Jonas Hofer, Remo Egli

Akoben: Penetration Tester Search Engine

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

[Frühjahrssemester 2011]

Autor(en):	Jonas Hofer, Remo Egli
Betreuer:	Ivan Bütler
Projektpartner:	Compass Security AG
Experte:	Walter Sprenger
Gegenleser:	Hansjörg Huser

3. Juni 2011

1 Erklärung

Wir, Jonas Hofer und Remo Egli erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.

Rapperswil, 30 Mai. 2011

Jonas Hofer

Remo Egli

2 Abstract

Abteilung	Informatik
Name[n] der Studierenden	Jonas Hofer
	Remo Egli
Studienjahr	FS 2011
Titel der Studienarbeit	Penetration Tester Search Engine
Examinatorin / Examinator	Ivan Bütler
Themengebiet	Internet-Technologien und Sicherheit
Projektpartner	Compass Security AG, Jona
Institut	Diverses

Die „Penetration Tester Search Engine“ sammelt aktiv Meta-Informationen über Geräte, welche mit dem Internet verbunden sind und legt diese in durchsuchbarer Form in einer Datenbank ab. Der Prototyp besteht aus dem Web-Frontend, realisiert mit Ruby on Rails, einem Job-Service, ebenfalls in Ruby geschrieben, welcher die Datenbeschaffung (Scanning) steuert und die Resultate aufbereitet, sowie einer MySQL-Datenbank zur Ablage der Scanning Ergebnisse. Als Datenquelle dienen externe Scanning Tools wie Nmap, Netsparker oder Curl, welche vom Job-Service über einen zu diesem Zweck konzipierten Plugin Mechanismus angesteuert werden. Diese Plugin Architektur ermöglicht die Einbindung weiterer Scan Tools.

Der Prototyp dient im aktuellen Entwicklungsstadium zur Automatisierung von Teilprozessen bei Penetration Testing Aufgaben, welche von Compass Security AG angeboten werden.

Die Search Engine erlaubt das Einlesen von Scan Ergebnissen die von externen Scanning Tools, ohne die Hilfe des Job-Services, erstellt wurden, beispielsweise aus dem Intranet einer Firma.

Das Web-Frontend, wie auch der Job-Service, sind voneinander unabhängig und tauschen alle Informationen über die zentrale MySQL Datenbank aus.

Das Projekt umfasst neben dem implementierten Prototyp ein erweitertes Konzept für die Realisierung zusätzlicher Funktionen und Use Cases (z.B. Autorisierung der Benutzer), welches auf Stufe der Datenbank bereits vollständig implementiert ist.

3 Inhalt

Jonas Hofer, Remo Egli

4 Management Summary

4.1 Was ist Akoben?

Akoben ist eine Mischung zwischen einer Suchmaschine und einer Automatisierungs-Software.
Der Auftraggeber, die Compass Security AG, ist spezialisiert auf Schwachstellenanalyse und Forensische Untersuchungen von IT-Systemen.
Im Bereich der Schwachstellenanalyse gibt es Prozesse, welche für jeden Kundenauftrag durchlaufen werden, die das Zielsystem mit einer Reihe von speziellen Programmen überprüfen, sogenannten Vulnerability Scannern (engl. Für „Verwundbarkeitsprüfer“).
Akoben automatisiert diese Analyse, indem der Benutzer das gewünschte Ziel über die Programmoberfläche erfasst und danach mit einem/mehreren Vulnerability Scanner(n) automatisch überprüfen lassen kann.
Die Analyse der so erhaltenen Informationen wird von Akoben vereinfacht, indem die Resultate in einer Datenbank so gespeichert werden, dass ein Vergleich der Informationen möglich wird. Der Benutzer muss dadurch nicht mehr die Resultate aller Vulnerability Scanner einzeln auswerten, sondern erhält auf einen Blick alle gespeicherten Informationen über ein gewünschtes System.

Inspiriert wurde diese Software durch „SHODAN“ (www.shodanhq.com), einer Suchmaschine für Netzwerk-Geräte (Server, Router, etc.), welche spezifische (Meta-)Informationen, z.B. über Betriebssystem und Version des jeweiligen Geräts, in einer Suchmaschine zur Verfügung stellt. (Beispiel einer SHODAN-Abfrage: http://www.shodanhq.com/?q=hsr.ch)

4.2 Wie ist Akoben aufgebaut?

Der Prototyp besteht aus zwei Hauptteilen, einer Benutzeroberfläche, zugreifbar über jeden modernen Internetbrowser (z.B. Firefox, Internet Explorer, Chrome) und einem Programm, welches im Hintergrund die Aufträge bearbeitet. Der Programmteil, welcher für die Abarbeitung der Aufträge verantwortlich ist, greift auf die Vulnerability Scanner zu, welche auf dem Akoben-Server installiert sind. Mit dem flexiblen Plug-In-Mechanismus können alle Vulnerability Scanner, welche über eine Kommandozeilen-Schnittstelle verfügen angebunden werden, was Akoben zu einem mächtigen Werkzeug für die Schwachstellenanalyse macht.

4.3 Die Vorteile auf einen Blick

- Web-Oberfläche
  Der Benutzer kann die Software einfach und bequem über einen gängigen Internetbrowser bedienen. Der Status der einzelnen Aufträge kann jederzeit über die entsprechende Verwaltungsansicht überprüft werden.
  Der Administrator kann, mit wenigen Ausnahmen, sämtliche Einstellungen auf der Web-Oberfläche machen zu welchen unter anderem folgende gehören:
  - Benutzerverwaltung
  - Plug-In-Verwaltung
  - Rechteverwaltung
  - Informationstypen-Verwaltung
  - Etc.

- „Fire & Forget“
  Der Computer des Benutzers wird während der Ausführung der Vulnerability Scanner nicht blockiert.
- Durchsuchbare Resultate
  Alle Informationen und Resultate werden in einer Datenbank abgespeichert, welche auf Wunsch mit verschiedenen Parametern und Filtern durchsucht werden können.
- Erweiterbare Informationsstruktur
  Die Datenbank bietet die Möglichkeit, zusätzliche eigene Informationstypen zu erfassen, ohne dass die Software umprogrammiert werden muss.
- Anbindung unzähliger Vulnerability Scanner und Tools
  Durch den Einsatz eines Plug-In-Mechanismus kann die Software nach Belieben und mit wenig Aufwand um zusätzliche Vulnerability Scanner und Tools erweitert werden. Der Mechanismus erlaubt die Anbindung jedes Programms, welches über eine Kommandozeilen-Schnittstelle verfügt.
- Offline Analyse (Parsing)
  Viele Vulnerability Scanner bieten die Möglichkeit, ihre Resultate in einer Datei abzuspeichern. Sollte der Benutzer eine manuelle Analyse mit einem Vulnerability Scanner durchführen, so kann er danach das Resultat abspeichern und nachträglich in Akoben einspeisen. Akoben lässt das entsprechende Plug-In die Datei analysieren und lädt das Ergebnis in die Datenbank, wo der Benutzer dann bequem die Resultate vergleichen kann.
- Download der Original-Resultate
  Sollten die Informationen aus der Datenbank einmal nicht genügen oder werden aus rechtlichen Gründen die originalen Resultate der eingebundenen Tools benötigt, so können diese direkt über die Web-Oberfläche heruntergeladen werden.

4.4 Ausblick

Akoben befindet sich noch im Prototyp-Stadium, wobei bereits viele weitere Funktionen und Möglichkeiten vorbereitet sind:

- Bewertung der Resultate
  Die Resultate welche von den Vulnerability Scannern an Akoben übergeben werden können manchmal Fehler enthalten. Durch ein eingebautes Bewertungssystem kann der Benutzer sämtliche Resultate bewerten und kommentieren.
- Mehrere Server
  Da die Architektur von Akoben bereits aus mehreren Programmteilen besteht, wird es in einer weiteren Version möglich sein mehrere Server zur Abarbeitung der Aufträge installieren zu können.
- Verbesserte Programmoberfläche
  Die vorhandene Oberfläche unterstützt bereits eine Vielzahl von Funktionen. In einer nächsten Version wird die Oberfläche noch besser auf die Bedürfnisse des Benutzers und an den Arbeitsfluss einer Schwachstellenanalyse angepasst sein.
- Auswertung und Statistik
  Momentan beschränkt sich das System auf die Darstellung der Auftragsresultate. In einer

zukünftigen Version soll eine statistische Auswertung über viele Resultate möglich sein. Dies ermöglicht neue Einblicke in den Zustand und die Sicherheit moderner Netzwerke und der angeschlossenen Geräte.

- Erweiterte Plug-In-Unterstützung
  Während der Entwicklung des Prototyps wurde nur eine kleine Anzahl von Plug-Ins für aktuelle Vulnerability Scanner geschrieben. Die Unterstützung aktueller Scanner wird bei jeder veröffentlichen Version erweitert und so das Angebot laufend verbessert.

5 Ziel

Ziel unserer Arbeit war es einen Prototyp eines Dienstes zur Verfügung zu stellen, welcher die Mitarbeiter der Compass Security AG in Jona in ihrer täglichen Arbeit unterstützt.

Die momentane Situation ist so, dass die Pentester in ihrer täglichen Arbeit eine Vielzahl von Tools für ihre Arbeit benötigen. Ein Grossteil dieser Programme wird über die Commandline gestartet und generiert dann einen Output, beispielsweise in XML, welcher Informationen, wie offen Ports, Protokolle oder Verletzlichkeiten etc., über verschiedene Hosts beinhalten.
Einerseits sind diese Dateien für das menschliche Auge nicht angenehm zu lesen anderseits muss der Pentester alle benötigten Programme auf seinem Rechner installiert haben.

Dieser Prozess soll vereinfacht werden.
Unsere Applikation besteht aus fünf Teilbereichen: Das erfassen von Pentests, die automatische Verarbeitung der Pentests, das Archivieren von Rohdaten, eine benutzerfreundliche Darstellung der Resultate und die Möglichkeit über die Daten zu suchen.

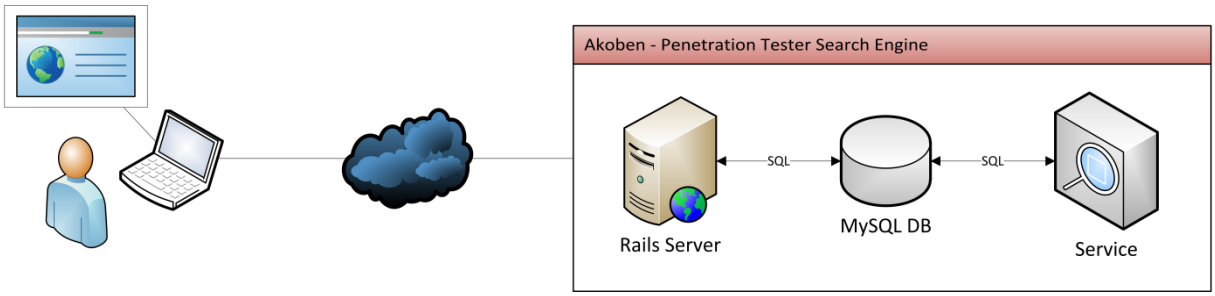


Abbildung 1: Konzept

6 Vorgaben & Einschränkungen

Die Aufgabenstellung wurde sehr offen definiert und wurde zusammen mit unserem Betreuer im Verlauf der Arbeit konkretisiert. Die einzige Einschränkung war, dass die Applikation auf einem Linux-System laufen muss und die Daten auf einem MySQL-Server abgelegt werden.

Ziel der Arbeit war, einen Prototypen zu entwickeln, welcher zeigt, dass unsere Architektur und das Design funktionieren und nicht eine vollumfängliche Applikation, welche produktiv eingesetzt werden kann.

7 Technologie

In diesem Kapitel werden die verwendeten Technologien und ihre Prinzipien kurz erläutert.

7.1 Ruby

7.1.1 Was ist Ruby

Ruby ist interpretiert und objektorientiert, unterstützt aber mehrere weitere Programmierparadigmen (unter anderem prozedurale und funktionale Programmierung sowie Nebenläufigkeit), bietet dynamische Typisierung, Reflexion und automatische Speicherbereinigung.

(Ruby (Programmiersprache), 2011)

7.1.2 Wieso Ruby

Der Job-Service der Applikation ist in Ruby geschrieben, da die Weboberfläche durch Ruby on Rails auch auf Ruby basiert. Der entscheidende Vorteil durch diesen Technologieentscheid besteht darin, dass die gemeinsam genutzten Komponenten(hauptsächlich die Entity-Klassen) in beiden Teilen verwendet werden können und so keine Redundanz entsteht.

7.2 Ruby on Rails

7.2.1 Was ist Ruby on Rails

Ruby on Rails, kurz Rails, früher auch oft kurz RoR, ist ein von David Heinemeier Hansson in der Programmiersprache Ruby geschriebenes und quelloffenes Web Application Framework. Es wurde im Juli 2004 zum ersten Mal der Öffentlichkeit vorgestellt.

Es basiert auf den Prinzipien „Don't Repeat Yourself" (DRY) und „Konvention vor Konfiguration", das heißt statt einer variablen Konfiguration sind Konventionen für die Namensgebung von Objekten einzuhalten, woraus deren Zusammenspiel sich automatisch ergibt. Diese Funktionen ermöglichen eine rasche Umsetzung von Anforderungen und damit agile Softwareentwicklung.

(Ruby On Rails, 2011)

7.2.2 Wieso Ruby on Rails

Ruby on Rails(RoR) ist ein sehr mächtiges Framework und durch das „Konvention vor Konfiguration"-Prinzip lässt sich damit sehr schnell entwickeln, da der Konfigurationsaufwand nur sehr gering ist und man sich auf das wesentliche konzentrieren kann.

Ein weiterer Vorteil an RoR ist, dass durch ActiveRecord eine komplette Abstraktion des Datenbanklayers ermöglicht wird, was bei einem sehr komplexen Datenbankmodel, wie in unserem viele Vorteile bezüglich dem Datenhandling bringt.

7.3 jQuery

7.3.1 Was ist jQuery

jQuery ist eine freie, umfangreiche JavaScript-Klassenbibliothek, welche komfortable Funktionen zur DOM-Manipulation und -Navigation zur Verfügung stellt. Die von John Resig entwickelte Klassenbibliothek wurde im Januar 2006 auf dem BarCamp(NYC) in New York veröffentlicht und wird laufend weiterentwickelt. Im September 2008 haben Microsoft und Nokia angekündigt, jQuery in ihren Produkten zu verwenden. Microsoft will jQuery in der Entwicklungsumgebung Visual Studio in Verbindung mit dem ASP.NET MVC Framework und ASP.NET Ajax verwenden und Nokia plant, es in seine Web-Runtime-Plattform zu integrieren. jQuery ist die meistverwendete JavaScript-Bibliothek.

(JQuery, 2011)

7.3.2 Wieso jQuery

Ruby on Rails kommt standardmässig mit dem JavaScript Framework „Prototype". Wir haben uns aber für jQuery entschieden, da die nächste Rails Version auch jQuery als JavaScript-Framework verwenden wird. Zudem findet man im Netz mehr Bespiele und die Dokumentation ist besser.

7.4 MySQL

7.4.1 Was ist MySQL

Der MySQL Server ist ein relationales Datenbankverwaltungssystem. Es ist als Open-Source-Software sowie als kommerzielle Enterpriseversion für verschiedene Betriebssysteme verfügbar und bildet die Grundlage für viele dynamische Webauftritte.

(Mysql, 2011)

7.4.2 Wieso MySQL

Ruby On Rails verwendet standardmässig eine SQL-Lite 3 Datenbank für die Persistenz.
Wir verwenden aber nach Vorgaben der Compass Security eine MySQL Datenbank.

8 Anforderungen

8.1 Nicht funktionale Anforderungen

8.1.1 Schnittstellen

8.1.1.1 Zielplattform

Als Zielplattform wurde Linux bereits in der Aufgabenstellung als Vorgabe beschrieben. Die Plattform benötigt ausserdem Ruby 1.9.2 und Rails 3.0.5 sowie die Installation der verwendeten Ruby-Gems.

Für die Ausführung der Plug-Ins wird ausserdem die jeweilige Software auf dem System benötigt (z.B. zur Verwendung des Nmap-Plugins ist eine Installation von Nmap zwingend).

8.1.1.2 Entwicklungsplattform

Basierend auf der Vorgabe der Zielplattform soll auch die Entwicklungsplattform alle genannten Kriterien erfüllen. Zur effizienten Entwicklung wird ausserdem eine gängige IDE (Integrated development environment) verwendet, welche die Technologie bestmöglich unterstützt. Im Fall von Ruby und Rails haben wir uns für Aptana Studio 3 entschieden.

8.1.2 Rails

8.1.2.1 Wartbarkeit

- Die Architektur der Applikation muss so aufgebaut sein, dass sie einfach um weitere Komponenten erweitert werden kann

8.1.2.2 Benutzbarkeit

- Das GUI muss den üblichen Web 2.0 Standards genügen.
- Das GUI sollte wenn möglich selbsterklärend sein.

8.1.3 Job-Service

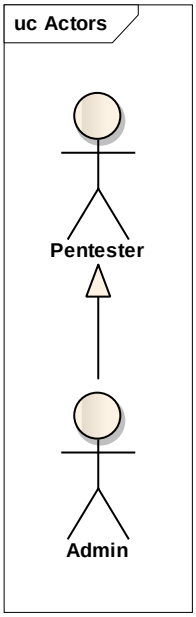
8.1.3.1 Wartbarkeit

- Da der Job-Service ohne das Zutun des Benutzers ausgeführt wird, muss es eine Möglichkeit geben, den Zustand der Software überwachen zu können.
- Diese Überwachung soll dem Benutzer die Möglichkeit geben zwischen verschiedenen Detailstufen der Zustandsinformationen wählen zu können.

8.1.3.2 Erweiterbarkeit

- Der Job-Service soll eine Schnittstelle bieten, über welche sich weitere Tools und Vulnerability Scanner einbinden lassen, ohne dass der Code des Job-Services angepasst werden muss.

8.2 Actors



Pentester

Ein Pentester ist ein im System erfasster User mit folgenden Möglichkeiten:

- Jobs erfassen und betrachten
- Upload-Job erfassen und betrachten
- Resultate von Jobs einsehen
- Hosts suchen
- Rohdaten downloaden.
- Die genauen Rechte eines Benutzers können durch unser rollenbasiertes Rechtesystem verwaltet werden.

Administrator

Der Administrator erbt vom Pentester und hat zusätzliche Rechte.

- User und Rechte verwalten
- Gruppen verwalten
- Kunden verwalten
- Plugins verwalten
- Infotypen verwalten

8.3 Use Cases

ID	1
Name	Login
Primary Actor	Pentester, Administrator
Secondary Actors	
Stakeholders	Kunden
Precondition	User ist im System erfasst und ausgeloggt und befindet sich auf der Login-Seite.
Postcondition	User befindet sich auf der Suchseite und ist eingeloggt.
Steps	1. User tippt sein Benutzername ein. 2. User tippt sein Passwort ein. 3. User bestätigt seine eingaben mit ENTER oder klickt auf den Login-Button.

ID	2
Name	Job erfassen
Primary Actor	Pentester
Secondary Actors	Administrator
Stakeholders	Kunden
Precondition	User ist im System erfasst und eingeloggt.
Postcondition	Job ist auf der Datenbank mit dem Status „todo“ gespeichert.
Steps	4. User ruft die Jobs-Übersicht auf. 5. User klickt auf „new Job“. 6. User wählt einen Kunden (optional). 7. User bestimmt ein Target. a. Eine bestimmte Anzahl IP-Ranges und/oder b. Eine bestimmte Anzahl Domainnames 8. User wählt Plugins aus, welche auf das definierte Target ausgeführt werden. a. User gibt optional Parameter an. 9. User bestätigt mit „Save Jobs“.

ID	3
Name	Job kopieren
Primary Actor	Pentester
Secondary Actors	Administrator
Stakeholders	Kunden
Precondition	User ist im System erfasst und eingeloggt.
Postcondition	Job ist auf der Datenbank mit dem Status „todo“ gespeichert.
Steps	1. User ruft die Job-Übersicht auf. 2. User klickt auf bei einem bestehenden Job auf das Copy Icon. a. Zeigt sich den Job an und klickt dann auf das Copy-Icon 3. Der User bekommt das Selbe Interface als würde einen neuen Job erfassen, mit bereits ausgefüllten Werten des zu kopierenden Jobs. 4. User passt den Job gegebenenfalls an und bestätigt mit „Save Jobs“.

ID	4
Name	Suchen
Primary Actor	Pentester
Secondary Actors	Administrator
Stakeholders	
Precondition	User ist im System erfasst und eingeloggt.
Postcondition	User sieht das Resultat seiner Suche.
Steps	1. User klickt auf „Search“ und kommt auf die Suchseite mit einer Suchleiste. 2. User tippt seine Suchkriterien ein und bestätigt mit ENTER oder klickt auf den Search-Button. a. Suche-Parameter können mittels „AND“ und/oder“ OR“ verknüpft werden. b. Mittels einem Scope-Parameter kann eine Suche spezifiziert werden: Scope suchen haben folgende Form: „scope:suche“. Beispiel: „ipv4:127“ sucht nach Hosts mit einer ipv4-Adresse die mit „127“ beginnt.

	Es gibt folgende Scopes: i. **ipv4** Suche nach einer ipv4 Adresse ii. **ipv6** Suche nach einer ipv6 Adresse iii. **banner** Suche nach einem Banner iv. **vul** Suche nach einer Vulnerability(Verletzlichkeit) v. **port** Suche nach einem Port vi. **prot** Suche nach einem Protokoll vii. **wasc** Suche nach einer WASC-Nummer viii. **owasp** Suche nach einer OWASP-Nummer ix. **cwe** Suche nach einer CWE-Nummer x. **capec** Suche nach einer CAPEC-Nummer 3. Die Host die den Kriterien entsprechen werden nach der IPv4 Adresse zusammengefasst und dargestellt.

ID	5
Name	User CRUD
Primary Actor	Administrator
Secondary Actors	
Stakeholders	
Precondition	User existiert noch nicht. Administrator ist eingeloggt.
Postcondition	User ist im System erfasst und kann sich einloggen.
Steps: Erfassen	1. Administrator klickt auf „Users“ 2. Administrator klickt auf „new User“ 3. Administrator tippt einen Namen eine 4. Administrator tippt ein Passwort ein 5. Administrator wählt die Rolle des Users 6. Administrator klickt auf speichern 7. User wird gespeichert angezeigt.

Jonas Hofer, Remo Egli

ID	6
Name	Userrole CRUD
Primary Actor	Administrator
Secondary Actors	
Stakeholders	
Precondition: Erfassen	Userrole existiert noch nicht. Administrator ist eingeloggt.
Precondition: Updaten, Löschen	Userrole existiert bereits. Administrator ist eingeloggt.
Postcondition: Erfassen	Userrole ist im System erfasst und kann in der Userverwaltung ausgewählt werden.
Postcondition: Bearbeiten	Userrole ist angepasst.
Postcondition: Löschen	Userrole ist gelöscht.
Steps: Erfassen	1. Administrator klickt auf „Userroles“ 2. Administrator klickt auf „new Userrole“ 3. Administrator tippt einen Namen eine 4. Administrator gibt optional eine Beschreibung an 5. Administrator wählt die Permissions aus 6. Administrator klickt auf speichern
Steps: Bearbeiten	1. Administrator klickt auf „Userroles“ 2. Administrator klickt auf „Edit“ 3. Administrator bearbeitet die Felder 4. User klickt auf speichern
Steps: Löschen	1. Administrator klickt auf „Userroles“ 2. Administrator klickt auf das Delete-Icon 3. Administrator bestätigt die Sicherheits-Meldung

Jonas Hofer, Remo Egli

ID	7
Name	Permission CRUD
Primary Actor	Administrator
Secondary Actors	
Stakeholders	
Precondition: Erfassen	Permission existiert noch nicht. Administrator ist eingeloggt.
Precondition: Updaten, Löschen	Permission existiert bereits. Administrator ist eingeloggt.
Postcondition: Erfassen	Permission ist im System erfasst und kann sich einloggen.
Postcondition: Bearbeiten	Permission ist angepasst
Postcondition: Löschen	Permission ist gelöscht.
Steps: Erfassen	1. Administrator klickt auf „Permissions“ 2. Administrator klickt auf „new Permission“ 3. Administrator tippt einen Namen eine 4. Administrator gibt optional eine Beschreibung ein. 5. Administrator wählt optional ein Plugin aus 6. Administrator klickt auf speichern.
Steps: Bearbeiten	1. Administrator klickt auf „Permissions“ 2. Administrator klickt auf „Edit“ 3. Administrator bearbeitet die Felder 4. User klickt auf speichern
Steps: Löschen	1. Administrator klickt auf „Permissions“ 2. Administrator klickt auf das Delete-Icon 3. Administrator bestätigt die Sicherheits-Meldung

Jonas Hofer, Remo Egli

ID	8
Name	Plugin CRUD
Primary Actor	Administrator
Secondary Actors	
Stakeholders	
Precondition: Erfassen	Plugin existiert noch nicht. Administrator ist eingeloggt.
Precondition: Updaten, Löschen	Plugin existiert bereits. Administrator ist eingeloggt.
Postcondition: Erfassen	Plugin ist im System erfasst und kann in den Permissions ausgewählt werden.
Postcondition: Bearbeiten	Plugin ist angepasst
Postcondition: Löschen	Plugin ist gelöscht.
Steps: Erfassen	1. Administrator klickt auf „Plugins“ 2. Administrator klickt auf „new Plugin“ 3. Administrator tippt einen Namen eine 4. Administrator gibt optional die Defaultparameter an. 5. Administrator klickt auf speichern.
Steps: Bearbeiten	1. Administrator klickt auf „Plugins“ 2. Administrator klickt auf „Edit“ 3. Administrator bearbeitet die Felder 4. User klickt auf speichern
Steps: Löschen	1. Administrator klickt auf „Plugins“ 2. Administrator klickt auf das Delete-Icon 3. Administrator bestätigt die Sicherheits-Meldung

Jonas Hofer, Remo Egli

ID	9
Name	Customers CRUD
Primary Actor	Administrator
Secondary Actors	
Stakeholders	
Precondition: Erfassen	Customer existiert noch nicht. Administrator ist eingeloggt.
Precondition: Updaten, Löschen	Customer existiert bereits. Administrator ist eingeloggt.
Postcondition: Erfassen	Customer ist im System erfasst und kann beim Job erfassen ausgewählt werden.
Postcondition: Bearbeiten	Customer ist angepasst.
Postcondition: Löschen	Customer ist gelöscht.
Steps: Erfassen	1. Administrator klickt auf „Customers“ 2. Administrator klickt auf „new Customer“ 3. Administrator tippt einen Namen eine 4. Administrator gibt optional eine Beschreibung an 5. Administrator klickt auf speichern
Steps: Bearbeiten	1. Administrator klickt auf „Customers“ 2. Administrator klickt auf „Edit“ 3. Administrator bearbeitet die Felder 4. User klickt auf speichern
Steps: Löschen	1. Administrator klickt auf „Customers“ 2. Administrator klickt auf das Delete-Icon 3. Administrator bestätigt die Sicherheits-Meldung

Jonas Hofer, Remo Egli

ID	10
Name	Groups CRUD
Primary Actor	Administrator
Secondary Actors	
Stakeholders	
Precondition: Erfassen	Group existiert noch nicht. Administrator ist eingeloggt.
Precondition: Updaten, Löschen	Group existiert bereits. Administrator ist eingeloggt.
Postcondition: Erfassen	Group ist im System erfasst und kann beim erstellen eins Users ausgewählt werden.
Postcondition: Bearbeiten	Group ist angepasst.
Postcondition: Löschen	Group ist gelöscht.
Steps: Erfassen	1. Administrator klickt auf „Groups“ 2. Administrator klickt auf „new Group“ 3. Administrator tippt einen Namen eine 4. Administrator gibt optional eine Beschreibung an 5. Administrator klickt auf speichern
Steps: Bearbeiten	1. Administrator klickt auf „Group“ 2. Administrator klickt auf „Edit“ 3. Administrator bearbeitet die Felder 4. User klickt auf speichern
Steps: Löschen	1. Administrator klickt auf „Groups“ 2. Administrator klickt auf das Delete-Icon 3. Administrator bestätigt die Sicherheits-Meldung

Jonas Hofer, Remo Egli

ID	11
Name	Infotypes CRUD
Primary Actor	Administrator
Secondary Actors	
Stakeholders	
Precondition: Erfassen	Infotype existiert noch nicht. Administrator ist eingeloggt.
Precondition: Updaten, Löschen	Infotype existiert bereits. Administrator ist eingeloggt.
Postcondition: Erfassen	Infotype ist im System erfasst und kann beim parsen verwendet werden.
Postcondition: Bearbeiten	Infotype ist angepasst.
Postcondition: Löschen	Infotype ist gelöscht.
Steps: Erfassen	1. Administrator klickt auf „Infotypes“ 2. Administrator klickt auf „new Infotype“ 3. Administrator tippt einen Namen eine 4. Administrator gibt optional eine Beschreibung an 5. Administrator klickt auf speichern
Steps: Bearbeiten	1. Administrator klickt auf „Infotypes“ 2. Administrator klickt auf „Edit“ 3. Administrator bearbeitet die Felder 4. User klickt auf speichern
Steps: Löschen	1. Administrator klickt auf „Infotypes“ 2. Administrator klickt auf das Delete-Icon 3. Administrator bestätigt die Sicherheits-Meldung

9 Grob-Architektur

9.1 Komponenten

Browser	Ein beliebiger Webbrowser, welcher JavaScript und CSS3 unterstützt.
Ruby on Rails	Ruby on Rails basierte Webapplikation
MySQL Database	MySQL Datenbank
Job-Service	Backgroundservice, welcher die Jobs verarbeitet
Pentest Tool	Ein beliebiges Pentest-Tool, welches über die Commandline ausgeführt werden kann (z.B. Nmap)
Linux	Betriebssystem

9.2 Workflow

In diesem Kapitel wird ein kurzer Überblick über den Workflow der Applikation gegeben und das Zusammenspiele der verschiedenen Komponenten.

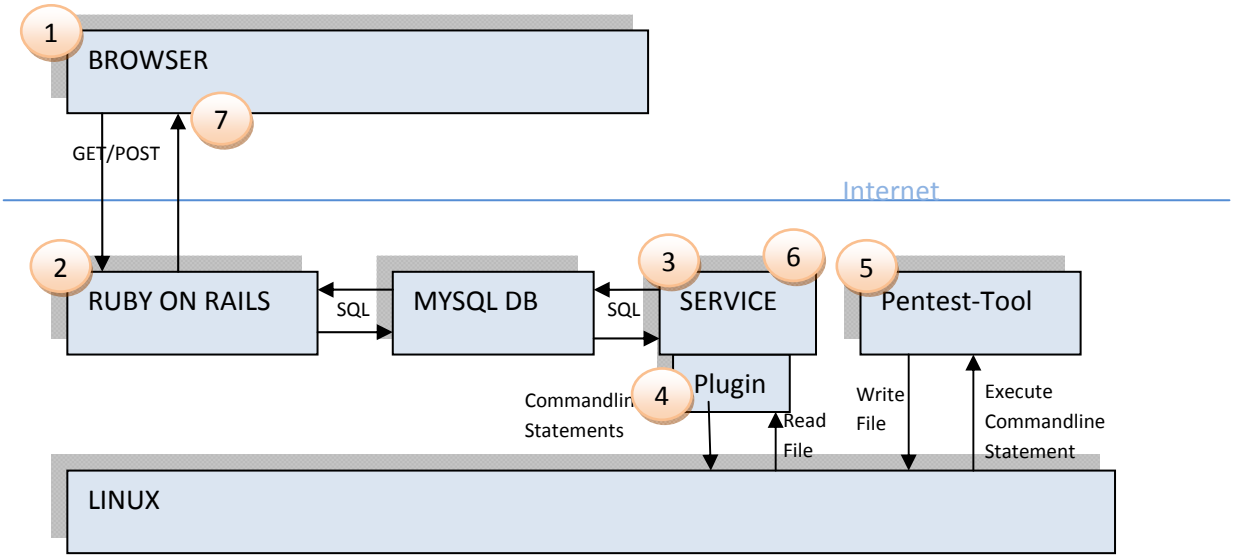


Abbildung 2: Workflow

1. Ein User ruft mit seinem Webbrowser die Ruby on Rails(RoR) Website auf und erfasst einen neuen Job mit einer beliebigen Anzahl Pluginexecutions.
2. Die RoR Applikation speichert den Job in der MySQL Datenbank ab mit dem Status „todo“ (mehr zu den verschiedenen States im Kapitel: „Bedeutung der Zustände“)
3. Der Job-Service im Hintergrund macht in regelmässigen Abständen Abfragen auf die Datenbank und prüft, ob sich neue Jobs in der Datenbank befinden. Falls ja, werden diese ausgeführt.
   Dazu werden die einzelnen Pluginexecutions mit dem passenden Plugin verarbeitet.
4. Aus den Informationen, welche der User beim erfassen des Jobs übergeben hat, werden vom Plugin, welches auf ein bestimmtes Pentest-Tool zugeschnitten ist, passende Commandline

Statements generiert. Diese werden dann vom Job-Service dem Betriebssystem übergeben, welches wiederum das Pentest-Tool zur Ausführung bringt.

5. Das Pentest-Tool wird ausgeführt und schreibt seinen Output in eine Datei, welche auf dem Dateisystem des Betriebssystems geordnet abgelegt wird.
6. Nach Erfolgreichem Ausführen des Pentest-Tools wird die Datei ausgelesen und verarbeitet. Die Resultate werden dann zurück in die Datenbank geschrieben.
7. Der User kann danach über die Webapplikation das Resultat seines Aufgegebenen Jobs einsehen.

9.3 Zustände der Jobs und Pluginexecutions

Jobs und Pluginexecutions durchlaufen jeweils 4 Zustände im Normalfall. Der Zustand wird im Statusfeld der Pluginexecution oder des Jobs gespeichert. Er ist einerseits dazu da, damit der User auf dem GUI erkennen kann, was im Hintergrund geschieht und andererseits kann bei einem Neustart des Job-Services am entsprechend Punkt wieder eingestiegen werden.

9.3.1 Bedeutung der Zustände

State Diagramm Pluginexecution

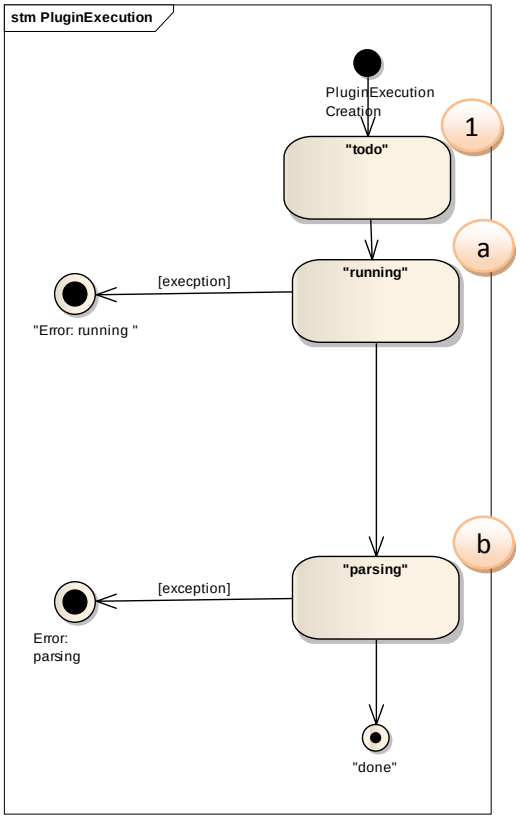


Abbildung 3: State Diagramm Pluginexecution

State Diagramm Job

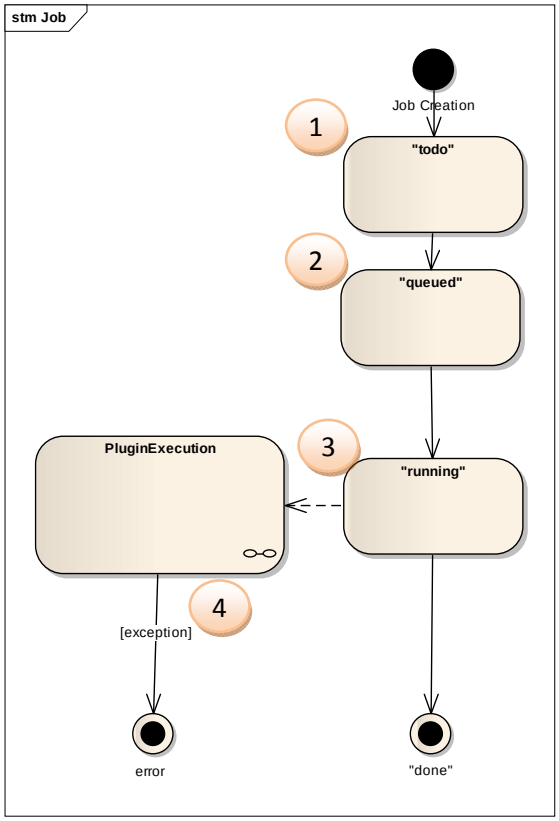


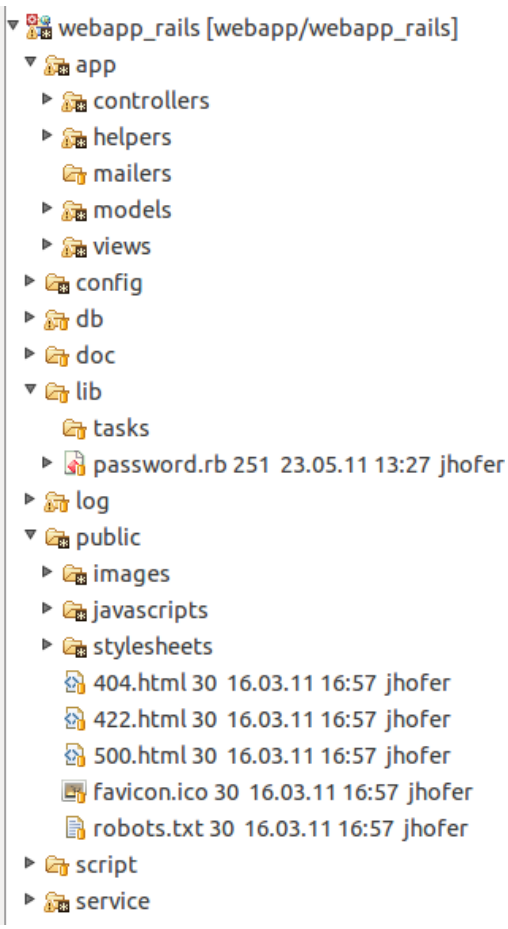
Abbildung 4: State Diagramm Job

1. Nach dem Erstellen eines Jobs hat der Job, so wie alle seine Pluginexecutions, den Status „todo“.
2. Der Job wird vom JobAggregator aus der Datenbank ausgelesen und in einer Liste abgelegt. Job erhält den Status „queued“, dies ist wichtig, damit der JobAggregator diesen Job nicht erneut ausliest.
3. Nun fordert der Job-Service einen neuen Job an, um ihn auszuführen. Dabei geht der Status über in „running“.
4. Darauf wird jede Pluginexecution des Jobs einem WorkerThread zugewiesen. Die Pluginexecution durchläuft, dann ein ähnliches Prozedere.
    a. wenn der Worker die Pluginexecution erhält ist der Status auf „todo“ und wird dann auf „running“ gesetzt und danach die Pluginexecution ausgeführt.

        i. falls beim Ausführen etwas schief läuft, wird der Status auf „Error: running“ gesetzt.
    b. Nach erfolgreichem Ausführen der Pluginexecution geht der Status über in „parsing“ und das Resultat wird geparst.
        i. Falls beim Parsen ein Fehler auftritt, wird der Status auf „Error: parsing“ gesetzt.
    c. Nach erfolgreichem Parsen wird der Status auf „done“ gesetzt.
5. Nach dem alle Pluginexecutions auf „done“ gesetzt wurden, wird abschliessend auch auf „done“ gesetzt,
6. Falls eine Pluginexecution fehlgeschlagen hat, dann wird auch der Job auf „error“ gesetzt.

9.4 Ruby on Rails

9.4.1 Struktur



models	Im Ordner „models“ befinden sich die Modelklassen der Entities.
controllers	Hier befinden sich die Controller der einzelnen Models.
views	Hierbefinden sich die erb.html Seiten die für die Darstellung zuständig sind
helpers	Hier wird komplexere Logik der Views ausgelagert
lib	Methoden die von den Models verwendet werden können.
db	Im Ordner „db“ sind Dateien, die die Datenbank betreffen. Auch die Migrationfiles werden darin abgelegt.

Abbildung 5: Dateistruktur

9.4.1.1 Models

In den Models werden die Kardinalitäten zwischen den Models definiert:
Beispiel:

```
class Host < ActiveRecord::Base
        belongs_to :pluginexecution
        belongs_to :provider
        belongs_to :geolocation
        has_many :hostinfos
        has_many :ports
         has_many :ratinghosts
         has_and_belongs_to_many :domainnames
end
```

belongs_to: entspricht n: 1
has_many: entspricht 1: n
has_and_belongs_to_many: entspricht n: m

Attribute, die in auf der Datenbank vorhanden sind, müssen in den Models nicht definiert werden, sondern sind implizit verfügbar.

9.4.1.2 Controller

In dem Controller werden die RESTful-Requests verarbeitet. Für jedes Model gibt es ein Controller. Die Standardmethoden sind:

- Index
- show
- new
- edit
- create
- update
- destroy

Diese werden standardmässig durch das Framework abgedeckt. Weitere Methoden müssen in der Routingconfig explizit angegeben werden.

9.4.1.3 Views

Passend zu den Methoden werden die entsprechenden Seiten eines Controllers benannt:

- index.html.erb
- new.html.erb
- show.html.erb
- edit.html.erb

9.4.2 Routing

Das RESTful-Routing wird über eine Konfigurationsdatei(.config/routes.rb) konfiguriert.
Die Standardmethoden werden mit der Zeile: resources:modelname (z.B.: resources:userroles) abgedeckt. Selbst definierte Methoden müssen selbst definiert werden. (z.B.: get "jobs/upload")
Wichtig: Eigendefinierte Methoden müssen **vor** den anderen Routen definiert werden.

9.4.3 Suche

Die Suche ist im Hostcontroller implementiert.

9.4.3.1 Algorithmus

Für die Suche wurde folgender Algorithmus implementiert

1. Die Suche wird bei den „OR“s aufgeteilt
2. Jedes OR wird bei den „AND“s aufgeteilt
3. Falls die „AND“-Teile über einen Doppelpunkt verfügen wird die linke Seite als Scope verwendet und die rechte Seite als Suchparameter
4. Nun wird die Schnittmenge der IDs der Hostobjekte genommen für die abfragen die mit AND verknüpft sind.
5. Diese aus diesen Schnittmengen wird wiederrum die Vereinigungsmenge genommen.

Achtung: AND bindet stärker als OR!

Beispielablauf zur Verdeutlichung:

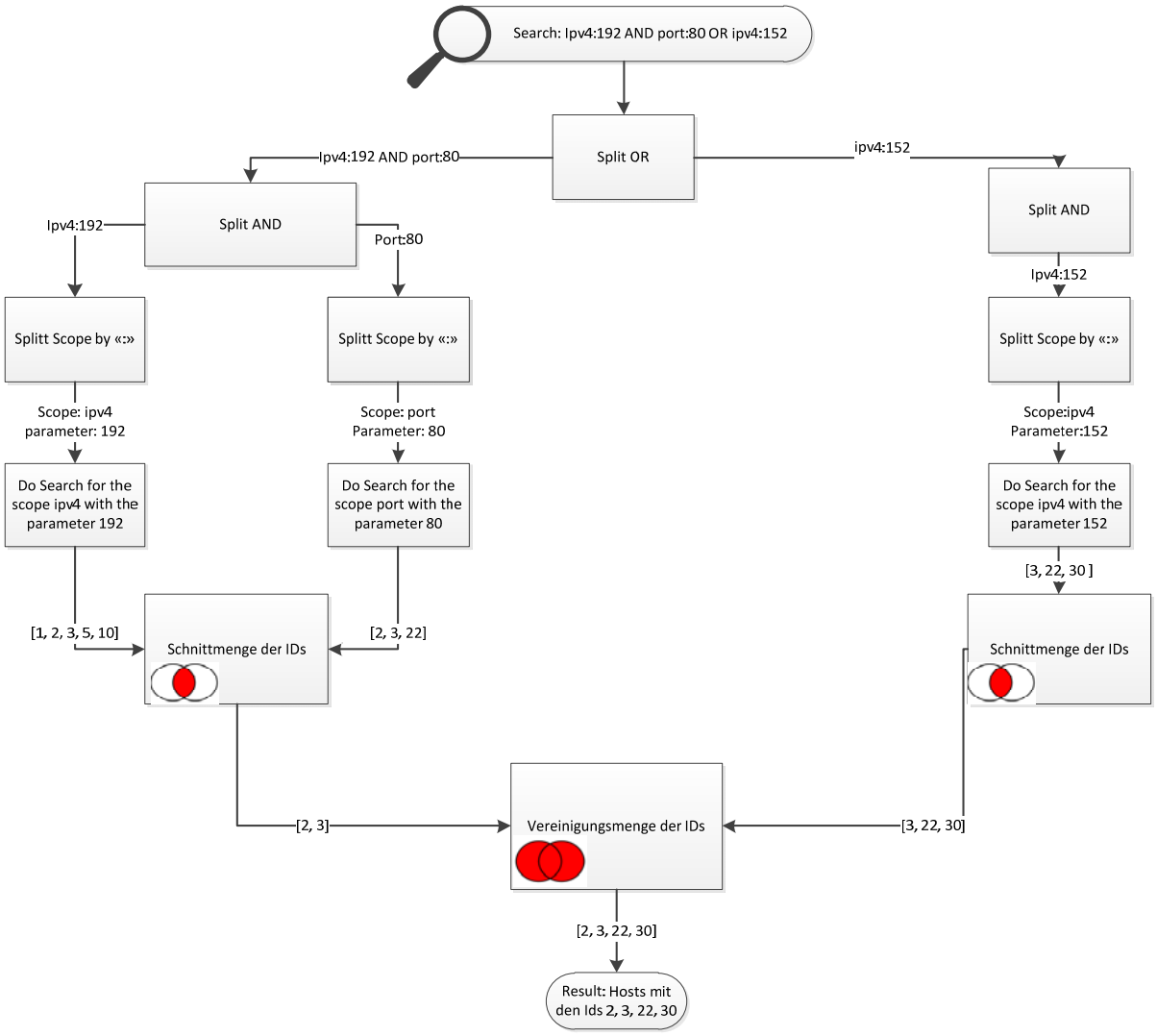


Abbildung 6: Suchalgorithmus

9.4.4 Userrechte

Vor jeder Methodenausführung der Controller wird geprüft, ob der aufrufende User eingeloggt ist. Umgesetzt wird dies mittels einem before Filter, welcher in der Superklasse (application_controller.rb) aller Controller, ausgenommen dem login_controller, definiert ist:

```
class ApplicationController < ActionController::Base
 protect_from_forgery

 before_filter :audit

 def audit
  user = User.where(:id => session[:user_id]).first
  if !(user)
    redirect_to :controller => 'login', :action=> 'index'
  else
    if !(user.userrole)
    redirect_to :controller => 'login', :action=> 'index'
    end
  end
 end
end
```

Achtung: Dabei wird noch nicht geprüft, ob ein User ein Admin ist. D.h. Sofern ein User den Link kennt, ist es ihm möglich die alle zu bearbeiten und neue User

9.4.5 Weiterführende Links

Für weitere und genauere Informationen zum Framework empfehlen wir die Guides der offiziellen Webseite(http://guides.rubyonrails.org/) und die Screencasts der Railscasts Website (http://railscasts.com/)

9.5 Job-Service

9.5.1 Struktur

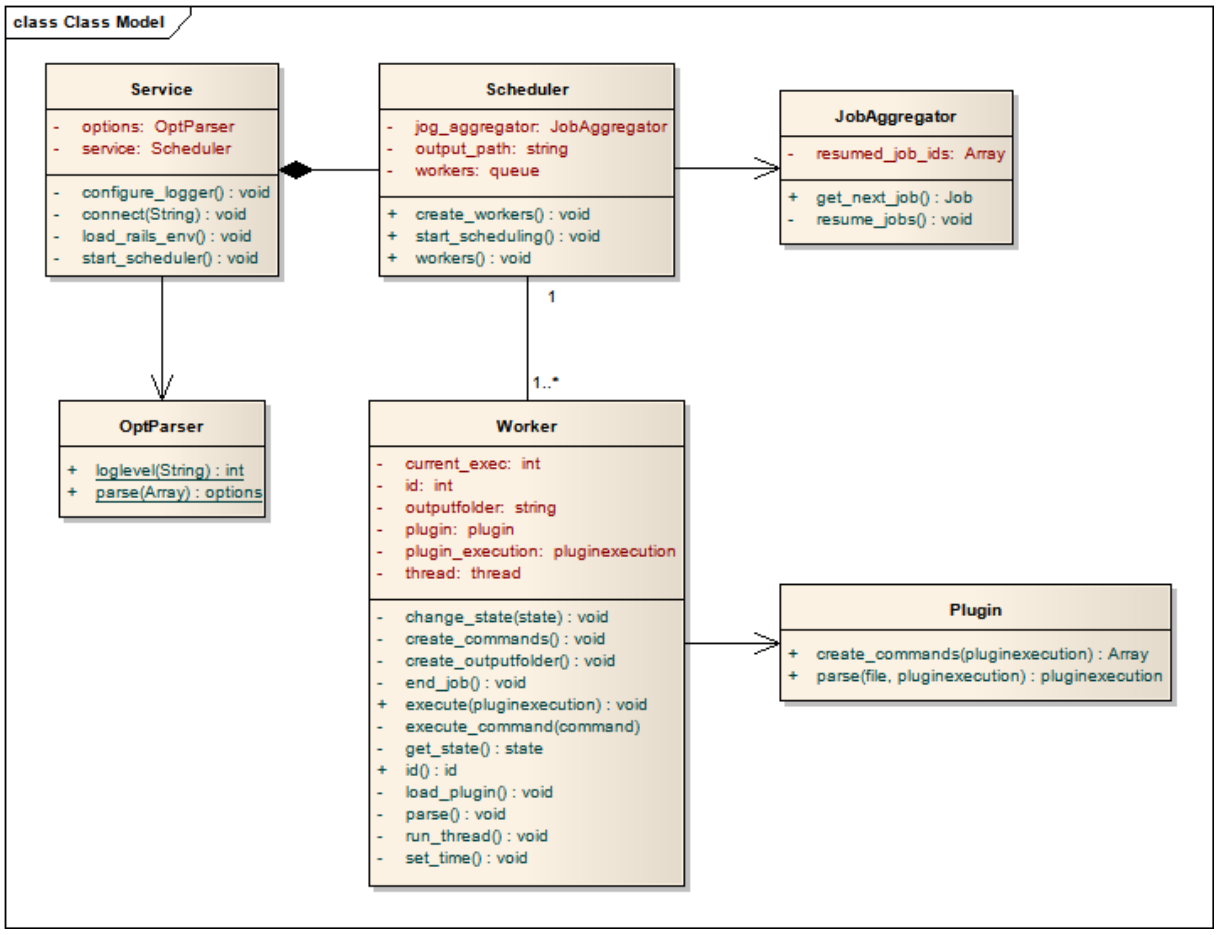


Abbildung 7: Job-Service Klassendiagramm

9.5.2 Klassenbeschreibung

9.5.2.1 Job-Service

Die Service Klasse ist verantwortlich für folgende Funktionen:

- Konfiguration des Loggers
- Laden des korrekten Rails Environments
- Verbindung mit der Datenbank herstellen
- Scheduler Klasse initialisieren und starten

Die Rails und die Job-Service Applikation laden für den Datenbank-Zugriff die gleichen Klassen und Module, wodurch für die Service Applikation kein eigener Data-Access-Layer entwickelt werden musste. Dies bedingt jedoch, dass der benötigte Code geladen werden muss, bevor der Scheduler mit seiner Arbeit beginnen kann.

9.5.2.2 Scheduler

Das Architekturkonzept mit Scheduler und Worker ist angelehnt an das bekannte Master-Slave-Pattern, musste jedoch aufgrund der Technologieunterschiede etwas angepasst werden. Deshalb fungiert der Scheduler als eine Art Master, welcher verantwortlich ist für die Verteilung der Jobs. Diese werden vom Scheduler beim JobAggregator abgeholt, die einzelnen Pluginexecutions ausgepackt und an verfügbare Worker zur Bearbeitung weitergeleitet.

9.5.2.3 JobAggregator

Der JobAggregator dient für den Scheduler als Schnittstelle zur Datenbank. Neben der Übermittlung der zu bearbeitenden Jobs dient der JobAggregator auch dazu, nach einem etwaigen Absturz des Systems nicht fertiggestellte Aufträge wieder an den Scheduler zu übergeben. Dies stellt sicher, dass jeder Job sauber abgeschlossen wird.

9.5.2.4 Worker

Diese Klasse dient als Schnittstelle zwischen dem Job-Service und dem Betriebssystem und ist für folgende Funktionen:

- Laden der benötigten Plugins und entsprechende Interaktion
- Generierung der Ordnerstruktur für die Output-Files der Pluginexecutions
- Durchführung der Systemaufrufe mit den vom Plugin erhaltenen Parametern
- Durchführung des Parsing mit dem entsprechenden Plugin
- Verwaltung der Pluginexecution Metainformationen wie Timestamps und Zustände
- Abschliessen des Jobs, sobald alle dazugehörigen Pluginexecutions ebenfalls abgeschlossen sind

9.5.2.5 Weitere Informationen

Eine detaillierte Klassenbeschreibung wurde mit Rdoc erstellt und findet sich in folgendem Quellcode-Verzeichnis:
[akoben root]/service/doc/index.html

9.5.3 Sequenzdiagramme

9.5.3.1 Initialisierung

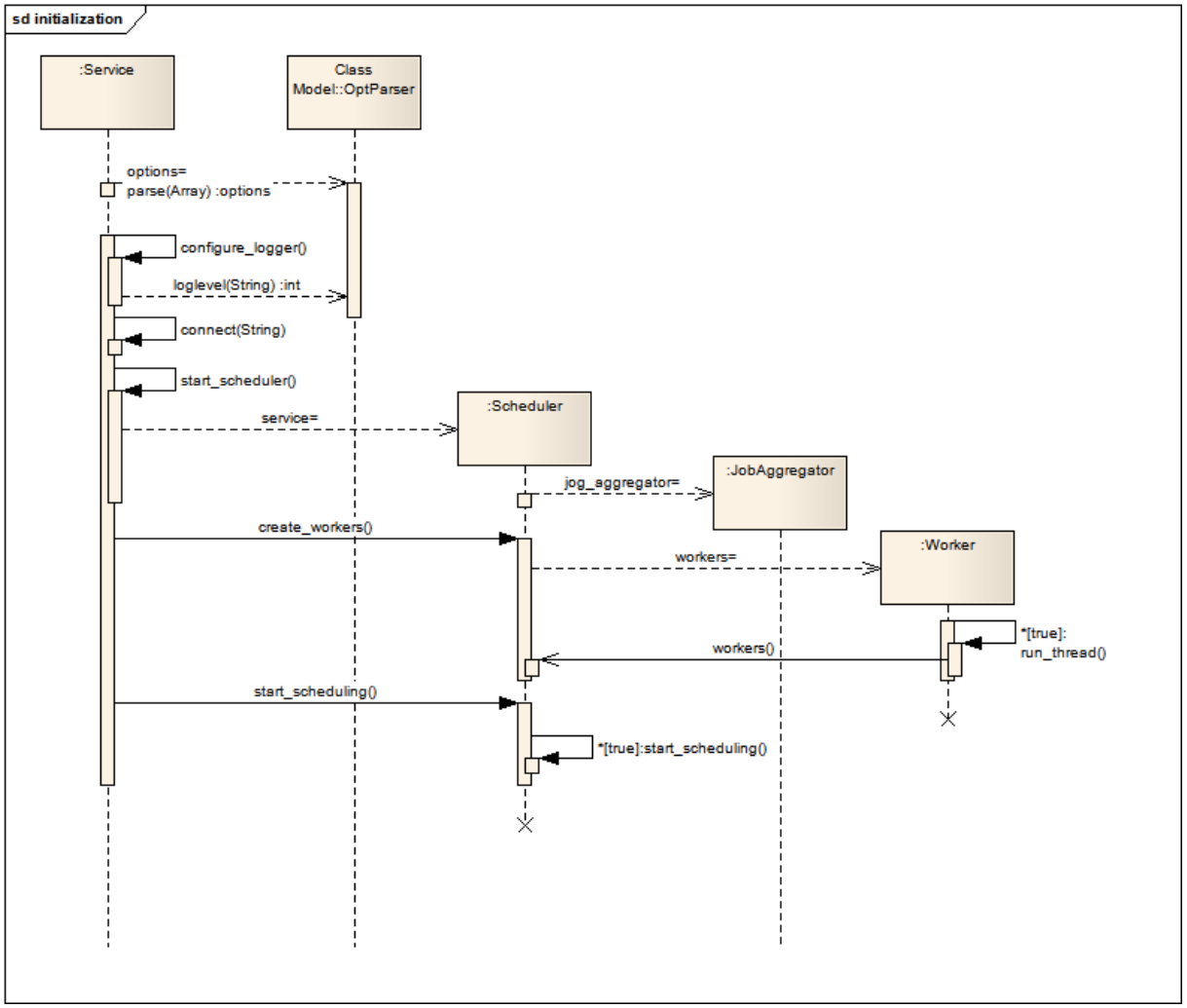


Abbildung 8: Sequenzdiagramm für die Initialisierung des Job-Services

9.5.3.2 Ausführung

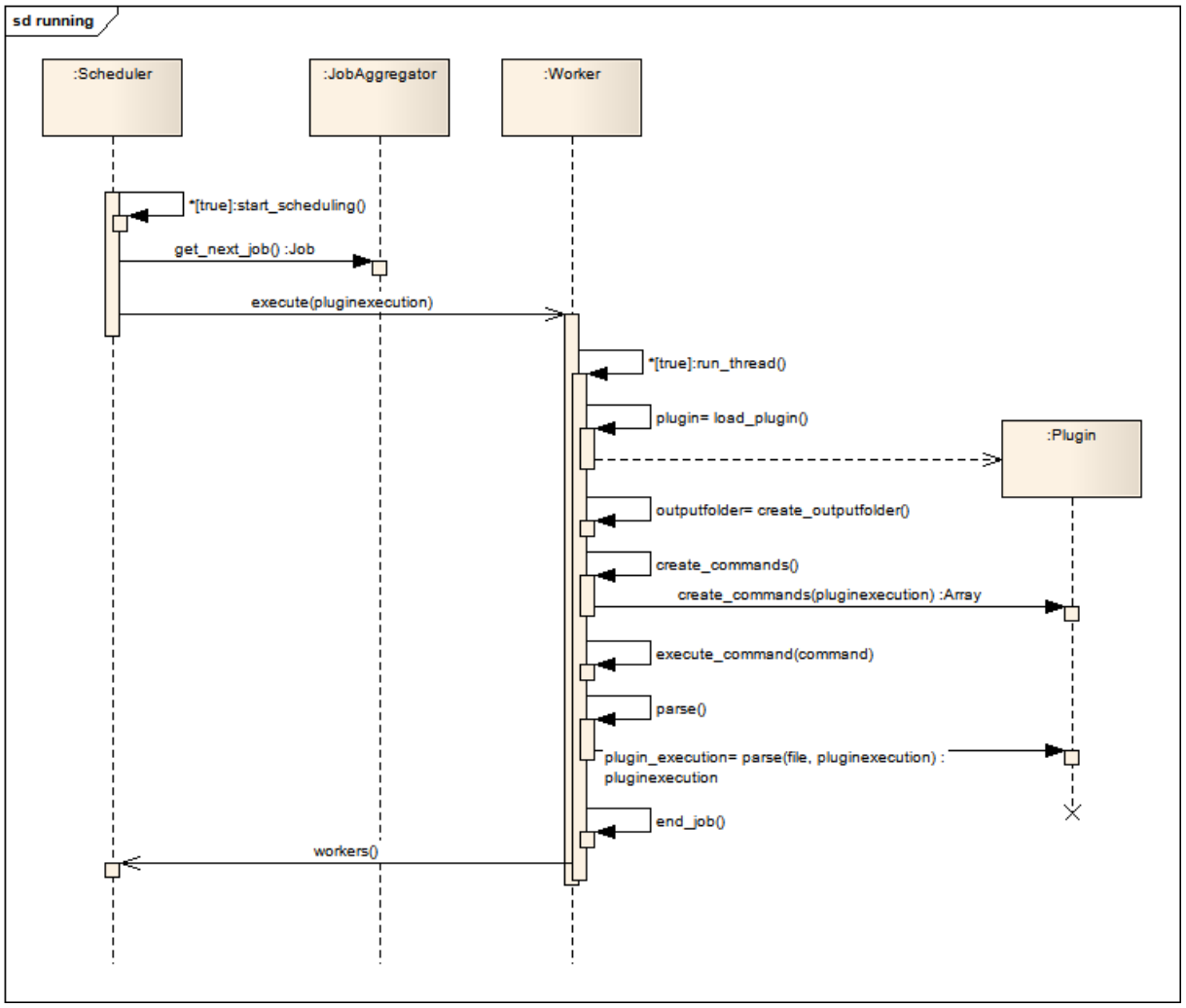


Abbildung 9: Sequenzdiagramm für die Ausführung des Job-Services

9.5.4 Plugins

9.5.4.1 Plugin-Mechanismus

Ein wichtiger Punkt ist die Modularität des Job-Services, dazu wurde ein Plugin-Mechanismus entwickelt, welcher es ermöglicht, einfach weitere Plugins für Tools zu schreiben, welche über die Applikation gestartet werden sollen.

Um ein weiteres Plugin der Applikation hinzuzufügen, muss es einerseits mit dem Namen in der Datenbank erfasst werden und andererseits muss eine Rubydatei (.rb) mit demselben Namen im Plugin-Ordner der Applikation abgelegt werden.

der Job-Service lädt sich dann über den Namen in der Datenbank das passende Plugin und führt dessen Methoden aus. Wobei der Dateiname kleingeschrieben sein muss und der Klassenname mit einem Grossbuchstaben beginnt und der Rest klein ist. (Bsp.: die Datei "nmap.rb" mit der Klasse "Nmap")

9.5.4.2 Laden des Plugins:

```
def load_plugin()
        ...
                require "./plugins/" + @plugin_execution.plugin.name.downcase + ".rb"
                plugin_name = @plugin_execution.plugin.name.titleize
                @plugin = Module.const_get(plugin_name).new
        ...
end
```

In der Pluginklasse müssen zwei Methoden Implementiert sein.

9.5.4.3 Plugininterface

In Ruby gibt es keine Interfaces im eigentlichen Sinne. Es wird einfach per Konvention definiert, welche Methoden die Plugins enthalten müssen:

Create Commands

Methodensignatur	Parameter	Rückgabewert
`def create_commands(pluginexecution)`	`pluginexecution` eine Instanz einer Pluginexecution die ausgeführt werden soll	Ein Array von Commandline Commands die vom Worker ausgeführt werden soll

Das Array mit den Commands muss Commands enthalten, die vom entsprechenden Tool ausgeführt werden und welche den Output in den Richtig Ordner ablegen. Der Pfad zu diesem Ordner lässt sich aus der ID des Jobs und der Pluginexecution ableiten: .....Output/Job_id/pluginexecution_id

In einem Job kann eine Grosszahl von Targets(IP-Bereiche, Domainnames) bestimmt werden. Einige Programme können all diese Targets nicht mit einem einzigen Command bewältigen, deswegen werden die Pluginexecutions in mehrere Commands auf gesplittet und als Array zurückgegeben.

Parse

Methodensignatur	Parameter	Rückgabewert

`def parse(pluginexecution, file)`	`pluginexecution`
	eine Instanz einer Pluginexecution an welche die Resulate angehängt werden
	
	`file`
	
	Die Datei, welche geparst werden soll
		Die Pluginexecution mit den Resultaten.

In der Parse-Methode muss die mit dem Parameter übergebene Datei in die von der Datenbank vorgegebene Form geparst werden. Vorzugsweise wird ein Streamparser(Bsp.: libxml) verwendet, damit auch grosse Dateien ohne Probleme geparst werden können.

10 Datenbank

10.1 Migration

Die Tabellen werden in Ruby on Rails in sogenannten „Migrationfiles“ definiert, die mit einem Zeitstempel beginnen. Auf der Datenbank hat es eine Tabelle „schema_migration“, in welcher die Zeitstempel eigetragen werden, wenn eine Migration durchgeführt wurde. Auf diese Weise kann jederzeit zu einem älteren Stand des Datenbankschemas zurückgekehrt werden.

Migriert wird über die Commandline mit dem Befehl: **rake db:migrate**

Beispiel: 20110316133735_create_jobs.rb

```
class CreateJobs < ActiveRecord::Migration
 def self.up
  create_table :jobs do |t|
   t.string :status
   t.datetime :timestampstart
   t.datetime :timestampend
   t.references :user
   t.references :customer
   t.string :linkhash
   t.timestamps
  end
 end

 def self.down
  drop_table :jobs
 end
end
```

10.2 Seeds

In der Seeddatei(./db/seed.rb) hat man die Möglichkeit einen Grundsatz von Daten für Datenbank zu definieren. Ausgeführt werden die Seeds via Commandline mit dem Befehl: **rake db:seed**

10.3 Domain Entities

Hier werden die verschiedenen Domain Klassen erläutert und ihren Zweck erklärt. Ein wichtiger Punkt der Anforderungen war, dass auch zukünftige Funktionalität bereits durch die Datenbank abgedeckt wird auch wenn sie in dieser Arbeit noch nicht implementiert werden. Wenn in diesem Kapitel über Funktionalität gesprochen wird, welche noch nicht oder nicht vollständig implementiert wurde wird Sie mit einem *Stern markiert.

Jonas Hofer, Remo Egli

10.3.1 Datenbankschema

Hier einen ersten Überblick über die gesamte Datenbank.

In den folgenden Kapiteln wird der Zeck der einzelnen Tabellen / Entities erklärt.

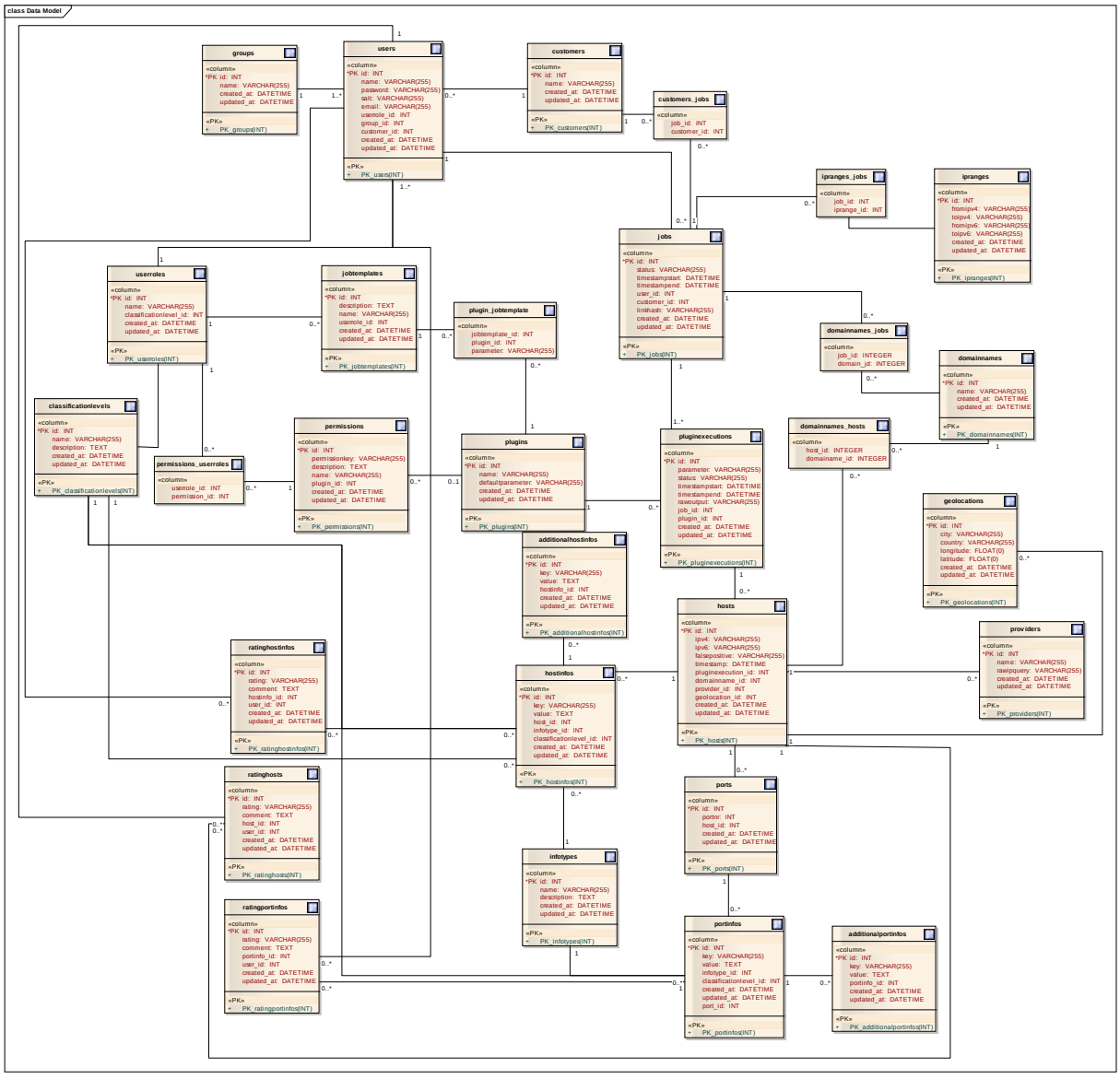


Abbildung 10: Datenbankschema

10.3.2 Userverwaltung

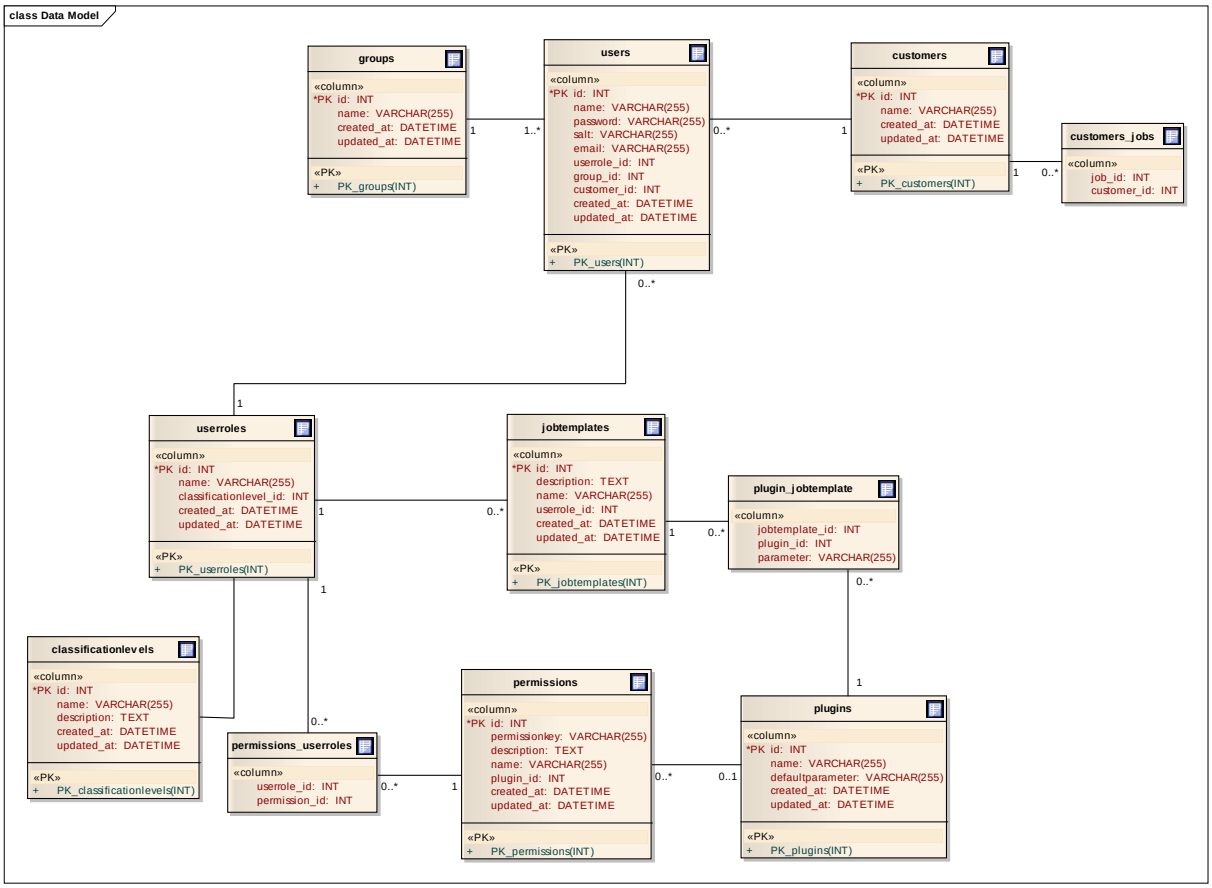
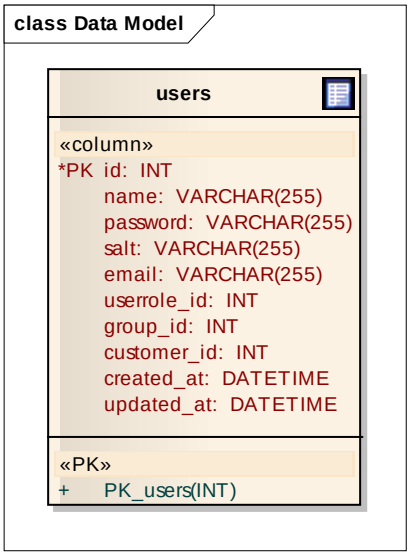


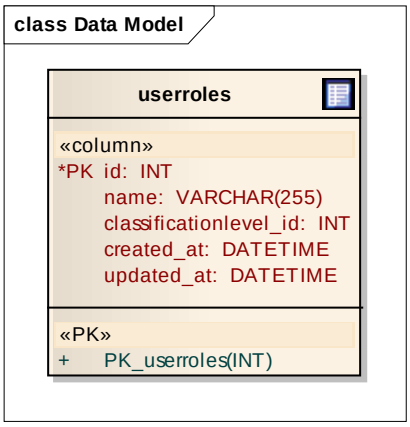
Abbildung 11: Userverwaltung

Ein Teil des Datenbankschemas betrifft die Userverwaltung und die Rechtevergabe. Es wird ein rollenbasiertes Rechtevergabesystem unterstützt.
*Zusätzlich kann ein User jeweils einer Gruppe angehören, damit verschieden User mit verschiedenen Rollen gemeinsame Daten einsehen können.
* Ein User kann einem Kunden zugewiesen werden. Dadurch ist es ermöglich, dass für einen Kunden einen Account erststellt werden kann und dieser dann die Möglichkeit hat über Pentest-Ergebnisse zu suchen, welche für Ihn durchgeführt wurden.
*Es können JobTemplates erstellt werden, um immer wiederkehrende Pentests vorzudefinieren. Diese Templates können dann Rollen zugeordnet werden, die es benützen dürfen.



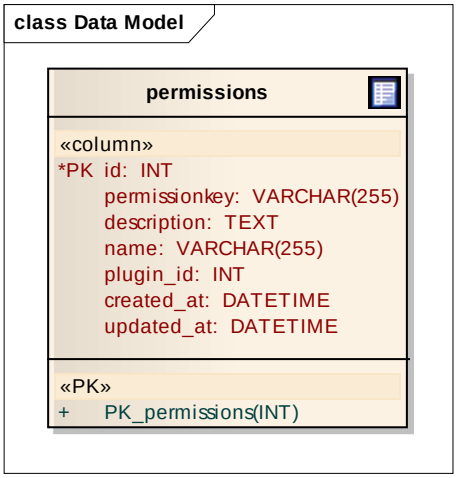
Benutzer werden im System als User erfasst mit den üblichen Angaben. Das Passwort wird mittels SHA512 inklusivem Salz verschlüsselt und abgelegt.

Abbildung 12: users



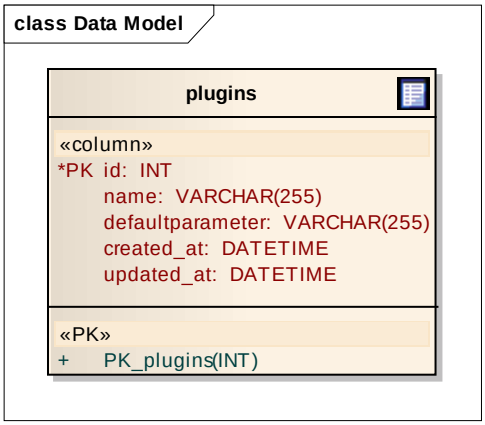
Das System unterstützt eine Rollenbasierte Rechtevergabe, in welchem jeder User eine Rolle einnehmen muss, es ist durchaus möglich und auch so gedacht, dass verschiedene User dieselbe Rolle im System einnehmen.

Abbildung 13: userroles



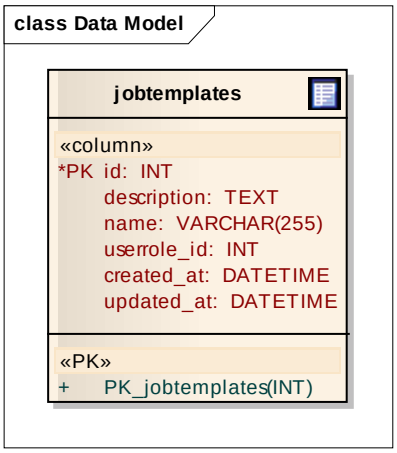
Je nach Rolle hat der User verschiedene Rechte(Permissions). Die Rechte können verschiedenere Natur sein. Sie können sich zum Beispiel auf ein Plugin beziehen.

Abbildung 14:permissions



Unter einem Plugin versteht man eine Schnittstelle zwischen dem System und einem externen Tool, dass über ein solches Plugin gesteuert wird. Ein konkretes Beispiel könnte das Scanning-Tool Nmap sein.

Abbildung 15:plugins



Oftmals ist es so, dass immer eine Reihe von Standardtests auf ein Ziel ausgeführt wird. Aus diesem Grund gibt es die Möglichkeit Jobtemplates zu erstellen, die eine gewisse Anzahl Plugins umfassen.

Abbildung 16: jobtemplates

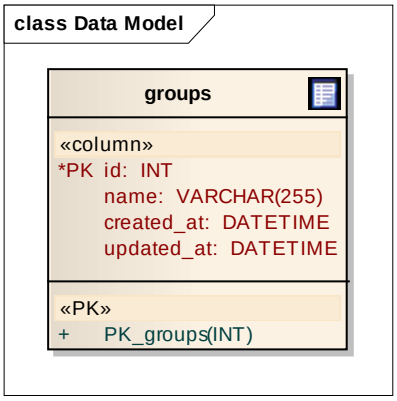


Abbildung 17: groups

Zusätzlich zu den Rollen ist es möglich User einer Gruppe zu zuweisen. Gruppen sind dazu da, dass User mit verschiedenen Rollen und entsprechend mit verschiedenen Rechten, die Möglichkeit haben, die Scanergebnisse mit Usern derselben Gruppe zu teilen.

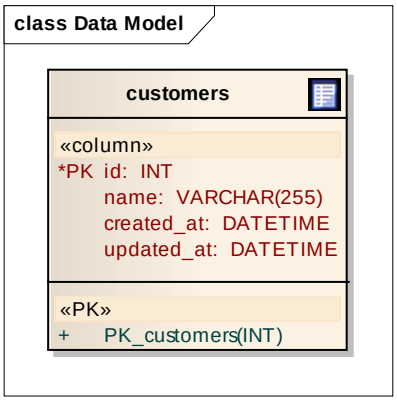


Abbildung 18: customers

Für den Kunden könne spezielle User erfasst werden, die sich auf einen Kunden beziehen. Dadurch hat der Kund die Möglichkeit Sich im System einzuloggen und alle Scans und Ergebnisse anzusehen die sich auf Ihn beziehen.

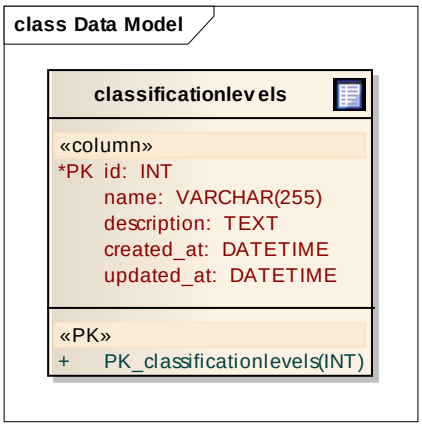


Abbildung 19: classificationlevels

Die Classificationlevels sind dazu da erfasste Daten in ihrer Vertraulichkeit zu gewichten. Die Idee dabei ist, dass bestimmte Daten nur für gewisse Userroles zugänglich gemacht werden.

10.3.3 Jobs

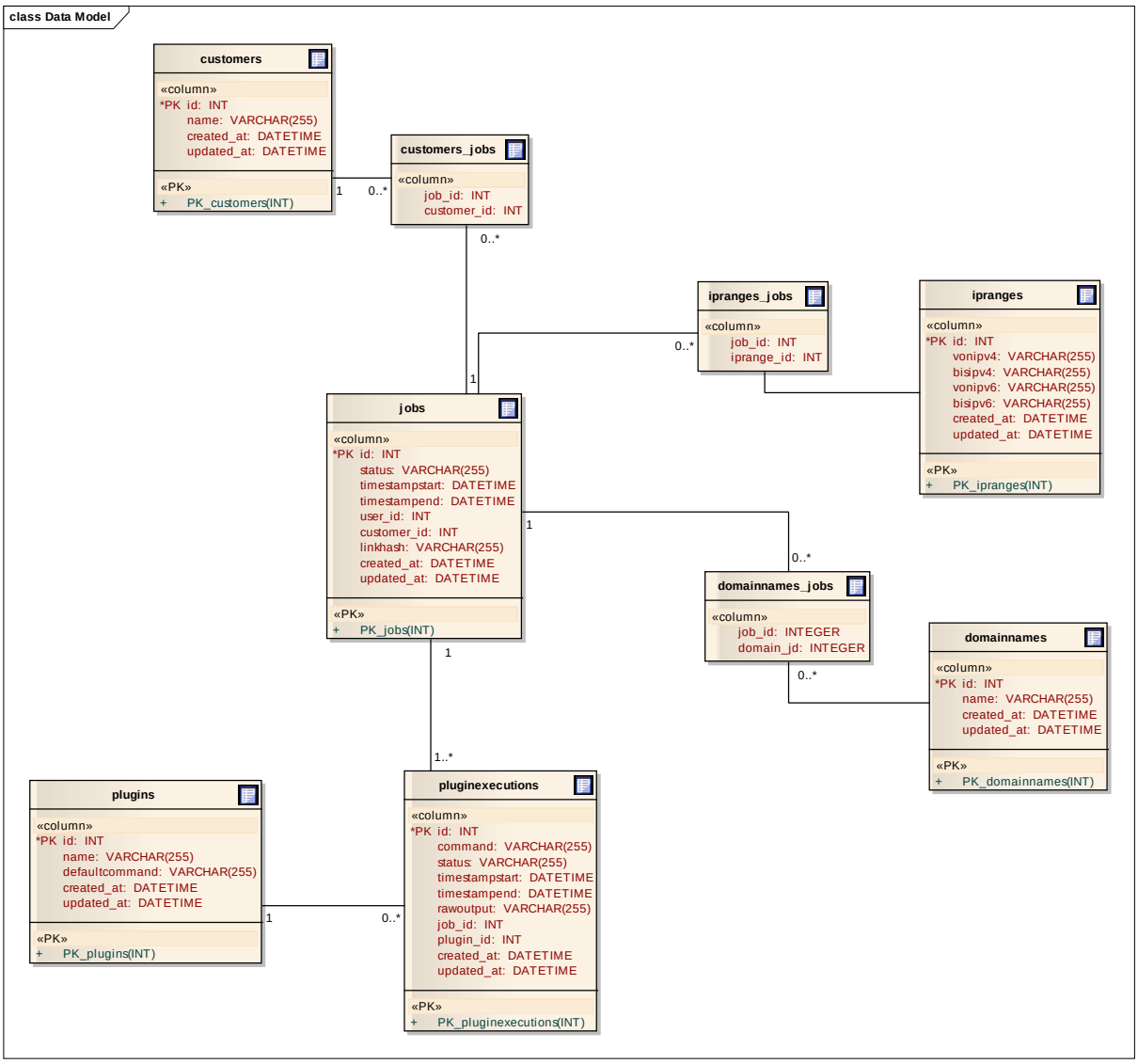
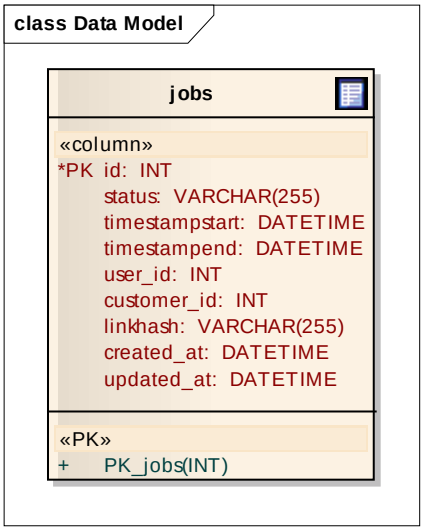


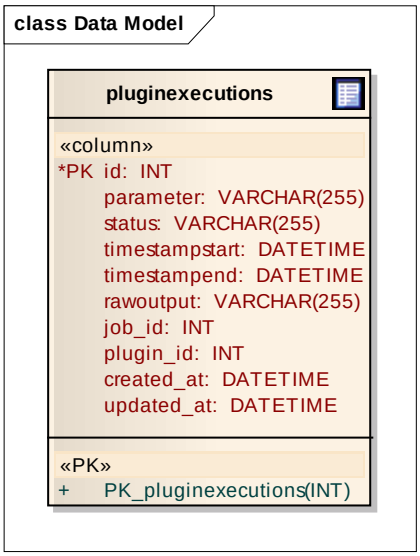
Abbildung 20: jobs

Ein weiterer Bereich des Datenbankschemas betrifft das erfassen von Jobs, welche von einem User erfasst werden können.



Unter einen Job versteht man eine gewisse Anzahl von Pluginexecutions, die sich auf ein gewisses Ziel beziehen, das kann eine bestimmte Anzahl von IP-Ranges und/oder Domainnames sein.

Abbildung 21: jobs



Eine Pluginexecution ist die Ausführung eines Plugins. Sie hat Parameter, die mitgeben werden können. Nachdem die Pluginexecution erfolgreich durchgeführt wurde wird das Resultat verarbeitet.

Im Status wird der Status gespeichert in welchem sich die Pluginexecution momentan befindet.

Abbildung 22: pluginexecutions

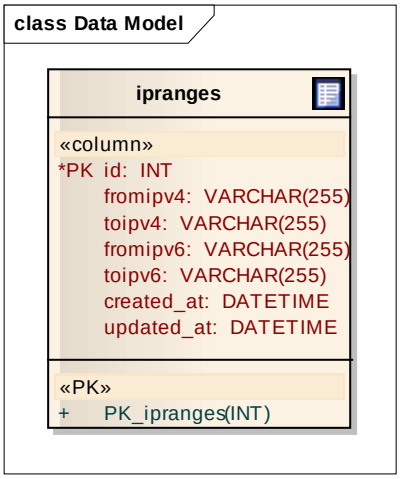


Abbildung 23: ipranges

Ein Job wird auf ein gewisses Ziel ausgeführt. Dieses Ziel wird durch Ipranges und Domainnames definiert.

Ipranges werden mit einer Start- und einer Endadresse definiert. Beispiel 152.168.10.12 bis 152.168.10.20

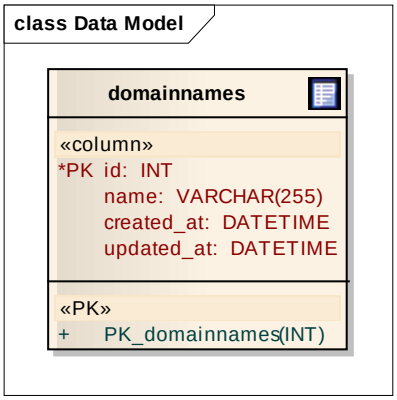


Abbildung 24: domainnames

Eine weitere Möglichkeit für die Zieldefinition ist ein Domainname. Beispielsweise: www.google.ch

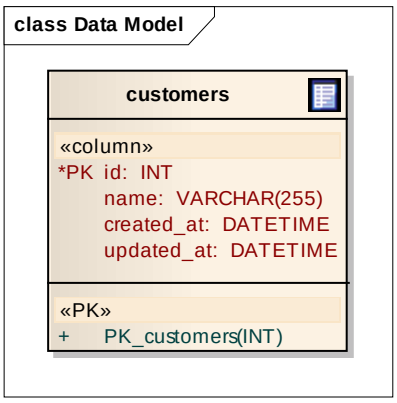


Abbildung 25: customers

Ein Job kann einer gewissen Anzahl Kunden(customers) zugewissen werden.

10.3.4 Suchresultate

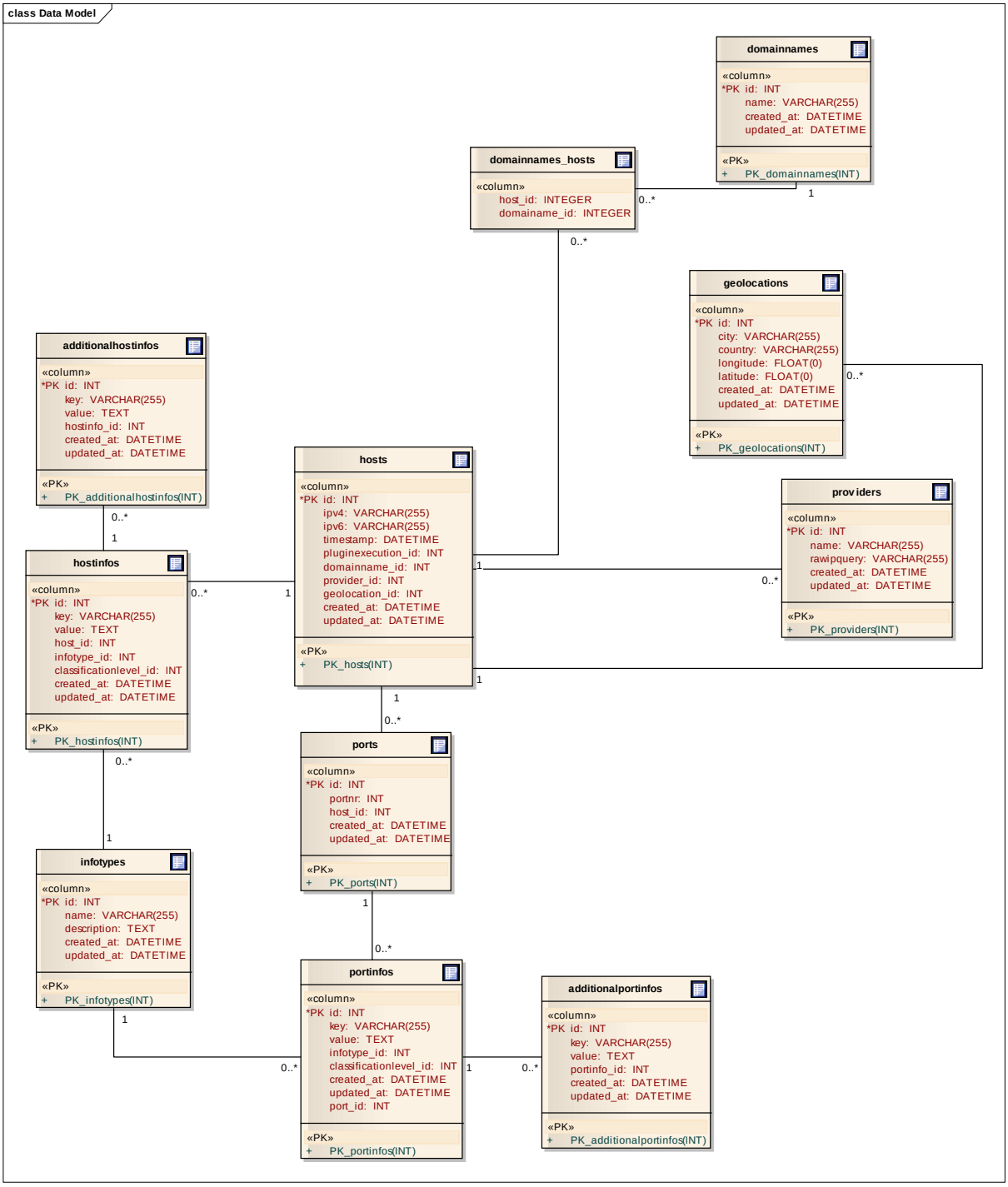
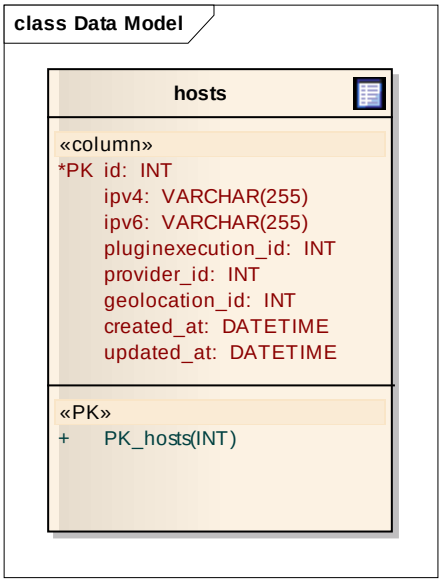


Abbildung 26: Suchresultate

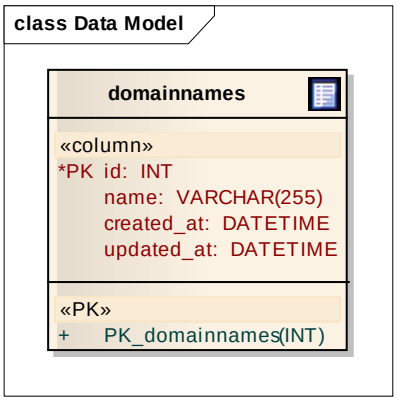
Die verschiedenen Pentest-Tools erzeugen verschiedene Arten von Outputs. Diese Informationen müssen in eine einheitliche Form gebracht werden, dazu wird der Output geparst und in der Datenbank abgelegt. Die Tabellen und Felder sind sehr generisch gehalten, damit eine Vielzahl von verschiedenen Informationen abgebildet werden können.



Ein Host verfügt immer über eine IPv4 Adresse

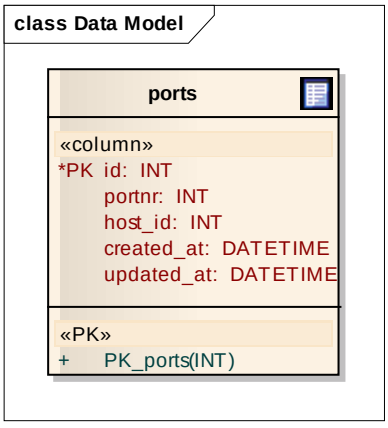
Zusätzlich besteht die Möglichkeit eine IPv6 Adresse zu speichern. Ein Host referenziert über pluginexecution_id die Pluginexecution mit welcher dieser Host und die damit Verbunden Informationen gefunden worden sind.

Abbildung 27: hosts



Falls URLs/Domainnames zu einem Host gefunden werden, können diese in Domainnames gespeichert werden.

Abbildung 28: domainnames



Jeder Port eines Hosts wird in Ports erfasst und referenziert den entsprechenden Host.

Abbildung 29: ports

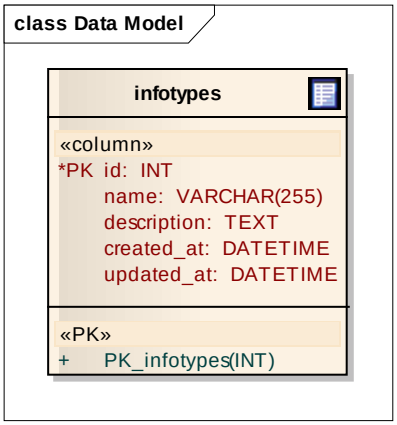


Abbildung 30: infotypes

In Infotypes werden Infotypen abgelegt, welche die Informationen zu einem Host oder einem Port klassifizieren.

Wir haben einige Typen vordefiniert:

Info: Generelle Informationen zu einem Host oder Port: z.B.: Das Protokoll auf diesem Port

Vulnerability Critical: Kritische Verletzlichkeit die gefunden worden ist.

Vulnerability Important: Wichtige Verletzlichkeit, die beachtet werden sollte

Vulnerability Low: Weniger wichtige Verletzlichkeiten.

Vulnerability Information: Verletzlichkeitsinformation

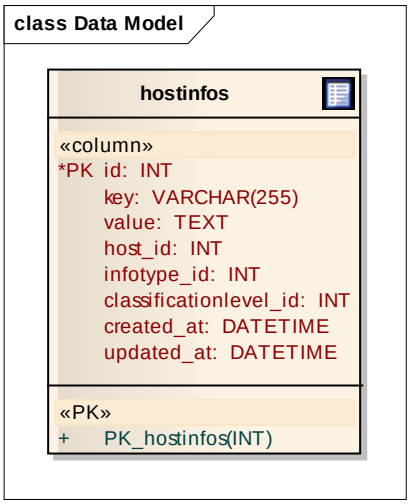
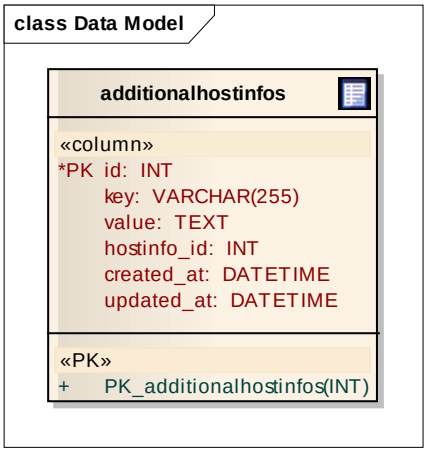


Abbildung 31: hostinfos

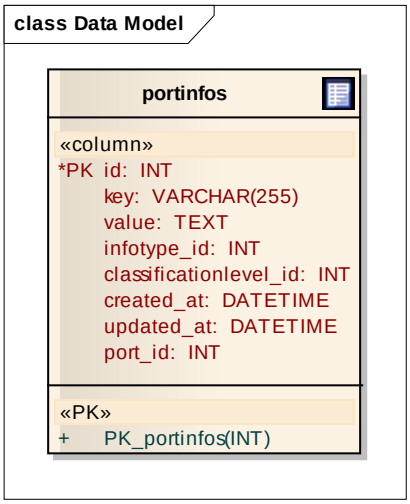
Hostinfos wie auch Portinfos sind sehr generische Tabellen. Die wichtigsten Felder sind „Key“ und „Value“. Alle Informationen werden in diesen Key-Value-Pair abgelegt. Dies hat den Vorteil, dass auch in Zukunft verschiedenste Informationen abgelegt werden können.

In Hostinfos werden Informationen abgelegt, welche nicht direkt einen Port angehängt werden können, werden dem Host angehängt.



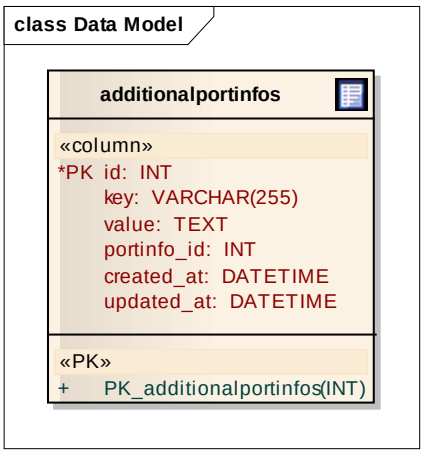
Falls zu einer Hostinfo noch komplexere Informationen abgebildet werden müssen, welche sich nicht nur in einem Key Value Pair abbilden lassen, so können diese in additionalhostinfos abgelegt.

Abbildung 32: additionalhostinfos



Genau gleich wie bei Hostinfos werden Informationen zu einem Port in einem Key-Value-Pair abgelegt.

Abbildung 33: portinfos

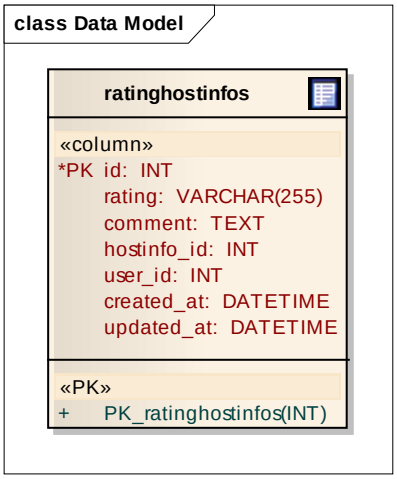


Die Additionalportinfos Tabelle enthält weitere Informationen.

z.B.: Kann in einer Portinfo folgende Vulnarability-Information abgelegt werden:
Key: vulnerability und Value: ConfirmedBlindSQLInjection

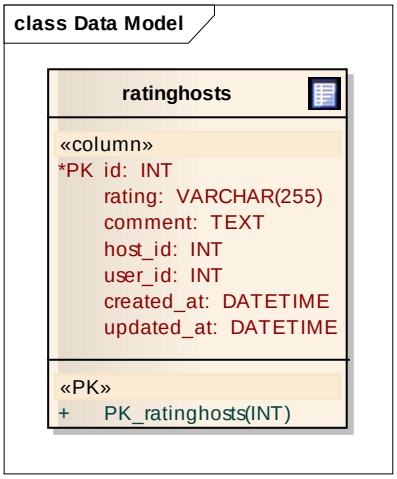
Additionalinformationen wären z.B. das Key-Value-Pair:
Key: CWE und Value: 89

Abbildung 34: additionalportinfos



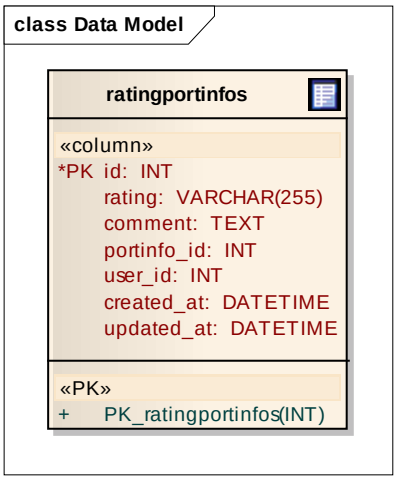
Ratinghostinfos ist dazu da, um Informationen, die durch ein Programm erzeugt worden sind zu bewerten. Falls ein Benutzer merkt, dass ein Resultat eines Tools falsch ist, kann er so einen Kommentar zu dieser Information scheiben und dies Vermerken.

Abbildung 35: ratinghostinfos



Ratinginfos ist dazu da um Informationen die durch ein Programm erzeugt worden sind zu bewerten Falls ein Benutzer merkt, dass ein Resultat eines Tools falsch ist, kann er so einen Kommentar zu dieser Information scheiben und dies Vermerken.

Abbildung 36: ratinghost



Ratingportinfos sind dazu da, um Portinformationen die durch ein Programm erzeugt worden sind zu bewerten. Falls ein Benutzer merkt, dass ein Resultat eines Tools falsch ist, kann er so einen Kommentar zu dieser Information scheiben und dies Vermerken.

Abbildung 37: ratingportinfos

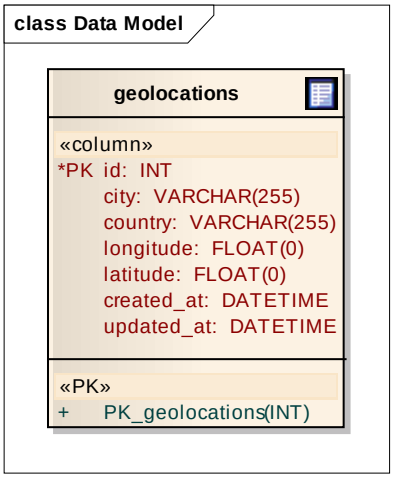


Abbildung 38: geolocation

Falls die Geolocation eines Hosts bekannt ist, kann diese in Geolocations abgebildet werden und einem Host angehängt werden.

11 Code Qualität

Um die Codeqalität zu sichern haben wir Coderichtlinen verwendet.

