the needed information can be mined to configure a simple web application firewall to secure such a system. This analysis is done with IBM Business Process Manger v.8.5.0.1 Standard. To process the results of this analysis, two small helper tools were implemented, which are documented in chapter 6.
As result, a structured collection of URL-paths is provided. A web application firewall that filters the HTTP-traffic based on URL-paths has to be configured to enable these URL-paths in a system that uses IBM Business Process Manager.
This examination follows an abstract analysis method that can be reused to mine for other web application firewall configurations.

If a customer decides to run a public available Business Process Management System as it is discussed here, he probably decides to reuse the old pattern of charging external consultants.
The modern Business Process Management products evolved from a type of technology that was just usable for high qualified IT specialists. This evolution is driven by enabling the customer to build the system he needs. The simplification of the securing of the public interfaces of a modern Business Process Management System is probably needed to preserve its market attractiveness. This is because smaller customers probably expect to be enabled to build such a system by themself.
This simplification can advance by handling the actual problems with the configuration. For example, configurations for web application firewalls could be exportable or the BPMN standard could be extended according to the needs of this scenario.

Bibliography

[AAD^{+}07] A. Agrawal, M. Amend, M. Das, M. Ford, C. Keller, M. Kloppmann, D. König, F. Leymann, R. Müller, G. Pfau, et al. WS-BPEL extension for people (bpel4people), version 1.0, 2007. *Organization for the Advancement of Structured Information Standards (OASIS)*, 2007. (Cited on page 58)

[AAG^{+}13] D. Ahukanna, V. P. A. de Almeida, V. Gucer, S. Narain, B. Pham, M. Salem, M. Warkentin, J. K. Wood, Z. Q. Xie, C. Zhang. RedBook - IBM Business Process Manager V8.0 Production Topologies. Technical report, IBM, 2013. URL `http://www.redbooks.ibm.com/abstracts/sg248135.html`. (Cited on pages 24, 25, 30 and 31)

[ABB^{+}13] F. Albertoni, J. Bajerski, D. Barillari, L. Cada, S. Hanson, G. L. Huang, R. Jain, G. K. Mendes, C. Mierlea, S. Narain, S. Pinto, J. Ricciuti, C. Sadtler, C. Steege. RedBook - WebSphere Application Server V8.5 Concepts, Planning, and Design Guide. Technical report, IBM, 2013. URL `http://www.redbooks.ibm.com/abstracts/sg248022.html`. (Cited on pages 32 and 34)

[BAB^{+}14] A. Buecker, S. Arunkumar, B. Blackshaw, M. Borrett, P. Brittenham, J. Flegr, J. Jacobs, V. Jeremic, M. Johnston, C. Mark, G. Marx, S. Van Daele, S. Vereecke. RedBook - Using the IBM Security Framework and IBM Security Blueprint to Realize Business-Driven Security. Technical report, IBM, 2014. URL `http://www.redbooks.ibm.com/abstracts/sg248100.html?Open`. (Cited on pages 35, 36, 37 and 38)

[BJM08] A. Barth, C. Jackson, J. C. Mitchell. Robust Defenses for Cross-site Request Forgery. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pp. 75–88. ACM, New York, NY, USA, 2008. doi:10.1145/1455770.1455782. URL `http://doi.acm.org/10.1145/1455770.1455782`. (Cited on pages 53 and 54)

[CKM^{+}10] L. Clément, D. König, V. Mehta, R. Mueller, R. Rangaswamy, M. Rowley, I. Trickovic. Web Services - Human Task (WS-HumanTask) Specification Version 1.1. *Organization for the Advancement of Structured Information Standards (OASIS)*, 2010. (Cited on pages 21, 22 and 40)

[HH94] D. Hollingsworth, U. Hampshire. Workflow management coalition the workflow reference model. *Workflow Management Coalition*, 68, 1994. (Cited on pages 12, 15, 16 and 17)

[KKL$^+$05] M. Kloppmann, D. Koenig, F. Leymann, G. Pfau, A. Rickayzen, C. von Riegen, P. Schmidt, I. Trickovic. Ws-bpel extension for people–bpel4people. *Joint white paper, IBM and SAP*, 183:184, 2005. (Cited on page 20)

[Kol14] N. Kolban. Kolban's Book on IBM Business Process Management. PDF document for to download on the authors personal website, 2014. URL http://neilkolban.com/ibm/. Neil Kolban is an IBM enigneer and his book is constantly refreshed about the understanding of IBM BPM. (Cited on page 24)

[Ley12] F. Leymann. Business Process Management. Lecture, 2012. Institute of Architecture of Application Systems. (Cited on pages 19 and 20)

[LR00] F. Leymann, D. Roller. *Production Workflow: Concepts and Techniques*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000. (Cited on pages 11, 12 and 13)

[LRST00] F. Lau, S. Rubin, M. Smith, L. Trajkovic. Distributed denial of service attacks. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 3, pp. 2275–2280 vol.3. 2000. doi:10.1109/ICSMC.2000.886455. (Cited on page 48)

[UP14] S. T. Uday Pillai, Nirmal Patil. RedBook - Business Process Management Deployment Guide - Using IBM Business Process Manager V8.5. Technical report, IBM, 2014. URL http://www.redbooks.ibm.com/abstracts/sg248175.html. (Cited on page 23)

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature