universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Attack Trees zur Analyse von Schwachstellen in WfMS“

verfasst von / submitted by

Amy Daublebsky-Sterneck BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Diplom-Ingenieurin (Dipl.-Ing.)
Master of Science (MSc.)

Wien, 2016 / Vienna 2016

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 926
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Wirtschaftsinformatik
Betreut von / Supervisor:	Univ.-Prof. Dipl.-Math. Dr. Stefanie Rinderle-Ma

Eidesstattliche Erklärung

Ich erkläre hiermit eidesstattlich, dass ich die vorliegende Masterarbeit selbständig und ohne fremde Hilfe verfasst, und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Weiters versichere ich hiermit, dass ich die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission weder im In- noch im Ausland vorgelegt und auch nicht veröffentlicht.

Datum Unterschrift

Danksagung

Herzlichst bedanken möchte ich mich an dieser Stelle bei meiner Betreuerin Frau Prof. Dr. Stefanie Rinderle-Ma, die es mir ermöglicht hat, längere Zeit an diesem Thema zu arbeiten, die mir und meiner Kollegin Erlaubnis zu dem anfänglichen und sehr wertvollen Workshop gegeben hat und immer für Fragen und Meetings zur Verfügung stand.

Für die Unterstützung bei meiner Diplomarbeit möchte ich mich außerdem bei meiner Kollegin Sabine Gottwald bedanken, die mich von Anfang an begleitet hat, die für die Idee des Workshops verantwortlich war, gute Ratschläge gegeben hat und mich stets bis zum Schluss motiviert hat.

Mein besonderer Dank geht auch an meinen Freund Marcel, der mir mit seinem umfangreichen Wissen zahlreiche und wertvolle Anregungen gegeben hat und mich auch in stressigen Zeiten mit Rat zur Seite stand.

Natürlich danke ich auch meiner Familie, die teilweise auch aus weiter Entfernung an mich geglaubt und mich motiviert hat.

Herzlicher Dank geht auch an meine Freunde, Kolleginnen und Kollegen, die oft ermutigende Worte fanden und mich mit interessierten Fragen zu dem Thema auf weitere Ideen brachten.

Amy Daublebsky-Sterneck, Juni 2016

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Problemstellung . . . 1

1.2 Verwandte Arbeiten . . . 2

1.2.1 Sicherheit in Process-Aware Information Systems (PAIS) . . . 2

1.2.2 Methoden der Sicherheitsanalyse . . . 3

1.3 Ziel der Arbeit . . . 3

1.4 Methodisches Vorgehen . . . 4

2 Grundlagen **7**

2.1 Geschäftsprozess vs. Workflow . . . 7

2.1.1 Geschäftsprozess . . . 7

2.1.2 Workflow . . . 8

2.2 Workflow Management System . . . 8

2.3 Workflow Referenz Modell . . . 9

2.4 Sicherheitsanforderungen in WfMS . . . 10

2.5 Sicherheitskontrollen . . . 11

2.6 Bedrohungen und Gefahren . . . 11

2.6.1 Angriff und Angreifer . . . 11

2.6.2 Schwachstelle . . . 12

2.7 Schwachstellenanalyse und Methoden . . . 13

2.7.1 Die Attack Tree Methode . . . 14

2.7.2 Erzeugung von Attack Trees . . . 14

2.7.3 Die ATLIST Methode . . . 15

2.7.4 Erzeugung von ATLIST Bäumen . . . 15

3 Arbeitsumgebung **17**

3.1 Activiti BPM Plattform . . . 17

3.2 Standard BPMN 2.0 . . . 18

3.2.1 Grafische Repräsentation . . . 18

3.2.2 Eclipse BPMN 2.0 Erweiterung . . . 19

3.3 Voraussetzungen . . . 20
3.3.1 Technische Anforderungen und Konfiguration . . . 21
3.3.2 Konfiguration Java Development Kit . . . 21
3.3.3 Konfiguration Apache Tomcat . . . 22
3.3.4 Konfiguration Activiti . . . 22
3.3.5 Konfiguration Apache Maven . . . 22
3.3.6 MySQL Datenbank . . . 22
3.4 Eclipse Activiti Projekt - Die Umsetzung . . . 23
3.4.1 Kritische Faktoren . . . 24
3.4.2 User und Groups . . . 24
3.4.3 Webservice . . . 25
3.4.4 DB Konzept . . . 26
3.4.5 Prozessaufruf . . . 27
3.4.6 Zusätzliche Funktionalitäten . . . 27
3.4.7 Geschäftsprozess-Gesamtbild . . . 28

4 Schwachstellenanalyse **30**
4.1 Methodologie . . . 30
4.1.1 Angriffsabsichten und Kategorie . . . 30
4.1.2 Häufige Angriffsarten . . . 32
4.2 Angreifer und Angriffsebenen . . . 33
4.2.1 Unterschiedliche Blickwinkel . . . 34
4.2.2 Teilhabende Entitäten . . . 34
4.2.3 Bedingungen und Einschränkungen . . . 35
4.3 Angriffsszenarien und Analyse . . . 37
4.3.1 Bestimmung der Wurzelknoten . . . 37
4.3.2 Strukturiertes Vorgehen . . . 38
4.4 Attack Tree 1 - Zugang verschaffen . . . 39
4.4.1 SQL injections . . . 40
4.4.2 Web-Seite nachahmen - Unsichtbaren Berührungspunkt platzieren . . . 42
4.4.3 Password erraten . . . 43
4.4.4 Daten aufzeichnen und wiederherstellen . . . 44
4.5 Attack Tree 2 - Datenbank . . . 44
4.5.1 Falsche Parameter und fehlende Validierung . . . 45
4.5.2 Verfügbarkeit . . . 46
4.6 Attack Tree 3 - Ressourcen . . . 47
4.6.1 Auslastung . . . 47
4.6.2 Ressourcen direkt angreifen . . . 50
4.6.3 Phishing . . . 52

4.7 Attack Tree 4 - Benutzerrechte . . . 53
4.7.1 Prozessdefinition - Rechte Implementierungen angreifen . . . 53
4.7.2 Rechte umgehen - systematische URL Manipulation . . . 54
4.7.3 Benutzerrechte in der Datenbank . . . 56
4.8 Attack Tree 5 - Instanz . . . 57
4.8.1 Dienstverweigerung . . . 58
4.8.2 Unterbrechung auslösen . . . 58
4.8.3 Verzweigungen erzwingen . . . 59
4.8.4 Aufgaben entziehen . . . 60
4.8.5 Abnormale Situationen . . . 62
4.9 Attack Tree 6 - Der Insider . . . 64
4.9.1 Passwort ablesen . . . 65
4.9.2 Absichtliche Fehler . . . 65
4.9.3 Fehler in der Konfiguration . . . 66
4.9.4 OWASP Zed Attack Proxy - ZAP . . . 66
4.9.5 Ausgelagerte programmatische Aufgaben . . . 69
4.10 Erkenntnisse im Überblick . . . 69
4.11 Nützlichkeit von Scanner . . . 71
4.12 Vergleich und Gemeinsamkeiten mit der ATLIST Methode . . . 72
4.13 Auffälligkeiten im laufenden Betreib . . . 72
4.14 Übertragbarkeit der Ergebnisse . . . 73

5 Zusammenfassung **75**

6 Fazit und Ausblick **77**

7 Anhang **85**

1 Einleitung

Aus heutiger Sicht ist die IT-Unterstützung in einem Unternehmen kaum weg zu denken. Unzählige Prozesse sind involviert und beeinflussen das Funktionieren innerhalb und außerhalb der Unternehmensgrenzen. Mit Hilfe eines Workflow Management Systems (WfMS) werden unterschiedlichste Arbeitsabläufe definiert, koordiniert und verwaltet. Diese Steuerung in nahezu allen Unternehmensbereichen wird transparent gehalten und involviert sowohl Menschen als auch IT-Systeme. Dabei werden Aufgaben erstellt, Daten von verschiedenen Rollen verarbeitet, Informationen generiert und über Abhängigkeiten miteinander in Verbindung gebracht. WfMS sind ein großer Bestandteil vieler Unternehmenswelten und tragen zur Wertschöpfung bei. Aus diesem Grund ist es auch essentiell die Sicherheit eines solchen Systems in Betracht zu ziehen, da neben optimalen Prozessstrukturen auch Daten und Informationen den Wert eines Unternehmens ausmachen. Zwar lassen sich keine Unmengen von bekannten Sicherheitslücken im Bereich des Einsatzes von WfMS nachlesen, was jedoch nicht ausschließen soll, dass keine Schwachstellen in diesem speziellen Bereich vorhanden sind. Deshalb ist an dieser Stelle eine genauere Betrachtung nicht unbedeutend.

In dieser Arbeit wird im Speziellen ein ausgewähltes WfMS auf Schwachstellen analysiert und dabei besonders auf folgende Fragen Wert gelegt: Auf welche Art und Weise können WfMS gefährdet werden. Welche Maßnahmen zur Sicherheitskontrolle müssen umgesetzt sein, um gegen mögliche Angriffe gewappnet zu sein? Wie kann das WfMS umfassend auf ein mögliches Vorhandensein von Schwachstellen überprüft werden? Nach diesen Fragen ist zunächst auch nicht belanglos zu erfahren, welche Anforderungen im Speziellen an die Sicherheit von WfMS gestellt sind. Sicherheitsmaßnahmen und sicherheitsrelevante Aspekte werden oftmals dokumentiert aber nur unzureichend umgesetzt. Der Entwickler des Systems sowie teilhabende Programmierer haben dafür Sorge zu tragen, dass bereits während der Designphase auf etwaige Gefahren Rücksicht genommen und das System auch schon im Anfangszustand gründlich getestet wird. Programmierfehler sind unter anderem die häufigste Ursache vieler Schwachstellen.

In den folgenden Abschnitten werden zuerst das zugrundeliegende Problem, die Zielsetzung dieser Arbeit und anschließend das methodische Vorgehen vorgestellt.

1.1 Problemstellung

Wie sicher sind WfMS? Aus [25], einer umfassenden Studie von über vierhundert Publikationen zum Thema Sicherheit und Sicherheitskontrollen in prozessorientierten Informationssystemen, geht hervor, dass ein grundlegendes und vor allem gemeinsames Verständnis für das Thema Sicherheit in prozessorientierten Informationssystemen fehlt und, dass es aufgrund der Fülle von Informationen erschwert ist, den Stand der Dinge in diesem Bereich eindeutig zu bestimmen. Eines der wesentlichen Resultate ist die Bestimmung von zwölf Sicherheitskontrollen, welche den folgenden fünf Kategorien zu zuordnen sind:

Das *Sicherheitskonzept* umfasst sicherheitsrelevante Kontrollen bei der Modellierung von Geschäftsprozessen.

Die *Zugriffskontrollen* sind die am häufigsten eingesetzten Kontrollmaßnahmen. Rollenbasierte Zugriffskontrollen stellen eine beschränkte Menge an spezifischen Berechtigungen dar.

Die *Verifizierung* der Geschäftsprozesse ist wesentlich, da bei mangelnder Korrektheit, Widerspruchslosigkeit und Regelkonformität Sicherheitsprobleme bestehen können.

Die *Fehlerbehandlung* bei Problemen während der Prozesslaufzeit verringert die Gefahr eines unerlaubten Zugriffs.

Die *Anwendung* umfasst sämtliche Implementierungen von Funktionen der Sicherheit.

Eine daraufhin eigene Literaturrecherche hinsichtlich kompromittierter Informationssysteme führte zu dem Ergebnis, dass es an Beschreibungen und Veröffentlichungen bezüglich des Erfolgs eines Angriffs auf ein WfMS mangelt. Es gibt kaum Informationen über Vorfälle, in denen eine Schwachstelle eines WfMS die Ursache eines erfolgreichen Angriffs war, was jedoch nicht bedeuten soll, dass Schwachstellen in WfMS ausgeschlossen sind. Mögliche Gefahren müssen beachtet und überprüft werden, denn es gilt: gerät das System in falsche Hände, kann es zu Schäden und Einbußen kommen - Geschäftsprozesse können manipuliert und möglicherweise nicht mehr, wie gewohnt, ablaufen, was wiederum zu operativen Verzögerungen oder Totalausfälle führen kann. Darüber hinaus können durch einen unbefugten Zugriff die involvierten Geschäftsdaten missbraucht werden. Geheime Geschäftsvorgänge auch in kooperativen Prozessen, die das Unternehmen zu seinem Erfolg verschaffen, können an die Öffentlichkeit bzw. an den Konkurrenten geraten. Das Thema Sicherheit ist somit auch hier von Bedeutung und wie auch das *Wall Street Journal* betont , gelten Cyber-Attacken als „Risikofaktoren" [23]. Aber nicht nur der Einsatz von unsicheren WfMS birgt Risiken für das Unternehmen, sondern auch die Tatsache, dass es allgemein unmöglich ist, ein System vor neuen Schwachstellen und neuen Angriffsarten zur Gänze zu schützen. [18]

Das Problem ist somit nicht die Unkenntnis über notwendige Sicherheitsmaßnahmen und das Vorhandensein von Gefahren bei schwachen oder fehlenden Sicherheitsvorkehrungen, sondern das bisherige Fehlen von systematischen Analysen zur ganzheitlichen Betrachtung der sicherheitsrelevanten Aspekte in prozessorientierten Informationssystemen.

1.2 Verwandte Arbeiten

1.2.1 Sicherheit in Process-Aware Information Systems (PAIS)

Das Thema „Sicherheit" spielt eine bedeutende Rolle in Process-Aware Information Systems (wie WfMS), in denen oftmals eine Vielzahl von Menschen an Prozessbearbeitungen beteiligt sind und entsprechend eine Menge an Informationen teilweise auch automatisiert verarbeitet werden. Ein WfMS ist ein Beispiel für solch ein Softwaresystem, welches Workflows verwaltet und ausführt, während Menschen, Anwendungen und Informationen auf Basis eines Modells involviert sind.[3]

Besonders in Prozess-orientierten Systemen werden unterschiedliche Sicherheitsanforderungen notwendig, verglichen zu Informationssystemen, in denen keine Prozessinstanzen im Zentrum stehen und wenige Menschen Zugriff auf sensible Informationen haben. Die Wichtigkeit wird besonders in der Untersuchung von [28] verdeutlicht, in der Sicherheitskontrollen und Regularien aber auch rechtliche Anforderungen untersucht wurden. Die Informationssicherheit bedeutet die Sicherheit und den Schutz von Informationen und die Einhaltung von Sicherheitsanforderungen, wie die Gewährleistung der Integrität und Vertraulichkeit, dient dazu, diesen Schutz aufrecht zu erhalten. Entsprechend kann ein WfMS erst dann als ein „sicheres" System betrachtet werden, wenn die sicherheitspolitischen Maßnahmen auch umgesetzt sind. Dazu zählen ebenso die Umsetzungen der Zugriffskontrollmechanismen aber auch die der Autorisierungsmechanismen. Aus [28] geht ebenso hervor, dass „Sicherheit" im Bereich der Prozess-orientierten Systeme nicht eindeutig definiert ist. Sie betrifft einerseits Bestimmung der sicherheitspolitischen Maßnahmen,

die maßgebend sind für die Entwicklung und Ausführung von Prozessen. Andererseits wird sehr viel Wert auf Autorisierung und Zugriffskontrollmechanismen gelegt. Erst durch korrekte Implementierung kann ein „sicheres" WfMS entwickelt und gewährleistet werden. [14]

Auch für die vorliegende Arbeit wird, noch bevor die eigentliche Analyse stattfindet, die Umsetzung von Authentifizierung, Autorisierung und Zugriffskontrollen untersucht und entsprechende (fehlende) Implementierungen realisiert. In diesem Bereich liegt der Fokus bei Zugriffskontrollmechanismen, die in die Prozessdefinition separat integriert werden müssen, da dieser Sicherheitsaspekt für das vorliegende WfMS nicht in einem vordefinierten Muster vorgesehen ist. Im Gegensatz dazu stehen die Konfigurationsmöglichkeiten der sogenannten *Authorization Constraints* im WfMS *AristaFlow*[1] dem Anwender zur Verfügung, sodass Zugriffskontrollen direkt bei Modellierung des Workflows bestimmt und integriert werden können [20].

1.2.2 Methoden der Sicherheitsanalyse

Eine Risikoanalyse gilt als weitere Methode der Sicherheitsanalyse. Aus Gründen, wie das Vorhandensein verschiedener Sicherheitsanforderungen in komplexen Unternehmensstrukturen und mangelnde Standardvorschläge für Maßnahmen der Sicherung, ist die Analyse der IT-Sicherheit besonders auch in einer Organisation sinnvoll . Durch Analyse der Risiken kann beispielsweise der Bedarf dieser Maßnahmen ermittelt werden, wodurch wiederum Sicherheitskonzepte und Sicherungsmaßnahmen erzielt werden können[16]. ISO 17799 und BS 7799 bieten eine Methode an, um Risiken zu identifizieren, die vor allem die Sicherheit der Informationen betrifft. Nach Bestimmung der Unternehmenswerte werden Risiken, Bedrohungen und Schwachstellen festgestellt, um in weiterer Folge ihre Auswirkung (auf die Schutzziele) abzuschätzen. [38] Zu weiterführenden Sicherheitsanalysen gelten neben der Risikoanalyse auch die sogenannten Penetrations Tests, bei denen die Prüfung auf Sicherheitsschwachstellen aus der Sicht eines tatsächlichen Angreifers unternommen wird [8]. Sämtliche (relevante) Informationen liegen dabei nicht vor, weshalb diese Tests auch „Black-Box-Tests" bezeichnet werden (im Gegensatz dazu die sogenannten „White-Box-Tests", bei denen für die Überprüfung des Systems alle relevanten Informationen zur Verfügung stehen). Der echte Penetrations Tester agiert mit Mitteln und der Motivation eines echten Angreifers, ohne dabei gegebenenfalls auf nicht vertrauenswürdige Werkzeuge zu verzichten. Der Unterschied zu einem normalen Sicherheitstest besteht darin, dass nach der Untersuchung von vorhandenen Informationen versucht wird, auch unbekannte Schwachstellen zu ermitteln (durch Kreativität und Wissen) und in weiterer Folge die Sicherheitslücken auch als solche aufzuzeigen und zu beweisen. [35]

Für die vorliegende Arbeit ist eine systematische Analyse von Schwachstellen in einem WfMS vorgesehen, wobei sämtliche Sicherheitsmaßnahmen überprüft werden. Die Untersuchung beschränkt sich somit auf ein bestimmtes System, welches jedoch mit weiteren Komponenten (und Personen) in Verbindung steht und somit ebenso zur gesamten IT-Sicherheit einer Organisation beiträgt. Die konkrete Zielsetzung wird im nächsten Abschnitt vorgestellt.

1.3 Ziel der Arbeit

Das Ziel dieser Arbeit ist eine Analyse eines ausgewählten WfMS auf Schwachstellen, um Antworten auf sicherheitsrelevante Fragen zu erhalten. Wie sicher ist das zugrundeliegende System vor unbefugtem Zutritt und welche Rolle spielt die Sicherheit in Bezug auf die beteiligten Systemkomponenten? Sind grundlegende Sicherheitsmaßnahmen umgesetzt, sodass Prozessbeteiligte keine potentielle Bedrohung darstellen können? Sind einzelne Prozessinstanzen von Gefahren und Bedrohungen ausgenommen? Eine mögliche Schwachstelle kann sich in vielen Bereichen

[1] Forschungsprojekt dessen Ergebnisse in die kommerzielle Business Process Management Suite mündete. Details sind unter www.aristaflow.com abrufbar, letzter Zugriff im Mai, 2016

befinden, die sich meist erst durch manuelle und oder automatisierte Tests ausfindig machen lässt. In dieser Arbeit wird eine Schwachstellenanalyse mithilfe von zuvor erarbeiteten Angriffsszenarien, dargestellt durch sogenannte „*Attack Trees*", also Angriffsbäume, durchgeführt und die dabei gewonnenen Ergebnisse abschließend zusammengefasst. Die Angriffsziele und Szenarien werden dabei ohne einer Kenntnis über (bekannte) Schwachstellen erarbeitet und umfasst das Gesamtsystem inklusive der teilhabenden Systeme und Komponenten. Auf diese Weise soll eine möglichst umfassende und systematische Analyse von Schwachstellen im WfMS durchgeführt werden.

In der parallel erstellten Arbeit von Sabine Gottwald[2] wird ebenso eine systematische Schwachstellenanalyse vorgestellt und es handelt sich um das gleiche WfMS, wobei die Angriffsszenarien anhand der ATLIST Tree Methode, auch „*Attentive Listener*" genannt, erstellt werden. Der Unterschied in dieser Arbeit liegt vorrangig darin, dass die ebenfalls baumorientierten Szenarien mit einer unterstützenden Anleitung erstellt werden und bekannte Schwachstellen die Grundlage dafür bilden (im Detail in Abschnitt 2.7.3).

Aus methodischer Sicht betrachtet, soll mit dieser Zusammenarbeit in Erfahrung gebracht werden, ob durch die Anwendung verschiedener Methoden im Endeffekt die gleichen oder doch unterschiedliche Schwachstellen gefunden und Erkenntnisse gewonnen werden. Aus diesem Grund wurde letztendlich auch entschieden, das gleiche WfMS als eine Art Grundlage und gemeinsame Ausgangslage für die systematische Analyse zu verwenden. Besonders aufgrund der in [28] beschriebenen Schwachstellenanalyse wurde das Anwenden zweier verschiedener Methoden, nämlich Attack Tree und ATLIST Tree, als zielführend erachtet, um den Effekt und auch die Durchführbarkeit der beiden Methoden zu vergleichen. Im Endeffekt wird das WfMS umfassend und mit zwei unterschiedlichen Methoden unabhängig voneinander auf Schwachstellen analysiert.

Basierend auf obiger Zielbeschreibung lassen sich folgende Forschungsfragen ableiten:

- Welche Vorgehensweise dient einer umfassenden Analyse von Schwachstellen in WfMS?
- Mit welchen Methoden wird die systematische Schwachstellenanalyse in WfMS ermöglicht?

In diesem Zusammenhang ergibt sich eine weiterführende Frage hinsichtlich der "Übertragbarkeit"der Resultate:

- Wie können die Resultate der Schwachstellenanalyse eines WfMS auf andere WfMS übertragen werden?

Der folgende Abschnitt gibt einen Überblick über den Aufbau und fasst die Vorgehensweise in dieser Arbeit zusammen.

1.4 Methodisches Vorgehen

Um eine Schwachstellenanalyse anhand vorher definierter Attack Trees durchführen zu können, werden in erster Linie die Systemkomponenten bestimmt und aufgesetzt. Ein detaillierter Installationsvorgang und die notwendige Konfiguration sämtlicher Komponenten wird in Kapitel 3.3 festgehalten.

Anhand von definierten System- und Prozessanforderungen wird das Prozessmodell erstellt, um anschließend das gesamte Szenario mit der Unterstützung und ausgehend von Attack Trees auf Schwachstellen hin zu analysieren. Die Mindestanforderungen an das Prozessmodell und dem

[2] Diplomarbeit von Sabine Gottwald mit dem Titel: „ATLIST Tree zur Analyse von Schwachstellen in WfMS", 2016

Workflow, sowie kritische Faktoren beeinflussen die Überlegungen bei der Konstruktion der Attack Trees. Diese fokussieren sich auf die angebundenen System-Komponenten und werden in Anlehnung an [11] und Schneier[3] kreiert.

Die konstruierten Angriffsbäume bilden zudem den anfänglichen Soll-Zustand und dienen dem Analysevorhaben, um zu überprüfen, ob relevante und notwendige Sicherheitsmaßnahmen für das WfMS umgesetzt sind. Die Durchführung der Angriffsszenarien und Testergebnisse (entspricht dem Ist-Zustand) werden schlussendlich mit den Angriffszielen gegenüber gestellt. Die nachfolgende Darstellung (Abbildung 1.1) fasst die Vorgehensweise dieser Arbeit grob zusammen.

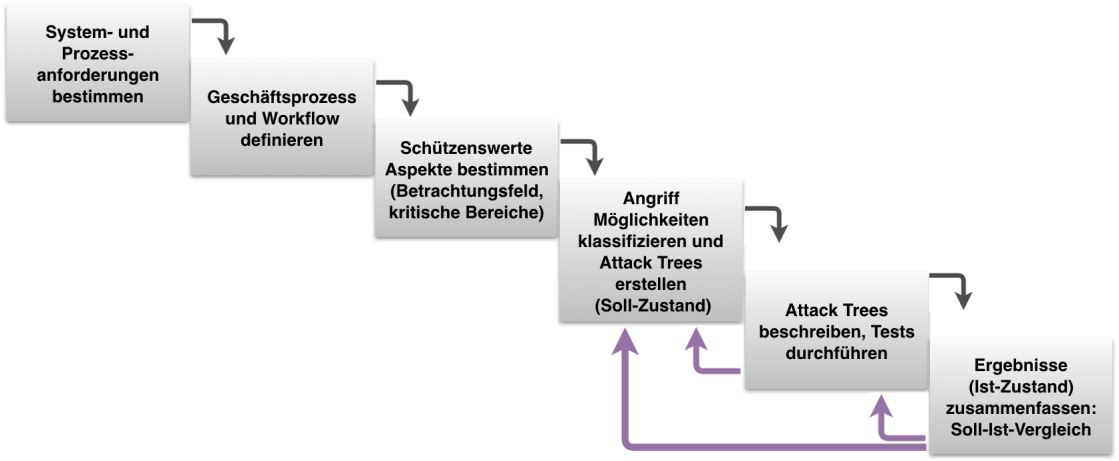

Abbildung 1.1: Vorgehensmodell - Aufbau der Arbeit (eigene Darstellung)

Die Attack Trees werden im Einzelnen untersucht, sowie Versuche und Tests durchgeführt, wodurch wiederum Erweiterungen oder Anpassungen an dem ursprünglichen Baum entstehen können. Auch die Tests führen zu Ergebnissen, die ebenso zu weiteren Testmöglichkeiten (oder Bedarf) führen können, aber auch zu Anpassungen und Erweiterungen von bereits bestehenden Attack Trees.

Nachdem die Problemstellung, die Ziele und das Vorgehen in dieser Arbeit vorgestellt wurden, folgt im zweiten Kapitel eine Beschreibung grundlegender Begriffe rund um Geschäftsprozesse, Workflows und Bedrohungen im Allgemeinen. Sie dienen einleitend für ein besseres Verständnis über die vorherrschende Thematik und der Analyse von Schwachstellen im Bezug auf WfMS.

Um eine praxisnahe Untersuchung durchzuführen, wird das WfMS Activiti BPM als Grundlage der Analyse und Durchführung installiert und wesentliche Schritte der Konfiguration notweniger Komponenten dokumentiert (siehe Kapitel 3 Arbeitsumgebung). Dabei werden vorbereitend wesentliche Aspekte des Prozessmodells und des Workflows festgehalten (beteiligte Entitäten, Ablauf, Abhängigkeiten, Aufrufe, Kontrollen), um anschließend einen Überblick über die Umsetzung und den Workflow im Gesamtzusammenhang zu erhalten.

Die Entscheidung, ein konkretes WfMS für die bevorstehende Analyse aus zu wählen, wurde gefällt, nachdem zunächst eine Recherche über potentielle WfMS durchgeführt wurde, in der sowohl kommerzielle Systeme (AristaFlow BPM[4], WebSphere[5]), als auch Open Source (Activiti BPM, jBPM[6]) sowie der Forschungsprototyp CPEE[7] betrachtet wurden. Für das frei verfügba-

[3] Attack Trees wurden von dem Sicherheitsspezialisten Bruce Schneier eingeführt: https://www.schneier.com/cryptography/archive.html, letzter Zugriff im April, 2016
[4] AristaFlow BPM Suite: http://www.aristaflow.com/de/bpm-suite/ueberblick.html, letzter Zugriff im Mai, 2016
[5] IBM WebSphere MQ: http://www-03.ibm.com/software/products/de/ibm-mq, letzter Zugriff im Mai, 2016
[6] jBPM von JBOSS: http://www.jbpm.org/, letzter Zugriff im Mai, 2016
[7] Cloud Process Execution Engine: http://cpee.org/, letzter Zugriff im Mai, 2016

re „Activiti BPM" wurde letztendlich entschieden aufgrund einer zahlreich vorhandenen Dokumentation, die eine sehr gute Unterstützung bei der Implementierung versprach, der sehr aktiven Benutzerforen, aus denen auch für „Anfänger" verständliche Sachverhalte entnommen werden konnten und letztendlich auch aufgrund vorangegangener Erfahrungswerte im anfänglichen Umgang mit diesem konkreten WfMS.

Das Kapitel 4 Schwachstellenanalyse befasst sich mit der Analyse des vorbereiteten Gesamtsystems auf etwaige Schwachstellen und Fehler. Dabei werden Analyseebenen mit beteiligten Rollen (Personen) und Systemen hervorgehoben sowie die zugrundeliegende Attack Tree Methode, mit der die anschließende Schwachstellenanalyse eingeleitet wird. Sechs Attack Trees bilden den Umfang und lassen das WfMS aus unterschiedlichen Blickwinkeln betrachten. Ausgewählte Angriffspfade werden analysiert und gegebenenfalls auch unter Zuhilfenahme von technischen und frei zugänglichen Hilfsmittel umgesetzt. Durch die zusammenfassende Soll-Ist Gegenüberstellung werden die Ergebnisse hinsichtlich der vorherrschenden Systemsicherheit nochmals kurz aufbereitet und das Kapitel mit weiteren Erkenntnissen, die während der Arbeit und Untersuchung des WfMS und der Prozesse entstanden sind, abgeschlossen.

Das abschließende letzte Kapitel sechs dient der Zusammenfassung der wesentlichen Erkenntnisse der Analyse sowie einem Ausblick auf mögliche und weiterführende Untersuchungsfelder im Bereich der Schwachstellen und Sicherheit in Workflow Management Systeme, die aufgrund dieser Arbeit entstanden sind.

2 Grundlagen

In diesem Kapitel der Grundlagen wird aufbauend zunächst ein Überblick über die wichtigsten Begriffe rund um die Themen Prozesse und Workflow Management Systeme und ihre Zusammenhänge zueinander verschafft. Dieses Verständnis dient als Basis für den Hauptteil dieser Arbeit, der Schwachstellenanalyse eines WfMS. Nach Anforderungen an und Bedrohungen gegen derartige IT-Systeme werden die Baum-orientierte Methoden vorgestellt, die für eine Planung und Durchführung derartiger Analysen eingesetzt werden können.[29]

2.1 Geschäftsprozess vs. Workflow

Diese beiden Begriffe „Geschäftsprozess" und „Workflow", liegen zwar eng beieinander, jedoch gibt es Unterschiede je nach Betrachtungsweise: die Unternehmenssicht und die technische Sicht. Ein Geschäftsprozess aus der Unternehmenssicht *beschreibt* Prozesse mit ihren Kern- und Hilfsprozessen. Während der Modellierung eines Geschäftsprozesses liegt der Fokus entsprechend auf der Wirtschaftlichkeit und der Dokumentation. Hingegen aus der technischen Sicht betrachtet, wird ein Geschäftsprozess von einem Workflow teilweise oder komplett *unterstützt*. Dies bedeutet, dass Geschäftsprozesse *automatisiert* und Informationen verarbeitet werden. Die Modellierung eines Workflows fokussiert sich demnach auf die technische Unterstützung des Ablaufs und die technische Realisierung der Zugriffe und Funktionalitäten. [12][29]

2.1.1 Geschäftsprozess

Für den Begriff *Geschäftsprozess* gibt es zwar keine einheitliche aber eine Fülle von Definitionen, doch an dieser Stelle wird zunächst die Bedeutung besonders unter Berücksichtigung des internationalen Standardisierungsgremiums, die Workflow Management Coalition (WfMC)[1], beschrieben. [29]

Bei einem Geschäftsprozess handelt sich um einen *Geschäftsvorfall* mit definiertem Anfang und Ende. Nach einem Ereignis als Initiator folgen eine Reihe von in logischer Verbindung stehenden *Aktivitäten* (oder Aufgaben). Diese Aktivitäten können manuell, teilautomatisiert und oder komplett automatisiert sein. Ein Geschäftsvorfall kann durch Einflussfaktoren, wie personelle und maschinelle Eingaben, unterschiedlichen Verlauf annehmen, da die Verarbeitung einer Eingabe zu einer bestimmten Ausgabe führt, welche für weitere Aktivitäten wiederum als Eingabe bereit steht.[19][6]

Wie [21] beschreibt, sind die Aktivitäten und Funktionen inhaltlich abgeschlossen, dienen zur Erreichung der Unternehmensziele und erbringen eine Wertschöpfung. Ein Geschäftsprozess bezieht sich auf Aufgaben (*tasks*), die Informationen erstellen, auf Informationen zugreifen, diese verarbeiten und verwalten. Beispiele für einen Geschäftsprozess sind eine „Überprüfung der Kreditwürdigkeit in einer Bank" oder die „Bearbeitung einer Reklamation in einem Warenhaus".

[1] Die 1993 gegründete WFMC gilt als primäres Standardisierungsorgan im Bereich des Workflows und entwickelt Standards und Softwarespezifikationen, http://www.wfmc.org/, letzter Zugriff Mai, 2016

2.1.2 Workflow

Ein Geschäftsprozess wird von einem Workflow teilweise oder gänzlich technisch unterstützt. Auch hier orientiert sich der Begriff *Workflow* an die von der WfMC veröffentlichten Beschreibung. Ein Workflow wird definiert als eine Menge von koordinierten Aktivitäten, welche zusammen ein Unternehmensziel erreichen. Die Aktivitäten werden aufgeteilt in einzelne zu bearbeitende Aufgaben, die von Menschen aber auch automatisiert durch Programme oder Systeme ausgeführt werden. Die Aufgaben in einem Workflow stehen in Beziehung zueinander und hängen von einander ab - diese Abhängigkeiten sind spezifiziert als *task dependencies* (Aufgaben-Abhängigkeiten). Es handelt sich bei einem Workflow um eine Verfeinerung des Prozesses „bis auf die organisatorisch-technische Ebene". Wenn es zu einer Ausführung eines Prozesses kommen soll, ist es der Workflow, welcher das *wann*, *wie*, *wer* und die benötigten Ressourcen dazu beschreibt. Dabei ist eines der Hauptziele die Optimierung der Geschäftsprozesse, damit diese beschleunigt werden können sowie die Sicherstellung des korrekten Ablaufs.
Ein ganzheitliches Software-System übernimmt dabei die Überwachung und Steuerung des Workflows: das WfMS. [37][2] Im nächsten Abschnitt werden die Eigenschaften sowie Zusammensetzung eines solchen Systems vorgestellt.

2.2 Workflow Management System

Bei einem WfMS handelt es sich um ein rechnergestütztes System, welches das Bearbeiten von strukturierter und unstrukturierter Vorgänge ermöglicht [15]. Ein WfMS trägt dazu bei die Geschäftsvorgänge „automatisiert" mit der Unterstützung von Rechnersystemen „ablaufen zu lassen" und somit arbeitsteilige Prozesse aktiv zu steuern. Es ist eine Gesamtheit von Werkzeugen, die die Prozessdefinition, die Initiierung von Workflow Engines und die Administration von Workflow Prozessen unterstützen (MfMC). Nach [4] ist das WfMS ein „aktives System, das mit Hilfe von vielen Bearbeitern in mehreren Schritten den Ablauf von Geschäftsprozessen durchführt." Dabei werden die gebrauchten Informationen und Daten, unter Berücksichtigung der Zeit und Reihenfolge, an die zugeordneten Personen weitergeleitet. Auf diese Weise wird ermöglicht, die Aufgabenlösung in mehrere Arbeitsschritte zu teilen, um Schritt für Schritt nacheinander ausgeführt zu werden. Mit einem WfMS wird eine funktionsübergreifende Unterstützung von Prozessen möglich. Die Korrektheit der Prozessausführung kann sichergestellt sowie eine durchgehende Bearbeitungskontrolle ermöglicht werden.[7]

Es gibt zwei grundsätzliche Phasen, die vom traditionellen Workflow-Management ausgehen. In der ersten Phase, auch Definitionsphase bezeichnet, wird die Prozessdefinition erstellt und bearbeitet, woraufhin in der zweiten Phase, die Ausführungsphase, der Prozess zunächst instanziiert und dann erst ausgeführt wird und kontrolliert werden kann. Das Ziel der Automation eines Workflows mittels eines WfMS kann durch die Beantwortung der folgenden vier Fragen verstanden werden: 1) Welche Informationen sind die richtigen Informationen? 2) Auf welche Art und Weise wird diese Information bereitgestellt und bearbeitet? 3) Welche Person erhält und welche Person bearbeitet die Informationen? und 4) Mit welchen Hilfsmitteln und Werkzeuge werden diese Informationen bearbeitet? [40]

Die Activiti Business Process Management Anwendung ist klassisch und transaktionsorientiert, wobei während des Prozesslaufs auch ad-hoc Aufgaben erstellt werden können. Der Ablauf eines Prozesses ist jedoch a-priori bekannt und durch die Prozessdefinition darstellbar. Während des Ablaufs sind Änderungen nicht vorgesehen (obwohl eine Person während des Ablaufs auch einzelne Aufgaben erstellen und erledigen kann, die sonst nicht vorgesehen sind). Grundsätzlich können nach konventioneller Art nur in der Prozessdefinition Änderungen vorgenommen werden, deren Auswirkung jedoch alle folgenden Instanzen betreffen und keine der bereits aktiven. Diese Art betreffen traditionelle WfMS [3].

Auch an dieser Stelle ist für das vertiefende Verständnis die weiterführende Beschreibung der

WfMC nützlich. Das System, welches die Verwaltung von Workflows technisch unterstützt, lässt sich nach der WfMC in folgende fünf Komponenten unterteilten: Die Modellierungs-, Steuerungs-, Überwachungs-, Schnittstellen- und Simulationskomponente.

Die Modellierungskomponente wird für die grafische Beschreibung von Prozessen genutzt. Die daraus resultierende Prozessbeschreibung wird anschließend von der Steuerungskomponente interpretiert und ausgeführt.

Die Steuerungskomponente ist das Herzstück des WfMS: die Workflow Engine. Ihre Aufgabe ist es, aus der Prozessdefinition eine Prozessinstanz (ein bestimmter Fall) zu erstellen, die gestartet und in weiterer Folge gesteuert und protokolliert wird. Hier werden Daten in Bezug auf die Instanz dargestellt, was eine Auswertung des Laufzeitverhaltens ermöglicht. Auch sollte es möglich sein fachliche Daten zu protokollieren, denn diese geben eine rundum Auskunft über den gesamten (bereits abgeschlossenen) Geschäftsvorfall ab.

Die Überwachungskomponente stellt neben den „instanzbezogenen" Daten ebenso „vorgangsbezogene" Daten dar, die ein Gesamtbild über den Prozessablauf bieten. Es ermöglicht zum Beispiel das Erkennen von Engpässen und dafür zeitgerechtes Handeln.

Die Schnittstellenkomponente ermöglicht die Anbindung weiterer Systeme und ist notwendig für die Verwendung von Daten aus anderen Anwendungen (Datenschnittstelle, vergleichbar mit einer „Tür zur Außenwelt"). Hier ruft die *Programmschnittstelle* externe Programme und Funktionen auf, die *Benutzerschnittstelle* wird meist schon bereitgestellt und ist als „Workflow-Client" leicht in bestehende Systeme integrierbar.

Die fünfte Komponente bezieht sich auf die Simulation des Ablaufs des modellierten Prozesses. Durch die Angabe der Durchläufe und Zeitdauer einzelner Aktivitäten können während und nach der Simulation Fehler und Schwachstellen (Verzögerungen, Engpässe) erkannt und ausgebessert werden.

E-Mail-Systeme oder kollaborative Software (Groupware) unterstützen zwar Teile des gesamten Ablaufs des Geschäftsprozesses, sie sind jedoch nicht gleichzusetzen mit WfMS. Gemäß der WfMC sind die einzelnen Produkt-Lösungen bestenfalls in WfMS *integriert*. [10] Seit den siebziger Jahren wurden viele unterschiedliche Lösungen entwickelt, die die Geschäftsprozesse technisch unterstützen sollen. Da diese Lösungen jedoch nicht für alle Aspekte des Geschäftsprozesses dienten, sondern nur für Teile davon, musste ein alles verbindendes System, das WfMS, entwickelt werden, um allumfassend alle Prozessaktivitäten steuern und unterstützen zu können. [19]

2.3 Workflow Referenz Modell

Das Referenzmodell der WfMC bezweckt eine Verknüpfung und somit die Möglichkeit der Kommunikation zwischen einzelner Komponenten und dem WfMS [10]. Sie definiert drei charakterisierende Funktionalitäten, wie die „Build-Time"-Funktionalität, die die Übersetzung des Prozesses übernimmt, sodass es von der Maschine interpretiert werden kann. Dies entspricht dem Modellieren und Simulieren von Workflows. Die zweite Funktionalität ist die „Run-Time Process Control" und ermöglicht die Instanziierung und Steuerung des interpretierten Prozesses (Process Engine). Dies entspricht der Instanziierung und Ausführung von Workflows. Damit ein Anwender letztendlich mit dem Prozess interagieren und zugeteilte Arbeitsschritte ausführen kann, wurde die Funktionalität der „Run-Time Activity Interaction" definiert. Hier findet die Überwachung statt - das Monitoring der laufenden Vorgänge und die Analyse entstandener Daten. Das Workflow Management hat zum Ziel die Ausführung der Geschäftsprozesse zu modellieren und zu steuern. Das WfMS unterstützt Prozessspezifikation, Erstellung, Koordination und Administration eines Prozesses über eine ausführende Komponente (Software), wobei die Ausführungsreihenfolge auf der Workflow Logik basiert.

2.4 Sicherheitsanforderungen in WfMS

Sicherheitsmaßnahmen und allgemeine Schutzziele für Informationssysteme gelten genauso auch für webbasierte WfMS und ihrer wertschöpfenden Objekte. Grundlage von Sicherheitsanalysen sind die zu wahrenden Schutzziele, denn ein Verlust dieser hat den Verlust der Sicherheit zur Folge. [35] Die häufigsten in der Literatur angeführten (technischen) Sicherheitsmaßnahmen (auch als Sicherheitsziele oder Schutzkriterien vorzufinden) werden in der nachstehenden Auflistung vorgestellt [16] [26]:

- Vertraulichkeit: (*confidentiality*) betrifft die unbefugte Bekanntgabe von sensiblen Informationen. Nur ausgewählte Personen und Systeme sollen auf vertrauliche Daten zugreifen können.
- Integrität: (*integrity*) betrifft die unerlaubte Modifikation von Informationen, sowie Datenmanipulation. Integer sind Daten dann, wenn diese nicht unbefugt modifiziert wurden.
- Verfügbarkeit: (*availability*) bezieht sich darauf, dass Daten und Ressourcen für die zuständigen Einheiten und die Abarbeitung der Aufgaben (tasks) zugänglich sind.
- Authentifikation: (*authentication*) betrifft die zuverlässige Identitätsprüfung der berechtigten und ausführenden Einheiten .
- Autorisierung: (*authorization*) betrifft die Durchführung von Zugriffskontrollen (Berechtigungen), um besonders die ersten beiden Schutzziele (Vertraulichkeit, Integrität) zu gewährleisten.
- Pflichtentrennung: (*separation of duties*) betrifft zusätzliche Restriktionen und Bedingungen in den Aufgabenbearbeitungen, um Arglist zu vermeiden.
- Kontrollierter Zugriff: (*controlled access*) um den korrekten Zugang zu Aufgaben und Informationen (Ressourcen) zu kontrollieren

Das WfMS erfüllt das Schutzziel der Vertraulichkeit, wenn Informationsgewinnung *unautorisiert* nicht möglich ist, der Integrität, wenn *unautorisiert* Datenänderung und oder -löschung nicht möglich ist und das Schutzziel der Verfügbarkeit, wenn alle *berechtigten* Zugriffe auf das System und die Ressourcen durchführbar sind. Diese Schutzziele gelten auch als die wichtigsten Sicherheitsziele, während weitere auf diesen aufbauen, und eignen sich als umfassende Anforderung, um die Informationssicherheit zu ermöglichen [16].

Die Informations-Sicherheitspolitik (*security policy*), wie sie in [13] beschrieben ist, legt organisatorische Maßnahmen fest (was ist erlaubt und was ist nicht erlaubt, wer hat Zugriff auf Daten und wer hat keinen Zugriff). [17] Die Sicherheitspolitik kann anhand Rollen innerhalb eines Unternehmens ausgedrückt werden. Rollen repräsentieren organisatorische Mittel, um definierte Funktionen im Unternehmen auszuführen. Den Rollen werden die im Unternehmen tätigen Personen je nach individueller Qualifikation zugeordnet. Neben der Zuteilung der Personen werden den Rollen ebenso Autorisierungen zugeteilt. Eine typische Restriktion einer Berechtigung ist das Prinzip des *separation-of-duties*, dessen Ziel es ist, das Risiko des Schwindels und falschen Angaben zu verringern. Eine komplexe Aktion wird dabei in kleinere Teilaufgaben unterteilt, die von unterschiedlichen Personen ausgeführt werden soll. Demnach muss ein entsprechender Zugriffs-Kontrollen-Mechanismus in der Lage sein, Rollen und die Spezifikation von Restriktionen zu unterstützen. Wie es jedoch auch im speziellen WfMS in dieser Arbeit der Fall ist, finden derartige Restriktionen selten bis gar keine Unterstützung in WfMS, weshalb sie explizit als externe Implementierung (Java-Programm) in den jeweiligen Aufgaben (tasks) implementiert werden müssen. Gleiches gilt für das Gegenstück des *separation-of-duties*, nämlich das *binding-of-duties*, wonach mindestens zwei aufeinander folgende Teilaufgaben von der gleichen Person bearbeitet

werden müssen. Auch dieses Verhalten ist für Demonstrationszwecke in dieser Arbeit manuell zu implementieren.

Grundlegende Sicherheitsmaßnahmen, wie sie auch in der International Standard Organisation (ISO) als „security services" spezifiziert sind, sollten vor allem für webbasierte Informationssysteme bereitgestellt werden. Neben Authentifizierung, Zugriffskontrolle, Vertraulichkeit und Datenintegrität wird auch die Nachweisbarkeit (*non-repudiation service*) beschrieben. Dabei geht es darum, alle Zugriffe im Überblick zu behalten. Das System muss stets darüber Bescheid wissen oder geben können, welche Person welche Aufgabe ausgeführt und wann die Ausführung statt gefunden hat. WfMS haben im Allgemeinen eine solche Kontrolleinheit, um die Effizienz und Effektivität der Prozessbeteiligten und den gesamten Prozessdesign zu evaluieren. [30][25][28]

2.5 Sicherheitskontrollen

Die umfassende Literaturrecherche aus [25] ergibt ingesamt zwölf Sicherheitskontrollen, aufgeteilt in fünf Kategorien, die vorrangig in Process-Aware Information Systems (PAIS) genützt sind. Diese Kategorien umfassen Sicherheitskonzepte (*security concepts*), Berechtigung und Zugangskontrollen (*authorization and access control*), Verifikation (*verification*), Fehlerbehandlung (*failure handling*) und den Bereich der Anwendung (*application*).

Die erste Kategorie der Sicherheitskonzepte betrifft angewendete Sicherheitskontrollen bei der Modellierung und Entwicklung von Geschäftsprozessen (zusätzliche sicherheits-relevante Informationen). Die Kategorie Berechtigungen und Zugangskontrollen betrifft Mechanismen zur Administration der User-Berechtigung, wie die (Rollen-basierte) Zugriffskontrolle als die am häufigsten angewendete Maßnahme. Die Kategorie der Verifikation betrifft den Geschäftsprozess an sich: Korrektheit (*correctness*), Stimmigkeit (*consistency*) und Regelkonformität (*compliance*) müssen bei der Ausführung gegeben sein, um etwaige Sicherheitsprobleme zu vermeiden (zB. unerlaubter Zugriff). Die Kategorie der Fehlerbehandlung betrifft Fehlermeldungen, die aufgearbeitet und behandelt werden müssen, um Gefahren durch Ausnahmezustände während der Laufzeit zu vermeiden. Die Kategorie Anwendung beinhaltet die Implementation von Sicherheitsfunktionen (Zugangskontroll-Modell). [25]

Eine Nicht-Einhaltung sämtlicher Aspekte birgt Gefahren besonders auch in WfMS. Vor allem wird hier der unbefugte Zugang zu Informationen hervorgehoben. Die genannten Kategorien werden in der Umsetzung des Geschäftsprozesses in dieser Arbeit berücksichtigt und ebenso in der Schwachstellenanalyse untersucht.

2.6 Bedrohungen und Gefahren

Eine existierende und mögliche Verletzung eines System, insbesondere die IT-Sicherheit dieses Systems, wird als Bedrohung oder auch Gefahr betrachtet und ergibt sich aus der Bewertung von Systemangriffen. Bekannte Angriffe können eine Gefahr darstellen und werden deswegen auch als mögliche Bedrohung betrachtet. Dies gilt vor allem dann, wenn die Angriffe auf das System tatsächlich technische Probleme, wie Systemausfälle, zur Folge haben. [13]

2.6.1 Angriff und Angreifer

Die Aktion eines Angreifers gegen Sicherheitsmaßnahmen eines Unternehmens oder Systems, um diese zu umgehen, wird von [13] als Angriff (*attack*) beschrieben. Dabei lassen sich Angriffe in zweierlei Hinsicht voneinander unterscheiden: Angriffe können internen aber auch externen Ursprungs und aktiver oder passiver Natur sein. Greift der Angreifer innerhalb der Systemgrenzen an, wird von einem internen Angriff gesprochen, andernfalls von einem externen Angriff.

Ein aktiver Angriff beeinflusst Systemressourcen und kann bestenfalls eine totale Beeinträchtigung des Systems (also einen Stillstand) verursachen, während ein passiver Angriff vorrangig die definierten Schutzziele zu umgehen versucht. Der passive Angriff beeinträchtigt dabei nicht die Funktionalität des Systems, sondern befasst sich vordergründig mit Spionage-ähnlichen Aktivitäten (abhören, beobachten aber auch stehlen von Daten und Informationen). Schafft es die angreifende Person die Sicherheitsmaßnahmen zu umgehen, hat sie damit die „IT-Sicherheit des Systems verletzt" [39].

Die Personen, die Angriffe an ein System ausführen, können je nach persönlichen Wissensstand grob in vier unterschiedliche Angreifertypen unterteilt werden, wobei die in der Literatur verwendeten Bezeichnungen der Typen und Gruppen auch variieren.

Personengruppen, die als „normale Benutzer" bezeichnet werden, sind zunächst nicht auf die leichte Schulter zu nehmen, da sie durch ihre „Unwissenheit" auch großen Schaden anrichten können, selbst wenn dies nicht beabsichtigt ist. Eine im zugrundeliegenden WfMS registrierte und „harmlose" Person kann durch ihr „fahrlässiges Verhalten" Zugangsdaten an fremde und unautorisierte Personen weitergeben, sodass sich der Angreifer mit diesen Informationen Zugang zum gesamten System verschaffen und von dort aus weitere Angriffstätigkeiten ausüben kann.

Die sogenannten Skriptkiddies, als zweite Wissensgruppe, können sich bereits offengelegter Algorithmen bedienen und Schwachstellen ausnutzen, sofern sie diese entdecken. Skriptkiddies zählen noch zu den Anfängern, die sich Schwachstellenscannern und andere im Internet auffindbaren Werkzeuge bedienen, bestmöglich einen Schaden dadurch erreichen aber nicht weiter tiefgründiges Wissen über die Hintergründe haben. Werden durch Skriptkiddies Schwachstellen ausgenutzt, können meist ihre „Fingerabdrücke" erkannt und so nachverfolgt werden. Sollten in der Activiti BPM Anwendung alt bekannte Schwachstellen vorhanden sein, wäre diese Gruppe eventuell auch in der Lage, diese Schwachstellen auszunutzen. Web-basierte Applikationen, wie das zugrunde liegende Activiti BPM sind ein beliebtes Ziel für derart nicht-professionelle Hacker. [35]

Halb-professionelle Angreifer haben bereits einen tiefergehenden Wissensstand, was besonders ihre Fachgebiete anbelangt. Sie wissen, wie sie ihren Bereich angreifen aber auch vor Angriffen schützen können.

So auch die letzte Wissensgruppe, nämlich die professionellen Angreifer, von denen die wohl größte Gefahr ausgeht [35]. Sogenannte „kriminelle Hacker" sind Experten in ihrem Gebiet und nutzen unter anderem selbst geschriebene Werkzeuge, um ihre Angriffe durchzuführen (diese selbst erstellten Programme sind es zum Beispiel, die teilweise frei zur Verfügung stehen und deshalb unter anderem von den Skriptkiddies verwendet werden). Darunter zählen die schadhafte Programme, wie Viren und Würmer. Im Gegensatz zu den „Anfängern" haben professionelle Angreifer auch Wissen darüber, wie sie ihre „Spuren verwischen" können (beispielsweise durch Manipulieren der Logfiles), um im kompromittierten System nicht aufzufallen und vor allem auch im Nachhinein nicht ausfindig gemacht werden können. [39]

2.6.2 Schwachstelle

Schwachstellen (*vulnerabilities*) sind in diesem Kontext Fehler in Informationssystemen, die missbraucht werden können, um die Sicherheitskonzepte (*security policies*) zu verletzen. Es wird dann von einer Schwachstelle gesprochen, wenn (fehlerhafte) Systemzustände die Chance ermöglichen missbräuchlich genutzt zu werden. Einerseits können Schwachstellen beabsichtigt in ein System eingeführt werden, andererseits auch durch Fahrlässigkeit (oder Unkenntnis durch eine Person, eines Mitarbeiters) entstehen.

Werden sie während der Entwicklung des Systems eingeführt, handelt es sich um konstruktionsbedingte Schwachstellen. Weitere Arten von Schwachstellen sind beispielsweise Sicherheitsfunktionen, die direkt angegriffen werden können (Passwortfunktion), offene Kommunikationskanä-

le, die ausgenutzt werden können und eine für die eigentliche Person unvorteilhafte Bedienung (schlechte Benutzerfreundlichkeit). [13][28]

Die folgende Abbildung (2.1) verdeutlicht den Zusammenhang zwischen den Begriffen Bedrohung, Angriff und Schwachstelle, wobei die internen aber auch die externen Angriffe als die größten Bedrohungen gelten.

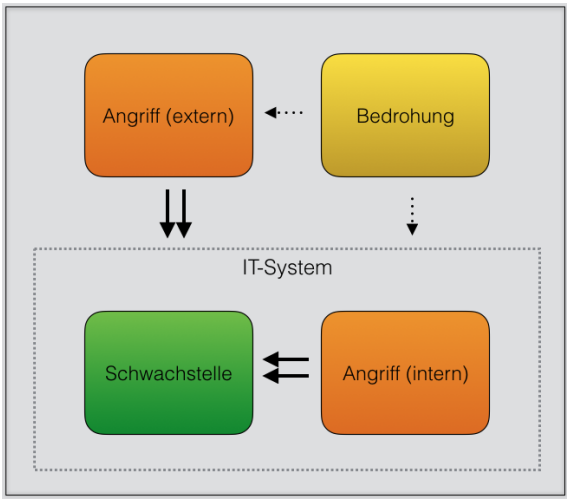

Abbildung 2.1: Zusammenhang: Bedrohung, Angriff und Schwachstelle (in Anlehnung an [13])

Grundsätzlich besitzt jedes IT-System gewisse Schwachstellen und kein System ist zu hundert Prozent sicher gegen jede Art von Angriffen, da sich die Angriffsarten im Laufe der Zeit oft abändern und dadurch ihre Gegenmaßnahmen meist erst entwickelt werden müssen.

In der durchgeführten Schwachstellenanalyse wird davon ausgegangen, das keine expliziten Schwachstellen „bekannt" sind. Das System wird daraufhin aus verschiedenen Blickwinkeln systematisch betrachtet, um dadurch eine etwaige Schwachstelle zu finden.

Schwachstellen liegen oftmals nicht offensichtlich vor, sondern können meist erst durch eine umfassende Systemanwendung und oder Tests „gefunden" werden. Was eine Schwachstellenanalyse ist, wird im folgenden Abschnitt beschrieben.

2.7 Schwachstellenanalyse und Methoden

In einer Schwachstellenanalyse wird versucht Schwachstellen und Fehler zu entdecken. Dabei handelt es sich um einen Vorgang (mit unterschiedlichen Ansätzen), bei dem „Eigenschaften" gefunden werden, die es erlauben oder ermöglichen, dass die Schutzziele verletzt und damit ausgenutzt werden können (*exploit*). Mit einer Schwachstellenanalyse soll untersucht werden, in wie weit Sicherheitsmaßnahmen eingerichtet und berücksichtigt wurden.[13].

Webbasierte Systeme lassen sich auf unterschiedliche Art und Weise untersuchen. Je nach Absicht (und auch Wissensstand) variieren die anwendbaren Methoden, der Umfang der Analyse und die Einbeziehung von etwaigen Teilnehmern. Ein Softwaretest beispielsweise prüft auch während der Entwicklungsphase die vorliegende Software hinsichtlich der Erfüllung der definierten Anforderungen. Haupteigenschaft solcher Tests sind neben der individuellen Entwicklung von Tests, die erfolgreichen oder fehlerbehafteten Testresultate. Die Bewertung der Systemqualität und der Prüfungsumfang hängen von den definierten Testfällen ab. Je umfangreicher die Tests gestaltet sind, umso genauer fällt die Gesamtbewertung aus, sie geben jedoch keine Bestätigung darüber, dass das untersuchte System fehlerfrei und dadurch auch als sicher gilt.

Mit einem Schwachstellen-Scan können automatisiert die aus dem Internet erreichbaren Systeme (das sind Webserver, Datenbankserver und weitere Systeme) auf Schwachstellen überprüft werden. Je nach Art sind zwei unterschiedliche Arbeitsweisen vorherrschend: Die Untersuchung des Zielsystems mit und ohne einer Authentifizierung. Das Resultat ist eine technische Zusammenfassung aller Untersuchungsergebnisse, welche dazu dient, auf lokalisierte Fehler und Schwachstellen aufmerksam zu machen, um diese zur Stärkung des Systems zu beseitigen.

Der Schwachstellen-Scanner als Computerprogramm bedient sich einer Schwachstellen Datenbank, in der bekannte Sicherheitsprobleme dokumentiert sind: unsichere Dienste, Fehler in diversen Konfigurationen, Fehler in Sicherheitsrichtlinien, und weitere Informationen[2]. Die beispielhafte Anwendung eines derartigen Scanners wird in Kapitel 4.9.4 vorgestellt.

Das Ziel dieser Vorgehen und Methoden ist es, herauszufinden, ob das entsprechende IT-System den Anforderungen entspricht und die definierten Sicherheitsmaßnahmen erfolgreich umgesetzt sind. Eine weitere Möglichkeit eine Schwachstellenüberprüfung durchzuführen, ist eine manuelle Untersuchung (nicht-automatisierbare Prüfungen).

In dieser Arbeit wird die Schwachstellenanalyse mittels einer baumorientierten Methode durchgeführt, die zunächst manuelle Tätigkeiten umfasst und in weiterer Folge teilweise auch durch Computerprogramme unterstützt wird. Die gegebene Infrastruktur wird dabei analysiert und schematisch aufbereitet. Nach eingehender Recherche hinsichtlich möglicher Methoden zur systematischen Untersuchung konnten besonders in [28] ansprechende Möglichkeiten gefunden werden. Darin wird neben der eher bekannten Attack Tree Methode vorrangig die sogenannte ATLIST Methode detailliert vorgestellt.

Aufgrund dieser doch sehr gegensätzlichen Methoden wurde daraufhin für die erwähnte Zusammenarbeit beschlossen die Attack Tree Methode für die vorliegende Arbeit anzuwenden, während für den Vergleich der Resultate die ATLIST Methode in der Arbeit von Sabine Gottwald[3] angewendet wird. In diesem Zusammenhang wird in den folgenden Abschnitten die Attack Tree Methode und die ATLIST Methode vorgestellt.

2.7.1 Die Attack Tree Methode

Die sogenannte „Attack Tree" (auch Angriffsbaum) Methode ist eine von Bruce Schneier entwickelte Analysemethode, um Angriffe mithilfe von Bäumen zu modellieren und schematisch aufzubereiten. Bezugnehmend auf IT-Systeme können so sicherheitsspezifische Aspekte dargestellt und beschrieben werden. Das Resultat ist ein Attack Tree Modell, welches auf verschiedene Aspekte angewendet (beispielsweise in einer Risikoanalyse) und auch wiederverwendet werden kann.

2.7.2 Erzeugung von Attack Trees

Die Hauptbestandteile eines Attack Trees sind zum einen die Wurzel und zum anderen die Blätter. Der Wurzelknoten entspricht dabei das vorher festgelegte Angriffsziel, wohingegen die Blätter einzelne Angriffsmethoden und Angriffsstrategien darstellen (auch Subziele), um das jeweils obere Angriffsziel zu erreichen. Es folgen von der Wurzel ausgehend Kanten zu den entsprechenden Blättern und von diesen Blättern wiederum Kanten zu weiteren Blättern oder Subzielen (Aufbau verläuft grundsätzlich von oben nach unten). Die zwei wesentlichen Kantenbeschriftungen trennen oder verbinden zwei oder mehrere Subziele. Die Unterscheidung liegt bei „Und" und „Oder" Verknüpfungen. Die Und-Verknüpfung verlangt, dass zwei oder mehrere mit „Und" verbundene Subziele von einer angreifenden Person erfüllt werden müssen, während eine Oder-Verknüpfung nur eines der nachfolgenden Subziele erzwingt dabei aber eine freie Wahl möglich

[2] Web App Security Consortium, http://www.webappsec.org/ , letzter Zugriff April, 2016

[3] Diplomarbeit von Sabine Gottwald mit dem Titel: „ATLIST Tree zur Analyse von Schwachstellen in WfMS", 2016.

ist, um das nächst höhere Blatt oder die Wurzel zu erreichen. Wesentlich hierbei ist, dass der „Angriffsweg" von unten nach oben, also von den Blättern zur Wurzel hinauf verläuft. In Anlehnung an [11] werden zur Kürzung von Bäumen, die ein unüberschaubares Ausmaß annehmen können, auch Unterbäume (oder Subbaum) hinzugefügt. Die Skizze in Abbildung 2.2 zeigt den groben Aufbau eines Attack Trees, wie er in dieser Arbeit angewendet wird.

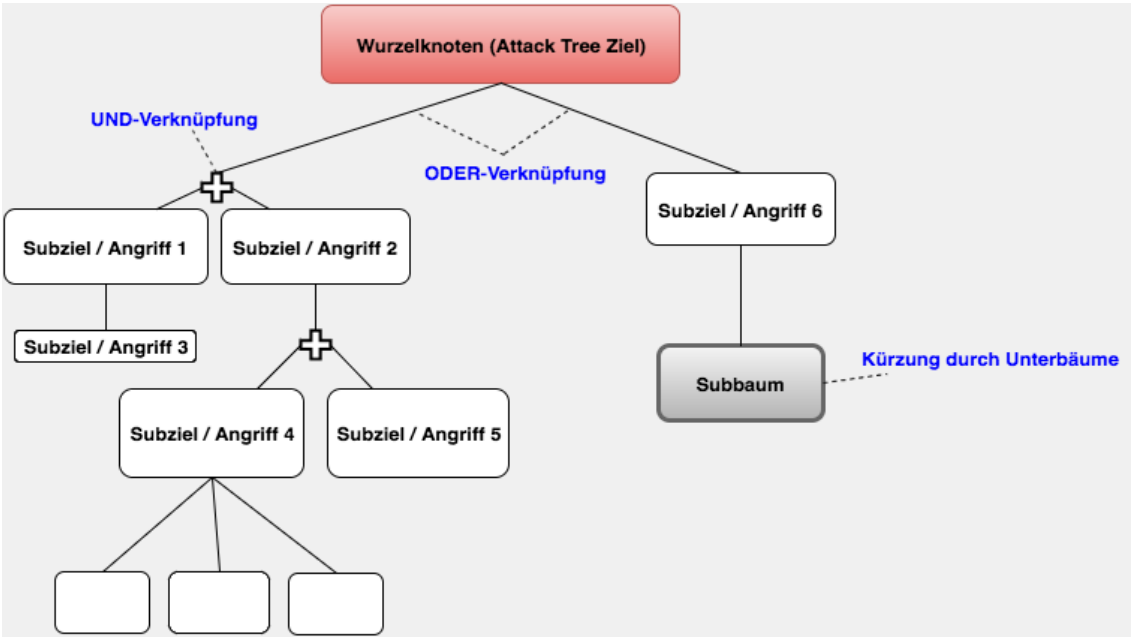

Abbildung 2.2: Attack Tree - Exemplarischer Baum (eigene Darstellung)

Die Angriffsbäume beschreiben als eine Art Flussdiagramm, wie potentielle Angreifer gegen das System vorgehen könnten. In dieser Arbeit wird anhand der Attack Tree Methode der entsprechende Angriffsbaum erstellt und dient für die nachfolgende Analyse der Schwachstellen als Grundlage.

2.7.3 Die ATLIST Methode

ATLIST wird als „attentive listener" vorgestellt und beschreibt eine weitere Schwachstellen Analysemethode, welche im Besonderen die Entdeckung von bekannten Schwachstellentypen und anhand dieser das Ableiten von Schwachstellen Muster ermöglicht. Die Besonderheit gegenüber den herkömmlichen Methoden ist die Tatsache, dass die Analyse richtungsweisend mittels Analyse-Elementen geleitet und geführt wird, um auf diese Weise einen ATLIST Baum zu erstellen. Dieser Baum wird im Hinblick auf Schwachstellen der einzelnen Elemente analysiert, woraufhin aufgrund der Schwachstellentypen bestimmte Muster abgeleitet und entwickelt werden können. Solche abgeleitete Muster können in automatisierten (musterbasierten) Schwachstellen-Suchprogrammen anschießend eingesetzt werden und ihre Anwendung finden.

2.7.4 Erzeugung von ATLIST Bäumen

Für jede Angriffsauswirkung wird ein ATLIST Baum erstellt, wobei der Wurzelknoten die Auswirkung (*attack effect*) darstellt. Anschließend werden die Blätter (hier als Kindsknoten bezeichnet) hinzugefügt, welche die aktiven Komponenten (*active components*) darstellen. Dabei handelt es sich um die als erstes vorkommenden Komponenten, in der die Schwachstelle verwertbar ist (wie etwa der Webserver). Diese Komponenten haben wiederum eine eigene Menge an

Kindsknoten, welche die involvierten Standards (*involved standards*) beinhalten, die bei falschem Gebrauch die Ausnutzung der Schwachstelle ermöglichen. Diese Standards haben ebenso Blätter, welche die sogenannten auslösenden Eigenschaften darstellen (*triggering properties*). Diese Eigenschaften sind es, die der Angreifer letzten Endes ausnutzt. Jeder Pfad, begonnen bei der auslösenden Eigenschaft bis hin zum Wurzelknoten, stellt einen Schwachstellentyp und jedes Schwachstellendetail einen Subtyp dar.

Wie es in [28] schlussendlich beschrieben wird, entsprechen die Schwachstellen-Muster die formale Representation der Subtypen. Ein Beispiel des fertigen ATLIST Baumes wird in Abbildung 2.3 vorgestellt.

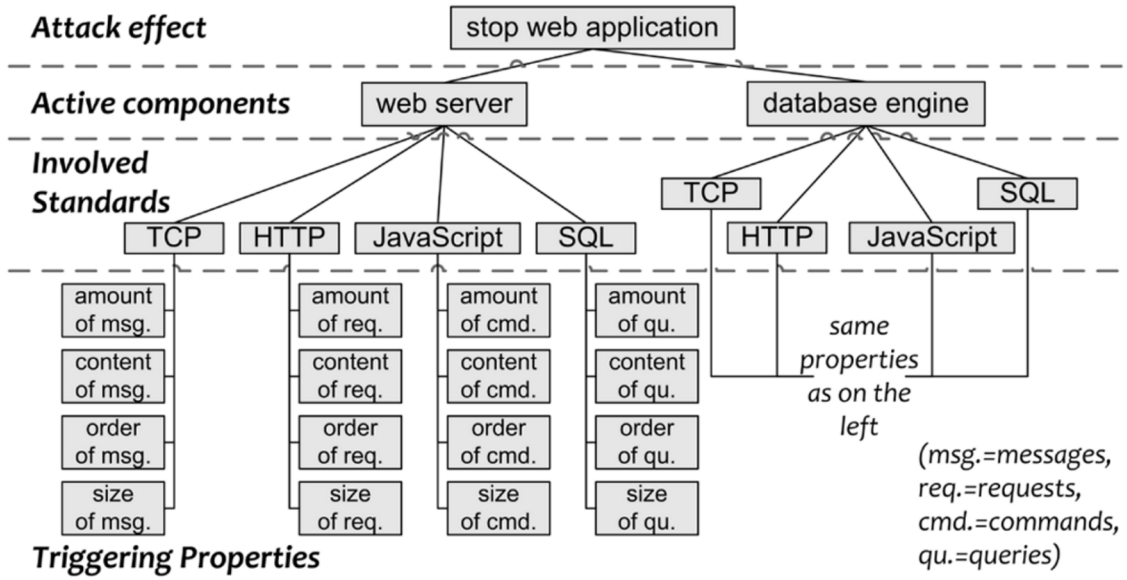

Abbildung 2.3: ATLIST - Exemplarischer Baum (entnommen aus [28])

Eine beispielhafte Anwendung der ATLIST Methode wird in der Schwachstellenanalyse von Sabine Gottwald[4] vorgestellt.

[4]Diplomarbeit von Sabine Gottwald mit dem Titel: „ATLIST Tree zur Analyse von Schwachstellen in WfMS", 2016

3 | Arbeitsumgebung

Das vorliegende Kapitel befasst sich mit der Vorstellung des angewendeten und in weiterer Folge analysierten WfMS, sowie die zugrundeliegende Spezifikationssprache mit graphischen Grundelementen, von denen Teile im Beispielmodell angewendet werden.

Zur Realisierung eines abwechslungsreichen Prozessmodells spielen sämtliche technische Aspekte und Komponenten eine Rolle und ihre korrekte Konfiguration ist wesentlich für eine reibungslose Anwendung. Die Umsetzung in Kapitel 3.4 (Eclipse Activiti Projekt - Die Umsetzung) umfasst die Bestimmung der Benutzer und Gruppen mit ihren Einschränkungen und Bedingungen, sowie weitere Implementierungen, welche dem Workflow maschinelle Aufgaben hinzufügen, um abschließend einen Gesamteindruck des zugrundeliegenden Prozessmodells vorzustellen.

3.1 Activiti BPM Plattform

Die in Kapitel 4 beschriebene Schwachstellenanalyse wird mit dem Open Source WfMS *Activiti BPM* [1] durchgeführt. Activiti ist eine Workflow- und BPM-Anwendung, die auf Java basiert und den Fokus auf das Modellieren und Ausführen von vorher definierten Prozessen setzt. Die schnelle und flexible Standard BPMN 2.0 (Business Process Model and Notation) Engine (auch Workflow Prozessor) im Kern der Anwendung sorgt für die Abarbeitung aller Arbeitsschritte und ermöglicht die Integration weiterer Geschäftsanwendungen. Folgende Bestandteile stellen einen groben Überblick über Activiti zusammen:

- In der Management Komponente befindet sich die Webanwendung, der sogenannte *Activiti Explorer*, über die der Zugriff auf die Activiti Engine (Laufzeit Komponente, die für die Ausführung der Prozessschritte zuständig ist) auch während der Laufzeit möglich ist. Dies gilt für alle im System registrierten Personen (zentraler Einstiegspunkt) und umfasst eine umfangreiche Sammlung von Aktivitäten, wie das Verwalten der Arbeitsschritte, die Durchsicht und Verwaltung der einzelnen Prozessinstanzen (starten, stoppen, suspendieren), diverse Managementaktivitäten (Benutzerregistrierung, Gruppenverwaltung), sowie die Erstellung von Reports (personenbezogen, prozessbezogen) und auch die Modellierung oder Abänderung eines Prozesses. Activiti ist webbasiert und wird demnach im Web-Browser bedient. Jede registrierte Person erhält nach der Authentifizierung die eigene Ansicht der verfügbaren Funktionen.
- Der webbasierte *Activiti Modeler* ist eine grafische Modellierungskomponente von und in Activiti und kann für die graphische Erstellung der Prozesse direkt im Activiti Explorer (also im Web-Browser) bedient werden (BPMN 2.0 konform). Allerdings wird diese Komponente nicht mehr aktiv weiterentwickelt, weshalb ein Plug-in (eine Software Erweiterung) für die externe Entwicklungsumgebung als Alternative zur Verfügung gestellt wurde.

[1] Activiti Hauptseite: www.activiti.org, letzter Zugriff im Mai, 2016

- Der *Activiti Eclipse designer* ist ein Zusatzmodul für die Entwicklungsumgebung Eclipse[2], mit dem direkt eine BPMN 2.0 konforme Erstellung von Workflows bewerkstelligt werden kann (sowohl die Prozessdefinition, als auch die graphische Prozessmodellierung). Diese Alternative wird in der vorliegenden Arbeit angewendet.
- Die Webanwendung *Activiti REST* stellt eine REST Schnittstelle zur Verfügung und ist ebenso ein Eingangspunkt zur Activiti Engine. Mit der Schnittstelle ist es externen Programmen möglich auf Prozesse zuzugreifen, aber auch externe Programme selbst können ausgeführt werden. [33]

Da für die graphische Darstellung der Prozesse der Standard BPMN 2.0 verwendet wird, beschreibt der folgende Abschnitt die wesentlichen Bestandteile dieser Notation und erleichtert das Lesen des im Anschluss vorgestellten und dieser Arbeit zugrundeliegenden Prozessbeispiels.

3.2 Standard BPMN 2.0

Mit der BPMN wird eine Spezifikationssprache in graphischer Form verstanden, wodurch Geschäftsprozesse und Workflows mit Hilfe von zur Verfügung gestellten Symbolen erstellt werden können. Diese graphische Notationssprache ermöglicht das Modellieren von Geschäftsprozesse, sodass diese von allen beteiligten Personen leicht zu verstehen und anzuwenden sind. Die verschiedenen Elemente und Konstrukte können in einem Geschäftsprozess-Diagramm (Business Process Diagram) abgebildet, um in weiterer Folge direkt in einer entsprechenden Process-Engine ausgeführt zu werden. Ableitend ist eines der Ziele der BPMN die Definition, wie erstellte Geschäftsprozess-Diagramme in Ausführungssprachen auf XML-Basis übertragen werden sollen. Die Version, BPMN 1.0, wurde erstmals im Jahr 2006 von der internationalen Organisation Object Management Group (OMG) als Standard anerkannt und gilt als eines der wenigen Standards, die hauptsächlich für die Modellierung von Geschäftsprozessen entwickelt wurde. Nach zahlreichen Weiterentwicklungen wurde im Jahr 2011 der Standard BPMN 2.0 verabschiedet. Seither wird BPMN als „Business Process Modeling and Notation" gelesen. Die Modelle in BMPN 2.0 Notation können nun als XML gespeichert und von einer BPMN 2.0 Process Engine ausgeführt werden.[5]

3.2.1 Grafische Repräsentation

Für eine erste Orientierung bei der Modellierung wurden Kategorien, auch BPMN Basis-Elemente genannt, geschaffen, die die Kernelemente zur Geschäftsprozess-Modellierung darstellen. Diese sind Flussobjekte (Flow Objects), Verbindende Objekte (Connecting Objects), Bahnen (Lanes, Pools) und Artefakte (Artefacts) und werden in Abbildung 3.1 zusammengefasst.

Ein Geschäftsprozess enthält Aktivitäten, die oft auch erst unter bestimmten Bedingungen abgearbeitet werden. Hierfür gibt es die sogenannten Gateways. Aber nicht nur Aktivitäten, sondern auch Ereignisse können auftreten. Diese Flussobjekte stehen nicht frei im Raum, sondern werden über verbindende Objekte, wie dem Sequenzfluss miteinander in Verbindung gebracht (vergleichbar mit Kanten). Der Nachrichtenfluss dient der Verbindung von Lanes, also wenn eine Verbindung außerhalb von Grenzen erfolgt. Keinen direkten Einfluss auf den Prozessablauf aber informativen Charakter haben Artefakte. Diese können an jedem Flussobjekt angebunden werden (Assoziationen) und so Informationen für ein besseres Verständnis bereits anbieten. Während der Prozessablaufs können Informationen ebenso Wichtigkeit erlangen, weshalb Datenobjekte ebenfalls mittels Assoziationen mit Aktivitäten verbunden werden können. Auf diese Weise wird die Erzeugung aber auch die Verarbeitung und Speicherung von relevanten Daten und Informationen darstellbar.

[2]https://eclipse.org/

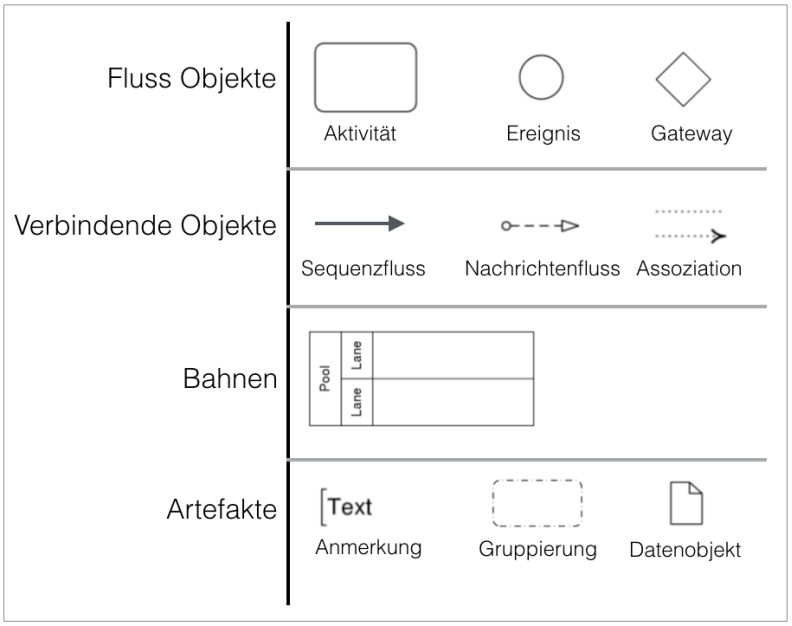

Abbildung 3.1: BPMN Basiselemente (in Anlehnung an [1] und [22])

In der vorliegenden Arbeit wird mit der Erweiterungskomponente, dem Activiti Eclipse designer direkt in der Eclipse Programmierumgebung gearbeitet, um den zugrundeliegenden Geschäftsprozess grafisch darzustellen und technisch umzusetzen.

3.2.2 Eclipse BPMN 2.0 Erweiterung

Auf den Seiten des online abrufbaren Benutzerhandbuchs zur Konfiguration der Activiti BPM-Anwendung wird erklärt, wie mittels der Eclipse Erweiterung graphische Modelle erstellt und im Weiteren getestet und angewendet werden können. Diese Erweiterung enthält für jede der obigen Kategorien eine Menge an Möglichkeiten, um den Geschäftsablauf originalgetreu abzubilden (siehe Zusammenstellung in Abbildung 3.2).

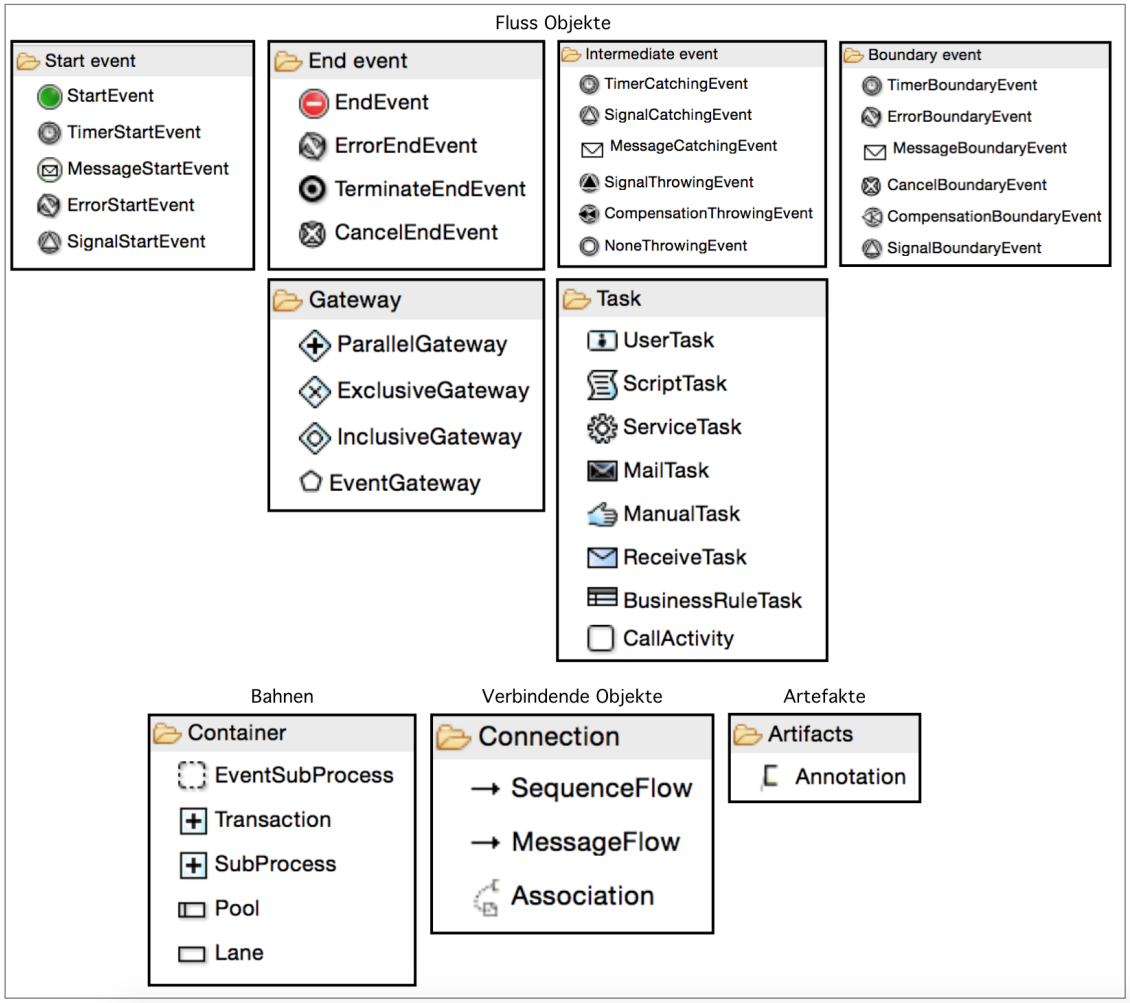

Abbildung 3.2: Activiti Eclipse designer - Angebotene BPMN Erweiterungen (entnommen aus der Activiti Eclipse designer Anwendung)

Nach der Erstellung des Geschäftsprozesses kann bereits die XML-Ansicht des Modells geöffnet und für den Import in die Activiti Explorer Webanwendung abgespeichert werden. Die Konfiguration und Vorgehensweise wird in den nächsten Abschnitten näher erläutert.

3.3 Voraussetzungen

Für Personen, die wenig bis keine Erfahrung mit Activiti BPM haben, gibt es eine Installationsvorlage im Anwenderleitfaden, der online abrufbar ist. Dabei handelt es sich um eine Installationsvorgabe, die im „Optimalfall" reibungslos angewendet werden kann[3]. Aus diesem Grund werden in diesem Kapitel zusätzliche und wesentliche Punkte des für die Arbeit durchgeführten Installations- und Konfigurationsvorgangs festgehalten. An dieser Stelle sei erwähnt, dass der Installationsvorgang aller Komponenten auf dem Betriebssystem Mac OS X Yosemite Version 10.10.5 durchgeführt wurde. Einige wenige Aspekte variieren im Vergleich zur Installation auf einem Windows Betriebssystem.

[3]Online-Dokumentation: http://activiti.org/userguide/index.html#_introduction

3.3.1 Technische Anforderungen und Konfiguration

Die allgemeinen Anforderungen betreffen Open Source Werkzeuge, die für die Inbetriebnahme des WfMS und speziell für die vorliegende Arbeit benötigt werden. Die folgende Auflistung zeigt zudem jeweils die installierte Version, die direkt von der im Internet abrufbaren Herstellerseite herunter geladen werden kann. Die Reihenfolge hat an dieser Stelle keine Bedeutung.

1. **Java** Java Development Kit[4] - Version 1.8.0_45
2. **Webserver** Apache Tomcat[5] - Version 8.0.32
3. **Build-Tool** Apache Maven[6] - Version 3.3.3
4. **Datenbank** MySQL[7] - Version 5.6.24 (Workbench Version 6.2.5)
5. **WfMS** Activiti BPM[8] - Version 5.19.02
6. **Programmierwerkzeug** Eclipse Java EE IDE[9] - Version Mars.1 Release 4.5.1
7. **Design-Tool** Activiti Eclipse BPMN 2.0 designer[10] - Version 5.18.

An dieser Stelle sei auch erwähnt, dass Activiti mehrere Datenbanken unterstützt, unter anderem wird die flüchtige „in-memory" H2 Datenbank bereitgestellt und kann für Testzwecke direkt verwendet werden. In dieser Arbeit wird jedoch die Datenbank MySQL angebunden. Die wesentlichen Punkte der Konfiguration obiger Komponenten werden in den anschließenden Abschnitten näher beschrieben und dient als wesentlicher Zusatz zu der im Netz befindlichen Installationsanweisungen der jeweiligen Hersteller.

3.3.2 Konfiguration Java Development Kit

Nach der Installation ist es vor allem notwendig den Home-Pfad, die Umgebungsvariable zu setzen, um sicherzustellen, dass die Erstellungsaufgaben sowie Implementierungsaufgaben fehlerfrei ausgeführt werden. Am einfachsten geschieht dies über die Befehlszeile mittels:

`edit .bash_profile` oder aber auch: `vi~/ .bash_profile`. Für die anschließende Überprüfung kann der Befehl: `echo $JAVA_HOME` ausgeführt werden. Ein Beispiel für Mac OS X wird in Listing 3.1 gezeigt:

Listing 3.1: Umgebungsvariablen setzen

```
1 //Home Pfade zeigen auf den Installations-Ordner
2 export JAVA_HOME=$(/.../java_home)
3 //
4 export ANT_HOME=/.../ant
5 //
6 export PATH=$PATH:${ANT_HOME}/bin
```

[4] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[5] http://tomcat.apache.org/index.html
[6] https://maven.apache.org/
[7] https://www.mysql.de/downloads/
[8] http://activiti.org/download.html
[9] https://www.eclipse.org/downloads/
[10] Plug-in, direkt über Eclipse installierbar, ermöglicht die Designentwicklung und Tests der erstellten BPMN 2.0 Prozesse

3.3.3 Konfiguration Apache Tomcat

Der Apache Tomcat Server kann mittels einem einfachen Kommandozeilen-Befehl (Shell) installiert werden: `brew install tomcat` (andernfalls über die Homepage des Herstellers herunter geladen und installiert werden). Nach dem der Webserver erfolgreich installiert wurde, können in der Datei `tomcat-users.xml` unter dem Attribut `<tomcat-users>` die Rollen, Benutzer und Passwörter für den Zugriff auf dem Server bestimmt werden. Der Webserver kann nun im Web-Browser mittels `localhost` getestet werden und im Normalfall wird darin der Text: *It Works!* ausgegeben. Sobald der Tomcat Server gestartet wird (über die Shell: `catalina start`), kann auf `localhost:8080` die Rubrik „Manager app" geöffnet werden, worin alle stationierten und laufenden Applikationen einzusehen sind. In weiterer Folge wird auch hier der Activiti Explorer hochgeladen und zu sehen sein.

3.3.4 Konfiguration Activiti

Die ausführbare Datei `activiti-explorer.war` wird auf dem Webserver stationiert, um ihre Funktionalität mit dem Neustart des Webservers zu testen und in weiterer Folge auch zu nutzen. Auch ohne weiteren Einstellungen können die Basisfunktionen des Activiti Explorers mit vordefinierten Benutzern bereits ausgeführt werden. Die Web-Oberfläche der BPM-Anwendung sowie der Einstiegspunkt hierfür ist stets `<servername>/activiti-explorer`. Mit dem vordefinierte Manager-Benutzer müssen für den eigenen Geschäftsprozess notwendige Benutzer und Benutzergruppen definiert werden. Für realitätsnahe und umfangreichere Geschäftsprozess-Definitionen müssen jedoch detaillierte Anpassungen getätigt werden, die in weiterer Folge in Abschnitt 3.4 am Beispiel des zugrunde liegenden Prozessszenarios beschrieben werden.

Neben der Installation von Activiti BPM wird in der Eclipse Programmierumgebung für die Modellierung des Geschäftsprozesses und die technische Umsetzung die Erweiterung Activiti Eclipse designer installiert. Dieses Software Erweiterung lässt sich im Eclipse selbst in nur wenigen Schritten über den Menüpunkt „Help" und „Install new Software" installieren. Hierfür wird die URL http://m2eclipse.sonatype.org/sites/m2e benötigt.

3.3.5 Konfiguration Apache Maven

Maven wird in dieser Arbeit im Zusammenhang mit Eclipse benötigt. Unter OS X lässt sich Maven sehr schnell mit dem Befehl `brew install maven` installieren (andernfalls auch hier über die Hersteller Seite herunterzuladen und zu installieren). Nachdem im Eclipse der Geschäftsprozess in einem sogenannten Maven-Projekt erstellt ist, kann die Funktion `Generate Unit Test` ausgeführt werden. Allerdings werden im ersten Anlauf sämtliche Fehlermeldungen unvermeidbar, da bestimmte Abhängigkeiten (dependencies) noch nicht gesetzt sind. Zur Behebung dieser Fehlermeldungen wird in der Shell das Kommando: `mvn dependency:tree` benötigt, um direkt von `maven.apache.org` fehlende Dateien zu laden. Wie in Abschnitt 3.4 zu sehen ist, werden in der im Eclipse Projekt entstandenen `pom.xml`-Datei weitere Abhängigkeiten manuell hinzugefügt. Je umfangreicher der Geschäftsprozess, desto mehr Abhängigkeiten werden benötigt und müssen manuell hinzugefügt werden. Die grundlegenden Abhängigkeiten sind jedoch mit diesem Schritt getan und im einfachen Prozessmodell (in den zugrundeliegenden Dateien) sollten keine Fehlermeldungen enthalten sein.

3.3.6 MySQL Datenbank

Mit der Auswahl und erfolgreichen Installation von der MySQL Datenbank werden weitere essentielle Einstellungen vorgenommen. Damit die korrekte und dauerhafte Verbindung zum Activiti BPM hergestellt werden kann, wird zunächst beispielhaft folgende Datenbank-Anweisung schrittweise durchgeführt (Listing 3.2 und 3.3).

Listing 3.2: SQL Datenbank–Anweisung für die Verwendung mit Activiti BPM

```
// Anmeldung ueber die Shell mittels eines definierten Benutzers
CREATE USER 'activiti'@'xxxx' IDENTIFIED BY 'activiti';
// CREATE Anweisung zur Erstellung der neuen Datenbasis
CREATE DATABASE IF NOT EXISTS `activiti_test`
DEFAULT CHARACTER SET `utf8`;

// Passwort muss hier gesetzt werden, Admin Rechte vergeben
GRANT ALL PRIVILEGES ON `activiti_database`.* TO 'activiti'@'xxxx';
```

Im Anschluss daran wird ebenso für Activiti die Datenbank betreffende Konfigurations-Datei `activiti-costom-context.xml` entsprechend der gewählten Datenbank angepasst:

Listing 3.3: Konfiguration in activiti-custom-context.xml

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
<property name="driverClassName" value="com.mysql.jdbc.Driver" />
<property name="url" value="jdbc:mysql://localhost:3306/activiti" />
<property name="username" value="activiti" />

<! -- Passwort muss hier gesetzt werden -->
<property name="password" value="xxxx" />
<property name="defaultAutoCommit" value="false" />
</bean>
```

Die Verbindung zwischen der Datenbank und dem Activiti BPM ist damit hergestellt und kann bereits genutzt werden. Alle Activiti Tabellen werden auf diesem Weg in die neue Datenbank übertragen. Sämtliche neu erstellen Tabellen können in der Datenbank mit der Prozesslogik verknüpft werden.

3.4 Eclipse Activiti Projekt - Die Umsetzung

Im folgenden Kapitel wird vorgestellt, welche besonderen Aspekte bei der Erstellung des Geschäftsprozesses im Eclipse mit der Erweiterung des Activiti Eclipse designers berücksichtigt werden müssen, um sowohl menschliche (manuelle) als auch maschinelle (automatisierte) Tätigkeiten zu realisieren. Für menschliche Aktivitäten werden im Activiti Explorer einzelne Personen und Personengruppen angelegt, die im Activiti Eclipse designer mit ihrer persönlichen Identifikation angesprochen werden können (Zuweisung der Benutzer und oder Benutzergruppen an ihre Aufgaben mittels eindeutiger Kennung, die *id*).

Für das fiktive Prozessszenario werden als Mindestanforderung im Hinblick auf die Bearbeitung von Aufgaben eine Benutzertrennung (*separation-of-duties*), sowie Benutzerbindung (*binding-of-duties*) ermöglicht, ein Webservice und eine Datenbankanbindung eingebunden, sowie ein zweiter Prozess aufgerufen werden. Des weiteren wurden Aspekte wie das Versenden elektronischer Nachrichten (e-Mail) und weitere automatisierte Aktivitäten umgesetzt, die auf der Programmiersprache Java beziehungsweise auf JavaScript basieren.

Nachdem die Umsetzung hinsichtlich der kritischen Faktoren abgeschlossen ist, wird am Ende des Kapitels das fertige fiktive Geschäftsprozess-Szenario vorgestellt. Der zugrundeliegende Workflow, sowie die Funktionen der Activiti BPM Platform selbst dienen anschließend als Grundlage für die nachstehende Schwachstellen-Analyse.

Inspiriert wurde das fiktive Szenario von [24], worin das Beispiel einer Kreditkartensperre wesentliche Bestandteile, wie die Zugriffskontrollen und beteiligte Entitäten (Rollen, Datenbank, Dienste, etc.), aufzeigt. In Anlehnung an erwähntes Beispiel handelt es sich auch in dieser Arbeit um das Sperren von Kreditkarten und gegebenenfalls eine Kartenerneuerung (auf Wunsch).

Die Zugriffskontrollen *separation-of-duties* und *binding-of-duties* werden zwar ebenfalls in diesem Prozess implementiert, allerdings wird der Aspekt *separation-of-duties* durch zwei getrennte und aufeinander folgende Aktivitäten deutlich gemacht. Die Person, die die erste Aktivität bearbeitet hat, darf die darauf folgende zweite Aktivität nicht bearbeiten (beide Personen sollen jedoch die gleiche Rolle besitzen). Auf diese Weise wird auch bei Ablauf des Prozesses deutlich, welche Aktivität noch nicht bearbeitet wurde.

3.4.1 Kritische Faktoren

Die sogenannten „kritischen Faktoren" im Hinblick auf den Geschäftsprozess sind besondere Gegebenheiten und Aspekte des Geschäftsprozessablaufes, welche bei nicht korrekter Handhabung oder Implementierung zu Fehlverhalten oder Fehlfunktionen führen und folglich eine Schwachstelle bedeuten kann.

Diese Aspekte sind im Groben, wie folgt:

- die definierten Rollen und Zuständigkeiten
- ihre Berechtigungen und Einschränkungen,
- das kurzfristige Abgeben der (menschlichen) Kontrolle,
- der angebundene externe Dienst, wie der Web Dienst (*web service*),
- die Entscheidungsknoten, vor allem nach automatisierter Aufgaben,
- interne Automatisierungen, wie Datenbankzugriffe und diverse Berechnungen (*JavaScript, Java*);

Im folgenden werden die kritischen Faktoren betreffend des zu erstellenden Prozessmodells mitsamt ihrer Umsetzung vorgestellt. Anschließend werden diese Aspekte aus zwei Blickwinkel betrachtet.

3.4.2 User und Groups

Zu den bereits vordefinierten Benutzer und Benutzergruppen (auch Demo Users genannt) im Activiti Explorer[11] werden weitere Personen hinzugefügt, die für das vorliegende Geschäftsprozess-Szenario und vorrangig für die strikte Trennung der Arbeitsabläufe notwendig sind. Damit wird eine hierarchische Unternehmensorganisation ersichtlich und folgende Personen sind im Prozess beteiligt (siehe Abbildung 3.3, mit Ausnahme der vordefinierten Benutzer).

Die Rolle „admin" ist als Initiator für den Prozessstart zuständig, während die Fachkräfte aus Abwicklung und Kontrolle manuelle Aktivitäten ausführen und somit ebenso, wie die maschinellen Aktivitäten, Einfluss auf den Kontrollfluss ausüben. Die Rolle „büro" teilt sich Tätigkeiten bei der Antragserstellung und Antragsüberprüfung. Hier spielt es eine Rolle, wer welche Tätigkeit zuerst übernimmt. Die Rolle „kontrollor" überprüft und kontrolliert sämtliche Unterlagen und ist zuständig für die Auftragsfreigabe der Kartensperre (und Karten-Erneuerung). Hier ist der entscheidende Aspekt dieser, dass die Kontrolle und Freigabe von nur einer Person durchgeführt wird. Jeder Teilnehmer hat eigene Zugriffsrechte im Activiti Explorer, wobei lediglich der „admin" erweiterte Administrationsfunktionen einsehen und ausführen kann ("Manage"-Funktionen). Diese Rollen und Berechtigungen werden in der Schwachstellenanalyse eine wesentliche Rolle spielen, da eine Ausnutzung oder Umgehung von Berechtigungen zu falschen Ergebnissen und unkontrollierbaren Prozessabläufen führen kann.

[11] http://www.activiti.org/userguide/activiti.setup

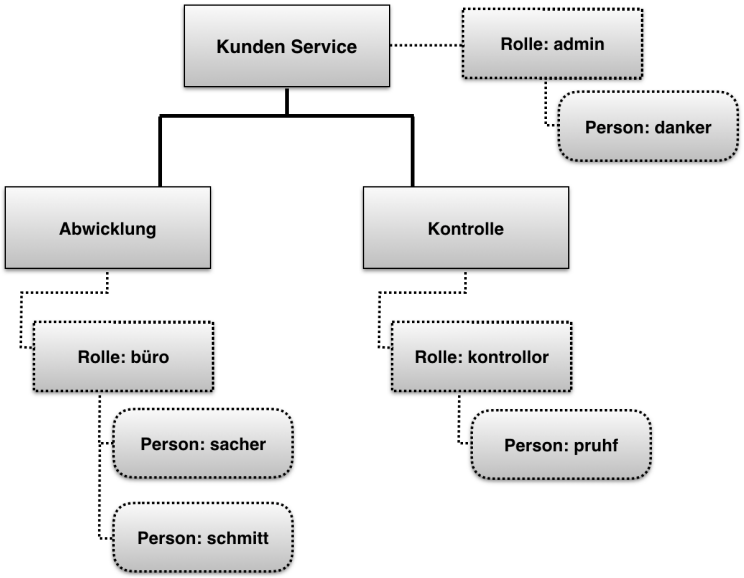

Abbildung 3.3: Fiktive Unternehmensorganisation (eigene Darstellung)

3.4.3 Webservice

Die Anbindung einer elektronischen Dienstleistung, eines externen Webservices, soll ermöglichen, dass maschinelle Aktivitäten ausgeführt und somit die Kontrolle von den teilnehmenden Personen kurzweilig abgegeben wird. Aus der strategischen Sicht werden Webservice-Anbieter für die Auslagerung von einzelnen hoch standardisierbaren Aufgaben bestimmt. Die Webservices bleiben dabei Bestandteil des Gesamtprozesses und übernehmen stets deutlich abgrenzbare Teilaufgaben, während sie in das WfMS (und generell in die Informationssysteme des Unternehmens) integriert werden.[34]

Nach Beendigung des Webservice-Aufrufs wird der Prozess, wenn keine andere Anweisung vorliegt, an dieser Stelle fortgeführt. Wesentlich ist dabei, dass der kurzweilig „wartende" Workflow auf eine Rückmeldung warten muss, damit er fortgeführt werden kann. Die Rückmeldung eines externen Services an dieser Stelle ist ein wichtiger Aspekt, der für die Schwachstellenanalyse eine Rolle spielt (schließlich muss nicht jede Rückmeldung harmlos sein).

Die Umsetzung ist nicht trivial, da stark unterschiedliche Herangehensweisen bei der Implementierung sowohl eines RESTful Dienstes als auch eines SOAP Services zur Verfügung stehen. Der Webservice in dieser Arbeit wurde mittels SOAP realisiert und ein öffentlich zugänglicher Dienst[12] in Anspruch genommen. Im Beispielszenario dieser Arbeit wird der Webservice dafür genutzt, um zu überprüfen, ob es sich bei den mitgelieferten Daten (die Bankleitzahl) unter anderem, um eine deutsche Bank handelt. Hierfür wird im Activiti Eclipse designer ein *ServiceTask* mit Informationen über die registrierte WSDL-Verbindung, die mitzuliefernden und zu empfangenden Werte in Form von Variablen ergänzt. Der folgende in der Programmiersprache Java geschriebene Programm-Ausschnitt zeigt den Aufruf des Webservices und die notwendigen Rückgabewerte für das im Prozessbeispiel nächstfolgende Flussobjekt (siehe Listing 3.4).

Listing 3.4: Webservice Aufruf - Ermitteln der Bankleitzahl

```
1 try{
2 //ueber Activiti Explorer uebergebene Parameter abrufen und weiterleiten
```

[12] http://www.thomas-bayer.com/axis2/services/BLZService?wsdl

```
3   String blz = (String) execution.getVariable("bankleitzahl");
4   DetailsType details = serv.getBLZServiceSOAP12PortHttp().getBank(blz);
5   response_collection = details.getBezeichnung() + " ";
6   //weitere Bank-Details moeglich zu empfangen
7   execution.setVariable("returnValue", response_collection);
8   ausgabe = true;
9   execution.setVariable("returnValue", ausgabe);
10 }catch(Exception exception){
11   System.out.println("Keine Deutsche BLZ - kein Bonus..."); execution.
         setVariable("returnValue", ausgabe);
12 }
```

Das anschließende Ergebnis des Webservice-Aufrufs ist somit entscheidend für den weiteren Prozessverlauf. Einerseits wird eine mögliche Fehlermeldung „abgefangen", sodass der Workflow an dieser Stelle nicht abbricht. Andererseits wird das Ergebnis für eine nachstehende Berechnung benutzt, weshalb `ausgabe=true` (Standardwert ist `false`) notwendig ist.

3.4.4 DB Konzept

Nachdem der Prozess kurz vor dem Abschluss steht, werden Daten in der MySQL-Datenbank festgehalten, um beispielsweise statistische Abfragen gewährleisten zu können. Realisiert wird dieser Datenbankaufruf- und ihre Eintragung ebenso mittels eines *ServiceTasks* und einer zugrundeliegenden Java-Klasse (siehe Listing 3.5).

Listing 3.5: SQL Datenbank-Anweisung für den Datenbank-Aufruf

```
1  connection();
2  kartenservice.setBooleanwert(ergebnis); // wird im Workflow empfangen
3  long kartennum = (Long) execute.getVariable("kartennummer");
4  try {
5    statement = connection.createStatement();
6    if(ergebnis == true){
7      sql = "insert into kartenservice values('1','" + kartennum + "','gesperrt
           ');";
8      statement.executeUpdate(sql);
9      statement.close();
10     connection.close();
11     System.out.println("Karte mit deutscher BLZ wurde gesperrt...");
12   }else{
13     sql = "insert into kartenservice values('0','" + kartennum + "','gesperrt
           ');";
14     statement.executeUpdate(sql);
15     statement.close();
16     connection.close();
17     System.out.println("Karte mit nicht-deutscher BLZ wurde gesperrt...");
18   }
19   dbantwort = true;
20   execute.setVariable("dbeintrag", dbantwort); //Antwort an den Workflow
21 } catch (Exception exception) {
22   System.out.println("Leider Fehler beim Verbinden! Admin beauftragen...");
23   //exception.printStackTrace();
24   execute.setVariable("dbeintrag", dbantwort);
25 }
```

In der Datenbank wird die mitgelieferte Kreditkartennummer gespeichert, sowie der Vermerk getätigt, ob es sich um eine deutsche Bank handelt, die Karte gesperrt und ob eine Erneuerung

erwünscht ist. Die Rückmeldung der Datenbankeintragung beeinflusst den weiteren Prozessverlauf. Entweder muss ein Administrator benachrichtig werden oder der Prozesslauf wird (erfolgreich) beendet.

3.4.5 Prozessaufruf

Da es sich bei dem Beispielszenario um einen Prozess handelt, der vermehrt administrativer Natur ist (Kundenservice), wird mittels eines Prozessaufrufs die eigentliche Sperrung und auf Wunsch die Erneuerung einer Kreditkarte veranlasst. Ein solcher Aufruf erfolgt ebenso automatisiert in einem *ServiceTask* ergänzt mit einem einfachen Java-Programm (Listing 3.6).

Listing 3.6: (Klassischer) Programmatischer Prozessaufruf

```
1  // Standard-Aufruf mit Uebergabe der Prozess-ID
2  public void execute(DelegateExecution execution) throws Exception {
3      RuntimeService runtimeService = execution.getEngineServices().
           getRuntimeService();
4      runtimeService.startProcessInstanceByKey("kreditkartensperren");
5  }
```

Der Prozess „Kreditkartensperren" wird als eigenständiger Prozess geführt und von einer weiteren Einheit bearbeitet. Dieser Aufruf kann ab den neuen Versionen des Activiti BPM ebenfalls vereinfacht mittels einer *CallActiviti* realisiert werden. Ein Aufruf eines weiteren Prozesses ist relevant für die Schwachstellenanalyse, da auch hier auf eine Rückmeldung „gewartet" wird. Eine Manipulation in diesem Bereich kann ein Fehlverhalten verursachen, sowohl im aufrufenden Prozess, als auch im aufgerufenen Prozess.

3.4.6 Zusätzliche Funktionalitäten

Die ausgewählte E-Mail-Funktionalität kann mittels eines *MailTask* realisiert werden und auch hier muss die Activiti Konfigurations-Datei `activiti-custom-context.xml` entsprechend ergänzt werden (Listing 3.7). Diese Datei ist für die Schwachstellen-Analyse relevant, weil darin wesentliche Informationen zur Verfügung stehen (wie beispielsweise das Passwort).

Listing 3.7: Konfiguration für den Mail-Task

```
1  <bean id="processEngineConfiguration"
2    % Anbindung an Google Mail
3    <property name="mailServerHost" value="smtp.gmail.com"/>
4    <property name="mailServerPort" value="587"/>
5    <property name="mailServerUseSSL" value="false"/>
6    <property name="mailServerUseTLS" value="true"/>
7    <property name="mailServerDefaultFrom" value="master.activiti@gmail.com"/>
8    <property name="mailServerUsername" value="master.activiti@gmail.com"/>
9    <property name="mailServerPassword" value="xxx"/>
10   % Passwort wird hier eingetragen
11 </bean>
```

Im Mail Task werden mindestens die Absender- und Empfängeradresse angegeben, sowie der Nachrichtentext, der vom Grundgerüst her während der Ausführung nicht mehr verändert werden kann. Änderungen können nur in der Prozessdefinition vor Einbindung in den Activiti Explorer vorgenommen werden (*a priori*). Auf der Seite des Mail-Servers muss in den Einstellungen eine Funktionalität bezüglich der Mail-Sicherheit „deaktiviert" werden, ohne diese die Verbindung zwischen Activiti und dem Mail-Server nicht zustande kommen kann. Bei dieser Funktionalität handelt es sich um „Zugriffe von weniger sicheren Apps" (Abbildung 3.4)

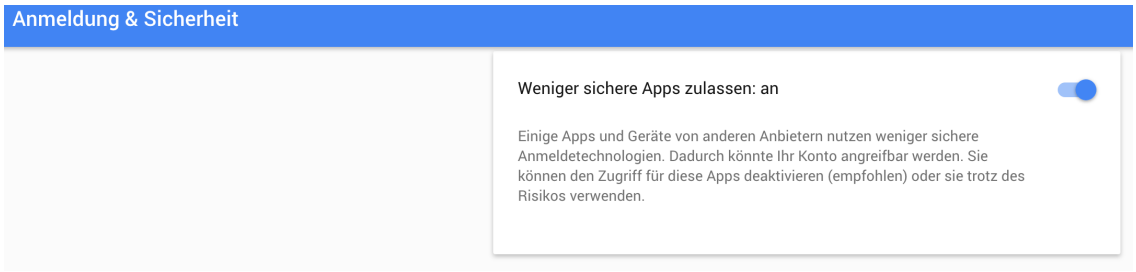

Abbildung 3.4: Sicherheit in den e-Mail Einstellungen (Screenshot)

Statt manuellen Tätigkeiten können jederzeit programmatische Funktionen implementiert werden, die keine Benutzereingaben benötigen. Diese Implementierungen können entweder in das Modellierungselement direkt als Skript (*Java Script, Groovy*) eingetragen oder in einer ausgelagerten Java-Datei umgesetzt werden. Im vorliegenden Beispielszenario wird nach dem Web-Dienst eine Bonus-Berechnung durchgeführt, welche kein manuelles Zutun mehr verlangt. Ein Beispiel sieht wie folgt aus (Listing 3.8):

Listing 3.8: JavaScript für Berechnungen

```
1 euro = "100Euro"
2 steine = "100Kiesel"
3
4 if(bankleitzahl > 12000000) {
5 execution.setVariable("bonuswahl", euro)
6 }
7 else {execution.setVariable( "bonuswahl", steine) }
8 java.lang.System.out.println("Script "Bonusbestimmung" ergibt: " + bonuswahl +
      "\n")
```

Sämtliche (teil-)automatisierte Aufgaben dieser Art müssen vor unbefugten Zugriffen geschützt sein, um einer Manipulation des Prozessablaufs entgegen zu wirken und die Ergebnisses nicht zu verfälschen.

3.4.7 Geschäftsprozess-Gesamtbild

In diesem fiktiven Szenario handelt es sich aus der externen Sicht betrachtet um ein schnelles, kostengünstiges und zeitunabhängiges Unternehmen, welches Kreditkartensperren und Erneuerungen auf Wunsch veranlasst. Ein Kundenservice stellt dabei die erste Anlaufstelle dar. In der Datenbank werden keine persönlichen Daten, wie Namen oder Adressen gespeichert (die eigentliche Person ist dem verarbeitenden Kreditinstitut bekannt) aber für statistische und informative Zwecke werden unter anderem die gesperrten Kartennummern abgespeichert.

Aus der internen Sicht betrachtet werden unmittelbar nach dem Start des Geschäftsprozesses die Kartennummer, Bankleitzahl und der Erneuerungswunsch entgegen genommen. Diese Pflichtdaten werden gegengeprüft (dabei soll die Prüftätigkeit nicht von der gleichen Person übernommen werden), und ein anschließender, öffentlicher Web-Dienst erledigt die Frage nach der Bank. Handelt es sich um eine deutsche Bank, so erhält der Kunde einen Bonus gutgeschrieben (in Form einer Geldgutschrift oder von Gutschein-Punkten), ansonsten nicht. Bevor der Auftrag an die Kreditkarten-Abteilung erfolgt, wird nochmals auf Korrektheit gegengeprüft (die prüfende Person ist anschließend auch zuständig für die Freigabe des Auftrages). Die Auftragserteilung löst einen weiteren Prozessaufruf aus (die Sperre der Karte und ggf. die Erneuerung), der bestenfalls „beendet" wird oder mit einer Nachricht antwortet, dass der Auftrag (aus diversen Gründen) „nicht beendet" werden konnte. Tritt dieser Fall ein, wird mittels Script automatisiert der

Manager informiert. Anschließend wird der Auftrag in der Datenbank persistiert. Sollte diese Eintragung fehlschlagen, muss ein Administrator beauftragt werden, der sich um das Problem kümmert. Der Prozess endet mit einer abschließenden Archivierung. Abbildung 3.5 zeigt den vereinfachten Gesamtumfang dieses Geschäftsprozesses.

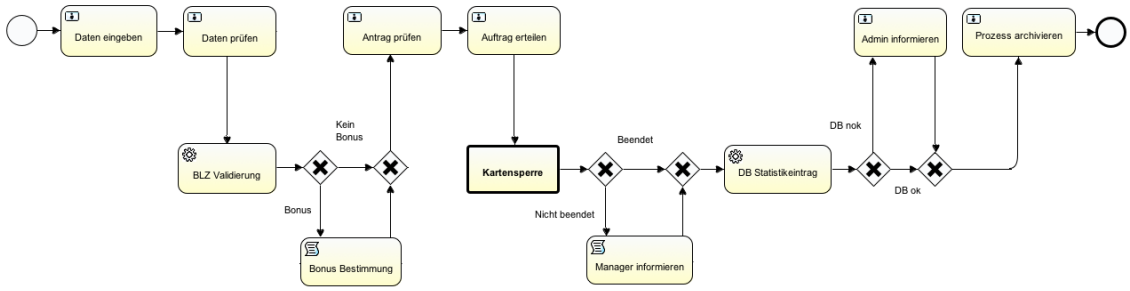

Abbildung 3.5: Geschäftsprozess - Fiktives Gesamtszenario (eigene Darstellung)

Die Vereinfachung zeigt den üblichen Verlauf des Prozesses und etwaige Ausnahmepfade wurden für das Beispielszenario entfernt. Nach einer „Datenüberprüfung" könnte ein Pfad zur vorangegangenen Aufgabe zurückführen, falls aufgrund der Überprüfung ein Fehler entdeckt wurde. Gleiches gilt für die Aufgabe „Auftrag prüfen". Diese Eventualitäten wurden zur Vereinfachung aus dem Modell entfernt.

Nachdem der Prozess im Activiti Eclipse designer[13] modelliert wurde, müssen nächste Schritte mittels der „Maven"-Funktionalität durchgeführt werden, wodurch wesentliche Abhängigkeiten (als Bibliotheken) dem Projekt hinzugefügt werden. Damit der modellierte Geschäftsprozess und der Workflow im Activiti Explorer reibungslos ausführbar ist, müssen sämtliche in der Eclipse Programmierumgebung erstellten Java-Klassen auch für den Activiti Explorer zugänglich gemacht werden. Anschließend kann der Workflow im Activiti Explorer von einer berechtigten Person zum Einsatz gebracht werden (*deploy*) und der Durchführung des Prozesses steht nichts mehr im Weg, vorausgesetzt alle notwendigen Einstellungen sind getätigt (die im Modell angeführten Benutzer und Benutzergruppen müssen bereits im Activiti Explorer zur Verfügung stehen. Eine Erstellung nach dem der Prozess eingeführt wurde, hat keinen Einfluss auf den bereitgestellten Prozess).

Bereits während der Modellerstellung müssen auf die Einhaltung von Regeln in der Modellierung, Abdeckung möglicher Fehlerpfade und korrekte Zuweisung von Personen an die jeweiligen Aktivitäten geachtet werden. Diese und weitere Aspekte sind, wenn sie unbeachtet bleiben, potentielle Fehlerquellen und können dadurch Schwachstellen im System bedeuten.

[13] Activiti Eclipse BPMN 2.0 designer plug-in für das Erstellen und Testen der BPMN 2.0 Prozesse

4 Schwachstellenanalyse

Potentielle Sicherheitslücken und Schwachstellen müssen so früh, wie möglich erkannt werden. Bei einer organisatorischen Überprüfung werden Prozesse des Unternehmens auf Mängel untersucht, während allgemeine Fehler bei der Betrachtung von konzeptionellen Bereichen gefunden werden können. Doch erst eine technische Überprüfung ermöglicht die eigentliche Bestimmung von Fehler und Schwachstellen. Solch eine Analyse kann sowohl auf Host-Ebene als auch auf Netzwerkebene durchgeführt werden. Das Ziel einer solchen Analyse ist durch Überprüfungen und intensive Untersuchungen mögliche Schwachstellen und Sicherheitslücken einer Lösung ausfindig zu machen.[35]

In den nachfolgenden Abschnitten wird die Methodik und die Vorarbeit zur Erstellung der Attack Trees beschrieben. Nach der Vorstellung der erarbeiteten Attack Trees werden die Analysen und Versuche für auserwählte Pfade durchgeführt. Ein abschließender Überblick über die Ergebnisse schließen das Kapitel der Schwachstellenanalyse ab.

4.1 Methodologie

Das vorliegende Kapitel beschäftigt sich mit einer Angriff-orientierten Schwachstellenanalyse des zugrunde liegenden WfMS, dem Activiti PBM, wobei auf die Methode der Attack Trees zurückgegriffen wird. Es wird als Herausforderung darüber hinaus davon ausgegangen, dass keinerlei Schwachstellen des vorliegenden Systems bekannt sind.

Um Attack Trees sinnvoll aufzustellen, wurde als Vorarbeit zunächst untersucht, welche Ziele von den Angreifern grundsätzlich verfolgt werden. Anschließend wurde der bereits definierte Geschäftsprozess aus zwei unterschiedlichen Perspektiven betrachtet, um daraus gemeinsam mit den Angriffsabsichten mögliche Ziele für die Attack Trees abzuleiten.

4.1.1 Angriffsabsichten und Kategorie

Mithilfe der zehn häufigsten Risiken aus OWASP [1], sämtlichen Angriffsmöglichkeiten aus [35] und den Informationen aus Microsoft Developer Network[2] konnte ein Überblick über mögliche Angriffsziele, Sicherheitsaspekte und Angriffe allgemein verschafft werden. Es wurden diverse Angriffsabsichten identifiziert, die aufgrund der Häufigkeit ihres Auftretens in Kategorien unterteilt werden können. Die Absichten dienten als Grundlage für mögliche Angriffsziele und Szenarien der Attack Trees.

Nach Zusammenführung und Auswertung obiger Literaturrecherche zielen die häufigsten Systemangriffe darauf ab, Zutritt zum entsprechend System zu erlangen (Passwort und Benutzername zu erraten oder die Authentifizierung generell zu umgehen). In erster Linie wird also versucht, sich Zutritt zum System zu verschaffen und aus Sicht der Anwendungssicherheit, handelt

[1] OWASP: www.owasp.org, letzter Zugriff Mai, 2016
[2] MSDN, https://msdn.microsoft.com/en-us/library/ff648032.aspx, letzter Zugriff Mai, 2016

	Ziel
A	Benutzerdaten, Zugang erlangen Passwort erraten IP-Authentifizierung umgehen Entschlüsselung
B	Berechtigung ändern Höhere Berechtigung erlangen Täuschen, Nutzerrechte umgehen Zugriff auf unautorisierte Daten Zugriff auf private Funktionen Unsichtbare Source Daten manipulieren
C	(DB) Befehle einschleusen Daten ausspähen, manipulieren
D	Benutzersitzung übernehmen Session ID stehlen
E	Seiteninhalt ändern
F	Umleiten
G	Ressourcen, Verfügbarkeit einschränken, Systemabsturz hervorrufen
H	Spionage, Abhören während Datenverkehr

Abbildung 4.1: Überblick der häufigsten Angriffsabsichten (eigene Darstellung)

es sich um wesentliche Aspekte der Eingabe-Validierung, Authentifizierung, Sensiblen Daten, des Konfigurationsmanagements sowie Session-Managements. Mit diesem ersten Schritt befasst sich der nachstehende Attack Tree „Zutritt und Zugangsdaten zu Activiti erlangen" in Abschnitt 4.4. Es werden mögliche Wege für dieses Szenario aufgedeckt, besonders auch die unterschätzten Möglichkeiten, bei dem kein aktives „angreifen" benötigt wird.

Die zweit häufigste Kategorie betrifft den Aspekt der Berechtigungen. Es stellt sich hier zum Beispiel die Frage, ob es möglich ist, dass eine Person mit bestimmten (eingeschränkten) Nutzungsrechten auch Funktionen ausführen kann, wofür sie sonst keine Berechtigungen besitzt (Rechte eines Administrators). Hier kann eine systematische Analyse der aufgerufenen Webseiten beider Berechtigungsgruppen durchgeführt werden. Auf der anderen Seite spielt auch hier ein „Insider" eine Rolle, der seine (eventuell auch übergeordneten) Berechtigungen ausnutzen kann, um den Prozesslauf durch Manipulation zu beeinflussen.

Die dritte Kategorie betrifft die Einschleusung (*injections*) von Daten und oder Quelltext und bezieht sich meistens auf die verwendete Datenbank. Dabei wird erprobt, ob Mechanismen zur Eingabeüberprüfung umgesetzt sind. Gelingt es einem Angreifer auf diese Art Daten einzuschleusen, kann dieser Erfolg auch zu einer Löschung der gesamten Datenbank führen und somit auch zu einem Fehlverhalten des Workflows, da wesentliche, für einen reibungslosen Ablauf notwendige Daten und Informationen dadurch verloren gehen. Die Thematik der zugrundeliegenden Datenbank und die betreffenden Gefahren werden mit dem Attack Tree „Durch Datenbank Manipulation Fehlverhalten auslösen" (Abschnitt 4.5) aufgegriffen.

Kategorie vier befasst sich mit der Übernahme von Funktionen bei dem die Sitzungsinformationen der im System angemeldete Personen übernommen werden, sodass dies einem „Zugang erlangen" ähnelt. Unterschied hierbei ist, dass dafür keine Passwörter oder Benutzerdaten benötigt werden. Diese können allerdings im Anschluss erlangt werden. Die Übernahme auf diese Weise wird in den Attack Trees als solches nicht dargestellt. Gleiches gilt für die nächst folgenden Kategorien fünf und sechs im Bezug auf Abänderung des Aussehens einer Web-Seite und die Umleitung auf eine nicht vorgesehene Web-Seite.

Kategorie sieben hingegen betrifft die Verfügbarkeit sämtlicher Komponenten, die mit dem Zielsystem in Verbindung stehen. Attack Tree „Verwendete Ressourcen manipulieren" (Abschnitt 4.6) geht auf diese Problematik ein. Activiti verwendet eine Datenbank, nimmt einen externen Web-Dienst an, läuft auf einem Webserver und beinhaltet sämtliche Implementierungen, die als Bibliothek-Ressourcen zur Verfügung stehen müssen, um reibungslos zu funktionieren. Werden diese Ressourcen nun manipuliert, kann dies mit unter zerstörerische Auswirkungen auf Activiti haben.

Die achte und letzte Kategorie betrifft passive Aktivitäten zur Erlangung von Informationen oder Daten. Daten können dabei währen der Laufzeit und des Datenverkehrs abgefangen und mitgeschnitten werden. Mittels eines geeigneten Schadprogrammes ist es möglich, Spionagefunktionen auszuüben, so zum Beispiel auch die Aufzeichnung von Tastatureingaben, um Zugangsdaten zu erlangen.

Um zu Erfahren mit welchen Mitteln Angreifer vorgehen, um eine gefundene Schwachstelle auszunutzen, werden im nächsten Kapitel die häufigsten Angriffe vorgestellt.

4.1.2 Häufige Angriffsarten

Die häufigsten Angriffsarten werden unter anderem auch in den Top 10 Risiken von OWASP vorgestellt (nähere Details, vor allem auch sämtliche Untertypen der im Folgenden aufgelisteten Angriffsarten, sind dort unter anderem in der Rubrik „Angriffe" nachzulesen)[3]. Die folgende Auflistung gibt einen groben Überblick über die in der Literatur am häufigsten angeführten Angriffsarten sowie eine kurze Beschreibung und dient als Anregung für die Konstruktion und Durchführung der anschließenden Angriffsbäume.

- SQL Injections (Structured Query Language injections)
  Angreifer schleusen (*inject*) schadhafte Eingaben über eine Web-Anwendung in ein Untersystem ein. Dabei werden die Eingaben nicht validiert oder unsicher benutzt, sodass besondere Sonderzeichen die eigentliche SQL-Abfrage zugunsten der Angreifer zu eigenen Abfragen umwandeln.

- Social Engineering
  Ein Vorgehen, um Menschen (Mitarbeiter) dazu zu bewegen, Informationen preiszugeben. Dabei können die betroffenen Personen auch unbeabsichtigt Aktionen ausführen, die Vorteile für die Angreifer bedeuten (öffnen eines schadhaften Anhangs in einer manipulierten e-Mail Nachricht)

- Cross Site Scripting (XSS)
  XSS Attacken sind ebenfalls eine Art Einschleusung, wobei schadhafter Code (scripts) in vertrauenswürdige Web-Seiten eingeschleust werden. Sobald bei einer Dateneingabe im Web-Browser keine Überprüfung stattfindet, kann solch eine Attacke unter anderem ausgeführt werden

- Clickjacking
  Die Darstellung einer Internetseite wird überlagert, um den Benutzer dazu zu veranlassen, scheinbar harmlose Mausklicks und oder Tastatureingaben durchzuführen. Es wird nicht auf das eigentliche (zu erkennende Feld) gedrückt, sondern auf die vorgelagerte (unsichtbare) Fläche, welche eine Weiterleitung auf eine andere Seite veranlasst. Dadurch können Passwort oder Kontendaten-Eingaben an Dritte weitergeleitet werden

- Malware (Malicious Software [41])
  Schadhafte Software, wie Würmer, Viren, Trojaner und Spionage-Programme, die oft auch unentdeckt bleiben. Jedes dieser Unterarten hat eine eigene Auswirkung auf das System

[3] Weitere Definitionen und Details unter www.techopedia.com, letzter Zugriff April, 2016

- Buffer Overflow
  Angriffe durch Pufferüberlauf sollen ein Programm dazu bringen, einen bestimmten Code auszuführen, dazu werden größere Datenmengen als vorgesehen übermittelt. Dabei werden Speicherfragmente überschrieben und das System oder Programm wird dadurch bestenfalls funktionsunfähig
- Brute-Force Attacke
  Angriffsmethode nach dem trial-and-error Prinzip (Versuch-und-Irrtum), um mithilfe von Programmen (da eine sehr große Anzahl an Versuche durchgeführt wird) sensible Daten herauszufinden
- Insecure direct object reference
  Ein Verweis auf ein internes Objekt (Verzeichnis, Datei) [9]; Durch das Fehlen einer Zugriffskontrolle wird es möglich durch manipulierte Verweise (URL, Parameter) auf Objekte zuzugreifen, für die der Angreifer nicht autorisiert ist
- Backdoor Attacke
  Backdoors sind Implementierungen, um eine Authentifizierung zu umgehen und können sich in Betriebssystemen und Anwendungen befinden. Online zugängliche Programme stehen zur Verfügung, um Backdoor Attacken zu Erstellen
- Phishing (Kunstwort aus Password und Fishing [42])
  Angreifer versucht durch Manipulation von Webseiten, e-Mails und Links sensible Daten zu erlangen (siehe auch Social Engineering)
- Password cracking
  Unterschiedliche Aktionen, um ein Passwort zu erraten. Dabei kann zum Beispiel ein Computerprogramm wiederholt Wortkombinationen ausprobieren bis das Losungswort gefunden wird
- Denial-of-Service
  Dienstverweigerung; Ein Angreifer versucht den Zugriff auf Systeme oder Ressourcen für legitime Benutzer durch Auslastung zu blockieren (der Netzwerk oder Server Dienst wird verweigert)
- Exploit
  Software oder andere Methoden zur Ausnutzung von Schwachstellen und Sicherheitslücken, um Zugriff auf das Zielsystem durchzuführen oder dort höhere Berechtigung zu erlangen. Exploit enthält schadhaften Quelltext, der nach der Ausnutzung der Schwachstelle auf das Zielsystem den böswilligen Effekt bewirkt. (Exploits sind ähnlich wie Einbrecher mit Brechstangen: sie verschaffen sich Zugang zu Dingen, für die sie keine Berechtigung besitzen)

4.2 Angreifer und Angriffsebenen

Alle vier in Abschnitt 2.6.1 beschriebenen Angreifer-Typen verfolgen zumindest ein gemeinsames Ziel, nämlich sich Zugang zu einem System zu verschaffen oder dieses auf ihre eigene Art zu manipulieren. Anschließend können beispielsweise schadhafte Programme installiert und oder Informationen und Daten entnommen oder verändert werden. Für ein WfMS bedeutet dies, dass neben den üblichen Gefahren gegenüber der Web-Applikation auch spezifische Funktionen gefährdet sind. Prozesse können derart manipuliert werden, sodass es zu fehlenden oder unkontrollierten Prozessaufrufen kommt, Datenbankeinträge fehlerhaft beendet, Personengruppen verändert oder falsche Pfade gewählt werden. Daraus folgt, dass die Schwachstellenanalyse verschiedene Ebenen umfasst und Gefahren nicht nur von externen Angreifertypen (Outsider), sondern auch von registrierten und berechtigten Personen (Insider) ausgehen kann. Auch dieser Aspekt wird in weiterer Folge in den Attack Tree Beispielen aufgenommen.[36][35]

4.2.1 Unterschiedliche Blickwinkel

In [24] wird das Thema Informationssicherheit in Process-aware Information Systems (PAIS) diskutiert, wonach eine ganzheitliche Herangehensweise zur Deckung der Sicherheit auf allen Ebenen und Rollen angesprochen wird. Für eine ganzheitliche Sicherheitsanalyse sind besonders folgende vier Ebenen relevant:

1. Die Geschäftsprozess-Ebene, in der nicht nur die teilhabenden Entitäten, sondern auch das Ergebnis und ein Kundennutzen eine wesentliche Rolle spielen.
2. Die System-Design-Ebene betrifft die Systemarchitektur, ihre Komponenten (zB. Anwendungen, Dienstleistungen, etc.), Schnittstellen und Daten.
3. Die Implementierungs-Ebene betrifft Technologien zur Realisierung des System-Designs und zur Implementierung und Ausführung von Geschäftsprozessen.
4. Die über alle Ebenen hindurch gehende Mensch-Ebene, welche die teilnehmenden Rollen bezeichnet und die Systemsicherheit auf aktive aber auch auf passive Art beeinflusst.

Diese Ebenen spielen eine Rolle in den nachfolgenden Attack Tree Szenarien. Soll ein Geschäftsprozess attackiert werden, sollte die angreifende Person zumindest Wissen darüber haben, wer oder was daran beteiligt ist und welche Aktivitäten ausgeführt werden. Für die nachfolgenden Szenarien sind sowohl „Insider-Wissen" als auch „Outsider-Wissen" vorhanden. Die zugrundeliegenden Technologien (Implementierungsebene) spielen eine wesentliche Rolle hinsichtlich der Sicherheit in dieser Domäne. Durch die Komponente Mensch erfährt die Sicherheit weitere negative Einflussfaktoren: Menschen stellen zusätzlich zu der erwähnten Fahrlässigkeit eine Schwachstelle dar, indem diese Fehler im Design, in der Implementierung, Konfiguration und in der Handhabung des Systems hinterlassen. Weiters sind Menschen als Angreifer (sowohl von Innen als auch von Außen) eine Gefahr für die gesamte Systemsicherheit. Aus diesem Grund müssen auch für die nachstehenden Attack Trees angreifende Personen und Rollen, im folgenden auch als Entitäten bezeichnet, identifiziert werden. In Anlehnung an [24] werden in der nächstfolgenden Analyse die Schwachstellen-relevanten Aspekte des fiktiven Geschäftsprozesses hervor gehoben.

4.2.2 Teilhabende Entitäten

Organisationale Aspekte, wie Rollen und Zuständigkeitsbereiche können aus Prozessmodellen direkt abgelesen werden. Andernfalls, wie es in der zugrundeliegenden Beispiel-Prozessdefinition der Fall ist, befinden sich diese Informationen auch in dem Prozess zugrundeliegenden XML-Dokument. Hier werden die teilhabenden Rollen ersichtlich: eine Führungskraft (*manager*), die verantwortlich dafür ist, dass der Prozess gestartet wird, eine Bürokraft (*büro*), die die Initialdaten angibt und verifiziert (diese Rolle muss von zwei Personen besetzt sein). Die dritte Rolle betrifft den Kontrollor (*kontrollor*), der die Anträge überprüft und Aufträge freigibt (beide Aktivitäten werden von der gleichen Person durchgeführt). Diese drei Rollen lassen sich in den jeweiligen Aktivitäten wiederfinden (siehe Listing 4.1).

Listing 4.1: XML-Prozessdefinition: Zuständige Rollen

```
1 \\Rolle manager: danker
2 <process id="prozessaufruf" name="Prozessaufruf" isExecutable="true" activiti:
      candidateStarterUsers="manager">
3 <startEvent id="startevent1" name="Start" activiti:initiator="manager">
4
5 \\Role buero: sacher, schmitt
6  <userTask id="pflichtdateneingeben" name="Daten eingeben" activiti:
       candidateGroups="buero">
```

```
7  <userTask id="pflichtdatenpruefen" name="Daten pruefen" activiti:
       candidateGroups="buero">
8
9  \\Rolle kontrollor: pruhf
10 <userTask id="antragpruefen" name="Antrag ueberpruefen" activiti:assignee="
       pruhf">
11 <userTask id="auftragerteilen" name="Auftrag erteilen" activiti:assignee="
       pruhf">
```

Dabei ist der *assignee* die Person, die die Aufgabe übernimmt (*claim*) oder zugeteilt bekommen hat. Es kann jedoch auch eine Person aus einer definierten Gruppe (candidateGroups) die jeweilige Aktivität für sich bestimmen (ebenso *claim*). Nach einem „claim" ist diese bestimmte Aufgabe für den Rest der Gruppe nicht mehr ersichtlich. Entsprechende Einschränkungen und Bedingungen werden in weiterer Folge beschrieben.

Zu den teilnehmenden Entitäten zählt eben so die Infrastruktur. Automatisierte Aktivitäten verbinden den Geschäftsprozess, und somit auch das WfMS, mit externen und öffentlichen Dienstleistung, genauer gesagt mit einem Webservice zur Ermittlung von bankspezifischen Informationen. Jedoch nicht nur extern zugängliche , sondern auch interne Funktionen werden im Prozess genutzt, wie die Ermittlung der Bonusart, nachdem der erfolgreiche Webservice-Aufruf stattgefunden hat und eine Rückmeldung empfangen wurde. Ein automatisierter Prozessaufruf startet einen weiteren Prozess, nach dem alle relevanten Daten positiv überprüft wurden. Den Anstoß dazu liefert eine manuelle Aktivität. Eine angebundene Datenbank zählt ebenso zu den teilhabenden Entitäten des Gesamtprozesses, bei denen die Aktivität automatisch ausgeführt werden. Sämtliche während der Laufzeit entstandenen Instanz-Variablen und Informationen werden am Prozessende in die Datenbank gespeichert.

Das Beispielszenario in Abbildung 4.2 beherbergt somit Aktivitäten, die automatisiert und vom Menschen manuell ausgeführt werden (manuelle Aktivitäten). Es wird eine Verbindung zwischen der Geschäftsprozess Ebene und der System-Ebene ersichtlich. Die maschinellen Aufgaben übernehmen stets für eine kurze Zeit die Kontrolle über den Prozess und verbinden sich einerseits mit externen Entitäten und andererseits rufen sie definierte Programme auf (in Form von Java-Klassen, die in der Prozessdefinition mit eingebunden sind), um die definierten Aufgaben zu erledigen. Diese jeweiligen Kommunikationsverbindungen gilt es vor etwaigen externen oder internen Angreifern zu schützen. Doch nicht nur die Verbindungen, sondern die Systeme und Programme selbst sind Gefahren ausgesetzt und verursachen bestenfalls eine fehlerhafte Weiterverarbeitung von essentiellen Informationen. In Folge hat das Ergebnis manipulierter Verarbeitung Einfluss auf den weiteren Werdegang des Geschäftsprozesses. Für das WfMS, welches für den Zugriff auf die Datenbank und anderen Programmen zuständig ist, bedeuten die Kommunikationskanäle zu anderen Systemen und Diensten kritische Faktoren und somit eine mögliche Schwachstelle.

4.2.3 Bedingungen und Einschränkungen

Im vorliegenden Prozess wird der Start erst durch den Manager ermöglicht und nur die Personen, die im Prozess der Rolle *büro* zugeteilt sind, dürfen die nachfolgenden Aktivitäten *Daten eingeben* und *Daten prüfen* durchführen. Diese Aktivitäten werden mit dem „4-Augen-Prinzip" durchgeführt, was dem „Seperation-of-Duties" (SoD) entspricht und verlangt, dass diese beiden Aufgaben von unterschiedlichen (aber der definierten Gruppe angehörigen) Personen durchgeführt werden müssen. Die darauffolgenden Aktivitäten sind automatisierte Aktivitäten, die kein menschliches Handeln benötigen. Erst in den Aktivitäten *Antrag prüfen* und *Auftrag erteilen* wird wieder menschliches Zutun erwartet und hier wird verlangt, dass beide Aufgaben von einer einzigen Person durchgeführt wird. Diese Bedingung entspricht dem Prinzip des „Bindung-of-Duties" (BoD). Die Prinzipen SoD und BoD sind sicherheitsorientierte Tätigkeitsbedingungen, wonach sichergestellt werden soll, dass beispielsweise „Betriebsblindheit" bei Überprüfungen

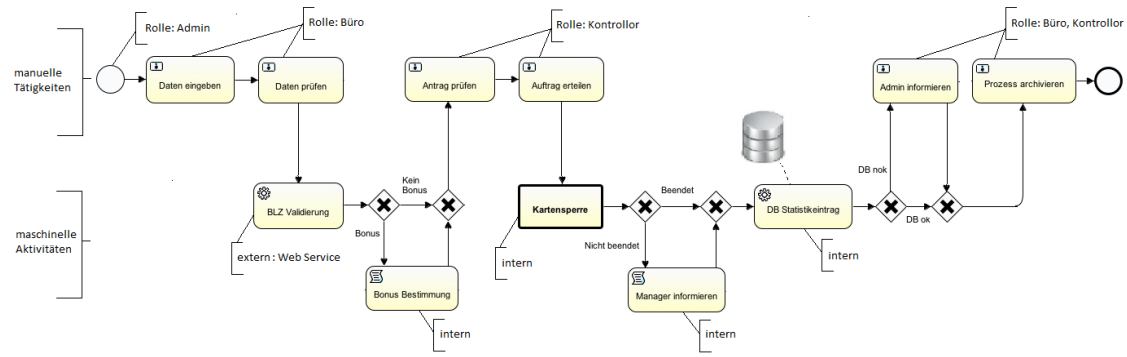

Abbildung 4.2: Gesamtprozess - Entitäten im Überblick (eigene Darstellung)

vermieden und korrekte Reihenfolgen der Abarbeitung bewahrt wird. Im Falle eines entdeckten Fehlers wird die überprüfende Person, die auch für den Start des Sendeauftrags zuständig ist (BoD), diesen Auftrag nicht erteilen.

Eine Person, die Zugang zu Berechtigungen und Authentifizierung der Prozessbeteiligten hat oder erlangt, hat die Möglichkeit den Prozess mithilfe von Berechtigungen zu gefährden. Können personenbezogene Einschränkungen und Bedingungen umgangen werden, kann jemand den Prozess auf einfache Art fortführen und manipulieren, ohne dass zum Beispiel eine Aktivität vorher notwendigerweise positiv überprüft wurde (Auftragsfreigabe, ohne Berechtigung). Bei unerlaubtem Zugriff auf Berechtigungen können weiters Prozessinstanzen mit falschen Angaben gestartet werden und unkontrollierte Prozessaufrufe sowie Datenbankeinträge verursachen. Solch eine Situation würde für ein WfMS auf eine mögliche Schwachstelle hindeuten, die sowohl von einem Insider aber auch von einem Outsider ausgenutzt werden kann. Abbildung 4.3 zeigt welche Aktivitäten von SoD und BoD betroffen sind.

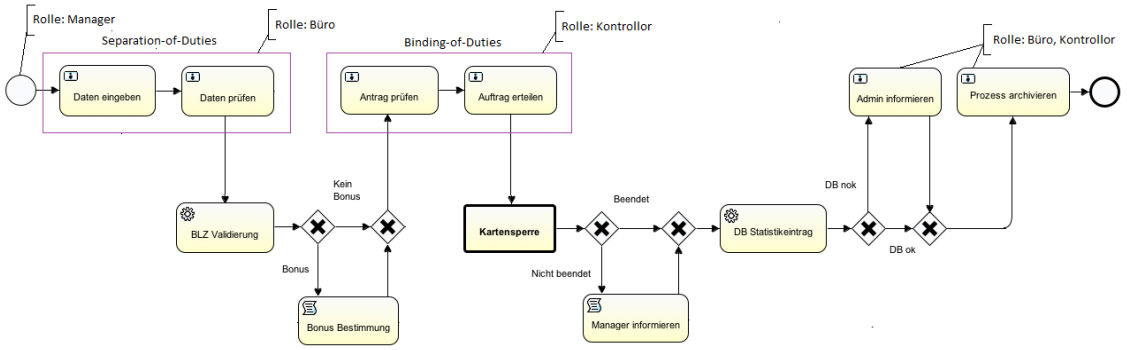

Abbildung 4.3: Gesamtprozess - Aktivitäten mit Bedingungen (eigene Darstellung)

Bei der weiteren Analyse wird neben den erwähnten Entitäten ebenso die Rolle des Administrators in Betracht gezogen, welcher eine übergeordnete Rolle inne hält. Relevant ist ebenso eine übergeordnete Ansicht, die die grobe Systemarchitektur (technische Sicht) betrifft, da auch in dieser Betrachtungsebene Schwachstellen vorhanden sein können. Eine Person kann sich über die entsprechende Benutzeroberfläche von Activiti Explorer im System authentifizieren (user login). Sowohl Personen mit Administrator-Rechten, als auch solche mit eingeschränkten Rechten (”normale” Personen) melden sich über die gleiche Anmeldemaske im Activiti Explorer an. Auch alle weiteren Interaktionen mit dem WfMS finden über den Web-Browser (Chrome) statt. Hier

ist es also der Browser, der auf Benutzeraktivitäten entsprechend reagiert. Die folgende Skizze (Abbildung 4.4) zeigt einen groben Aufbau dieses Szenarios.

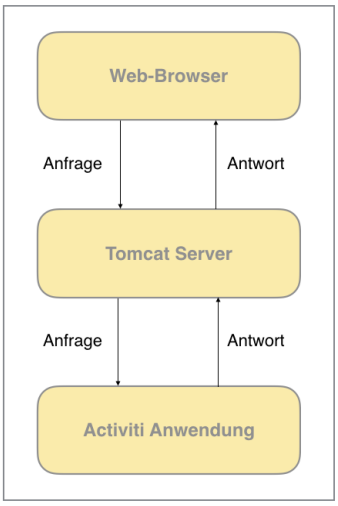

Abbildung 4.4: Potentielle Schwachstellen mit Einfluss auf das WfMS (eigene Darstellung)

Die möglicherweise gefährdenden Bereiche des WfMS befinden sich demnach auf allen drei Ebenen: Im Bereich der Benutzerauthentifizierung, rund um den Tomcat Web-Server, beispielsweise aufgrund fehlerhafter Konfiguration oder veralteter Version, und auf Prozess-Ebene (WfMS) selbst.

4.3 Angriffsszenarien und Analyse

Anhand der vorangegangenen Untersuchungen über kritische Faktoren sowie der allgemeinen und häufigen Angriffsziele werden identifizierte Angriffsszenarien mithilfe von Attack Trees textuell und graphisch vorgestellt. Dabei werden Aspekte, wie Komponenten, Implementierungen, Berechtigungen oder Kontrollen mit untersucht, die einen möglichen Angriffspunkt darstellen [31].

4.3.1 Bestimmung der Wurzelknoten

Die Angriffsziele wurden im Hinblick auf die Komponenten rund um das WfMS und das System selbst kombiniert und festgelegt, um auch einflussnehmende Aspekte des WfMS in Betracht zu ziehen. Die dabei entstandenen Angriffsziele entsprechen den „Wurzelknoten" oder Zielen der nachfolgenden Attack Trees, die anschließend grob in zwei Gruppen, nämlich „allgemeine" und „PAIS spezifische" Attack Trees (Tabelle 4.1) unterteilt wurden. Ein Angriff auf die Web-Oberfläche von Activiti (Activiti Explorer), einem Web-basierten System, wird als „allgemein" betrachtet, da auch andere Web-basierte Systeme auf ähnliche oder gleiche Art angegriffen werden können. Gleiches gilt zunächst auch für die Zielsetzung betreffend der Datenbank, da die meisten informationsverarbeitenden Systeme eine Datenbankanbindung haben. Die Attack Trees mit spezifischen Eigenschaften betreffen Prozesse und Instanzen, sowie Berechtigungen der am Prozess teilhabenden Benutzer und Dienste. Diese Bereiche werden als wesentlich für ein WfMS betrachtet. Eine Trennung ist nicht eindeutig, da die Erreichung der allgemeinen Ziele durchaus auch spezifische Auswirkungen haben können. Alle Bäume zusammen decken die wesentlichen Bereiche des WfMS ab und ermöglichen dadurch eine möglichst umfassende Analyse des Systems.

Tabelle 4.1 gibt einen Überblick über der Angriffsziele mit Informationen über das betrachtete Umfeld (*point of view*), die vorrangigen Schutzziele, die bei erfolgreichem Angriff verletzt werden und die soeben durchgeführte Gruppierung.

Attack Tree Ziel/Wurzelknoten	**Umfeld**	**Schutzziel**	**Gruppe**
Zutritt und Zugangsdaten zu Activiti erlangen	Activiti BPM Anwendung	Vertraulichkeit Integrität Authentizität	allgemein
Durch Datenbank-Manipulation Fehlverhalten auslösen	Datenbank	Vertraulichkeit Integrität Authentizität Verfügbarkeit	allgemein
Verwendete Ressourcen manipulieren	Geschäftsprozess Activiti BPM Server	Verfügbarkeit Integrität	spezifisch
Berechtigungen der Benutzer manipulieren (Rechte umgehen, ausnutzen, ändern)	Activiti BPM	Authentizität Vertraulichkeit Integrität	spezifisch
Prozesse und Ablauf der Instanzen gefährden (Fehlverhalten hervorrufen)	Geschäftsprozess Activiti BPM	Integrität Authentizität Verfügbarkeit	spezifisch
Als Insider die Sicherheit von Activiti gefährden (Allgemeinen Betrieb und Sicherheit)	Activiti BPM Geschäftsprozess Datenbank Server	Integrität Authentizität Verfügbarkeit Vertraulichkeit	spezifisch

Tabelle 4.1: Attack Tree Ziele und Kategorisierung

Im ersten Angriffsbaum wird versucht einen Zugang zum WfMS zu erlangen. Dabei wird die „Oberfläche" von Activiti, genauer der Zugang zum Activiti Explorer untersucht. Der zweite Attack Tree befasst sich mit der zugrundeliegenden Datenbank, als eigenständige Komponente. Ein Zugriff auf die Datenbank kann von Innen (zum Beispiel durch Prozessinstanzen), aber auch von Außen ermöglicht werden. Der dritte Attack Tree behandelt die eigentlichen Prozesse und Instanzen, womit die meiste Interaktion zwischen teilhabenden Entitäten und Activiti stattfindet. Die Benutzerrechte werden im vierten Attack Tree analysiert. Dabei können Berechtigungen umgangen und auch ausgenutzt werden. Der fünfte Baum beschäftigt sich mit den benötigten Diensten und Ressourcen und somit auch mit dem Workflow selbst, welche durch Manipulation Fehlverhalten auslösen können. Der letzte Attack Tree zeigt, welchen Einfluss besonders ein Insider auf das Gesamtsystem ausüben kann. Es wird nicht möglich sein, jeden einzelnen Baum strikt von einander zu trennen, da manche Aspekte mehrere Bäume betreffen können.

4.3.2 Strukturiertes Vorgehen

Nach der Bestimmung sämtlicher Angriffs-Ziele, wurden sukzessive alle „Blätter" als Strategien und Taktiken zur Erreichung des jeweiligen Ziels bestimmt. Auch hier wurden die Inhalte aus OWASP und die Angriffsmöglichkeiten aus [35] als Anregung hinzugezogen. Das Grundgerüst der Bäume wurde größtenteils in Anlehnung an Schneiers Vorgehensweise zur Konstruktion von Attack Trees aufgestellt, wonach zur Erreichung des Ziels (Wurzelknoten) von oben nach unten stets die Überlegung, wie das folgende Subziel zu erreichen ist, angestellt wurde.

Mit der Vorgehensweise der Attack Tree Methode wird nun untersucht, ob durch diese Szenarien Schwachstellen im WfMS gefunden werden können. Diese Vorgehensweise entspricht einem „top-down"-Ansatz. Aufgrund der Ergebnisse kann letzten Endes ein Soll-Ist-Vergleich aufgestellt werden. Der Soll-Zustand wird durch den anfänglich erstellten Attack Tree, im Speziellen

der ausgewählte Pfad, beschrieben, während das Resultat der Tests und der Durchführungsversuche den tatsächlichen Ist-Zustand beschreibt. Auf diese Art (Testfälle und Szenarien) kann eine möglichst übergreifende Analyse sowohl des vorliegenden Prozessmodells, als auch des zugrunde liegenden WfMS und seinen Komponenten durchgeführt werden.

Das WfMS und seine Komponenten werden somit nicht auf bereits „bekannte" Schwachstellen hin analysiert, wie es in der Arbeit von ATLIST Bäumen[4] der Fall ist. Dort wird in erster Linie beschrieben, welche Schwachstellen allgemein in WfMS bekannt sind. Daraufhin wird mit einer abweichenden Methode, der *Attentive Listener* Methode (kurz ATLIST [28]), analysiert, ob bekannte Schwachstellen auch auf das vorliegende WfMS zutreffen und durch Angriffe ausgenutzt werden können. Hierbei handelt es sich um ein sowohl „bottom-up" als auch „top-down" Ansatz inklusive einer unterstützender Anleitung, um im Anschluss daran ein Suchmuster abzuleiten. Solch ein Suchmuster kann anschließend programmatisch umgesetzt und in Schwachstellen-Scanner beispielsweise eingesetzt werden.

Ein strukturiertes Vorgehen (Abbildung 4.5) startet mit der Erstellung der jeweiligen Angriffsbäume. Anschließend werden bestimmte Pfade für eine tiefergehende Analyse und Tests ausgewählt und sämtliche Zwischenergebnisse dabei dokumentiert, um abschießend ein Gesamtergebnis festzuhalten.

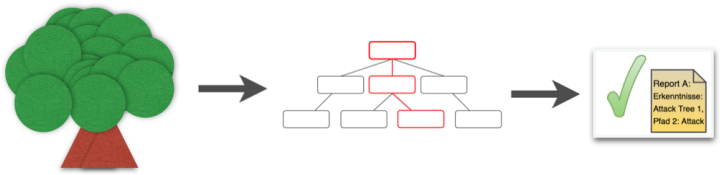

Abbildung 4.5: Vorgehensmodell - Attack Tree Analyse (eigene Darstellung)

Die jeweilige Dokumentation umfasst allgemein die Erreichbarkeit des Wurzelknotens, die Erkenntnisse durch die Behandlung der Zwischenziele (Blätter) und die möglichen Auswirkungen.

4.4 Attack Tree 1 - Zugang verschaffen

Die Beobachtungen in Tabelle 4.1 decken auf, dass die häufigsten Angriffsziele auf die „Oberfläche", genauer gesagt den Einstiegspunkt zum System abzielen (die Log-in Maske). Dieser allgemeine Baum dient als „Startschuss" der Angriffe von Außen und kann auch für andere Ziele wieder verwendet werden, wie zum Beispiel bei einem Zugriffsversuch auf die Datenbank, da auch dort eine Autorisierung benötigt wird. Eine Wiederverwendung dieses Attack Trees wird dann als „Subbaum" dargestellt, beispielsweise zur Darstellung des Vorganges zur Erlangung von Admin-Rechten. Der ordnungsgemäße Zugang zu Activiti geht über die Activiti Explorer Einstiegsseite mittels einer Benutzername-Passwort Kombination. Der folgende Attack Tree in Abbildung 4.6 zeigt einige Möglichkeiten, wie sich ein Angreifer Zutritt zum System verschaffen kann.

Am meisten macht es Sinn, sich Zutritt zu einem höherrangigen Admin-Konto zu verschaffen, obwohl eine untergeordnete Berechtigungsgruppe ebenso einen eigenen Einfluss auf den zugrundeliegenden Prozess ausübt. Selbst wenn mittels Phishing-Mails, Nachahmung von Einstiegs-Seiten (gefälschte Seite, die mit der ursprünglichen Seite fast identisch ist) oder programmatischem Ausspionieren von Tastatureingaben mit versteckter Software keine Erfolge verzeichnet werden kann, gibt es noch die Möglichkeit eine berechtigte (registrierte) Person auf zwischenmenschlicher Basis zu manipulieren (auch bezeichnet als *social engineering*, linker Ast). Wie auch

[4]Diplomarbeit von Sabine Gottwald mit dem Titel: „ATLIST Tree zur Analyse von Schwachstellen in WfMS"

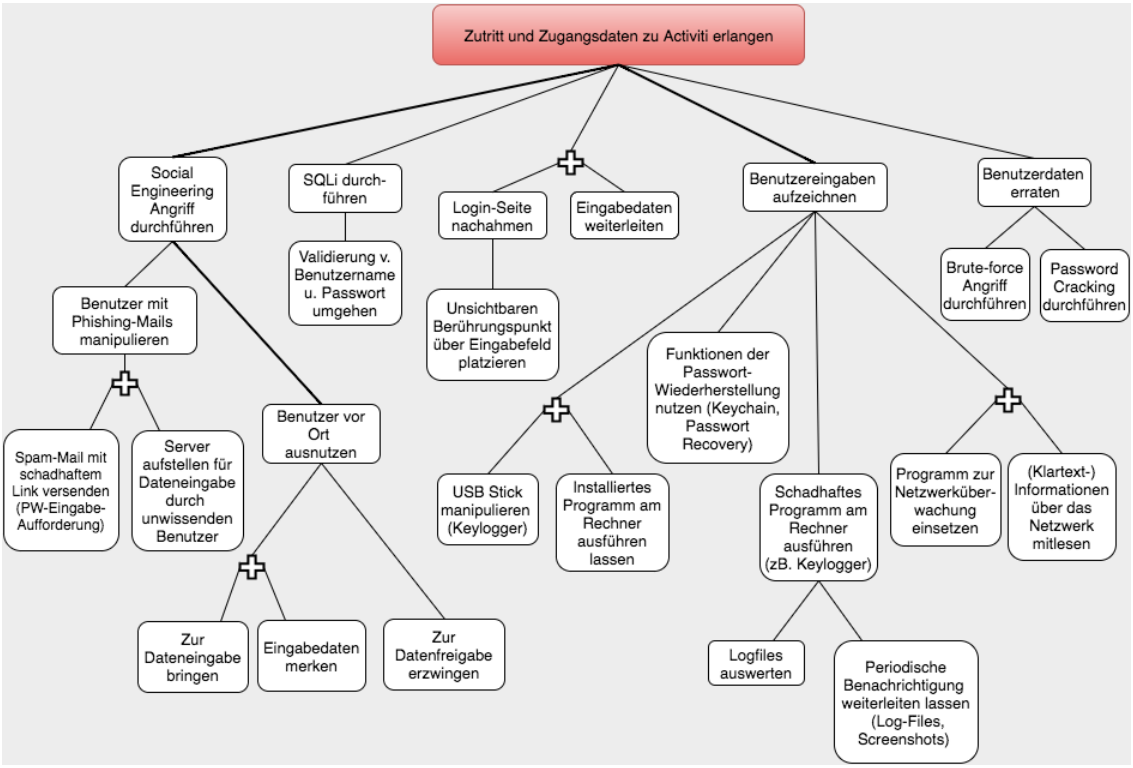

Abbildung 4.6: Attack Tree: Zugang verschaffen (eigene Darstellung)

bei den weiteren Attack Trees zu sehen ist, gibt es nicht nur einen Weg, um das Ziel zu erreichen aber nicht jeder Weg ist für jede angreifende Person durchführbar.

Im *social engineering* ist der unwissende und gutgläubige Mensch im Vordergrund, der zum Beispiel durch eine manipulierte e-Mail, in der auf unterschiedlichste Weise zur Datenfreigabe oder -eingabe gebeten wird, getäuscht werden kann. Er kann darin aufgefordert werden, die Benutzerdaten zwecks einer (falschen) Sicherheits-Überprüfung weiterzuleiten oder einem Link zu folgen, der die Einstiegs-Seite nachahmt. An dieser Stelle kann der Angreifer, der die Benutzerdaten zu erlangen versucht, auch aus weiter Entfernung agieren. Aus nächster Nähe ist das *social engineering* ebenso durchführbar, indem zum Beispiel eine im WfMS registrierte Person durch Zureden, Ablenkung und oder Zwang (Bedrohung) dazu gebracht wird, den Benutzernamen und das Passwort ein- oder direkt preiszugeben.

4.4.1 SQL injections

Um unauthorisierten Zugang zu Activiti zu erhalten und die korrekte Passworteingabe zu umgehen, wurde der Versuch unternommen, die erforderlichen Eingabefelder (User ID und Passwort) mittels *sqlmap*[5] automatisiert zu prüfen, um einen etwaigen Validierungsfehler auszunutzen und somit in das System einsteigen zu können[6]. Das Test-Werkzeug *sqlmap* ist im Internet frei zugänglich (Open Source) und automatisiert das Entdecken und Ausnutzen einer SQL injection (SQLi) Lücke, bei der Anfragen an den Interpreter (Programm, welches Quellcode einliest, analysiert und ausführt) manipuliert werden. *sqlmap* kann Datenbanken und das zugrundeliegende Betriebssystem infiltrieren.

[5]Details zu sqlmap unter http://sqlmap.org/, letzter Zugriff Mai, 2016
[6]Siehe auch Structured Query Language Injection unter www.owaps.org, letzter Zugriff Mai, 2016

Zuerst wurden jedoch sämtliche Möglichkeiten in Anlehnung an OWASP[7] manuell durchgeführt. Folgende Auflistung (4.2) zeigt mögliche Eingaben, die als Versuch zur Umgehung der Autorisierung durchgeführt wurden.

Eingaben	Anwendung
AND 1=1	im Feld „User ID" und oder „Password"
AND 1=2	im Feld „User ID" und oder „Password"
OR 1=1	im Feld „User ID" und oder „Password"
OR 1=2	im Feld „User ID" und oder „Password"
' OR '1'='1	nur im Feld „User ID"
' OR '1'='1	im Feld „User ID" und „Password"
' OR 1=1– -	im Feld „User ID"
' OR 1=1	im Feld „User ID"
' OR 1=1/*	im Feld „User ID"

Tabelle 4.2: SQLi - manuelle Testversuche

Es stellte sich heraus, da bei jeder dieser Eingaben die gleiche „Fehlermeldung" ausgegeben wird, nämlich „Could not log you in" (jeglicher Versuch, auf diese Weise die Anmeldung zu passieren, scheiterte). Es werden somit keine weiteren Informationen preisgegeben (Fehlermeldung ohne Zusatzinformationen), die bei der weiteren Untersuchung hätte nützlich sein können.

Die nächst folgende Darstellung beinhaltet sämtliche Optionen und Parameterangaben, die für eine automatisierte Ausführung mittels *sqlmap* durchgeführt wurden. Um die korrekte Adresse anzugeben, wird der Quelltext der Anmeldeseite von Activiti durchsucht. Dort kann im Anmeldebereich unter dem Stichwort „action" die gesuchte Adresse entnommen werden. Sie lautet: `http://localhost:8080/activiti-explorer/ui/1/loginHandler`.

Abbildung 4.7: Activiti Einstiegsseite - Erforderliche URL und Parameter (Screenshot)

Die hinzugefügten Optionen zur Durchführung und Ausführung mittels *sqlmap* werden in der Tabelle 4.3 vorgestellt.

[7] Siehe dazu https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OWASP-DV-005), letzter Zugriff April, 2016

Optionen	Information
–level –risk	Angabe über gewünschte Test-Mengen-Level (1-5) und Risiko-Level (1-4) der durchzuführenden Tests
–dbms=MySQL –os=Linux	Angaben über zugrundeliegende Datenbank u. Betriebssystem
–banner –is-dba –dbs –tables	Abfragen zum Datenbank-Administrator, Ausweisen der Datenbank und ihrer Tabellen
–technique=BEUST	Angabe über Spezifikation der Einschleusungs-Arten
–batch	Keine Benutzer-Befragung während der Tests, Standardauswahl wird vom System automatisch getroffen
–user-agent=SQLMAP –delay =1 –timeout=15 –retries=2 –keep-alive –threads=5	Angaben über die Zieladressen-Verbindung, (delay) Verzögerung in Sekunden zwischen HTTP-Anfragen, (timeout) Wartezeit bei Zeitüberschreitung und (retries) Neuanfragen, (keep-alive) persistente HTTP(s)-Verbindungen und (threads) max. Anzahl simultaner HTTP(s)-Anfragen
–wizard	Unterstützende Eingabehilfen für definierte Tests, ausgehend von der mitgelieferten URL

Tabelle 4.3: SQLi - automatisierte Testversuche

Die Tabelle zeigt eine Auswahl möglicher Optionen, die Schritt für Schritt nacheinander aber auch in unterschiedlichen Kombinationen über die Kommandozeile angegeben wurden. Weitere Kombinationen und Verfeinerungen durch andere Optionen und Parameter[8] führten neben Aufzählungen der fehlgeschlagener Zugriffe zu keinerlei ausschlaggebende Resultate.

Mit der Durchsicht der Einstiegsseite von Activiti Explorer (Quelltext im Web-Browser) ist ebenso ersichtlich, dass es um eine „POST"-Anfrage handelt, die Benutzereingaben übernimmt, nämlich einen Benutzernamen und ein Passwort (username, password). Aufgrund dieser Informationen kann ein erneuerter Test über *sqlmap* mit der Option „–data" und den Parametern „username" und „password" erfolgen, wobei die frei wählbaren Werte zunächst als Platzhalter gelten.

```
–url=http://localhost:8080/activiti-explorer/ui/1/loginHandler
–data=username=aaa,password=bbb –dbms=MySQL –level=5 –risk=3
```

Neben etlichen Fehlermeldungen und Warnungen konnten jedoch keine erfolgreichen *SQL Einschleusungen* durchgeführt werden. Es wurde schnell ersichtlich, dass auch auf diesem Weg keine Umgehung des herkömmlichen Einstiegs-Prozesses möglich ist und die Datenüberprüfung erfolgreich umgesetzt ist. Beim automatisierten Test ist vor allem auch ersichtlich, dass die angegeben URL keine Identifikations-Parameter in gewünschter Form aufzeigt (wie etwa „id=1"), was die *sqlmap*-Vorgänge und eine gewünschte Einschleusung auf diesen Wert erschwert.
Die häufigsten Fehler- und Warnmeldungen dabei sind zum Einen die serverseitigen Meldungen bei Einschleusungsversuchen auf der Einstiegsseite des Activiti Explorers mit „IllegalArgumentExceltion: bad parameter" und nach etlichen Durchführungen mittels *sqlmap*: „all tested parameters appear to be not injectable".

4.4.2 Web-Seite nachahmen - Unsichtbaren Berührungspunkt platzieren

Eine weitere Möglichkeit der Täuschung ist die Nachahmung der Startseite von Activiti, um den Benutzer in dem Glauben zu lassen, dass er sich auf der sicheren Seite befindet. Dabei kann die komplette Seite überdeckt sein oder nur ein bestimmter Bereich, wie etwa die Schaltfläche zur Freigabe der eingegebenen Benutzerdaten. Nach Freigabe dieser Daten wird entweder ein schadhaftes Programm ausgeführt oder die Daten im Hintergrund an einen vordefinierten Server

[8] Details abrufbar unter https://github.com/sqlmapproject/sqlmap/wiki/Usage, letzter Zugriff Mai, 2016

weitergeleitet. Der Angreifer hat auf diese Weise die Möglichkeit sämtliche Benutzerdaten zu erhalten. Dieser Vorgang wird beispielsweise durch die Technik des *clickjacking* ermöglicht und wird durch eine unzureichend konfigurierte Browser-Einstellung verursacht.

Zur Überprüfung, ob solch eine besteht, wurde mit *Zed Attack Proxy (ZAP)* eine erste Analyse von Activiti Explorer gestartet. Bei diesem Versuch wurden neben der erkannten Schwäche der Einstellungen weitere „unzureichende" Einstellungen entdeckt, die jedoch nach ZAP als weniger „gefährlich" gelten (siehe dazu Abbildung 4.8).

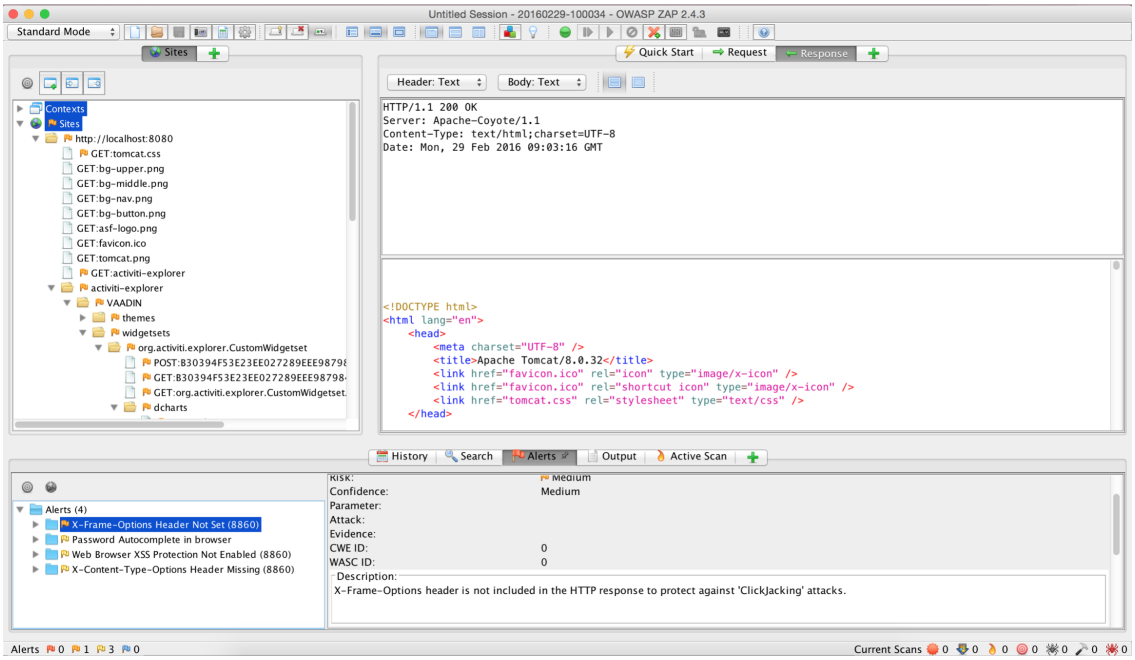

Abbildung 4.8: Zed Attack Proxy - Bestehende Clickjacking Gefahr (Screenshot)

4.4.3 Password erraten

Das Passwort ist ein geheimes Losungswort oder eine Zeichenkombination, die für die Authentifizierung in Anwendungen, wie Activiti Explorer verwendet wird und vor unbefugten Zugang schützt. Erst durch ein richtiges Passwort (und meist in Kombination einer weiteren Identifikation, wie Benutzername) kann der Zugriff auf Konten und Ressourcen gewährleistet werden. Der Vorgang des *password cracking* dient dazu ein Passwort wieder herzustellen oder zu erraten. Es kann angewendet werden, um unerlaubten Zugriff zu erhalten oder allgemein betrachtet die Sicherheit einer Applikation zu überprüfen. Um mittels *password cracking* die Sicherheit von Activiti Explorer zu prüfen, kann beispielsweise *PasswdFinder* herangezogen werden. Diese und weitere ähnliche Programme können im Internet kostenlos heruntergeladen werden. Die Software versucht sich mit dem System zu authentifizieren, in dem zum Beispiel eine große Anzahl an Wort-Kombinationen ausprobiert werden. Je komplizierter das Passwort zusammengestellt wurde, umso länger die Probier-Zeit des Programmes.

Die *brute force* Methode ist ebenfalls dazu da, um Passwörter zu erraten. Je komplexer das Passwort, umso schwieriger und länger dauert es, bis die eingesetzte Software in der Lage ist, zufällig das richtige Passwort zu erraten. Hinzu kommt die Anzahl der zur Verfügung gestellten und notwendigen Wortlisten, die größtenteils ebenfalls frei zugänglich sind. Die Anfälligkeit für sogenannte „Wörterbuch-Angriffe" bedeuten eine Schwachstelle und wird durch gute Zahlen-Buchstaben-Sonderzeichen Kombinationen vermieden.

4.4.4 Daten aufzeichnen und wiederherstellen

Software, wie diverse Keylogger (zB. *REFOG*[9]) und Wiederherstellungs-Werkzeuge (wie *keychain* auf OS X) sind insofern eine hilfreiche Alternative, um Passwörter und Benutzernamen zu erhalten, ohne des aktiven Zutuns des Mitarbeiters und deshalb wertvoll, da die Daten nicht verschlüsselt (*plain text*) wiedergegeben werden. Der Angreifer muss dabei lediglich Zutritt zum Rechner haben (zum Beispiel als Insider). Im Falle eines entsprechend ausreichend konfigurierbaren Programmes zur Aufzeichnung kann es möglich sein, sich diverse Daten sogar periodisch als e-Mail zusenden zu lassen. In den Grundeinstellungen werden sämtliche Daten in eine Log-Datei gespeichert. Netzwerk überwachende Programme arbeiten auf ähnliche Weise. Sie nehmen sämtliche Bewegungen im entsprechenden Netzwerk aus und geben sensible Informationen wieder, sobald sich eine Person in Activiti anmeldet (nähere Details dazu in Abschnitt 4.9.4 unter Verwendung von ZAP).

4.5 Attack Tree 2 - Datenbank

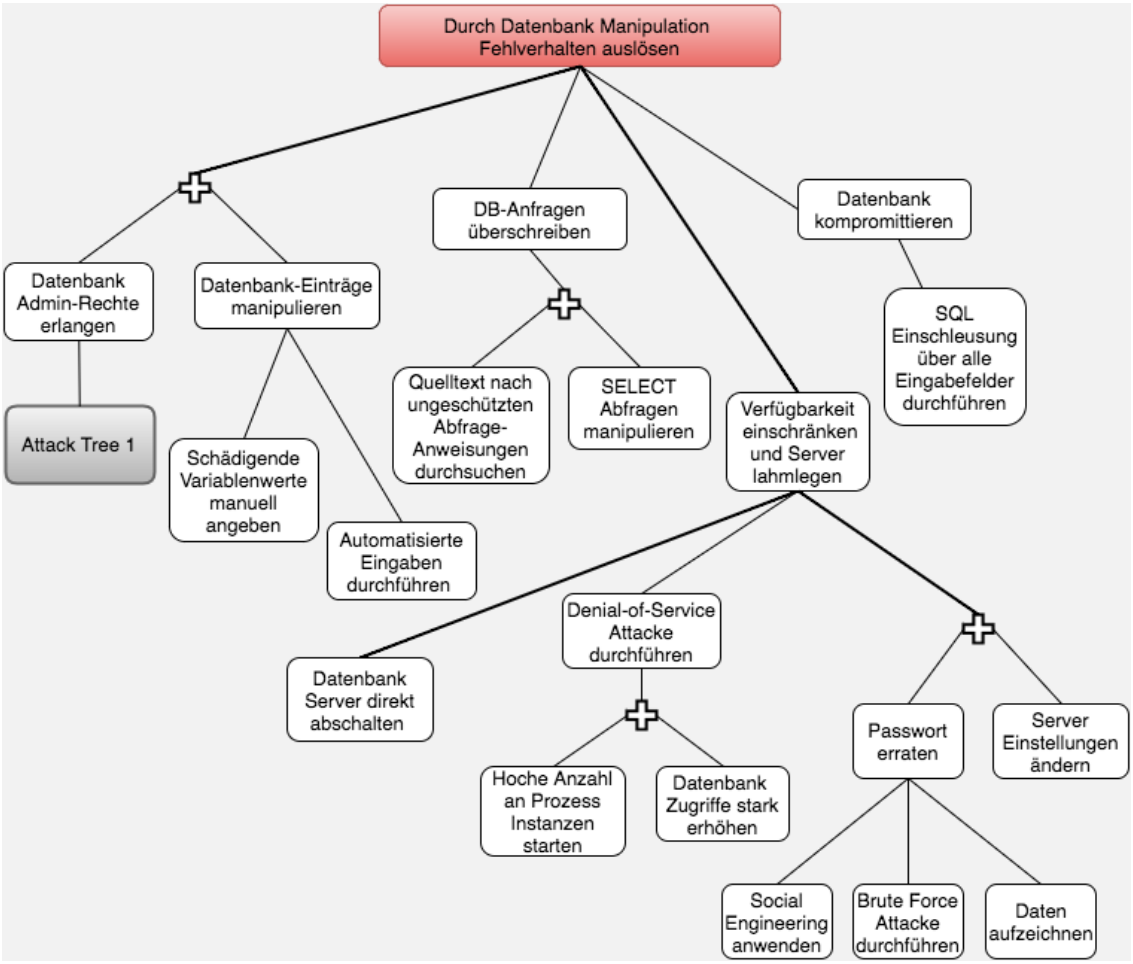

Abbildung 4.9: Attack Tree - Datenbank (eigene Darstellung)

Der zweite „allgemeine" Baum beschäftigt sich mit der zugrundeliegenden Datenbank und zeigt,

[9]REFOG Keylogger abrufbar unter www.refog.com, letzter Zugriff Mai, 2016

auf welche weise diese beeinflusst werden kann. Einerseits wird die Datenbank mit voreingestellten Parametern belegt, die jederzeit zumindest von einer Person mit der Rolle *admin* eingesehen werden können (Benutzerdaten, Prozessdefinitionen, archivierte Daten). Andererseits werden auch während der Laufzeit sämtliche Parameter, wie Instanz-Variablen, Aktivititäts-Zustände sowie verarbeitende Personen und Rollen abgespeichert. Es werden somit vor allem während der Laufzeit und der Instanziierung durchgehend Datenbankzugriffe durchgeführt.

Wesentlich dabei ist sicher zu stellen, dass die gespeicherten Daten nicht manipuliert werden, da dies für eine etwaige und vor allem korrekte Weiterverarbeitung essentiell ist. Ein verheerender Schaden kann entstehen, wenn Daten nicht ordnungsgemäß gespeichert oder beispielsweise im Nachhinein modifiziert werden. Der Prozess kann zum Stillstand gezwungen oder zu einem falschen Ablauf verleitet werden, da auch externe Dienste, die Prozess-Parameter entgegennehmen, entsprechend manipulierte Antworten zurück liefern.

4.5.1 Falsche Parameter und fehlende Validierung

Es ist wesentlich, dass Parameter-Werte korrekt weiterverarbeitet werden. Dazu zählt ebenso die Überprüfung unmittelbar nach der Eingabe durch eine bearbeitende Person. Bei sensiblen Daten kann auf organisatorischer Seite mittels „Vier-Augen-Prinzip" (*separation-of-duties*) überprüft werden, welche Daten in Formularfelder zur Weiterverarbeitung eingetragen wurden. Aus technischer Sicht muss ein Validierungsmechanismus hinterlegt sein, der nur bestimmte Eingaben zulässt (Zahlen, Buchstaben, deren Zeichen-Anzahl, etc.). Dateneingaben sind ein wesentlicher Bestandteil von Wf-Prozessen, weshalb eine Validierung und Überprüfung dringend notwendig sind. Jedoch wird in Activiti bei manuellen Eingaben sehr oberflächlich überprüft, ob die angegebenen Zeichen den Anforderungen entsprechen. Jede fehlende Eingabe-Überprüfung von manuellen Parametereingaben wäre zusätzlich zu implementieren. Einschränkungen zu „Text" oder „Zahlen" sind gegeben (*string, int, long*) jedoch ist beispielsweise bei Zahlenangaben keine Möglichkeit vorgesehen die Größe oder Länge dieser Zahl einzuschränken.

Sämtliche Versuche fehlerhafte Feldeingaben zu tätigen, um Informationen aus der Datenbank zu erhalten, führten jedoch zu keinem brauchbaren Ergebnis. Diese Versuche wurde ebenso im am System nicht angemeldeten Zustand durchgeführt.

Datenbankeinträge können während der Laufzeit über Activiti Explorer nicht mehr abgeändert werden. Allerdings konnte mittels *Activiti REST* durchaus der Versuch durchgeführt werden, bereits gespeicherte Daten während der Laufzeit abzuändern, noch bevor der Prozess beendet wird. Wesentliche Erkenntnis dabei ist ebenso, dass keine „Warnmeldungen" entstehen, trotz abgeänderter Daten, die weiterhin von noch nicht bearbeitenden Aufgaben verwendet wurden. Im laufenden Prozess bedeutete dies, dass die Bankleitzahl, noch bevor diese an den externen Webservice weitergeleitet werden konnte, abgeändert werden kann (somit ist eine manuelle Überprüfung hinfällig) und so immerzu „kein Bonus" als Rückmeldung zurück geliefert wurde, obwohl der Kunde in dem Fall sehr wohl einen Bonus erhalten hätte. Die im Anschluss daran vorgesehene Person, kann dieses Ergebnis nur weiterverarbeiten und erfährt nie, dass Daten „im Hintergrund" kurz vorher noch abgeändert wurden.

Im zweiten Szenario wird eine Kreditkartennummer korrekt eingegeben und die anschließende manuelle Kontrolle fällt positiv aus. Eine registrierte Person hat in weiterer Folge diese Aufgabe zu freizugeben, um die Stilllegung oder Löschung der Karte in die Wege zu leiten. Kurz vorher jedoch kann diese Kartennummer (der Parameter) von einer arglistigen Person ausgetauscht werden, sodass im Endeffekt die falsche Kartennummer stillgelegt wird. Hinzu kommt durch fehlende Validierung, dass bei SQL-Befehlen mit „`SELECT` "-Anfragen auch schadhafte Parameter-Werte übergeben werden können, die schlimmstenfalls zu einer ungewollten Löschung von Datenbankeinträgen führen. Mittels eines Anhanges, wie `'; drop table TABELLE; -`, könnte dies durchgeführt werden, vorausgesetzt, es bestehen Datenbank-Schreibrechte.

```
SELECT name
FROM kartenservice
WHERE kartennr = ,125623451222‘; DROP TABLE kartenservice; –,
```

Grundsätzlich beenden Strichpunkte einen SQL-Befehl und weitere Befehle können angeführt werden. Auf diesem Weg könnte die gesamte Tabelle „kartenservice” gelöscht werden

4.5.2 Verfügbarkeit

Während des Prozesslaufes ist es wesentlich, dass die Datenbank zur Verfügung steht. Eines der einfachsten Wege, eine Unterbrechung hervorzurufen ist das Trennen der Verbindung zur Datenbank, etwa durch das bloße Abschalten oder Stoppen des Datenbank Servers, wie dies etwa ermöglicht werden kann mittels dem Kommando `mysql.server stop`.

Als der Datenbank Server noch vor der Instanziierung getrennt wurde, konnte in den Log-Dateien des Servers beobachtet werden, dass ständig ein Versuch durchgeführt wurde, eine Verbindung mit dem Datenbank Server aufzubauen. Kurze Zeit später wurde auch im Activiti Explorer eine Warn-Meldung ausgegeben, aus der abgelesen werden konnte, dass es sich in erster Linie um eine MySQL Datenbank handelt und aber keine Verbindung aufgebaut werden konnte, da gesendete Nachrichten nicht empfangen wurden (Abbildung 4.10).

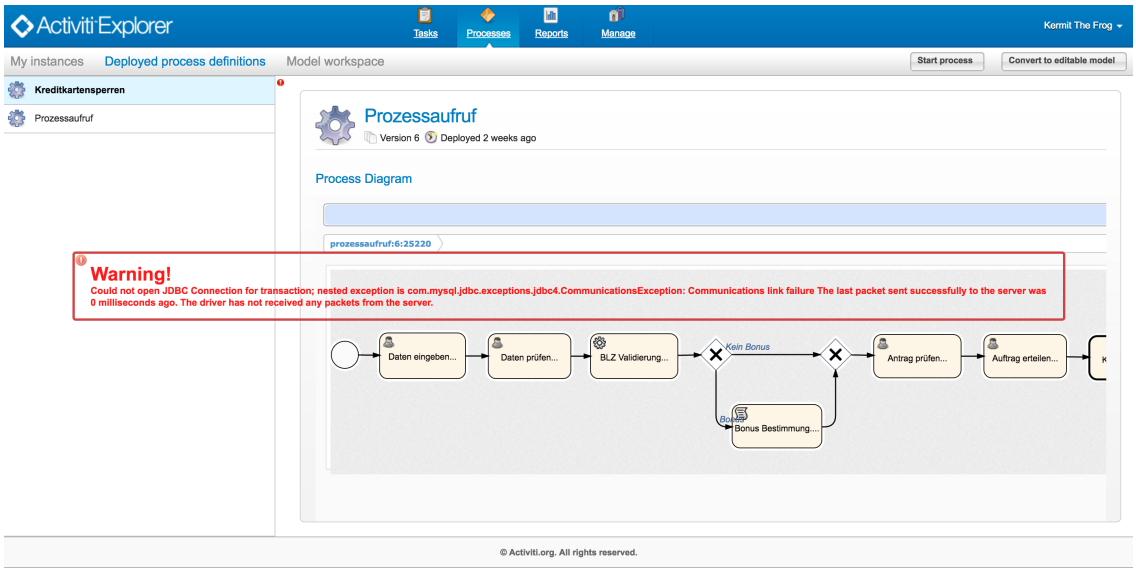

Abbildung 4.10: Warnung - Nach Trennung der Datenbank Verbindung (Screenshot)

Die Fortsetzung der Arbeit mit Activiti wurde dadurch sofort und komplett behindert. Nach erneuertem Start der Datenbank konnte die Warnmeldung beseitigt und der Prozess, wie gewohnt, gestartet werden. Nach einem zweiten Versuch jedoch folgte eine alles zum Stillstand bringende Fehlermeldung: *Internal Error*, welche auch erst beseitigt werden konnte, sobald die Datenbankverbindung wieder hergestellt wurde. Als der Datenbank Server während der Laufzeit getrennt wurde, erfolgten die gleichen Fehler- und Warnmeldungen und die Arbeit konnte nicht fortgeführt werden.

Es stellte sich zudem auch heraus, dass grundsätzlich, wenn die Datenbankverbindung getrennt ist, kein Zutritt zum System gewährleistet werden kann. Während einem Anmeldeversuch bei getrenntem Datenbank Server wird jedoch kein Hinweis auf eine mögliche Datenbanktrennung

bekannt gegeben. Diese Information erhält man nur mit Einblick in die Log-Dateien des Webservers.

Ein Trennen der Datenbank ist vor allem als Administrator ein leichtes Spielt und bewirkt großen Schaden im Hinblick auf die Bearbeitung der Aktivitäten, da selbst die Anmeldung ins System dadurch blockiert wird.

Eine weitere Möglichkeit derartige Verzögerungen auszulösen, ist programmatisch eine Dienstverweigerung hervorzurufen (*denial-of-service*), indem beispielsweise eine sehr hohe Anzahl an Verbindungsversuche durchgeführt wird. Dabei könnte durch das Initiieren von Prozessen bereits eine hohe Anzahl von Datenbank-Zugriffe erstellt werden.

Mit Zugang zu Konfigurationsdateien wird es möglich sein, Einstellungen für das reibungslose Funktionieren abzuändern.

Durch die Ausnutzung sämtlicher Eingabefelder kann versucht werden, die Datenbank direkt zu kompromittieren. Im am System angemeldeten Zustand ist dieser Vorgang vorteilhaft, da der Zugang zu sämtlichen Eingabefeldern gegeben ist, selbst wenn keine Instanz zu der Zeit aktiv ist. Doch auch in diesem Fall konnten keine Eingabefelder manipuliert werden. Eine explizite Suche nach speziellen SQL-Anweisungen führte ebenso zu keinem Erfolg.

4.6 Attack Tree 3 - Ressourcen

Für das WfMS spielen neben Berechtigungen, Rollen und Zugriffskontrollen auch (System-) Ressourcen eine wesentliche Rolle, ohne die das System nicht funktioniert. Es bestehen Abhängigkeiten von diversen Server (Web-Server, Datenbank-Server), externen Dienstleistungen (sofern diese angebunden sind) und weiteren Implementierungen, sofern in Verwendung, wie im vorliegenden Prozessbeispiel der Fall ist. Der folgender Attack Tree befasst sich mit dem Aspekt der zugrundeliegenden Ressourcen (Abbildung 4.11).

Aus gegebenen Anlass ist es stets ratsam in Erfahrung zu bringen, welche Dienste in Verwendung sind, denn diese können veraltet sein und diverse Schwachstellen beinhalten, die den Weg eines Angriffs auf das System ermöglichen[10]. Um generell Dienste und offene Ports (Anschlüsse, Zugänge zum System) zu untersuchen, kann ein sogenannter Port-Scan durchgeführt werden, der überprüft, welche angebotenen Dienste außerhalb des eigenen lokalen Netzwerkes erreichbar sind und Daten oder Informationen nach außen weitergeben. Dadurch, dass diese Ports auch weitergeleitet werden können, ist eine fehlerhafte Weiterleitung auch möglich zu untersuchen. Alle gefundenen Dienste beziehungsweise ihre Versionen können auf Schwachstellen hin in einer definierten Schwachstellen Datenbank, wie NVD[11], nachgeschlagen werden. Ein gekonnter Angreifer kann offene Ports missbrauchen und auf diesem Weg Zugriff auf Activiti erlangen oder anhand der Version mögliche Schwachstellen ausnutzen.

4.6.1 Auslastung

Aktivitäten, die im Besonderen viel Ressourcen benötigen, können gezielt ausgenutzt werden, um durch Ressourcenauslastung ein Fehlverhalten von Activiti zu verursachen. Im zugrundeliegenden Beispielsprozess können solche Aktivitäten zum Beispiel der Webservice-Aufruf oder der Speicher-Vorgang in der Datenbank sein.

Um nun ein Fehlverhalten (Dienstverweigerung) hervorzurufen, kann versucht werden, eine sehr hohe Anzahl an entsprechenden Aktivitäten auszuführen. Das System ist entweder in der

[10] Debian Security Advisory meldet Schwachstelle in Tomcat7: https://www.debian.org/security/2016/dsa-3552, letzter Zugriff April, 2016

[11] Nation Vulnerability Database: https://nvd.nist.gov/, letzter Zugriff April, 2016

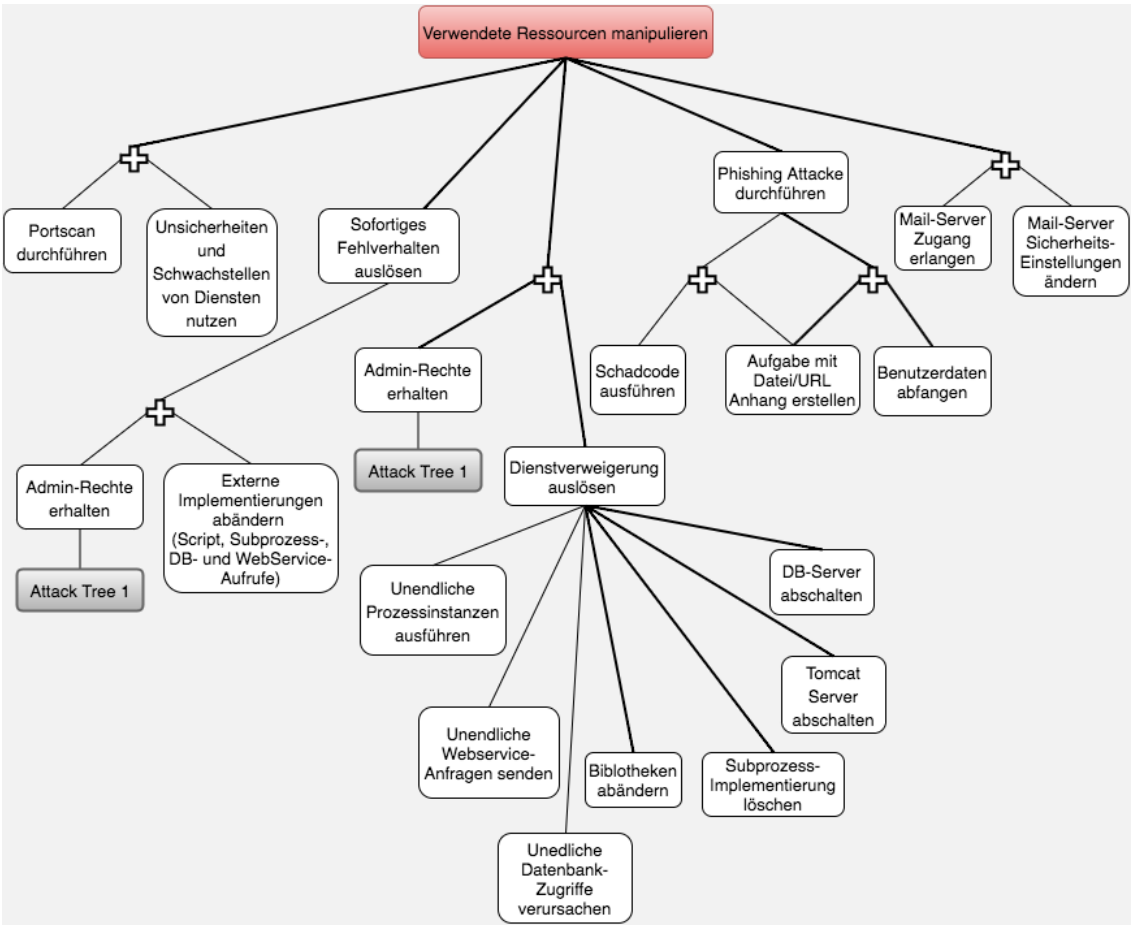

Abbildung 4.11: Attack Tree - Ressourcen (eigene Darstellung)

Lage die hohe Anzahl zu verarbeiten oder wehrt die Aktivitäten ab. Wesentlich dabei ist zu untersuchen, ob das System bloß „abstürzt" und damit weitere Angriffe erlaubt oder eine entsprechenden Fehler-Behandlung durchführt.

Verursachen eine Menge von Prozessinstanzen einen Systemabsturz, so ist der gesamte Prozesslauf oder Activiti allgemein unterbrochen. Wird der externe Webservice mit Anfragen überflutet, kann dies in einer Dienstverweigerung enden, was das Gesamtsystem jedoch nicht zum Stillstand bringt. Sollte der Webservice seine Dienste verweigern, kann eine Implementierung einer Fehlerbehandlung Abhilfe schaffen (eine manuelle „ad-hoc" Aktivität könnte dann den fehlenden Zwischenschritt bearbeiten). Sollte die Datenbank-Eintragung am Ende des Prozesslaufes nicht funktionieren, wird ein Administrator kontaktiert und die Instanz kann dennoch beendet werden.

Um durch Auslastung einer möglicherweise nicht-funktionierenden Datenbank-Eintragung einen Prozess-Stillstand zu vermeiden, wurde zum Beispiel in der zugrundeliegenden Java Implementierung (*SQLDelegate.java*) eine vereinfachte Abhilfe geschaffen (siehe Listing 4.2).

Listing 4.2: Auszug aus SQLDelegate.java - Auffangen von möglichem Fehler

```
1  try{
2  (...) // mit der Datenbank verbinden und Eintragungen durchfuehren
3  // falls die Eintragung nicht funktioniert, darf der Prozess nicht abstuerzen
4  } catch (Exception exception) {
5    System.out.println("Leider Fehler beim Verbindung und Einfuegen! Admin
         beauftragen...");
6    exception.printStackTrace();
7    // Bei Fehler wird die Variable auf ,,false'' gesetzt, damit der Prozess
         nicht stillsteht
8    execute.setVariable("dbeintrag", dbantwort);
9    System.out.println("Variable dbantwort ist: " + dbantwort);
10 }
```

Eine Unterbrechung des Workflows kann auch dann eintreten, wenn der Web-Dienst, genauer gesagt, der externe Server aufgrund einer Auslastung nicht mehr reagiert. Im Activiti Explorer wird dies ersichtlich, in dem eine Warnmeldung angezeigt wird, sobald die Aktivität gestartet wurde. Anschließend bleibt der Workflow an dieser Stelle stehen und keine weiteren Aktivitäten sind durchführbar. In den Log-Dateien wird jedoch der Hauptgrund genauer angegeben (wie zum Beispiel „..kann nicht verbunden werden"), sodass die Suche nach dem Fehler eingeschränkt werden kann.

Selbst wenn mit dem externen Server eine Verbindung aufgebaut werden kann, kann durch fehlende (interne) Implementation ein Stillstand „erzeugt" werden. Dies erfolgte zunächst durch Angabe einer nicht-deutschen Bankleitzahl. Um dadurch dennoch den Workflow fortzuführen und keinen Stillstand zu verursachen, wurde ähnlich wie bei der Datenbank-Anbindung die entsprechenden Java Implementation um eine Fehlerbehandlung angepasst (siehe Listing 4.3).

Listing 4.3: Auszug aus WsDelegate.java - Abhilfe bei falscher Bankleitzahl

```
1  try{
2  (...) // Verbindung aufbauen und Bankleitzahl uebergeben
3  // falls die Bankleitzahl falsch ist oder fehlt, darf der Prozess nicht
       abstuerzen
4  } catch (Exception exception) {
5    System.out.println("Keine Deutsche BLZ - kein Bonus zu vergeben!");
6    ausgabe = false;
7    execution.setVariable("returnValue", ausgabe);
8    // Variable ist auf false gesetzt und wird im Workflow entsprechend
         uebernommen
9  }
```

Auf diese Weise wird zumindest ein „absichtlicher" Versuch, durch eine falsche oder fehlende Bankleitzahl einen Stillstand zu verursachen, entgegengewirkt. Natürlich gilt auch hier, dass die Implementierungen (oder die Dateien) verändert werden können, um Schaden zu verursachen. Ein „Insider", der Zugriff auf diverse Implementierungen hat, könnte diese entsprechend abändern. Dies betrifft zusätzlich Skripte und Subprozess-Aufrufe, die programmatisch ausgeführt werden. Skripte können in der Prozessdefinition eingesehen und abgeändert werden. Ohne den beispielshaften Java-Implementierungen wäre der Workflow gefährdet und eine simple Aktion, wie eine falsche Eingabe einer Bankleitzahl, würde den gesamten Prozessablauf zumindest unterbrechen.

4.6.2 Ressourcen direkt angreifen

Die wohl einfachste Art Activiti zu beeinflussen ist das Abschalten und Trennen der Verbindungen zu diversen Server, was allerdings nicht immer ein „Angriff" bedeutet muss (Wartungen und Updates zählen nicht zu einem Angriff, es sei denn eine Schadsoftware ist zum Beispiel als Update getarnt).

Wird am e-Mail-Server eine Wartung durchgeführt und die e-Mail Einstellungen dabei erneuert, kann dies bedeuten, dass wesentliche Einstellungen für ein reibungsloses Funktionieren mit Activiti geändert werden und dadurch eine Fehlfunktion während der Laufzeit verursacht wird. Dieser Fall ist zwar spezifisch und betrifft in diesem Fall die Verbindung zwischen Activiti und Google Mail[12], kann jedoch auch beabsichtigt zu einer Dienstverweigerung und so zu einer Fehlfunktion führen.

Der Versuch während der Laufzeit bestätigte, dass bei einer einzigen Umstellung in den e-Mail-Sicherheitseinstellungen ("Unsichere Apps zulassen", siehe Abbildung 3.4) weder eine e-Mail versendet noch empfangen werden kann. Der Prozess bleibt an dieser Stelle (eine *MailTask*) stehen und die Instanz ist somit bereits unterbrochen. Lediglich eine Fehlermeldung wird ausgegeben, doch ein Fortsetzen des Workflows ist nicht mehr möglich. Die Fortsetzung dieser Instanz ist vor allem unterbrochen und der Schaden besteht bereits. Dadurch, dass die „e-Mail"-Aktivität eine automatisierte Ausführung hat, ist der Schritt bereits durchgeführt - und konnte nicht erfolgreich beendet werden. Die bestehende Problematik ist also, dass eine bereits gestartete (automatische) Aktivität nicht mehr wiederholt werden kann. Der Fehler besteht und kann nur durch eine neue Instanziierung (nach dem die richtigen e-Mail Einstellungen getätigt wurden) ersetzt werden. Ein Angreifer hat somit ein relativ leichtes Spiel, sofern er Zugang zu den e-Mail Einstellungen hat (als Insider oder durch Zugang zum e-Mail Konto).

Gleiche Problematik gilt auch für den aufzurufenden Subprozess „Kartensperre", welcher im Regelbetrieb eine Rückmeldung an den Hauptprozess zurücksendet. Werden diese Subprozesse derart manipuliert, sodass keine Antwort zurück gesendet wird, muss in dem Fall im Hauptprozess eine Sicherheitskontrolle eingeführt werden, damit an dieser Stelle nicht für unbegrenzte Zeit auf eine Antwort „gewartet" wird, was durchaus der Regelfall wäre, sofern solch eine Situation keine eigene Implementierung hat. Diese Situation kann zum Beispiel mittels eines *TaskListener* implementiert werden, in dem definiert wird, wie lange auf eine Antwort gewartet wird, beziehungsweise, wie viele Male ein Versuch gestartet werden kann, bis ein Abbruch eingelenkt werden muss. Diese Implementierung muss von der programmierenden Person ebenfalls, wie das Auffangen von Fehlern aus Abschnitt 4.6.1, realisiert werden.

In der Prozessdefinition wird auch hinterlegt, welcher Subprozess an welcher Stelle (Aufgabe) aufgerufen wird. Ein Prozessaufruf kann programmatisch in einer Java Implementierung bereit gestellt aber auch direkt bei der Modellierung im Activiti Eclipse designer bestimmt werden. Ein Angreifer, der hier einen Absturz einer laufenden Instanz hervorrufen möchte, hat die Möglichkeit die entsprechende Datei zu manipulieren, sodass der Zugriffspfad nicht mehr korrekt ist, oder den bereitgestellten Prozess grundsätzlich aus Activiti zu entfernen. Während der Laufzeit

[12] E-Mail Dienst von Google Inc., www.gmail.com, letzter Zugriff April, 2016

bedeutet dies, dass der Workflow erst an der bestimmten Stelle stehen bleibt und es wird vorher nicht darüber informiert (siehe Abbildung 4.12).

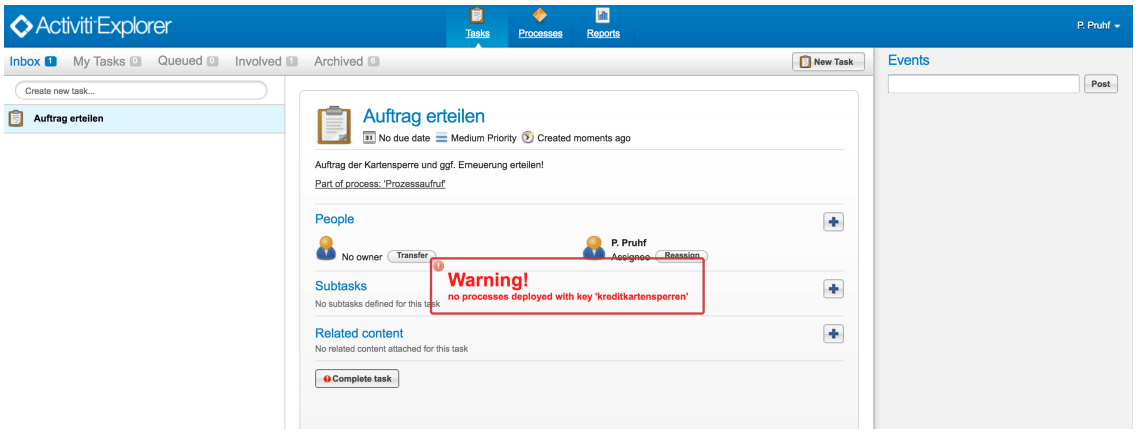

Abbildung 4.12: Warnung bei entfernter Prozessdefinition während der Laufzeit (Screenshot)

Sofern nicht weiter implementiert, kann der Workflow an dieser Stelle nicht mehr fortgeführt werden und der fehlende Prozess muss nachgeladen werden, damit die Unterbrechung aufgehoben werden kann. Selbst wenn der Angreifer keinen Zugang zu Activiti Explorer hat und keine bestehenden Prozesse einsehen kann, hat er die Möglichkeit zu erfahren, welcher Prozess von welchem Prozess aufgerufen wird und diesen entsprechend manipulieren. Ein kurzer Einblick in die Java-Implementierung verrät bereits den Namen des aufzurufenden Prozesses (Listing 4.4).

Listing 4.4: Programmatischer Prozessaufruf

```
1  package services;
2  import org.activiti.engine.RuntimeService;
3  import org.activiti.engine.delegate.DelegateExecution;
4  import org.activiti.engine.delegate.JavaDelegate;
5
6  public class ProzessAufruf1 implements JavaDelegate {
7    public void execute(DelegateExecution execution) throws Exception {
8          RuntimeService runtimeService = execution.getEngineServices().
               getRuntimeService();
9          // der zu startende Prozess hat die ID "kreditkartensperren"
               runtimeService.startProcessInstanceByKey("kreditkartensperren");
10     } }
```

An dieser Stelle wird somit bereits der Name (die Prozess-Identifikation) preisgegeben und kann für die weitere Suche von Informationen dienen.

Wird der Prozessaufruf nicht mittels Java implementiert sondern direkt in der Prozessdefinition integriert (siehe Listing 4.5, einfachste Variante in der Implementierung, ohne individuelle Bedingungen), so können sämtliche Informationen darüber in der Prozessdefintion (XML-Format) eingesehen werden, wie die übergebenen Variablen *kartennummer* und *fertiggestellt*.

Listing 4.5: Prozessaufruf - Daten direkt in der Prozessdefinition

```
1  </script>
2      </scriptTask>
3      <callActivity id="callactivity1" name="Kartensperre" calledElement="
           kreditkartensperren">
4        <documentation>Prozessaufruf mit Weitergabe der Kreditkartennummer.</
             documentation>
5        <extensionElements>
```

```
6           <activiti:in source="kartennummer" target="knr"></activiti:in>
7           <activiti:out source="fertiggestellt" target="beendet"></activiti:out>
8         </extensionElements>
9       </callActivity>
10      <sequenceFlow id="flow41" sourceRef="auftragerteilen" targetRef="
            callactivity1"></sequenceFlow>
11      <scriptTask id="kontaktieren" name="Manager informieren" scriptFormat="
            groovy" activiti:autoStoreVariables="false">
12        <script>java.lang.System.out.println("Legitime Meldung: P2 wurde nicht
              beendet! \n")
```

Die angreifende Person, die über die XML-Datei eine Unterbrechung hervorrufen möchte, muss die Datei zunächst abändern und dafür Sorge tragen, dass die abgeänderte Datei im Activiti Explorer bereitgestellt wird. Dieser Weg mittels eines Prozessaufrufs den Workflow zu unterbrechen ist somit komplizierter.

4.6.3 Phishing

Während diversen Bearbeitungen von Aufgaben können neben der vordefinierten Abfolge ebenso zwischen den bearbeitenden Personen kleinere „Aufgaben" erstellt, weitergeleitet aber auch für die eigene Bearbeitung bereitgestellt werden. Diese „New Task" kann mit Informationen und Dokument-Anhänge bereitgestellt werden. Abbildung 4.13 zeigt eine solche Aufgabe mit einem Anhang, der auf eine Tutorial-Seite weiterleitet. Dieser Anhang wird nie gegengeprüft und kann sowohl auf eine bestehende Internetseite, ein Dokument im lokalen Speicher aber auch ein gefälschter Aufruf einer Internetseite sein, woraufhin im Endeffekt ein Skript ausgelöst werden kann.

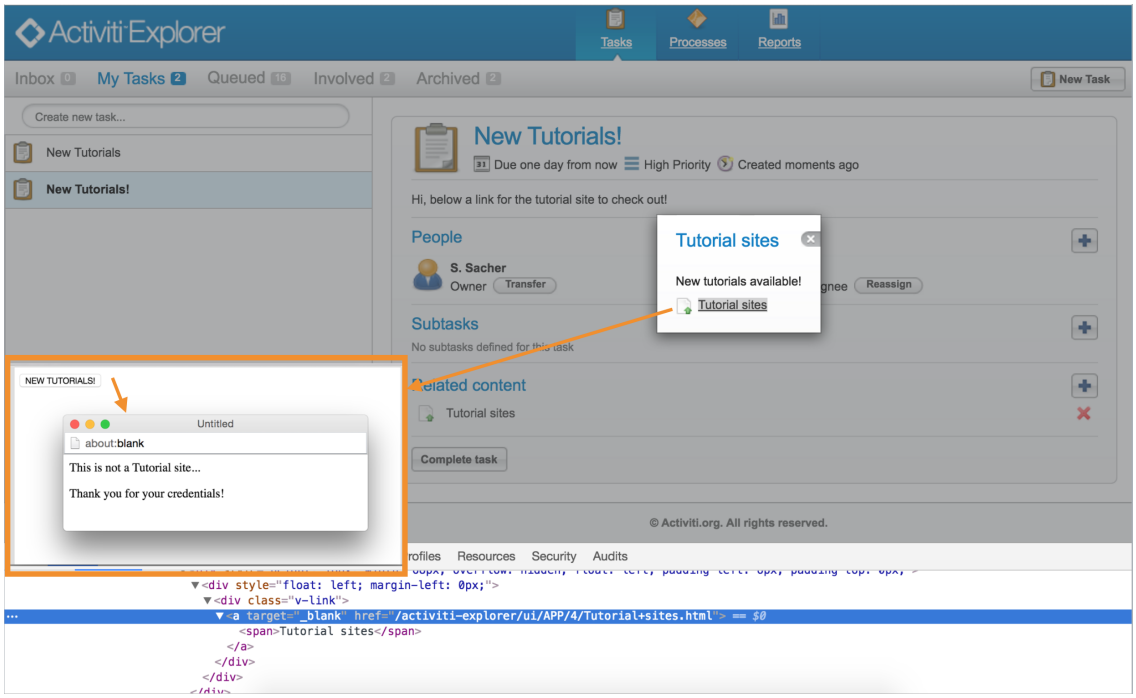

Abbildung 4.13: Phishing Versuch - Manipulierbare Anhänge (eigene Darstellung mit Screenshot)

Auch eine ursprünglich harmlose Datei kann manipuliert werden (lokaler Dateiinhalt wird aus-

getauscht oder der Aufruf führt zu einer manipulierten Internetseite), sodass ein Phishing Angriff (aber auch eine Ausführung von schadhaften Programmen) durchgeführt werden kann (Verdacht auf einer permanenten Schwachstelle).

4.7 Attack Tree 4 - Benutzerrechte

Die Rechtezuordnung hat besonders im WfMS und in der Bearbeitung einer Instanz eine große Bedeutung. Nicht jede Person soll die Berechtigung besitzen Instanzen zu starten und nicht jede Person soll jede Aufgabe bearbeiten können oder dürfen. Bedingungen und Einschränkungen wie das *separation-of-duties* (Aufgaben-Trennung) und das *binding-of- duties* (Aufgaben-Bindung) sind klassische Aspekte, in denen die Rechteverteilung zu Tragen kommt, um eine sichere Bearbeitung der Aufgaben zu bewerkstelligen.

Aus diesem Grund soll der Gesichtspunkt der „Berechtigung" noch einmal näher betrachtet werden, obwohl diese in den vorangegangen Angriffsbäumen bereits erwähnt wurde. Berechtigungen sind ein durch alle weiteren Aspekte hindurch gehendes Thema und der folgende Baum (Abbildung 4.14) zeigt einige wichtige Aspekte.

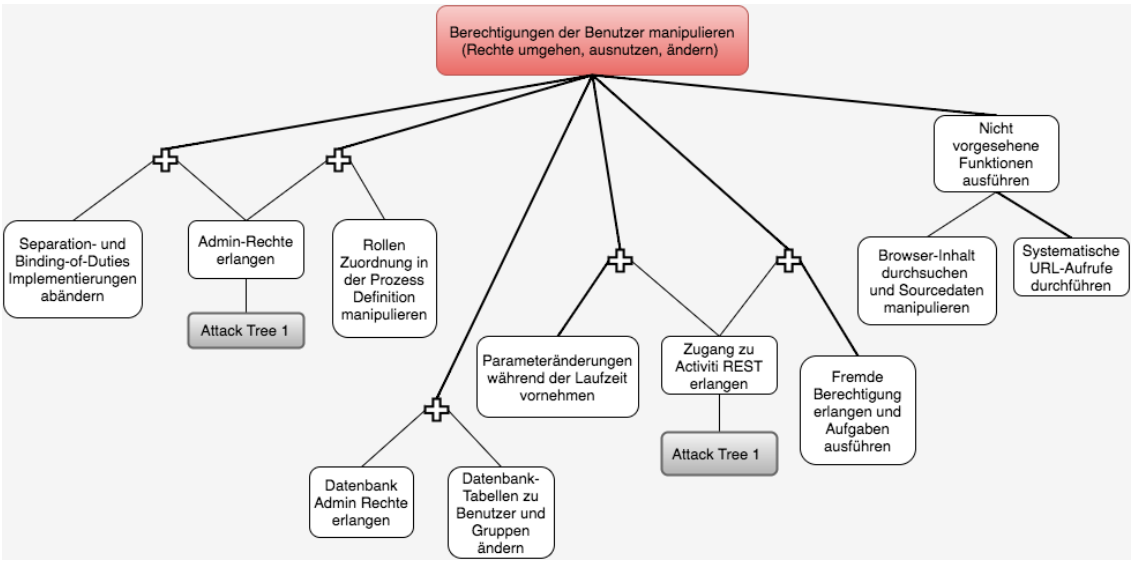

Abbildung 4.14: Attack Tree - Benutzerrechte (eigene Darstellung)

4.7.1 Prozessdefinition - Rechte Implementierungen angreifen

In der zugrundeliegenden Prozessdefinition werden erstmals Benutzerrollen (*assignee*) für bestimmte Aufgaben bestimmt. Sofern diese Rollen im Activiti Explorer existieren, kann die entsprechende Aufgabe bearbeitet werden. Ohne Rollen- oder Gruppenzuteilung in der Prozessdefinition kann keine Instanz gestartet, noch Aktivitäten bearbeitet werden. Berechtigungen sind daher von Anfang an essentiell, doch nicht jede Person soll in der Lage oder berechtigt sein, jede Aufgabe zu bearbeiten.

Eine arglistige Person, die Zugang zu Prozessdefinitionen hat, hat dort die Möglichkeit Berechtigungen entsprechend anzupassen, damit die entsprechenden Personen ungeplante Instanzen starten und Aufgaben bearbeiten können. Die Prozessdefinitionen geben ebenso Auskunft über externe Implementierungen, die nach Auffinden ebenfalls abgeändert werden können. Dies betrifft vor allem die Implementierungen zu *separation-of-duties* und *binding-of-duties* aber auch Skrip-

te und Definitionen zu externen Dienstleistungen. In der Prozessdefinition wird festgehalten, wo die benötigten Bibliotheken zu finden sind und wie diese bezeichnet wurden.

Ein Angreifer, der direkten Zugang zu den Prozessdefinitionen hat, ist also in der Lage herauszufinden, wo wesentliche Implementierungen gespeichert sind und kann diese (als triviales Beispiel) verlegen oder abändern. Der Prozess wird während der Laufzeit garantiert Fehlermeldungen zeigen und nicht mehr durchführbar sein.

Werden jedoch die zugeteilten Rollen und Gruppen in der Prozessdefinition abgeändert, wird es in erster Linie keine Auswirkung auf die im Activiti Explorer befindlichen Prozesse haben. Das heißt, dass der Workflow zunächst nicht unterbrochen wird. Das liegt daran, dass eine modifizierte Prozessdefinition erst in den Activiti Explorer geladen (*deploy*) werden muss, damit die Änderungen eine Auswirkung haben. Erst dann wird durch eine Instanziierung geprüft, welche Rollen an dem Prozess beteiligt oder erforderlich sind. Fehlt dabei eine bestimmte Rolle im Activiti Explorer, so wird von Anfang an eine Fehlermeldung geworfen und eine Instanziierung kann nicht durchgeführt werden, bis die fehlende Rolle oder Gruppe im System angelegt ist.

Die Manipulation von Rollen und Berechtigungen kann somit einerseits im Activiti Explorer selbst, andererseits durch die Prozessdefinition (dem XML Dokument) bewerkstelligt werden. Egal welchen Weg die angreifende Person wählt, eine Unterbrechung des Workflows oder eine fehlgeschlagene Prozess-Bereitstellung ist die Auswirkung eines solchen Eingriffs.

Die ausgelagerte Java-Implementierung des *separation-of-duties* soll berechtigte Personen daran „hindern", bestimmte nacheinander folgende Tätigkeiten zu bearbeiten. Der Grund liegt darin, dass eine Kontrolle durch eine weitere Person durchgeführt werden soll, um vorangegangene Aktivitäten zu bestätigen und gegebenenfalls zu korrigieren und vor allem auch der „Betriebsblindheit" nicht zu verfallen. Hat ein Angreifer nun die Absicht solch eine Kontrolle zu umgehen (weil er beispielsweise gefälschte Daten weiter verarbeiten lassen möchte), kann er ebenso versuchen die zugrundeliegende Datei abzuändern. Hierfür muss beispielsweise nur an einer Stelle ein Zeichen abgeändert werden (Listing 4.6).

Listing 4.6: Separation-of-duties - Leicht manipulierbar

```
1 String claimer = delegateTask.getAssignee();
2 String previousAssigneee = historicTask.getAssignee();
3
4  //wenn es sich um die gleichen Assignees handelt, Fehlermeldung auswerfen!
5 if(claimer.equalsIgnoreCase(previousAssigneee)) {
6 throw new ActivitiException("Assignee of previous task is not allowed to claim
      this task. SEPARATION OF DUTY!");
7 }
```

Sollte nun die Aufgabentrennung unterbunden werden, so muss regelrecht die Fehlermeldung entfernt werden oder die *if*-Anweisung mit einem einfachen Rufzeichen ergänzt werden, sodass die gesamte Klammer „negativ" wird. Nach derartiger Abänderung muss jedoch anschließend gewährleistet werden, dass diese externe Java-Klasse zumindest als Bibliothek erneuert zur Verfügung steht.

4.7.2 Rechte umgehen - systematische URL Manipulation

Für kurzweilige Berechtigungen, die am Besten nicht auffallen sollten, oder grundsätzlich fremde Berechtigungen zu erlangen, kann wiederum Activiti REST ausgenutzt werden. Gleiches gilt für die Parameteränderungen und das allgemeine Bearbeiten während der Laufzeit, welches nur durch das Umgehen von Berechtigungen möglich ist (siehe Abschnitt 4.8.4). Während gleichzeitiger Nutzung von Activiti Explorer werden außerdem keine Warnungen angezeigt, um auf etwaig falsche Berechtigungen (oder das Erlangen dieser) aufmerksam zu machen. Nach bereits

durchgeführter Aktionen, die nicht hätten durchgeführt werden dürfen, besteht der Schaden bereits.

Nun wurde untersucht, ob Aufgaben und Funktionen einer höher berechtigten Person freigesetzt werden, wenn keine korrekte Ressourcen-Referenzierung stattfindet. Um dies zu testen, wurden sämtliche Funktionen eines *admin*-Kontos systematisch aufgerufen und überprüft, ob diese auch für weniger berechtigte oder generell nicht-berechtigte Personen ebenso aufrufbar sind. Dazu wurden zunächst alle möglichen Funktionen einer Person mit *admin*-Rechten aufgezeichnet, um anschließend mit den Funktionen anderer Personen mit niedrigerer Berechtigung gegenüberzustellen. Es stellte sich dabei heraus, dass die wesentlichen Unterschiede in der Funktionsgruppe *Manage* vorzufinden sind, die nur eine Person mit der Rolle *admin* zur Verfügung hat. Dem normalen Benutzer fehlen somit alle „Manage"-Verbindungen (*Links* und Funktionen), alle weiteren Funktionen im Activiti Explorer sind in beiden Berechtigungsklassen identisch (Abbildung 4.15).

Manage	**Database**	-	/activiti-explorer/#database
	Database, erste Tabelle ACT_EVT_LOG	-	/activiti-explorer/#database/ACT_EVT_LOG
	Deployments / Show all	Zeigt direkt auf den ersten Eintrag:	/activiti-explorer/#deployment/60074 /activiti-explorer/#deployment/60069
	Active Processes	Zeigt direkt auf den ersten Eintrag: Zweiter Eintrag:	/activiti-explorer/#activeProcessDefinition/kreditkartensperren:1:60077 /activiti-explorer/#activeProcessDefinition/prozessaufruf:1:60072
	Suspended Processes	-	/activiti-explorer/#suspendedProcessDefinition
	Jobs	-	/activiti-explorer/#job
	User	Zeigt direkt auf den ersten Eintrag: Eintrag master: Eintrag gonzo:	/activiti-explorer/#user/fozzie /activiti-explorer/#user/master /activiti-explorer/#user/gonzo
	Groups	Zeigt direkt auf den ersten Eintrag: Eintrag Büro: Eintrag Engineering:	/activiti-explorer/#group/admin /activiti-explorer/#group/b%C3%BCro /activiti-explorer/#group/engineering
	Administration	Zeigt auf aktive Prozess Instanzen: Abgeschlossene Prozess Instanzen: Datenbank Einstellungen:	/activiti-explorer/#admin_management/0 /activiti-explorer/#admin_management/1 /activiti-explorer/#admin_management/2

Abbildung 4.15: Funktionen ausschließlich für die Rolle: admin (eigene Darstellung)

Es stellte sich heraus, dass keines der *Manage*-Ressourcen von einer weniger berechtigten Person aufgerufen werden kann. Gleiches gilt für zugeteilte Aufgaben mit entsprechender Aufgaben-`id`. Beim Aufruf von Admin-Funktionen mit Aufgaben-`id` gelangt ein normaler Benutzer ohne Fehlermeldung oder Warnung in die eigene Aufgabenverwaltung (*inbox*), auch wenn sich darin keine offenen Aufgabe befinden (leere *inbox*). Somit ist es nicht möglich Links in der Browser-Leiste zu manipulieren (ID oder Referenznummer direkt im Link anpassen), um Berechtigungen zu umgehen. Diese Analyse wurde auf Funktionen durchgeführt, die unabhängig von einer Instanz sind.

Weitere Untersuchungen in umgekehrter Reihenfolge ergaben das gleiche Ergebnis: Auch ein *admin* kann auf diesem Weg keine Links folgen, welche von einem weniger berechtigten Benutzer ausführbar sind. Das System lässt unberechtigte Funktionen und Ressourcen mittels Links im Web-Browser nicht aufrufen. Selbst ein Insider, der sich Zutritt zu einem *admin*-Konto verschafft hat, kann nicht auf die zu erledigende Aufgabe anderer Personen zugreifen, solange keine explizite Zuteilung bestimmt ist (assign). Der Activiti Explorer ist in dieser Hinsicht nicht zu manipulieren. Wie jedoch ersichtlich wird, ist mittels Activiti REST mehr erreichbar.

Die Versuche bestätigten, dass die Aufgaben während der Laufzeit bearbeitet werden können, auch wenn diese bereits einer anderen Person von vornherein zugeteilt wurde. Diese Ergebnisse haben eine Auswirkung auf den Prozesslauf. Es wurden Parameter während des Ablaufs verändert, obwohl hierfür in einer Weise eine Berechtigung im Activiti Explorer vorhanden ist. Hier ist es als weniger berechtigte Person nicht möglich gewesen, bestehende Instanzvariablen einzusehen (wenn nicht selbst Initiator) oder zu verändern, obwohl diese bis zum Prozessende im Hintergrund mitgeführt werden.

Einem Benutzer kann eine Aufgabe und somit Rechte entzogen werden, ohne dass diese eine Information über diese Änderung erhält (keine Warnungen). Der Benutzer im Activiti Explorer kann dadurch schnell manipuliert werden: eine kleine Ablenkung reicht aus, um eine Benutzerzuordnung zu verändern, um „fremde" Aufgaben zu bearbeiten. Die ursprüngliche berechtigte Person wird nie erfahren, dass eine Aufgabe zu erledigen war. Dieses Beispiel wird in Abschnitt 4.8.4 vorgestellt.

4.7.3 Benutzerrechte in der Datenbank

Die Datenbank ist ein wesentlicher Bestandteil und ist ebenso ausschlaggebend für ein Funktionieren von Activiti. Ein Zugang zur Datenbank und ein wenig Wissen über die Struktur verhilft einem Angreifer Berechtigungen zu manipulieren, die keine störenden Meldungen während der Laufzeit verursachen, es sei denn, einer Person werden Rechte entzogen, die zu der Zeit benötigt werden. In der Tabelle `ACT_ID_MEMBERSHIP` werden die registrierten Personen mit ihren zugeteilten Gruppen (entspricht den Rollen) aufgelistet (Abbildung 4.16). Eine Änderung der Gruppenzuordnung zu der Gruppe „admin" ist möglich, ohne dadurch die Aufmerksamkeit anderer auf sich zu lenken. Selbst wenn diese Berechtigung nur für eine kurze Zeit bestehen sollte, wäre dies eine leichte Möglichkeit zur Zielerreichung. Mittels `social engineering` kann ein Datenbank Administrator seine Zugangsdaten hierfür freigeben oder die Aufgabe selbst erledigen - die Auswirkung ist sofort im Activiti Explorer ersichtlich.

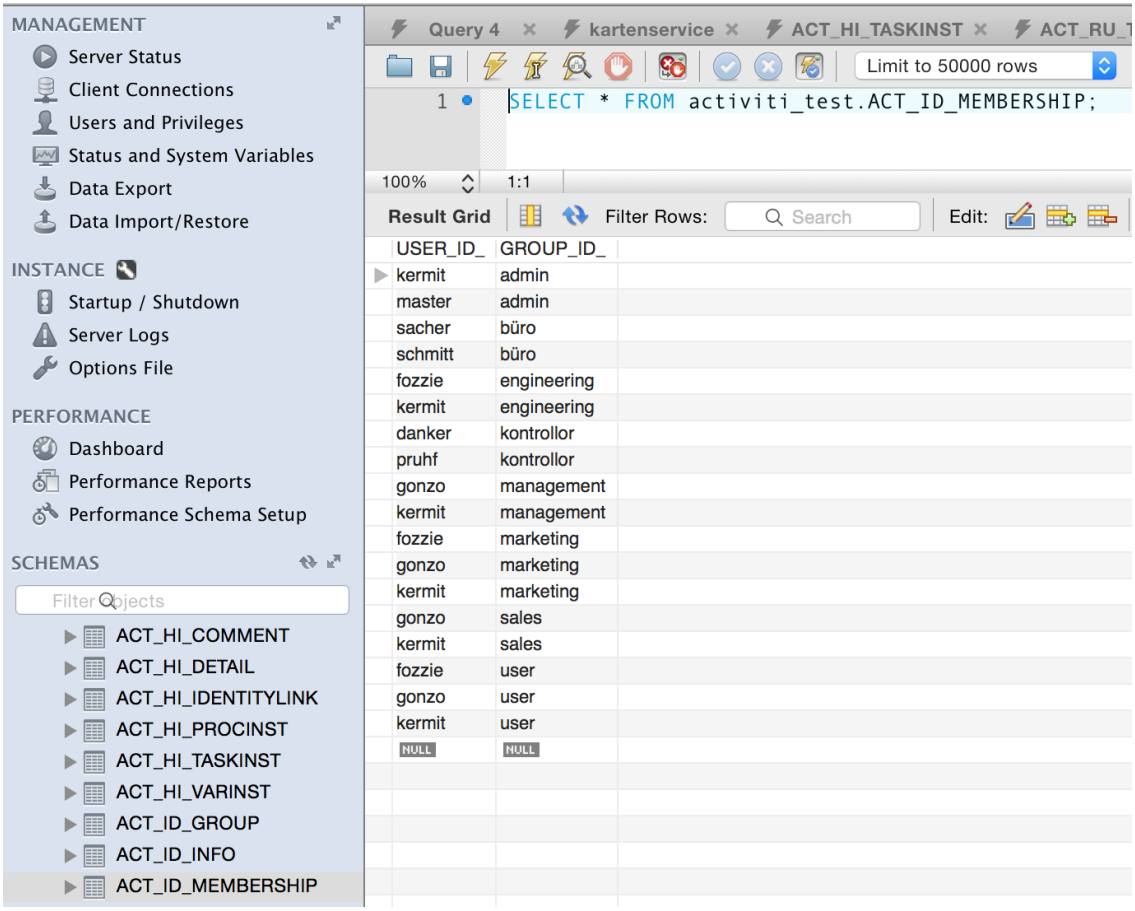

Abbildung 4.16: Datenbank manipulieren - Rollen Anzeige (Screenshot)

Ein Eingriff in diese Tabelle reicht, um Benutzer die Berechtigung zu geben, „fremde" Aufgaben

zu bearbeiten, ohne dabei durch warnende Benachrichtigung daran gehindert zu werden oder andere zu informieren. Ein Beispiel wird in Abschnitt 4.8.4 vorgestellt.

4.8 Attack Tree 5 - Instanz

Voraussetzung für die Erzeugung einer Prozessinstanz ist die Bereitstellung der entsprechenden Prozessdefinition (*deploy*) im Activiti Explorer, sowie eine berechtigte Person (vorrangig die Rolle *admin*). Der Instanzen betreffende Angriffsbaum (Abbildung 4.17) zeigt verschiedene Wege, um mögliche Angriffe vor und während der Instanziierung durchzuführen.

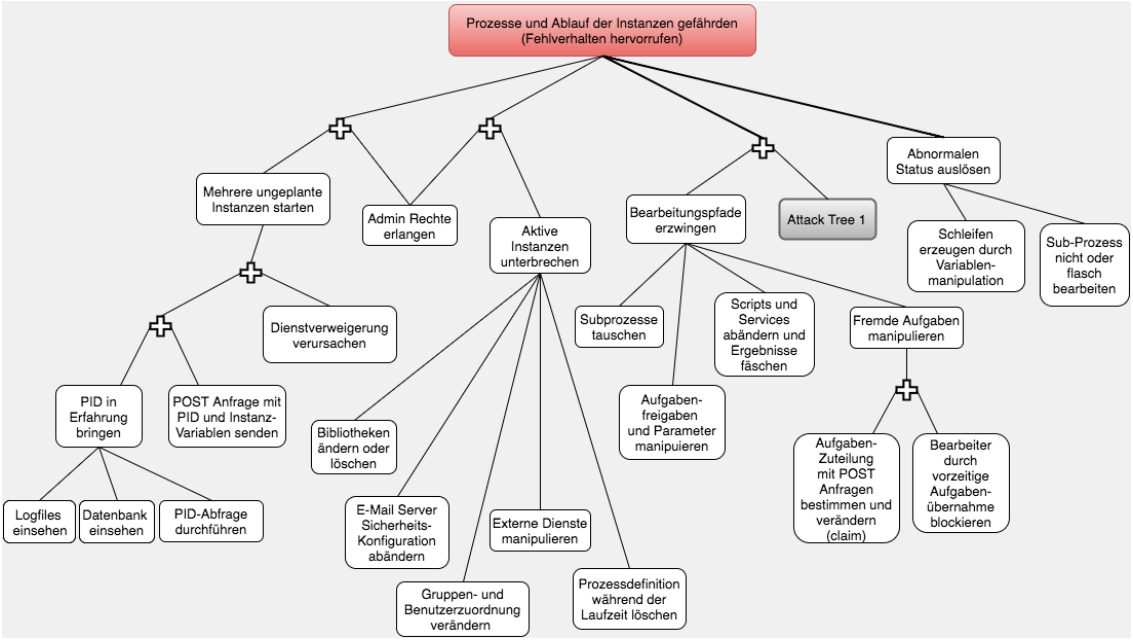

Abbildung 4.17: Attack Tree - Instanz (eigene Darstellung)

In manchen Fällen wird vorausgesetzt, dass der Zutritt zum System, entweder über Activiti REST oder auf herkömmlichem Weg über Activiti Explorer gegeben ist. Es wurde festgestellt, dass der Zugang über Activiti REST mehr Vorteile mit sich bringt, da hier beispielsweise Aufgabenzuteilungen durchgeführt werden können, die im Activiti Explorer nicht möglich sind, selbst wenn die berechtigte Person die Rolle eines Administrators besitzt.

Automatisiert oder manuell lassen sich Instanzen auch über Activiti REST starten. Prozessinstanzen können nur von definierten Personen gestartet werden. Diese Personen sind in der Prozessdefinition definiert („*a priori*") und können im Activiti Explorer nach dem die Prozessdefinition bereitgestellt wurde, nicht mehr abgeändert werden. Doch auch wenn der Zugang zur Manager- oder Admin-Ansicht im Activiti Explorer nicht gegeben ist, gibt es die Möglichkeit über Activiti REST diverse Prozessdaten zu erfragen, mitunter die Prozess-Identifikation, welche benötigt wird, um Instanzen zu starten. Mit einem Zugang zu Activiti REST und der entsprechenden `GET` Anfrage: `activiti-rest/service/runtime/process-instances` kann eine Instanz gestartet werden und ihre Identifikationsnummer unter dem Schlüsselwort `id` abgelesen werden. Sollte der entsprechende Prozess noch nicht in Activiti bereitgestellt sein, kann dies auch über eine entsprechende `POST` Anfrage: `activiti-rest/service/repository/deployments` mit der entsprechenden Prozessdefinition bewerkstelligt werden (*deploy*). Die strikte Trennung der Prozess-Initiatoren kann auf diese Weise umgangen werden.

4.8.1 Dienstverweigerung

Ein Angreifer verfolgt meist das Ziel, durch diverse Angriffsversuche einen Schaden, wie Datenverlust oder Dienstverweigerung zu verursachen. Eine Dienstverweigerung tritt dann auf, wenn durch eine sehr hohe Anzahl an Anfragen der entsprechende Server nicht korrekt reagiert, der Dienst also nicht mehr ausgeführt oder bearbeitet werden kann und somit arbeitsunfähig gemacht wird. Dies kann auch dann bewerkstelligt werden, wenn Programmierfehler ausgenutzt werden können, um diese Fehlfunktion auszulösen.

Durch eine Manipulation einer externen Java-Implementierung zum Starten eines zweiten Prozesses, kann mittels einer Endlosschleife, zum Beispiel durch entfernen einer Abbruchbedingung, das mehrfache Starten eines Prozesses nicht gestoppt werden. Eine unkontrollierbar hohe Anzahl von Prozesses kann dadurch gestartet werden, wodurch die erste Bearbeitung dazu führt, dass der erste Prozess ordnungsgemäß fortgesetzt wird. Alle weiteren Bearbeitungen haben keine Auswirkung auf den Auftrag-gebenden Prozess, können aber eine Auswirkung auf den Geschäftsprozess bewirken (so zum Beispiel eine Mehrfach-Erstellung einer neuen Kreditkarte). Gleiches gilt bei einer Manipulation eines *ScriptTasks*. Eine „wartende" Aktivität wird dadurch behindert, gestartet zu werden, wenn zehn-tausende Berechnungen durchgeführt werden und der Prozess erst nach Beendigung der Berechnung fortgesetzt wird.

4.8.2 Unterbrechung auslösen

Eine Prozessinstanz kann durch unterschiedliche Wege unterbrochen werden, wie im vorhergehenden Abschnitt beschrieben wurde, aber vor allem auch dann, wenn keine ausreichende Verzweigung für entstandene Fehler bereitgestellt wurde. Zusätzlich zu erwartenden Verzweigungen müssen unvorhergesehene Pfade berücksichtigt sein, damit ein reibungsloser Ablauf stattfinden kann.

Für jeden externen Dienst oder zusätzliche Funktionalität muss eine entsprechende Implementation bereitgestellt sein. Diese können einerseits jeweils als separate Bibliotheken, andererseits zusammen mit der Prozessdefinition in die Webanwendung eingefügt sein. Ihre Abhängigkeiten sind jedenfalls sehr stark, sodass ein Fehlen oder Abändern dieser Dateien (*.jar Dateien*) zu einem Nicht-Funktionieren der Prozess-Bereitstellung führt (*deployment*). Auf der anderen Seite wird auch dann die Prozessinstanz zur Unterbrechung gezwungen, wenn externe Services (wie etwa der Webservice zur Ermittlung des Bankinstitutes) inaktiv gestellt sind. Eine Person, die Zugang zu den externen Diensten hat, kann auf diese Weise den Prozessablauf und sämtliche Bearbeitungen unterbrechen oder zum Stillstand bringen. Eine Unterbrechung der e-Mail Funktionalität wird rasch hervorgerufen, sobald eine einzige Sicherheitsmaßnahme (Sicherheitseinstellung) des Mail-Servers deaktiviert wird (Abbildung 3.4).

Ein Angreifer kann den Prozessablauf auch dann gefährden, wenn er vor oder während der Laufzeit einen Benutzer und oder Benutzergruppen entfernt oder abgeändert. Werden Benutzer *vor* der Instanziierung entfernt, wird sogleich eine Warnmeldung angezeigt, die nur Personen mit der Rolle *admin* erhalten - teilweise auch mit detaillierter Fehlermeldung. Die restlichen „*non-admin*" Rollen erhalten keine Meldung, wenn eine benötigte Person nicht im System registriert ist. Wird somit während der Laufzeit eine Person gelöscht, kann der Prozesslauf solange unterbrochen werden, bis eine Person mit *admin*-Rechte Kenntnis darüber erhält und eine entsprechende Person (wieder) hinzufügt. Anschließend kann die Prozessinstanz fortgesetzt werden. Gleiches gilt für die jeweiligen Benutzergruppen, die noch vor Instanziierung gelöscht werden. Dadurch, dass in der Prozessdefinition definiert wird (*a priori*), das bestimmte Benutzergruppen einer oder mehreren Aufgaben zugeordnet sind, diese Gruppen jedoch nicht im Activiti Explorer registriert sind, wird die zu bearbeitende Aufgabe in keiner der möglichen „Aufgaben Postkörbe" (*inbox*) erscheinen, nicht einmal in dem der Person mit *admin*-Rolle. Das bedeutet, dass auch in diesem Fall die Instanz nicht fortgesetzt wird, bis die entsprechenden Gruppen erstellt werden.

Auffallend ist im Gegensatz zu Benutzer, die gelöscht wurden, dass nach Löschung der Gruppen in keiner Rolle eine entsprechende Warnung angezeigt wird. Durch das Löschen von Gruppen werden sämtliche Aufgaben aus der entsprechenden *inbox* nicht mehr sichtbar und erst wieder ersichtlich, sobald die Gruppenzuordnung abgeschlossen ist.

Löscht oder verändert ein Angreifer Benutzer und oder Gruppen *während* der Laufzeit, werden fehlende Personen in einer Fehler-Meldungen angezeigt während die registrierten und bearbeitenden Personen keine Nachricht in ihrer *inbox* erhalten und der Prozess an dieser Stelle stehen bleibt. Es wird auch dann keine Fehlermeldung oder Warnung angezeigt, wenn mittels *separation-of-duties* bestimmt wurde, dass eine Person aus einer bestimmten Gruppe erforderlich ist. Dies ist vor allem dann verheerend, wenn dadurch jede teilhabende Person der Ansicht ist, dass ihre Aufgabe getan und sie sich auf die andere Person verlassen kann.

Fehlt eine zugeteilte Person, kann auch eine Person mit *admin*-Rechten über Activiti Explorer keine Neuzuordnung veranlassen, da eine Neuzuordnung nur von zugeordneten Personen durchgeführt werden kann (diese Funktion ist jedoch in der vorliegenden Konstellation zwar vorgesehen aber nicht anwendbar aufgrund einer fehlenden Teilnehmer-Liste, um eine andere Person zu bestimmen. Die Funktionalität „*re-assign*" ist in der verwendeten Version fehlerhaft). Um in solchen Fällen eine reibungslose Fortsetzung während der Laufzeit zu gewährleisten, muss in der Prozessdefinition definiert sein, wie fehlende Entitäten berücksichtigt und behandelt werden müssen.

Weitere Möglichkeiten eine Instanz direkt anzugreifen ist das Löschen der zugrundeliegenden Prozessdefinition (im Activiti Explorer oder über Activiti REST), wobei nur die dazu berechtigte *admin*-Person einmalig eine Warnmeldung erhält. Alle beteiligten weiteren Personen erhalten keine Nachricht darüber, auch wenn alle beteiligten und aktiven Instanzen dadurch auch gelöscht werden.

Damit können bis zu einem bestimmten Punkt Instanzen ausgeführt (Datenbankabfragen durchführen, Informationen abfragen, und weitere Aktivitäten) aber vorzeitig noch gelöscht werden. Dadurch wird es möglich den Workflow und die Instanzen auszunutzen, um Informationen abzufragen. Aktive Parameter-Werte werden zwar beim Löschen oder nach Beendigung der Instanz verworfen, allerdings geht eine etwaige Datenbank-Eintragung oder Service-Anfrage damit nicht verloren. Eine Person, die sämtliche Datenbank-Abfragen oder Dienstleistungen ausnutzen und aber verhindern möchte, dass alle benötigten Instanzen für den Vorgesetzten ersichtlich bleiben, kann auf diese Art und Weise das Vorhaben durchführen, ohne dabei aufzufallen (da keine Instanzen abgeschlossen und archiviert werden).

Ein Angreifer kann Subprozesse, die eine Rückmeldung generieren, damit der wartende Prozess fortgesetzt wird, suspendieren oder gar löschen und damit eine Unterbrechung des Gesamtprozesses auslösen, da nie eine Rückmeldung generiert wird. Auch an diesem Beispiel wird ersichtlich, dass Implementierungen notwendig sind, um solche Unterbrechungen zu verhindern. Nichtsdestotrotz entfallen die mitgelieferten Parameter dadurch und die Weiterbearbeitung ist mit Zusatzaufwand verbunden.

4.8.3 Verzweigungen erzwingen

Im zugrundeliegenden Workflow ist ersichtlich, dass durch gewisse Entscheidungen oder Ergebnisse unterschiedliche Pfade angenommen werden können (unterschiedliche Ergebnisse lenken den Prozess). Aufgrund der Vereinfachung des Prozesses werden wesentliche Gabelungen beibehalten, obwohl Verzweigungen an mehreren manuellen Aufgaben ebenso erdenklich (und in einem Realbetrieb erforderlich) sind, wie etwa in der manuellen Aufgabe *Antrag überprüfen*. Sollte bei der Überprüfung ein Fehler entdeckt werden, könnte von dieser Aufgabe ein Pfad in Richtung „Daten korrigieren" zeigen.

Eine Instanz kann auch dann leicht manipuliert werden, wenn falsche Subprozesse aufgerufen, von bereits abgearbeitete Aufgaben die Parameter oder die programmatische Aufgaben abge-

ändert werden, sodass das Resultat aus gefälschten Werten besteht. Die Instanz kann dennoch unauffällig und ohne Unterbrechung beendet werden, sofern die Parameter belegt sind, jedoch entspricht das Gesamtergebnis nicht dem Ergebnis, wie es bei korrekten Parametern der Fall wäre.

Über Activiti REST ist ein Angreifer fähig, bereits bearbeitete Aufgaben und ihre Parameter zu verändern, während solch eine Änderung über Activiti Explorer keinesfalls möglich ist. Eine Person ohne *admin* Berechtigung kann grundsätzlich keine Variablenwerte und Ergebnisse einsehen, sofern sie keine Prozessinstanz gestartet hat. Nur wenn eine eigene Instanz aktiv ist, können angefallene Parameter abgelesen werden. Über Activiti REST können nun Parameter im Nachhinein auch abgeändert werden, ohne, dass die beteiligten Personen darüber informiert werden. Sollte beispielsweise eine Bearbeitung zu einer (korrekten) Ablehnung einer Bonusbestimmung führen, kann der korrekt eingetragene Parameter kurz vorher über Activiti REST entsprechend angepasst werden, sodass die antragsstellende Person dennoch einen Bonus erhält, obwohl dies nicht vorgesehen wäre. Damit am Ende des Workflows die richtigen Werte in die Datenbank eingetragen werden, kann der Vorgang zur Parameteränderung natürlich wieder rückgängig gemacht werden.

4.8.4 Aufgaben entziehen

Während in manchen Situationen eine bestimmte Aufgabe einer bestimmten Person zugeteilt ist, können andere Aufgaben einer oder mehreren Gruppen zugeteilt sein. In dem Fall kann eine Person, die der Gruppe zugeordnet ist, eine Aufgabe für sich übernehmen (*claim*). Sobald eine Aufgaben zur Bearbeitung übernommen wurde, kann diese von den anderen potentiellen Bearbeitern nicht mehr eingesehen werden. Sie wird von deren *inbox* entfernt. Jemand, der nicht in den definierten Gruppen zugeteilt ist, sieht diese Aufgabe erst gar nicht und sollte sie auch nie bearbeiten können. Anders ist dieser Zustand jedoch über Activiti REST. Dort kann eine nicht zugeteilte Aufgabe angenommen werden, ohne dass eine Person über Activiti Explorer darüber in Kenntnis gesetzt wird. Auf diese Weise kann jemand, der nichts mit dieser Aufgabe zu tun hat, zugeteilt werden und willkürliche Daten angeben, um eine Instanz zu manipulieren. Die Personen, die zuständig gewesen wären, werden nie über diese Aktion erfahren und bearbeiten anschließend die manipulierten Ergebnisse. Das Beispiel wurde mit dem Bearbeiter „Pruhf" (Rolle *kontrollor*) durchgeführt, der nicht Teil der *separation-of-duties* Aufgaben ist (das sind die Aufgaben *Daten eingeben* und *Daten prüfen*, durchzuführen von der Rolle *büro*). Die Übernahme war erfolgreich und der Prozess wird, wie gewohnt, weitergeführt (Abbildungen 4.18 und 4.19).

Listing 4.7: Nicht zugeteilte Aufgabe übernehmen

```
1 //Anfrage mit zuvor identifizierter Aufgaben-Kennung
2 POST  activiti-rest/service/runtime/tasks/25215
3
4 //Payload - Person pruhf soll uebernehmen
5 {"action" : "claim",
6 "assignee" : "pruhf"}
```

Nachdem im Activiti REST Client[13] mittels `POST` Anfrage die Aufgabe zugeteilt wurde (Listing 4.7), konnte „Pruhf" die sonst nie zur Verfügung stehende Aufgabe bearbeiten. Die für die Übernahme der Aufgabe benötigten Informationen werden im JSON-Format der `POST` Anfrage mitgegeben.

Die Aufgabe, die die Person „Pruhf" sonst nie zu Gesicht bekommt, ist nun in seiner *inbox* zur Bearbeitung bereitgestellt. Er kann diese bearbeiten, ohne dass es zu einer Warnmeldungen oder zu einem Fehler führt. In weiterer Folge konnte im Activiti Explorer die bearbeitete Aufgabe eingesehen werden.

[13] Advanced REST Client als Google Add-on anwendbar

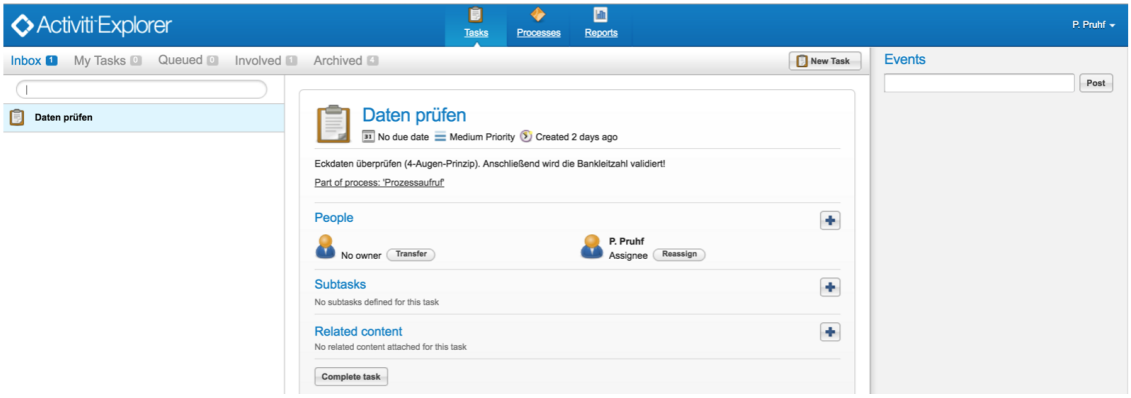

Abbildung 4.18: Aufgabe einer anderen Gruppe steht in der eigenen Inbox bereit (Screenshot)

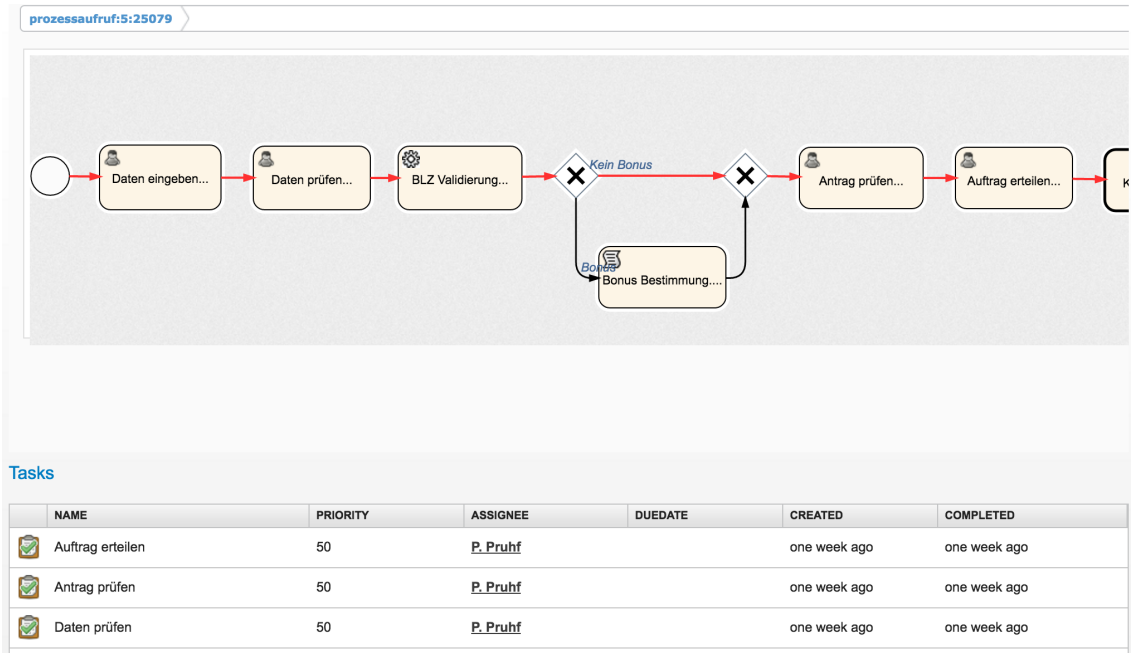

Abbildung 4.19: Zugriffskontrolle fehlgeschlagen - Fremde Aufgabe bearbeitet (Screenshot)

Selbst, wenn dieser Fehler im Nachhinein doch entdeckt wird, ist der Schaden bereits entstanden und die Manipulation erfolgreich durchgeführt. Der Angriff auf das Schutzziel der Zugriffskontrolle des Systems konnte auf diesem Weg erfolgreich durchgeführt werden, trotz korrekter Implementierung. Im Gegenteil dazu wird im Activiti Explorer eine legitime Fehlermeldung geworfen, wenn die Restriktionen zu umgehen versucht werden (Abbildung 4.20).

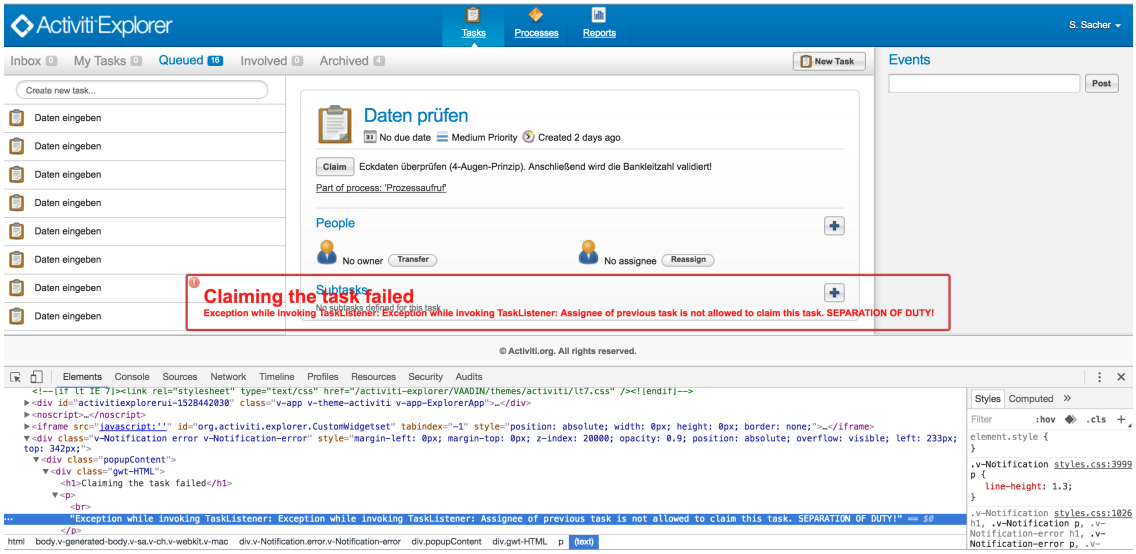

Abbildung 4.20: Zugriffskontrolle funktioniert - Warnung bei Übernahmeversuch (Screenshot)

4.8.5 Abnormale Situationen

Abnormale Situationen, sei es auch nur, um von dem Eigentlichen abzulenken, können vor allem dann herbeigeführt werden, wenn bei Durchlauf von sämtlichen Verzweigungen stets der gleiche Pfad angenommen wird. In einer erweiterten Prozessansicht könnte solch ein Pfad in einer Schleife und somit in einen untypischen Zustand enden. Nachdem *Daten prüfen* beendet wurde und das Ergebnis nicht akzeptabel ist, könnte hier eine Verzweigung zurück zur Aufgabe *Daten eingeben* führen, oder eine weitere Aufgabe entsprechend hinzugefügt werden, wie *Daten korrigieren*. Auf diese Weise können überall derartige Rückführungen implementiert werden, die, wenn ausgenutzt, zu einer *Schleife* führen.

Besonders bei einem Zugriff auf die Prozessdefinition können an den jeweiligen Verzweigungen Änderungen vorgenommen werden, die bei Betrachtung im Activiti Explorer nicht ersichtlich sind (Abbildung 4.21).

Nach Änderungen an den Verzweigungen, die möglicherweise keine Alternative mehr zulassen (beide Pfade werden auf „true" oder „false" gesetzt) wird der Workflow zu einem Stillstand gezwungen und die Instanz kann nicht mehr fortgesetzt werden. Abbildung 4.22 macht deutlich, dass für das Fortsetzen des Prozesses kein anzunehmender Pfad bereitgestellt ist.

Dieser Prozesszustand kann selbst durch erneuerte Bereitstellung (*deployment*) der richtig gestellten Prozessdefinition (als Administrator) nicht mehr aufgehoben werden und die Instanz muss gelöscht werden.

Für das Ausbleiben von ausgelagerten Prozessen kann jeder zuständiger Bearbeiter verantwortlich sein. Bei automatisierter Instanziierung und Erstellung von einer großen Menge an Prozessaufrufen zur Erneuerung der Kreditkarte, verursacht das Ignorieren oder gar Löschen des Subprozesses einen Stillstand im Hauptprozess. Ohne entsprechender Fehlerbearbeitung (bei Aus-

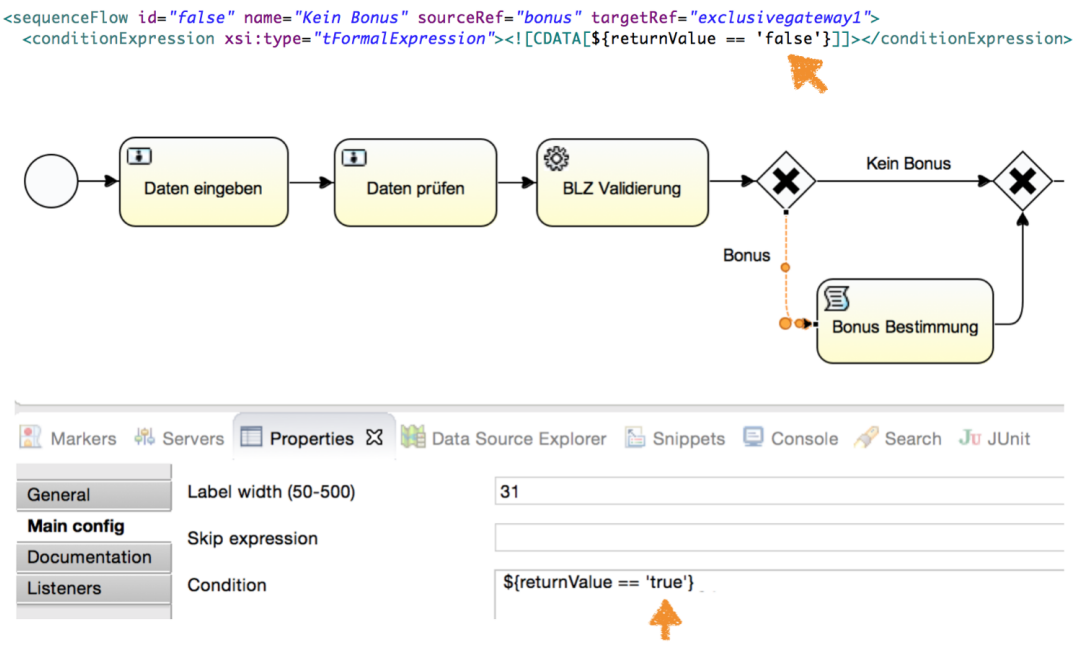

Abbildung 4.21: Ansicht aus der XML Prozessdefinition und Activiti Eclipse designer - Bezeichnungen der Pfade (eigene Darstellung mit Screenshot)

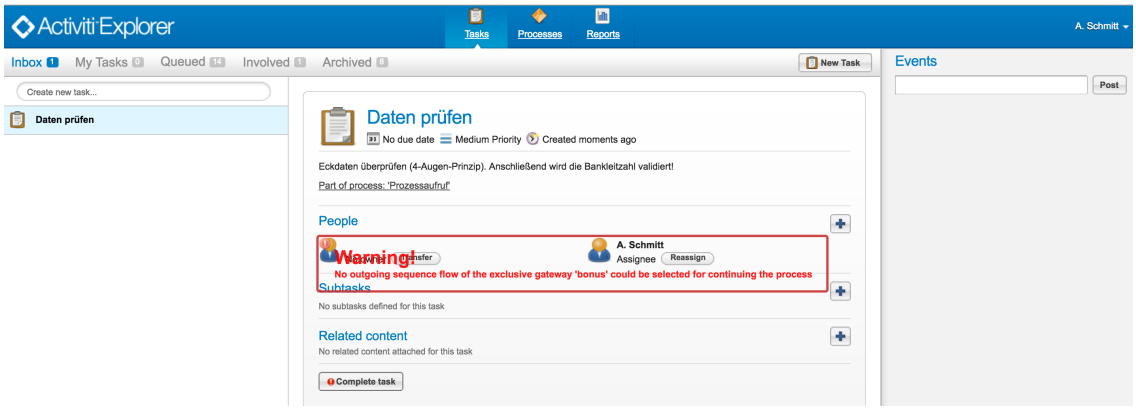

Abbildung 4.22: Warnung - Kein entsprechender Pfad vorhanden (Screenshot)

bleiben von Subprozess Rückmeldungen) kann eine arglistige Person einen abnormalen Status herbei rufen, welcher aber auch zum Verlust sämtlicher Daten führen kann.

4.9 Attack Tree 6 - Der Insider

Der Mensch, der neben maschinellen Tätigkeiten einen Großteil der Aktivitäten ausführt und an dem Workflow beteiligt ist, soll nicht außer Acht gelassen werden, da er durch seine diversen Zugriffserlaubnisse und -möglichkeiten selbst zu eine Gefahr für das System werden kann. [27]

Wie auch in Abschnitt 4.4 „Attack Tree 1 - Zugang verschaffen" bereits angesprochen, können Menschen durch *social engineering* eine Gefahr darstellen, sowohl passiv als auch aktiv. Eine registrierte Person kann dadurch zu einem *Insider* (auch Innentäter) konvertieren und seine Berechtigung ausnutzen, sei es für seinen eigenen Vorteil oder als Gefälligkeit für andere (arglistige) Personen. Nicht um sonst wird der Mensch als „Schwachstelle" bezeichnet, da er auch unbeabsichtigt bereits großen Schaden anrichten kann - etwa durch Absicht, Unachtsamkeit und oder Unwissenheit.

Die am Prozess beteiligten Personen besitzen die meiste Kontrolle während der Laufzeit und nur an wenigen Stellen wird diese Kontrolle kurzweilig abgegeben (etwa bei einem Webservice-Aufruf, einer Datenbankverbindung, Script-Ausführung, und weitere). Der zugrundeliegende Workflow selbst ist somit größtenteils abhängig von manuellem (menschlichem) Zutun. Der folgende Baum in Abbildung 4.23 fasst die wesentlichen Bereiche der Gefahren, die durch eine „Insider-Person" ausgelöst werden kann, nochmals zusammen.

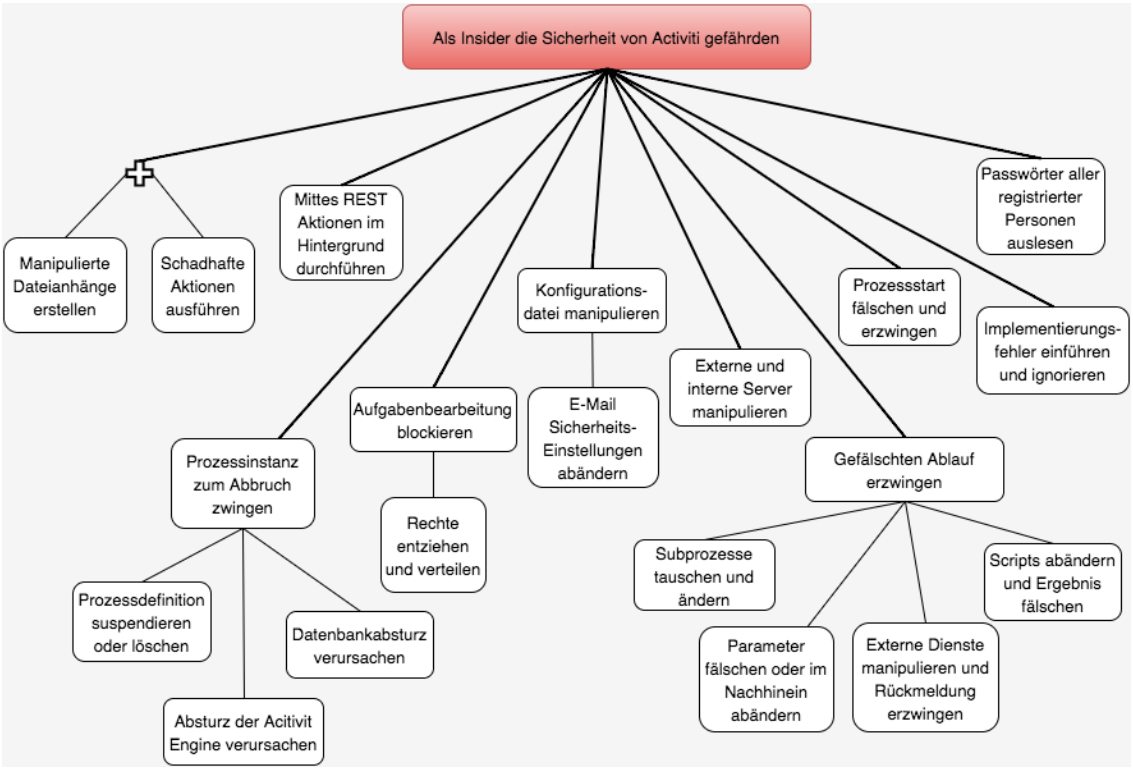

Abbildung 4.23: Attack Tree - Insider (eigene Darstellung)

Insider können Instanzen, Prozessteilnehmer (sowohl menschliche Rollen und Gruppen, als auch maschinelle Entitäten) aber auch externe Implementierungen sowie Systemkomponenten (Webserver, Datenbank) als potentielles Angriffsziel betrachten.

Teilhabende Menschen sind nicht immer in der Abarbeitung im Workflow beteiligt, sondern haben eine rein administrative Rolle. Diese Menschen haben beispielsweise direkten Zugang zu diversen Server und Einblicke in benötigte Bibliotheken und wissen, wo diese vorzufinden sind. Werden hier die Rechte missbraucht, können die bereits beschriebenen Systemunterbrechungen mit Leichtigkeit ausgelöst werden (Server abschalten, Bibliotheken entfernen).

4.9.1 Passwort ablesen

Die zugrunde liegende Datenbank beherbergt sensible Daten, wie das Passwort und die jeweilige Benutzerkennung in Kombination. Details hinsichtlich der Datenbank sind in Kapitel 4.5 zu sehen. Ein Insider, der einen Datenbank Zugriff hat (Datenbank Administrator), findet in der von Activiti vordefinierten Tabelle *ACT_ID_ USER* alle personenbezogenen Daten als *plain text* gespeichert vor (also nicht verschlüsselt) und kann diese kopieren und bestenfalls an Dritte weitergeben. Allerdings reicht es ebenso die Rolle *admin* zu haben, um an die gleichen Daten zu gelangen, da die gleiche Tabellenansicht auch über Activiti Explorer einzusehen ist (Abbildung 4.24). Dies ist ein wesentlicher Aspekt, der einen Einfluss auf die Sicherheit von Activiti ausübt und eine große Schwachstelle bedeutet, welche bereits durch einen Social Engineering Angriff zur Erlangung der Zugangsdaten eines *admin*-Kontos ausgenutzt werden kann.

Alle weiteren Rollen können nur ihre eigenen Daten abändern, wobei die Passwort-Eingabe bei Erneuerung vor einem Mitlesen geschützt ist (die eingegebenen Zeichen werden mittels Platzhalter ersetzt).

ACT_ID_USER

ID_	REV_	FIRST_	LAST_	EMAIL_	PWD_	PICTURE_ID_
danker	2	S.	Danker		danker	
fozzie	2	Fozzie	Bear	fozzie@activiti.org	fozzie	22
gonzo	2	Gonzo	The Great	gonzo@activiti.org	gonzo	18
kermit	2	Kermit	The Frog	kermit@activiti.org	kermit	7
pruhf	2	P.	Pruhf		pruhf	
sacher	3	S.	Sacher		sacher	
schmitt	3	A.	Schmitt		schmitt	

Abbildung 4.24: Activiti Explorer admin-Ansicht: Sichtbare Passwörter (Screenshot)

Es ist nicht der Regelfall, selbst in der Rolle des Administrators, dass Passwörter nicht verschlüsselt einzusehen sind. Diese Situation erhöht also die Wahrscheinlichkeit, dass Personen mit dieser Rolle ausgenutzt oder eben selbst als Insider tätig werden.

4.9.2 Absichtliche Fehler

Unter den beabsichtigten Aktionen wurde bisher der Aspekt der Fehler in der Implementierung noch nicht berücksichtigt. Solche Absichten können das Fehlen von Eingabeüberprüfung, das Fehlen von Zugangskontrollen und das Fehlen von Sicherheitskontrollen beinhalten, welche nach OWASP eine der häufigsten Ursachen von Schwachstellen darstellen und bedeutend sind für ein WfMS.

Absichtliche Fehler können über all eingeschleust werden, besonders dann, wenn der Insider der zuständige Programmierer oder Systemadministrator ist. Selbst die (fahrlässige) Anwendung veralteter Software kann für externe Angreifer eine offene Tür darstellen, sofern die alte Version eine Schwachstelle besitzt und bekannt ist.

Das Eingreifen in den Workflow über Activiti REST wird besonders auch für Insider begünstigst. Hier kann die Person ohne sofort aufzufallen Instanzen angreifen und diese mit einer Leichtigkeit manipulieren. Die Aktionen, die nach erfolgreicher Ausführung keinerlei Fehlermeldung im Activiti Explorer verursachen, bleiben meist (zumindest für längere Zeit) unentdeckt, so die Übernahme von Aktivitäten, für die nur eine andere abgegrenzte Gruppe Zugang hätte. Werden keine Kontrollen implementiert, können unberechtigte Funktionen und Aktionen ausgeführt werden, während niemand davon erfährt.

Mit einem manipulierten Dateianhang oder einer Weiterleitung auf eine schädigende Webseite wird es sogar für eine „normale Person", die im System angemeldet ist, möglich anderen angemeldeten Personen beabsichtigt Schaden zuzufügen. Datei Anhänge und Webseiten-Aufrufe werden vom System nicht auf Schadsoftware überprüft und können mit Leichtigkeit in das Activiti BPM geladen werden. Dabei spielt das Format der Datei keine Rolle. Dies kann als eine wesentliche Schwachstelle betrachtet werden.

4.9.3 Fehler in der Konfiguration

Sämtliche Standard-Benutzer und Prozessdefinitionen können anfänglich in der Activiti Konfigurationsdatei `activiti-custom-context.xml` eingesehen und abgeändert werden. In einer Organisation mit einer großen Anzahl an registrierten Personen würde ein Aktiv-setzen dieser Standard Benutzer nicht auffallen. Sofern diese Funktion für den Betrieb nicht komplett entfernt ist, wird die Gefahr zu jeder Zeit bestehen.

Notwendige Einstellungen betreffen auch die Browser- und Webserver Einstellungen. Im Bereich des direkten Arbeitsplatzes (*social engineering*) kann beobachtet werden, dass die automatische Nutzererkennung oft aktiviert ist, weil es das schnelle Einsteigen in Activiti erleichtert. Auf diese Weise ist es umso leichter, sich über einen fremdes Gerät anzumelden, da nur der Benutzername erforderlich ist. Oftmals reicht bei solch einer Browser Einstellung der erste Buchstabe eines Namens um die Autovervollständigung auszunutzen.

Es ist grundsätzlich möglich, versteckte Gefahren durch eine anfänglich automatisierte Überprüfung der Weboberfläche (*scan*) durchzuführen. Mittels einer aufzeichnenden Zwischenkomponente können dadurch Informationen eingeholt werden, welche im nächsten Abschnitt vorgestellt werden.

4.9.4 OWASP Zed Attack Proxy - ZAP

Wie der Name bereits verrät ist Zed Attack Proxy (ZAP)[14] ein von OWASP entwickeltes und im Internet frei zugängliches Programm. Diese plattformunabhängige Anwendung zielt darauf ab, Webanwendungen zu testen und mögliche Schwachstellen aufzudecken. Zu seinen Eigenschaften zählen unter anderem eine alle durchgeführten Aktivitäten im Web-Browser aufzeichnende Schnittstelle (Einsatz als *proxy*) und sämtliche Arten automatischer und passiver Durchsuchungen der gewählten Seiten und ihren Verbindungen (*scanner*).

In dieser Arbeit wurde ZAP als Proxy eingesetzt, um zunächst die Weboberfläche von Activiti Explorer und entsprechende Aktivitäten aufzuzeichnen, um anschließend eine umfassende automatisierte Überprüfung sämtlicher Seiten durchzuführen. Bei der initialen Durchführung

[14] Zed Attack Proxy abrufbar unter www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project, letzter Zugriff Mai, 2016

wurden einige Warnmeldungen (es werden vier Risikostufen unterschieden) und sicherheitsrelevante Aspekte aufgedeckt, die nach ZAP jedoch als nicht risikoreich (*medium*) eingestuft wurden. Bestätigt wurde beispielsweise die Gefahr der bestehenden Autovervollständigung (siehe Abbildung 4.25)

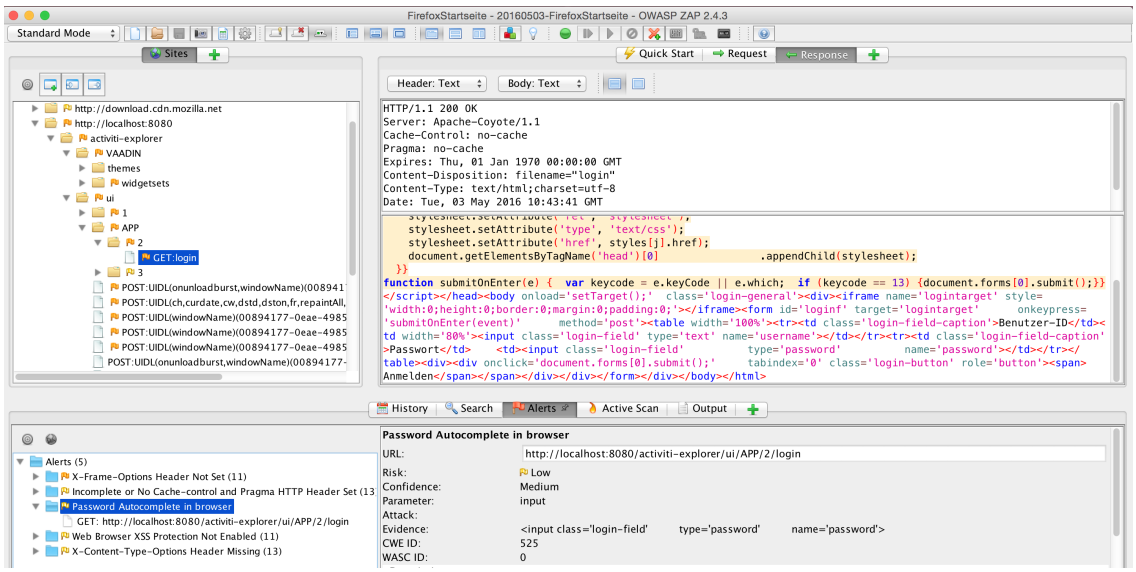

Abbildung 4.25: ZAP - Passwort Autovervollständigung aktiviert (Screenshot)

Ein Angreifer hat darüber hinaus anhand dieser Aufzeichnungsmethode auch die Chance sämtliche Zugangsdaten, also eingegebener Benutzername und das Passwort, zu erlangen (Abbildung 4.26). In dem Fall hat sich die Person „sacher" mit dem Passwort „sacher" im System identifiziert.

Anhand der weiteren Risiken, die durch ZAP ermittelt werden konnten, ist ersichtlich, dass aufgrund mangelnder Konfiguration (teilweise auch Implementierung) ein Cross Site Scripting (XSS) Angriff auf die Webanwendung möglich ist. Es handelt sich dabei um die Konfiguration des Web Servers, welcher Einfluss auf den XSS Schutzmechanismus des Browsers ausübt (Schutz aktivieren durch „`X-XSS-Protection: 1;`"). Gleiches gilt für den Schutz gegen sogenannte „Clickjacking"-Angriffe. Bei einem „Clickjacking" wird über einem durch Knopfdruck ("click") auswählbaren Bereich auf der aufgerufenen Webseite (beispielsweise ein Bereich „Jetzt absenden!") ein gefälschter und unsichtbarer Bereich platziert, sodass unbeabsichtigt und vor allem unwissentlich auch der gefälschte Bereich ausgewählt wird. Ist das „Opfer" dabei bereits im Activiti Explorer angemeldet, können damit sämtliche Informationen (Sitzungs-Identifikation) an einen fremden Server weitergeleitet werden [15] (Kreditkartennummer, PIN). Es besteht somit aufgrund derzeitiger Einstellung die Gefahr durch einen Clickjacking Angriff Daten an Dritte weiterzugeben und damit auch den Zugang zum Activiti Explorer.

Angriffsversuche auf Activiti Explorer können Fehlermeldungen herauslocken, die weiterführende und sensible Informationen enthalten. Meist betreffen diese Informationen genau die Dateien im System, aus denen die Fehlermeldung generiert werden und tragen dazu bei, dass weitere Angriffe auf die Webanwendung durchgeführt werden können. ZAP bezeichnet dieses bestehende Risiko als „Application Error Disclosure" (siehe Abbildung 4.27).

Daraus lässt schließen, dass alleine die unzureichende Konfiguration des Web Browsers und auch des Web Servers die Sicherheit des Gesamtsystems bereits gefährdet. Nicht ohne Grund zählt OWASP die „fehlerhafte Konfiguration" zu den zehn häufigsten Risiken.

[15] Clickjacking Abwehr bei MSDN: blogs.msdn.microsoft.com/ie/2009/01/27/ie8-security-part-vii-clickjacking-defenses/, letzter Zugriff Mai, 2016

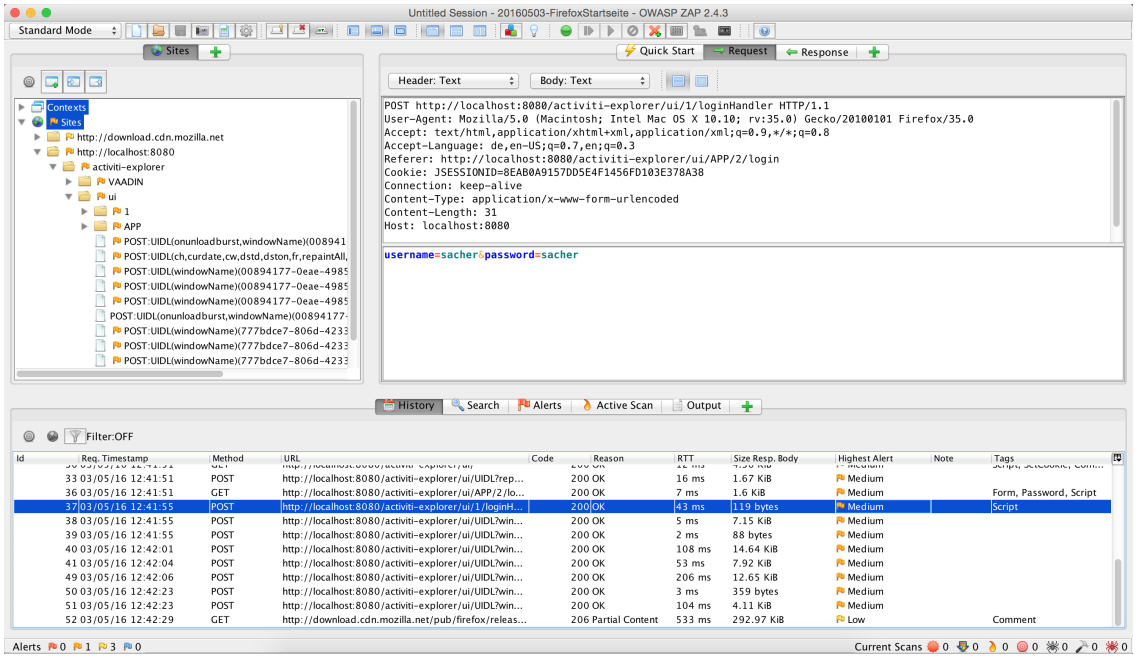

Abbildung 4.26: ZAP - Benutzername und Passwort ablesen (Screenshot)

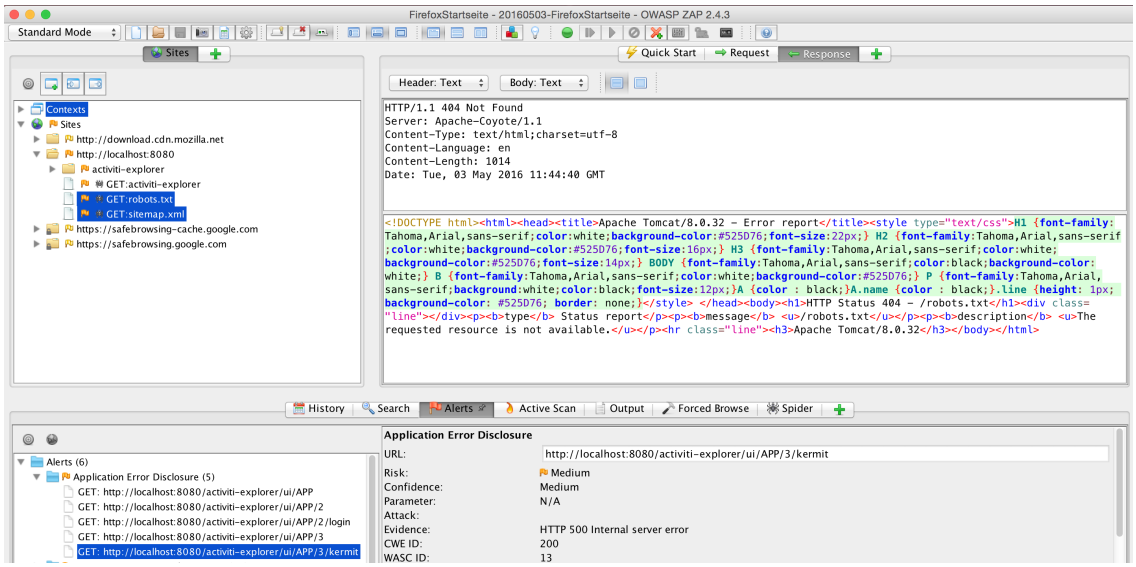

Abbildung 4.27: ZAP - Offenlegung von Fehlermeldungen (Screenshot)

4.9.5 Ausgelagerte programmatische Aufgaben

Das Auslösen externer Aktivitäten und das Empfangen von Rückmeldungen ist wesentlich für einen Prozess mit maschinellen Tätigkeiten. Alle im laufenden Prozessbeispiel ausgelagerte Aktivitäten liegen als Java Implementierung im Dateisystem vor. Aufgerufene Skripte sind direkt in den Prozessdefinitionen enthalten. Der Aufruf dieser automatischen Aktivitäten bedeutet eine kurzweilige Unterbrechungen der menschlichen Kontrolle über den Prozess und auf erwartete Ergebnisse nach Bearbeitung wird „blind" vertraut. Der Administrator kann somit, wie auch im Abschnitt 4.8 beschrieben, während der Laufzeit auf verschiedenste Art zumindest eine Unterbrechung der Instanz hervorrufen. Aus externer Sicht betrachtet ist eine derartige Manipulation effektvoll, sobald der Webservice funktionsunfähig gemacht oder so manipuliert wird, sodass falsche Ergebnisse an den Prozess zurückgeliefert werden. Im Beispielprozess wird nur überprüft, ob der zurückgelieferte Datentyp und Parameterwert den Anforderungen entspricht. Entsprechen die Werte nicht den Anforderungen bleibt der Workflow an dieser Stelle entweder stehen oder die unpassende Rückmeldung wird anderweitig bearbeitet, damit der Workflow an dieser Stelle zumindest weitergeführt werden kann, auch wenn eine korrekte Antwort ausbleibt.

4.10 Erkenntnisse im Überblick

Eine Zusammenfassung dieses Kapitels gibt die wesentlichen Erkenntnisse der Analyse kompakt wieder. Jeder Angriffsbaum wird dafür noch einmal für einen Soll-Ist Vergleich betrachtet, um damit einen Gesamteindruck über Analysen und die mögliche Angriffsfläche des vorliegenden WfMS zu erhalten. Durch die erarbeiteten Angriffsbäume wurden neben der Webanwendung selbst, die dazugehörenden Komponenten, wie die externe Datenbank, externe sowie interne Benutzer, externe Dienste und der Workflow als Gesamtheit in Betracht gezogen.

Attack Tree 1 - Soll: Zutritt und Zugangsdaten zu Activiti erlangen, mittels:

- Social Engineering, in dem Mitmenschen manipuliert werden
- SQL Einschleusung, um etwaige Eingabeüberprüfungen zu umgehen
- Manipulation der Einstiegsseite, um eingegebene Formular-Daten abzufangen
- automatisiertes Erraten des Passwortes
- Aufzeichnen der Authentifikationsinformationen
- eines am Rechner vorinstallierten Programmes zur Passwort-Wiederherstellung

Attack Tree 1 - Ist: Anmeldeinformationen können auf unterschiedliche Wege erreicht werden, am einfachsten jedoch über Social Engineering. Bei Zugang zum Rechner ist das Verteilen von Spionageprogramme ebenso ein einfaches Mittel, um an die Daten des Prozesses als auch der Benutzer zu gelangen. Wie in Attack Tree 6 „Der Insider" beschrieben, besteht derzeit Gefahr unzureichender Konfiguration, die Nachahmung der Login-Seite und eine „Clickjacking"-Attacke ist durchführbar. Die Analyse mit ZAP liefert alle Berechtigungsinformationen, indem aufgerufene Seiten analysiert werden. Ein Versuch über die SQL Einschleusung die Eingabe der korrekten Passwort-Benutzername Kombination zu umgehen, führte mit den durchgeführten Parameter-Kombinationen zu keinem positiven Ergebnis.

Attack Tree 2 - Soll: Durch Datenbank Manipulation Fehlverhalten auslösen, mittels:

- direktem Angriff auf die Datenbank-Inhalte und Umgehen einer Eingabeüberprüfung auf der Einstiegsseite und im angemeldeten Zustand
- manuelle und automatisierte Eingaben und Änderungen von Parameter-Werten
- Datenbank Anfragen mit SELECT Anweisung überschreiben

- Verfügbarkeit einschränken

Attack Tree 2 - Ist: Der Zugang zur Datenbank ist als Administrator gegeben aber die Einschleusen von gefälschten Werten ist mit den durchgeführten Mitteln der automatisierten Übertragung nicht möglich. Sämtliche manuelle Eingaben werden entweder am Eingabeort zugelassen (Datentyp) oder als einfacher Text gespeichert. Manuelle Datenänderung über die Instanz ist möglich und wird nicht durch Warnmeldungen behindert, der Zugang und Änderungen von Daten über Activiti REST ist ebenfalls möglich. Die Verfügbarkeit kann durch Unterbrechung des Servers eingeschränkt werden. Der Prozess aber auch die Activiti Platform selbst wird an der entsprechenden Stelle unterbrochen und kann nicht weiter bedient werden.

Attack Tree 3 - Soll: Verwendete Ressourcen manipulieren, mittels:

- Ausnutzen von bekannten Schwachstellen in externen Diensten
- Überbelastung der Dienste zur Erreichung von Dienstverweigerung
- direkte Manipulation der Dienste, wie Server, Bibliotheken und Prozessdefinitionen
- Mail-Server Sicherheits-Einstellungen manipulieren
- Gefährliche Dateianhänge hinzufügen und ausführen lassen

Attack Tree 3 - Ist: Unsichere Dienste werden aus aktueller Sicht nicht verwendet und es besteht kein Fund in der Schwachstellen Datenbank. Eine automatisierte (oder manuelle) Dienstverweigerung durch eine hohe Anzahl von gestarteten Prozessinstanzen, Webservice Anfragen und Datenbankzugriffe ist nicht ermöglicht, aber Abänderung externer Implementierung und Änderung der Sicherheitsfunktion des e-Mail Servers führen zum erzielten Ergebnis. Das Abschalten sämtlicher Dienste zur direkten Manipulation ist mit entsprechendem Zugang möglich. Das Anhängen von Dateien und Internetseiten an ad-hoc Aufgaben kann genutzt werden, um schadhaften Code auszuführen oder Benutzer auf eine manipulierte Internetseite zu locken (Social Engineering, Phishing Attacke).

Attack Tree 4 - Soll: Berechtigungen der Benutzer manipulieren (Rechte umgehen, ausnutzen, ändern), mittels:

- Manipulation der Zuordnungen innerhalb der Prozessdefinition des zugrundeliegenden Prozessmodells
- Ausnutzung fremder Rechte zur Ausführung von Aktivitäten, die nicht im persönlichen Zuständigkeitsbereich liegen
- Abänderung der Implementierungen zur Zugriffskontrolle (separation-of-duties, binding-of-duties)
- Änderung in den Datenbanktabellen zu Gruppen und Benutzer
- durch systematische URL Aufrufe als unberechtigte Person
- Zuteilung der Berechtigung als Administrator
- Abänderungen von Parameter während der Laufzeit über Activiti REST

Attack Tree 4 - Ist: Mit Zugang zu Prozessdefinition und Implementierungen sind Änderungen bezüglich der Berechtigung möglich. Als Datenbankadministrator werden die Zuteilungen der Rechte durch Hinzufügen oder Entfernen einfach manipuliert, ohne Informationen an die betreffenden Personen weiterzuleiten. Eine systematische Überprüfung der Funktionsaufrufe führte nicht zum gewünschten Ziel und der Aufruf einer anderweitig zugeteilten Aufgabe führt in die leere Aufgabenliste (teilweise mit Warnmeldung).

Attack Tree 5 - Soll: Prozesse und Ablauf der Instanzen gefährden (Fehlverhalten hervorrufen), mittels:

- Verursachen einer Dienstverweigerung über Activiti REST
- Unterbrechung aktiver Instanzen mithilfe von Abänderungen von Bibliotheken
- Server Einstellungen, Gruppen- und Benutzerzuordnungen und Prozessdefinitionen
- Bearbeitungspfade erzwingen durch Manipulation von Aufgabenfreigaben, Parametern, Subprozessen, externen Diensten und Implementierungen sowie der zuständigen Rolle
- Auslösen von abnormalem Prozess Status

Attack Tree 5 - Ist: Auch nicht vorgesehene Initiatoren können einen Prozess starten (Activiti REST) und ein endloses Starten externer Aktivitäten kann durchgeführt werden, jedoch eine hohe Anzahl zur Erreichung einer Dienstverweigerung konnte nicht umgesetzt werden. Eine Unterbrechung kann herbeigeführt werden durch Verändern der benötigten Implementierungen, inaktiv setzen externer Dienste, sowie auch die Sicherheitseinstellung des e-Mail Servers. Eine Gefährdung wird während der Instanz durch Änderungen an den Beteiligten ermöglicht. Dabei ist der Stillstand des Workflows einer der Folgen. Über Activiti REST sind bereits bearbeitete Parameter veränderbar, um falsche Pfade dadurch zu erzwingen. Eine Person, die keinen Zugriff auf Aktivitäten haben dürfte, kann diese dann bearbeiten. Abnormale Situationen werden auch durch Änderungen in der Prozessdefinition realisiert.

Attack Tree 6 - Soll: Als Insider die Sicherheit von Activiti gefährden, mittels:

- Activiti REST
- Manipulation der Server, Benutzerrechte und Konfigurationsdateien
- Fälschung von Prozessabläufen,
- ungeplanter und nicht legitimer Instanziierung
- absichtlichen Implementierungsfehler
- Ausgabe der Benutzeridentifikationen aller registrierter und somit berechtigter Benutzer

Attack Tree 6 - Ist: Der Insider kann neben Social Engineering zu Administratorrechten gelangen (oder diese bereits besitzen) und somit sämtliche Aktionen ausführen, die bereits in den bisherigen Attack Trees vorgestellt wurden. Selbst, wenn der Insider nicht am täglichen Geschäft, also am Prozess beteiligt ist, kann er diesen gezielt gefährden und Ergebnisse verfälschen. Darüber hinaus konnten durch automatisierte Untersuchungen weitere Gefahrenpunkte ausfindig gemacht werden, die vorrangig die Konfiguration anbelangen. Einzig der externe Webservice zur Bestimmung der Bank-Informationen ist außer Reichweite, ohne direkt mit der bereitstellenden Person oder Organisation in Kontakt zu treten. Der Insider kann durch beabsichtigte Fehler (und Fehlkonfigurationen) den Prozessablauf manipulieren aber auch durch schadhaften Dateianhang Daten freigeben oder Mitarbeiter gefährden. Eine Überprüfung wird dabei nicht in Betracht gezogen.

4.11 Nützlichkeit von Scanner

Sowohl ein Angreifer von Außen, als auch der Insider hat die Möglichkeit eine automatisierte Überprüfung der Webanwendung durchzuführen. Ein Systemadministrator, der nicht als Angreifer betrachtet wird, sollte, bevor es der eigentliche Angreifer unternimmt, eine solche Überprüfung vornehmen, um mögliche und übersehene Schwachstellen frühzeitig zu erkennen.

Der eingesetzte Scanner ZAP konnte allein bei der Initialüberprüfung der Einstiegsseite bereits vier (milde) Risiken an mehreren Stellen aufdecken. Es konnten zwar keine direkten Angriffe durchgeführt werden aber das Risiko bestand zu dem Zeitpunkt bereits.

Manuelle Überprüfungen und automatisierte Tests können gemeinsam verwendet werden, um jeweils auch solche Schwachstellen zu entdecken, die auf manuellem Weg nur sehr schwer zu finden sind und die mit automatisierten Tests möglicherweise übersehen werden, da bestimmte Muster angewendet werden und somit eventuell nicht jedes Detail berücksichtigt wird.

Grundsätzlich jedoch gehören auch für die Anwendung und Durchführung von automatisierten Überprüfungen ein Grad an Wissen dazu. Tests können zwar anzeigen, dass keine Fehler gefunden oder Angriffe durchgeführt werden konnten, dies soll jedoch nicht zur Gänze bedeuten, dass tatsächlich keine Möglichkeit besteht. Die Bedienung solcher Werkzeuge sollte im Detail nachlesbar und anwendbar sein, da sonst die Gefahr besteht, nur mit der Grundeinstellung des Werkzeuges, bloß die Oberfläche zu überprüfen.

4.12 Vergleich und Gemeinsamkeiten mit der ATLIST Methode

Die eingangs kurz beschriebene und etwas abweichende zweite Schwachstellenanalyse bedient sich der ATLIST Methode und hat zum Ziel das zugrundeliegende WfMS auf bereits „bekannte Schwachstellen" hin zu analysieren. Dabei bilden folgende fünf ATLIST Bäume die Grundlage der Analyse[16]:

1) Prozess stoppen
2) Prozess starten
3) Prozess steuern
4) Server überwachen und
5) Anwendung überwachen

Der wesentliche Unterschied zur Attack Tree Methode ist die Tatsache, dass in diesem Fall die *Schwachstellen bekannt* sind und sich der Aufbau und die Durchführung daher an diesem Aspekt orientiert. Für die Attack Tree Methode besteht zu der gleichen Zeit das Wissen über die *Ziele*, weshalb diese den Fokus darstellen.

Die Herausforderung beider Methoden ist letztendlich jedoch der Angriff selbst. Einerseits die bekannte Schwachstelle tatsächlich auszunutzen und andererseits den Angriff auf das System, um eine etwaige Schwachstelle zu finden. Ein hohes Maß an Wissen über Angriffsmöglichkeiten wird in beiden Methoden vorausgesetzt, um ein tiefgründiges und detailreiches Analyseergebnis darbieten zu können.

Die Analyse mittels der ATLIST Methode endet jedoch nicht bei der Ausnutzung potentieller Sicherheitslücken. Durch die Ergebnisse der Analysen sind etwaige Muster ableitbar, die wiederum in automatisierte Schwachstellen-Scanner eingebettet werden können. Dieser Aspekt wird jedoch als besonders schwierig erachtet, nicht zuletzt aufgrund der Ergebnisse, die einen hohen Detaillierungsgrad aufweisen müssen, um als Muster weiterverarbeitet werden zu können.

4.13 Auffälligkeiten im laufenden Betreib

Während der Bedienung von Activiti Explorer und vor allem bei der Durchsuchung aller möglichen Funktionen zur Überprüfung der Zugriffskontrollen und der erlaubten URL Aufrufe wird erkenntlich, dass auf keiner Funktionsseite die Möglichkeit angeboten wird, einen Arbeitsschritt zurück zu nehmen oder einzusehen ("Zurück"-Schaltfläche). Es ist lediglich das Löschen oder Suspendieren eines Prozesses möglich, was jedoch nur die Rolle *admin* durchführen kann.

[16]Sämtliche Erkenntnisse wurden von Sabine Gottwald, der Autorin dieser zweiten Schwachstellenanalyse, zur Verfügung gestellt. Für detaillierte Informationen kann in ihrer Arbeit mit dem Titel „ATLIST Tree zur Analyse von Schwachstellen von WfMS" nachgeschlagen werden.

In der Rubrik der archivierten Instanzen ist die Möglichkeit gegeben, Aktivitäten in Form von Ereignissen zu erstellen. Allerdings können diese nicht abgesendet werden und es entsteht dabei stets eine Fehlermeldung mit einer *task id*, die „nicht gefunden" werden kann.

Sofern eine Person nicht selbst für eine Instanz verantwortlich ist (Person ist nicht Initiator der Instanz), können auch keine Informationen, wie Parameter und zuständige Rollen eingesehen werden. Der Administrator kann einsehen, welcher Parameter welchen Wert hat, er kann jedoch die Zugehörigkeit der nächsten und unbearbeiteten Aktivität nicht einsehen (nur unter Zuhilfenahme der zugrundeliegenden XML Prozessdefinition ist diese Information einzusehen).

Als letzten Punkt wird die Aufgabenverteilung näher betrachtet. Die zuständige Person wird *assignee* genannt (zugeteilte und bevollmächtigte Person), welche durchaus auch der Initiator der Instanz sein kann. Eine Aufgabe kann, wenn diese angenommen aber noch nicht bearbeitet wurde, einer anderen Person zugeordnet werden (*re-assign*). Allerdings wird im Standard-Umfeld keine Möglichkeit geboten, eine andere Person auszuwählen, um dieser eine Aufgabe zuzuteilen. Dieser Aspekt führt zu einer Fehlermeldung, nachdem wiederum nur die eigene Person wieder gewählt werden kann. Eine Zuordnung ist nicht möglich und kann nur von einem Administrator beispielsweise über Activiti REST durchgeführt werden.

Die elementaren Bausteine des Prozesses und Workflows wird in der Modellierungsumgebung erstellt. Die Zuordnung von Gruppen und Personen zu Aktivitäten, sowie die Anbindung von externen Diensten und Skripte ist ebenso bereits während der Modellierung möglich, doch wesentliche Implementierung zur Gewährleistung der Zugriffskontrollen sowie Fehlerbehandlungen und die Ausgabe von Warnungen, bei auffälligem oder untypischem Verhalten hängen von der implementierenden Person ab. Hier entstehen somit (wenn überhaupt) die ersten Sicherheitslücken, die im weiteren Verlauf ausgenutzt werden können.

4.14 Übertragbarkeit der Ergebnisse

In der hier durchgeführten Analyse handelt es sich um eine systematische Schwachstellenanalyse, die auf ein konkretes WfMS durchgeführt wurde. Nach Betrachtung der Durchführung, der angewendeten Methode sowie der Resultate kann diese Analyse auch auf andere (webbasierte) WfMS übertragen werden und ist nicht an das hier untersuchte WfMS gebunden. Es wurden sämtliche beteiligte Entitäten betrachtet, die Einfluss auf das System ausüben. Dazu zählen ebenso die Zugriffskontrollmechanismen, die jedes WfMS benötigt.

Die angewendete Attack Tree Methode erlaubt es ebenso die systematische Analyse zu verallgemeinern und jedes weitere WfMS auf diese Weise zu untersuchen. Je nach vorhandener Funktionalität des Systems aber auch Erfahrung der Person, die die Angriffsbäume erstellt und Angriffe durchführt, variierte das Aussehen des Angriffsbaums, jedoch kann die Vorgehensweise beibehalten werden.

Auch die Resultate der Tests zeigen, dass diese Analyse anhand eines WfMS nicht an dieses gebunden ist. Sobald das System von mehreren Personen genutzt wird und diese sich am System anmelden müssen, spielt die sichere Speicherung der sensiblen Daten eine wesentliche Rolle. Sind an einem komplexen Prozess mehrere Personen beteiligt, spielt auch hier die Trennung der Rollen und die dazugehörenden Sicherheitsmechanismen eine wesentliche Rolle. Der Angriff auf einflussreiche Systemkomponenten, Zugriffskontrollen, sensible Daten , Ressourcen und Instanzen sind nicht auf nur ein WfMS beschränkt und die Attack Tree Methode kann in jedem erdenklichen Bereich angewendet werden, um das System auf ihre Sicherheit zu überprüfen.

Nicht jedes Resultat aus dieser Arbeit ist auf jedes WfMS übertragbar, aber der Angriffsbaum sowie der Angriffsweg und Tests können angewendet werden, um in Erfahrung zu bringen, ob das gleiche Resultat folgt. Während der Bearbeitung von Aktivitäten ist im Activiti BPM zum Beispiel die Möglichkeit gegeben, schadhafte Dateianhänge und Referenzen zu manipulierten Webseiten hinzuzufügen. Werden in anderen WfMS in diesem Bereich verstärkte Sicherheitsmaßnahmen

implementiert, besteht dort womöglich keine Schwachstelle, die durch derartige Dateianhänge ausgenutzt werden können.

5 | Zusammenfassung

Nach einer eingehenden Recherche über unterschiedliche Angriffsabsichten wurden Angriffsszenarien gegen das WfMS graphisch mithilfe der Attack Tree Methode erstellt. Die definierten Ziele umfassen dabei auch die Komponenten rund um das WfMS Activiti BPM und deckten das Umfeld auf diesem Weg ab. Die Funktionsweise des WfMS selbst, die beteiligten Personen mit ihren Rollen, die angebundene MySQL Datenbank sowie Java-Implementierungen mitunter zur Realisierung einer Teilaufgabe des Workflows, welches mit einem externen und öffentlichen Webservice kommuniziert, wurden mit in die Analyse aufgenommen, da sie Teil dieses Gesamtsystems darstellen.

Nach sowohl manuellen als auch automatisierten Testversuchen schadhafte Daten einzuschleusen, wurde schnell festgestellt, dass diese Art und Weise, dem System Schaden zuzufügen, nicht möglich ist. Auch ist die Ausgabe sämtlicher Datenbank Einträge dadurch verwehrt geblieben. Der Oberflächen Scan mittels Zed Attack Proxy hingegen stellte nach Authentifizierung mit dem System sämtliche Zugangsdaten bereit. Außerdem konnte durch diesen Scan erkannt werden, dass sämtliche Fehlkonfigurationen (vorwiegend Server-seitige Einstellungen) vorliegen, sodass unter anderem Clickjacking Angriffe, begünstigt sind.

In der Datenbank ist außerdem ersichtlich, dass sensible Daten, wie Passwörter und Benutzername nicht verschlüsselt vorliegen. Dies ist auch in der Rolle eines Administrators, der am Workflow beteiligt sein kann, der Fall und zeigt eine Schwachstelle auf. Auch diese Person hat somit die Möglichkeit sämtliche Zugangsdaten an Dritte weiter zu geben, beispielsweise als Opfer eines Social Engineering Angriffs. Glückt der Angriff auf die zugrundeliegende Datenbank, kann das gesamte System kompromittiert und auf jedes Konto zugegriffen werden.

Durch Manipulation externer Implementierung zur automatisierten Ausführung von Aufgaben, konnten zwar eine hohe Anzahl von Aufrufe und Berechnungen erfolgen, erzielten jedoch nicht das gewünschte Ziel einer tatsächlichen Dienstverweigerung (Denial of Service). Lediglich den Prozess betreffende Komponenten, die größtenteils als externe Bibliotheken zur Verfügung gestellt sind und die Prozessdefinition selbst konnten abgeändert werden, sodass es zu Fehlverhalten einer Instanz führte (Unterbrechungen, sowie Stillstand der Instanz). Zu grobe Änderungen, wie das gänzliche entfernen von wesentlichen Komponenten (wie der benötigte zweite Prozess), führt dazu, dass die Prozessdefinition von Activiti BPM nicht akzeptiert wird. Gleiches gilt in umgekehrter Form, wenn über Activiti BPM Rollen und Zuständigkeiten entfernt werden, die jedoch von der Prozessdefinition vorausgesetzt werden. Der Prozessstart wird dadurch behindert.

Es stellte sich schnell heraus, dass als Insider einige Optionen zur Verfügung gestellt sind, um auf diverse Arten Informationen einzusehen, Daten zu verändern und weiter zu geben. Über Activiti REST sind diverse solcher Aktivitäten umsetzbar und blieben im Activiti Explorer unentdeckt, da keine Warnmeldungen erscheinen und implementierte Zugriffskontrollen umgangen werden können. Rollen haben die Möglichkeit bereits an andere Rollen oder Gruppen zugeteilte Aufgaben zu übernehmen, ohne dass die zuständige Person davon in Kenntnis gesetzt wird. Bereits verarbeitete Parameter können auf diesem Weg abgeändert werden, sodass der Workflow die neue Abarbeitung entsprechend ohne Überprüfung übernimmt. In sensiblen Situationen, in denen es beispielsweise um einen Geldtransfer an ein vorgegebenes Konto handelt, könnte dieser

Eingriff einen enormer Schaden verursachen. Sämtliche sicherheitsrelevante Implementierungen, die nicht im Activiti BPM voreingestellt sind, betreffen vorrangig Separation-of-Duties, sowie Binding-of-Duties und die Einschränkung zu Datentypen. Die Implementierung dieser Aspekte trennt und regelt sämtliche Zugriffskontrollen, werden jedoch durch den Eingriff über Activiti REST nichtig. Administrator Rechte können somit mit Leichtigkeit ausgenutzt werden.

Activiti BPM kann entweder im eigenen Quellcode Fehler aufweisen, die zuerst zu finden sind, um ausgenutzt werden zu können oder durch die umgebenen Komponenten beeinträchtigt und gefährdet werden, da ein geglückter Angriff auf diese einen großen Einfluss auf das eigentliche WfMS hat.

6 | Fazit und Ausblick

Der professionelle Angreifer weiß wonach er suchen muss, sobald er ein Ziel vor Augen hat. Findet er tatsächlich die entsprechend gesuchte Schwachstelle, ist er vermutlich auch in der Lage diese auszunutzen, um den gewünschten Schaden zu erzielen. Je mehr Erfahrung vorhanden ist, umso anspruchsvoller auch die Angriffsmöglichkeiten und die Suche nach Schwachstellen im betreffenden System. Gleiches gilt auch für Personen, die für die Entwicklung und Umsetzung verantwortlich sind. Je mehr Wissen über sicherheitsrelevante Aspekte in der Implementierung vorhanden ist, um so mehr ist das Produkt im Endeffekt gehärtet und weniger angreifbar. Für diverse Tests schon während der Entwicklungsphase können entsprechende automatisierte Werkzeuge eingesetzt werden, die auf mögliche Fehler bereits in der Implementierung hinweisen. Wesentliche Vorteile für die Sicherheit des WfMS verschafft die Verwendung des Spring Frameworks [32], welches unter Anderem auf das Prinzip der aspektorientierten Programmierung basiert. Demnach wird im Speziellen der Bereich Sicherheit isoliert behandelt und ermöglicht die Zugriffskontrolle auf Instanzebene.

Auch für die vorliegende Ausarbeitung gilt, dass nach den erarbeiteten und untersuchten Angriffsbäumen erst der erste Schritt in die Richtung umfassende und systematische Schwachstellenanalyse gesetzt wurde. Dieser Schritt kann als Sprungbrett für zukünftige und umfangreichere Analysen verwendet werden.

Abbildungsverzeichnis

1.1 Vorgehensmodell - Aufbau der Arbeit (eigene Darstellung) 5

2.1 Zusammenhang: Bedrohung, Angriff und Schwachstelle (in Anlehnung an [13]) . 13
2.2 Attack Tree - Exemplarischer Baum (eigene Darstellung) 15
2.3 ATLIST - Exemplarischer Baum (entnommen aus [28]) 16

3.1 BPMN Basiselemente (in Anlehnung an [1] und [22]) 19
3.2 Activiti Eclipse designer - Angebotene BPMN Erweiterungen (entnommen aus der Activiti Eclipse designer Anwendung) 20
3.3 Fiktive Unternehmensorganisation (eigene Darstellung) 25
3.4 Sicherheit in den e-Mail Einstellungen (Screenshot) 28
3.5 Geschäftsprozess - Fiktives Gesamtszenario (eigene Darstellung) 29

4.1 Überblick der häufigsten Angriffsabsichten (eigene Darstellung) 31
4.2 Gesamtprozess - Entitäten im Überblick (eigene Darstellung) 36
4.3 Gesamtprozess - Aktivitäten mit Bedingungen (eigene Darstellung) 36
4.4 Potentielle Schwachstellen mit Einfluss auf das WfMS (eigene Darstellung) 37
4.5 Vorgehensmodell - Attack Tree Analyse (eigene Darstellung) 39
4.6 Attack Tree: Zugang verschaffen (eigene Darstellung) 40
4.7 Activiti Einstiegsseite - Erforderliche URL und Parameter (Screenshot) 41
4.8 Zed Attack Proxy - Bestehende Clickjacking Gefahr (Screenshot) 43
4.9 Attack Tree - Datenbank (eigene Darstellung) 44
4.10 Warnung - Nach Trennung der Datenbank Verbindung (Screenshot) 46
4.11 Attack Tree - Ressourcen (eigene Darstellung) 48
4.12 Warnung bei entfernter Prozessdefinition während der Laufzeit (Screenshot) . . . 51
4.13 Phishing Versuch - Manipulierbare Anhänge (eigene Darstellung mit Screenshot) . 52
4.14 Attack Tree - Benutzerrechte (eigene Darstellung) 53
4.15 Funktionen ausschließlich für die Rolle: admin (eigene Darstellung) 55
4.16 Datenbank manipulieren - Rollen Anzeige (Screenshot) 56
4.17 Attack Tree - Instanz (eigene Darstellung) 57
4.18 Aufgabe einer anderen Gruppe steht in der eigenen Inbox bereit (Screenshot) . . . 61

4.19 Zugriffskontrolle fehlgeschlagen - Fremde Aufgabe bearbeitet (Screenshot) 61
4.20 Zugriffskontrolle funktioniert - Warnung bei Übernahmeversuch (Screenshot) . . . 62
4.21 Ansicht aus der XML Prozessdefinition und Activiti Eclipse designer - Bezeichnungen der Pfade (eigene Darstellung mit Screenshot) 63
4.22 Warnung - Kein entsprechender Pfad vorhanden (Screenshot) 63
4.23 Attack Tree - Insider (eigene Darstellung) 64
4.24 Activiti Explorer admin-Ansicht: Sichtbare Passwörter (Screenshot) 65
4.25 ZAP - Passwort Autovervollständigung aktiviert (Screenshot) 67
4.26 ZAP - Benutzername und Passwort ablesen (Screenshot) 68
4.27 ZAP - Offenlegung von Fehlermeldungen (Screenshot) 68

Tabellenverzeichnis

4.1 Attack Tree Ziele und Kategorisierung 38
4.2 SQLi - manuelle Testversuche 41
4.3 SQLi - automatisierte Testversuche 42

Listings

3.1 Umgebungsvariablen setzen 21
3.2 SQL Datenbank–Anweisung für die Verwendung mit Activiti BPM 23
3.3 Konfiguration in activiti-custom-context.xml 23
3.4 Webservice Aufruf - Ermitteln der Bankleitzahl 25
3.5 SQL Datenbank-Anweisung für den Datenbank-Aufruf 26
3.6 (Klassischer) Programmatischer Prozessaufruf 27
3.7 Konfiguration für den Mail-Task 27
3.8 JavaScript für Berechnungen 28
4.1 XML-Prozessdefinition: Zuständige Rollen 34
4.2 Auszug aus SQLDelegate.java - Auffangen von möglichem Fehler 49
4.3 Auszug aus WsDelegate.java - Abhilfe bei falscher Bankleitzahl 49
4.4 Programmatischer Prozessaufruf 51
4.5 Prozessaufruf - Daten direkt in der Prozessdefinition 51
4.6 Separation-of-duties - Leicht manipulierbar 54
4.7 Nicht zugeteilte Aufgabe übernehmen 60

Literaturverzeichnis

[1] *Einführung in BPMN 2.0*. Foliensatz erstellt von Orientation in Objects (www.oio.de). `http://www.oio.de/seminar/orientierungs/punkte/archiv/OrPoint2013/BPMN-2-0-Orientierungspunkt-9-13.pdf`. Version: 2013, Abruf: 12.5.2016

[2] AALST, W. van d. ; HEE, K. van: *Workflow Management - Models, Methods and Systems*. 1. Cambridge, Massachusetts, London England : MIT Press, 2004

[3] In: AALST, W. can d.: *Process-Aware Information Systems: Lessons to be Learned from Process Minding*. P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands : Springer Berlin Heidelberg, 2009, S. 1–26

[4] ABECKER, A. ; HINKELMANN, K. ; MAUS, H. ; MÜLLER, H. J.: *Geschäftsprozess-orientiertes Wissensmanagement*. 1. Springer Verlag, Heidelberg, 2002

[5] ALLWEYER, T. : *BPMN 2.0: Introduction to the Standard for Business Process Modeling*. Books on Demand, 2011

[6] ANZELETTI, B. ; LEWANDOWSKI, R. ; MESSNER, K. : *Geschäftsprozess- und Workflowmanagement*. University, 2004

[7] BECKER, J. ; KNACKSTEDT, R. ; HOLTEN, R. ; HANSMANN, H. ; NEUMANN, S. : *Konstruktion von Methodiken: Vorschläge für eine begriffliche Grundlegung und domänenspezifische Anwendungsbeispiele*. Leonardo-Campus 3, 48149 Münster : Becker and Grob and Klein, 2001. – 29 S.

[8] DINGER, J. ; HARTENSTEIN, H. : *Netzwerk- und IT-Sicherheitsmanagement - Eine Einführung*. Universitätsverlag Karlruhe, 2008

[9] ECKERT, C. : *IT-Sicherheit Konzepte - Verfahren - Protokolle*. Bd. 9. Oldenbourg Wissenschaftsverlag GmbH, München, 2014

[10] FICHTER, M. : *Workflow 2000 Von Business Process Management bis eBusiness - Produkte und Middleware-Komponenten (EAI) - Band 1 Einführung in die Workflow Thematik*. Bd. 1. Oderfelder Straße 17, 20149 Hamburg : Project Consult GmbH Hamburg, 2000

[11] FLECHAIS, I. : *Webinos, Secure WebOS Application Delivery Environment*. Deliverable, Februar 2001

[12] FREUND, J. ; GÖTZER, K. : *Vom Geschäftsprozess zum Workflow: ein Leitfaden für die Praxis*. Hanser Verlag, 2012

[13] GÖRL, H. : *Systematische Analyse und Konstruktion integrierter Sicherheitsarchitekturen für mobile verteilte Systeme*, Fakultät für Informatik der Technischen Universität München, Dissertation, Juni 2005

[14] GUDES, E. ; OLIVIER, M. ; RIET, R. P. d.: Modelling, Specifying and Implementing Workflow Security in Cyberspace. In: *Journal of Computer Security* 7 (1999), Nr. 4, S. 287–315

[15] *Kapitel* Konzepte und Einsatzmöglichkeiten von Workflow-Management-Systemen. In: HASENKAMP, U. ; SYRING, M. : *Wirtschaftsinformatik '93*. Springer, 405–422

[16] HEID, D. : *IT-Sicherheit in Organisationen: Analysebegriffe und Konzeptionsmethoden*. Hamburg : Bachelor + Master Publishing, Diplomica Verlag GmbH, 2012

[17] HOLLINGSWORTH, D. : Workflow Management Coalition; The Workflow Reference Model. In: *The Workflow Management Coalition Specification* 1.1 (1995), Jan, Nr. TC00-1003, S. 55

[18] INFOSECURITY, M. : WebSphere MQ Security White Paper - Part 1 / MWR InfoSecurity. 2008. – Forschungsbericht

[19] JOACHIM, M. : *Workflow-based Integration / Grundlagen, Technologien, Management*. Springer, 2005

[20] KELBERT, F. : *Authorization Constraints in Workflow-Management-Systemen*. 89069 Ulm, Deutschland, Universität Ulm Fakultät für Ingenieurwissenschaften und Informatik, Diplomarbeit, Juni 2010

[21] KIRK, W. : *Public Management, Die Gestaltung von Dienstleistungen im allgemeinen Interesse, Prozessmanagement*. Norderstedt : Books on Demand GmbH, Norderstedt, 2010

[22] KLINGBEIL, C. ; ARNDT, P. : *BPMN 2.0 Modellierungssprache für Geschäftsprozesse*. `http://www1.hft-leipzig.de/bpmn/index.php?option=com_content&view=article&id=5:impressum&catid=9:uncategorised&Itemid=518`. Version: 2014, Abruf: 12.5.2016

[23] KREBS, B. : *Online Cheating Site Ashley Madison Hacked*. `http://krebsonsecurity.com/2015/07/online-cheating-site-ashleymadison-hacked/`, Abruf: 11.09.2015

[24] LEITNER, M. ; MA, Z. ; RINDERLE-MA, S. : A Cross-Layer Security Analysis for Process-Aware Information Systems. (2015), Juli

[25] LEITNER, M. ; RINDERLE-MA, S. : A systematic review on security in Process-Aware Information Systems - Constitution, challenges, and future directions. In: *Information and Software Technology* 56 (2014), S. 273–293

[26] LIMMER, T. ; SOMMER, C. : *Computer Networks and Communication Systems Netzwerksicherheit*. Universität Erlangen-Nuremberg, Deutschland, November 2007

[27] LIPSKI, M. : *Social Engineering - Der Mensch als Sicherheitsrisiko in der IT*. Hamburg : Diplomica Verlag GmbH, 2009

[28] LOWIS, L. ; ACCORSI, R. : Vulnerability Analysis in SOA-based Business Processes. In: *IEEE*

[29] MERTENS ; BODENDORF ; KÖNIG ; PICOT ; SCHUMANN ; HESS: *Grundlagen der Wirtschaftsinformatik*. Bd. 11. Auflage. Springer Gabler Verlag, Berlin Heidelberg, 2012

[30] MILLER, J. ; FAN, M. ; SHETH, A. ; KOCHUT, K. : *Security in Web-Based Workflow Management Systems*. `http://LSDIS.cs.uga.edu`

[31] MOSER, W. : *Methoden zur Sicherheitsanalyse web-basierter Anwendungen*. Fachhochschule Köln, Campus Gummersbach, Fachhochschule Köln, Institut für Informatik, Diss., 2006

[32] MULARIEN, P. : *Spring Security 3 Secure your webapplication against intruders with this easy to follow practical guide*. 32 Lincoln Road, Olton, Birmingham, B27 6PA, UK : Packt Publishing Ltd., 2010

[33] RADEMAKERS, T. : *Executable business processes in BPMN 2.0*. Manning Publisher, 2012

[34] REICHMAYR, C. : *Collaboration und WebServices: Architekturen, Portale, Techniken und Beispiele*. Springer Berlin Heidelberg, 2013 (Business Engineering)

[35] RUEF, M. : *Die Kund des Penetration Testing - Handbuch für professionelle Hacker*. Zavelsteiner Straße 20, 71034 Böblingen : Computer und Literatur Verlag GmbH, 2007

[36] SAALBACH, K. : *Cyberwar Grundlagen-Methoden-Beispiele*. `http://www.dirk-koentopp.com/downloads/saalbach-cyberwar-grundlagen-geschichte-methoden.pdf`. Version: März 2015, Abruf: 12.5.2016

[37] SCHMID, K. : *Prozess-Optimierung im Output-Management*. Books on Demand, 2009

[38] SCHOOLMANN, J. : *Praxishandbuch IT-Sicherheit - Risiken, Prozesse, Standards*. Holger Rieger, Symposion Publishing GmbH, Düsseldorf, 2005

[39] SCHUMACHER, M. ; RÖDIG, U. ; MOSCHGATH, M.-L. : *Hacker Contest Sicherheitsprobleme, Lösungen, Beispiele*. Springer-Verlag Berlin Heidelberg, 2003

[40] SCHWARZ, S. ; ABECKER, A. ; MAUS, H. ; SINTEK, M. : Anforderung an die Workflow-Unterstützung für wissensintensive Geschäftsprozesse. DFKI GmnH, 67608 Kaiserslautern, Deutschland, (CEUR Workshop Proceedings)

[41] SILLER, H. : *Gabler Wirtschaftslexikon - Das Wissen der Experten*. Springer Gabler Verlag (Herausgeber),

[42] WESTPHAL, M. : *Phishing (Password harvesting fishing)*. Bd. 10. Genios WirtschaftsWissen Verlag, 2004

7 | Anhang

Kurzfassung

In dieser Arbeit wird dargestellt, wie mittels der baumorientierten Attack Tree Methode das Workflow Management System Activiti BPM auf Schwachstellen untersucht werden kann. Mittels Activiti Eclipse designer wird die zugrundeliegende Workflow Modellierung inklusive manueller und automatisierter Aktivitäten, Anbindung einer MySQL Datenbank und externer Webdienste durchgeführt. Die externen und automatisierten Aktivitäten werden großteils mittels Java Implementierungen realisiert und anschließend dem Workflow zur Verfügung gestellt. Die Analyse beinhaltet manuelle und automatisierte Tests insbesondere im Hinblick auf die Gewährleistung technischer Schutzziele. Sie umfasst sämtliche Bereiche auch rund um die Activiti BPM Plattform selbst, wie die angebundene MySQL Datenbank, weitere Ressourcen und Dienste, der Webserver, die beteiligten Rollen und Gruppen sowie ihrer Zugriffsrechte und Kontrollen, um auch eine mögliche Auswirkung auf das Grundsystem mit einzubeziehen. Auf diese Weise wird das grundlegende Ziel verfolgt, eine systematische Schwachstellenanalyse in Workflow Management Systemen durchzuführen.

Abstract

This work shows how a certain workflow management system namely *Activiti BPM* undergoes a vulnerability analysis with the help of the *Attack Tree* method. The underlying model is developed with the Activiti Eclipse designer plugin including manual as well as automated activities with connections to a MySQL database and an external web service. Several implementations written in Java are provided and represent a part of the automated tasks. The analysis embraces the Activiti BPM platform as well as the database, resources and services, the web server and all participating roles and groups including their access rights and constraints. The vulnerability analysis is conducted with manual as well as automated tests, includes the outside as well as inside perspective and the surrounding components in order to finally also cover their impact of attacks on the main system. In this way the aim to conduct a systematic analysis of vulnerabilities in workflow management systems can be pursued.