11.1 Coderichtlinien

11.1.1 Übersicht

Der Code wird in der gewohnten Standardformatierung (wie sie von Eclipse vorgegeben wird) und mit den Namenskonventionen, die sich in der Ruby-Programmierung weit verbreitet und durchgesetzt haben, geschrieben. Der Code soll weitgehend selbsterklärend formuliert werden, sodass Kommentare sparsam und gezielt für die Datei-/Klassenbeschreibung und komplizierte Methoden/Codebereiche eingesetzt wird.

11.1.2 Reihenfolge

Die Reihenfolge der verschiedenen Codekomponenten soll sich folgendermassen strukturieren:

1. Header Block mit gemäss Quellcode-Header (siehe Kommentare)
2. „require“ Anweisungen
3. „include“ Anweisungen
4. Klassen und Modul Definitionen
5. Hauptprogrammteil
6. Testcode

11.1.3 Formatierung

Die Codeformatierung wie Einzug, Klammersetzung etc. wird von der Eclipse-Standardeinstellung übernommen. Eclipse ermöglicht neben der Eingabeunterstützung und –Vervollständigung auch eine automatische Formatierung, welche auf geschriebenen Code anzuwenden ist.

11.1.4 Namenskonventionen

Die Namen aller Bezeichner sind in englischer Sprache.
Folgende Schreibkonventionen sollen von den Programmierern eingehalten werden:

Sprachelement	Schreibweise	Beispiel
Ordner/Dateinamen	- Nur Kleinbuchstaben - Worttrennung durch Unterstriche	app, service, config, test_helper
Klassen/Module	- Nomen - CamelCase	Worker, JobAggregator
Methode	- Verben - Nur Kleinbuchstaben - Worttrennung durch Unterstriche - Abfrage-Methoden für Feldwerte vermeiden -> Accessor verwenden - Boolsche Variablen abfragen mit: *?	parse, create_commands, empty?
Attribute	- Nomen - Nur Kleinbuchstaben - Initialisierung immer mit *.new	jobs = Array.new
Konstanten	- Nur Grossbuchstaben - Worttrennung durch Unterstriche	SCHEDULER_DELAY
Blöcke	- Blöcke mit einer einzelnen Anweisung mit geschweiften Klammer definieren - Blöcke mit mehreren Anweisungen mit „do“ und „end“ definieren	array.each { |element| element.do() } array.each do |element| element.do_a() element.do_b() end

11.1.5 Kommentare

Alle Kommentare sind kompatibel zu RDoc und in englischer Sprache. Grundsätzlich sind die Bezeichner so zu wählen, dass sich der Code selbsterklärend und ohne Kommentar lesen lässt.

11.1.5.1 Header

Jede Ruby-Quellcodedatei ausgenommen wird mit einem einheitlich aufgebauten Kommentar-Header versehen. Der Header enthält die wesentlichen Informationen über die Datei.

```
# Project::   Akoben
# File::      [filename].rb
# Version::   [fileversion]
#
# Author::    Jonas Hofer, Remo Egli
# Copyright:: Copyright (c) 2011
#
# Last Change: [Date]
# Review:      [Date]
```


12 User Interface

In diesem Kapitel werden die wichtigsten Screens und teilweise deren Paperprototypes erklärt.

12.1 Login

Der Login Screen.
Falls ein User bereits eingeloggt ist, wird er weiter geleitet.

Falls eine andere Seite aufgerufen wird und der User nicht eingeloggt ist, dann wird er auf die Loginseite umgeleitet.

Abbildung 39: Login

12.2 Navigationbar

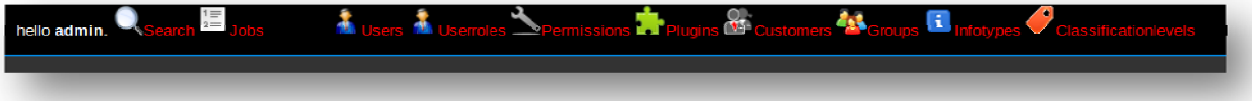


Abbildung 40: Navigationbar

Die Navigation oberhalb der Website ist jederzeit sichtbar:
Die Menupunkte „Search“ und „Jobs“ sind für alle sichtbar. Die anderen Menupunkte stehen nur dem Administrator zur Verfügung.

12.3 Hints

Auf manchen Seiten erscheinen auf der rechten Seite blaue Fragenzeichen. Durch anklicken werden dem User nützliche Hinweise eingeblendet.

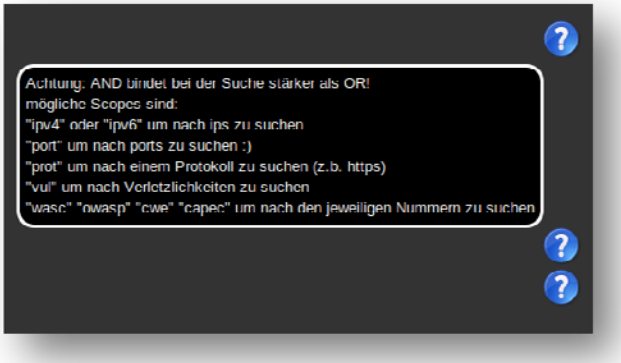


Abbildung 41: Hints

12.4 Suche

12.4.1 Paperprototype

Mittels diesem Paperprototype wurde festgestellt, dass die Suche mittels Suchstring im Stil von Google definiert wird und nicht via Drop-down.

Abbildung 42: Paperprototype Suche

12.4.2 Screens

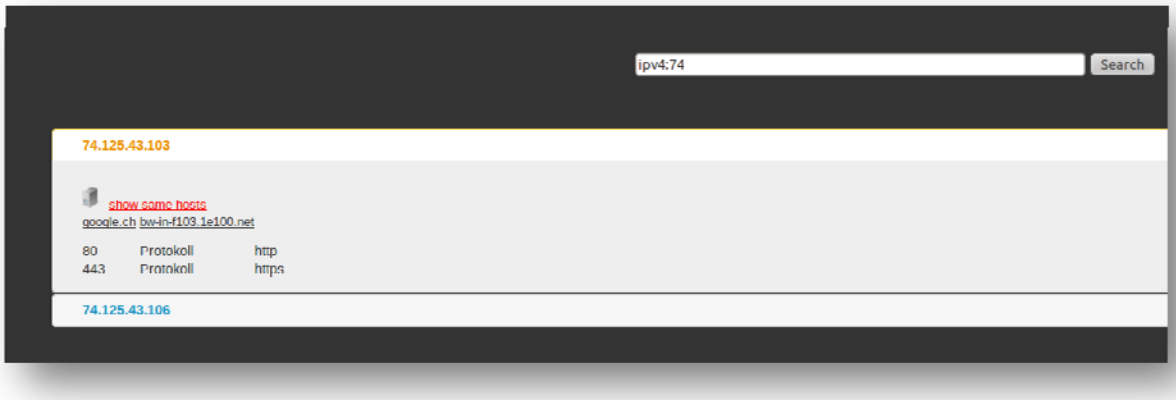
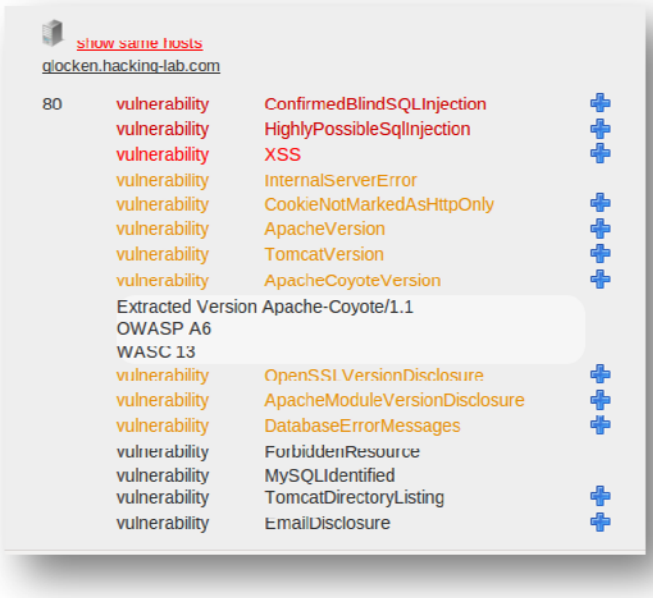


Abbildung 43: Suche

Nach dem eine Suche abgeschickt wurde, werden unterhalb der Suchleiste die Resultate eingeblendet.
Durch das anklicken einer IP werden die Informationen zu diesem Host angezeigt.



Gewisse Informationen haben noch weitere Informationen. Diese können mit dem Plus-Icon eingeblendet werden.

Abbildung 44: Hostdetails

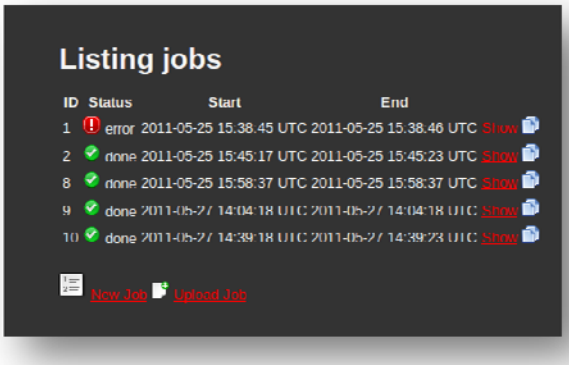
12.5 Jobs

12.5.1 Paperprototype

Mit diesem Paperprototype wurde klar, dass sowohl IP Ranges wie auch Domainnames gleichzeigt erfasst werden können müssen und eine beliebige Anzahl von Domainnames.

Abbildung 45: Paperprototype Job erfassen

12.5.2 Screens



Unter Jobs werden alle Jobs angezeigt, die vom eingeloggten User erfasst worden sind. In der Übersicht sieht man den Status des Jobs, wann er gestartet wurde und wann er fertig durchgelaufen war.

Abbildung 46: Jobs Übersicht

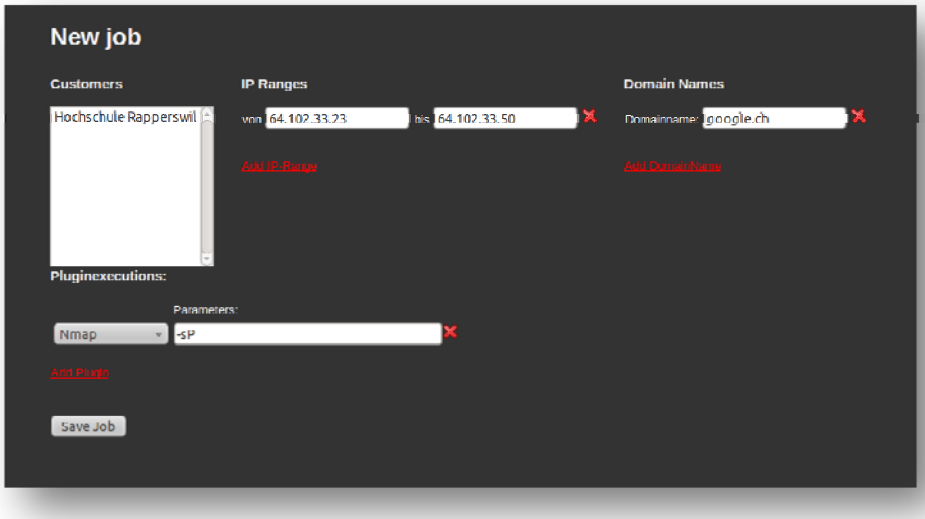


Abbildung 47: Job erfassen

Beim Erfassen eines neuen Jobs können dynamische eine beliebige Anzahl IP-Ranges und Domainnames definiert werden und eine beliebige Anzahl Pluginexecutions.

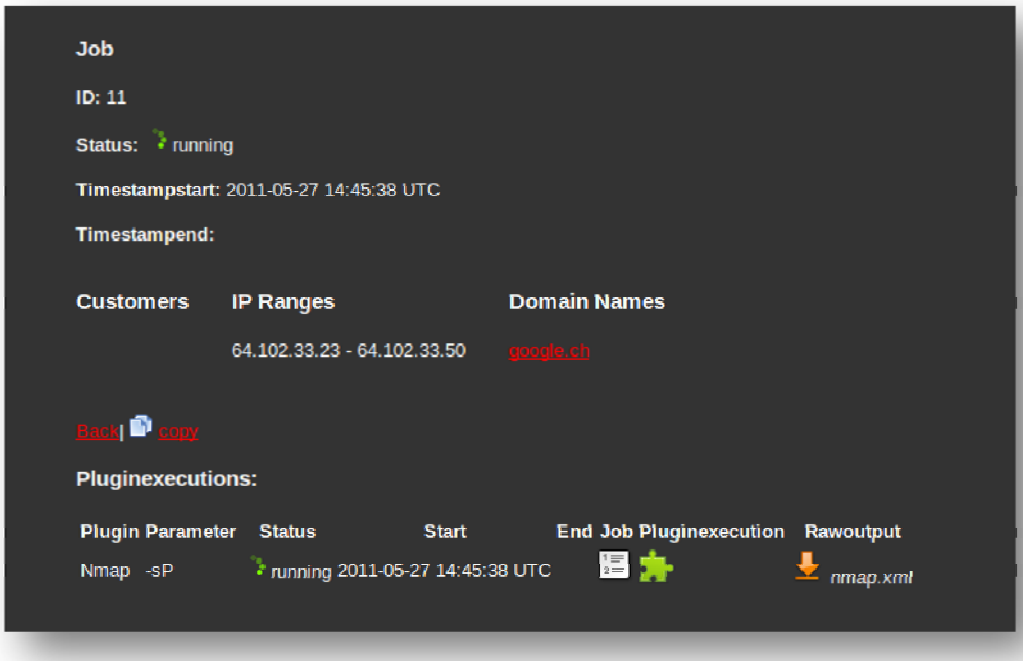
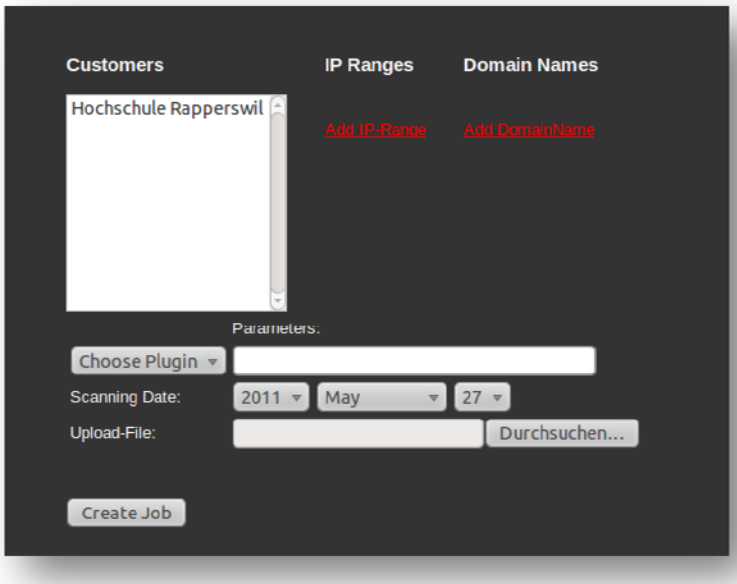


Abbildung 48: Job Details

Nach dem Speichern wird der Job nochmals angezeigt.

12.5.3 Upload Job



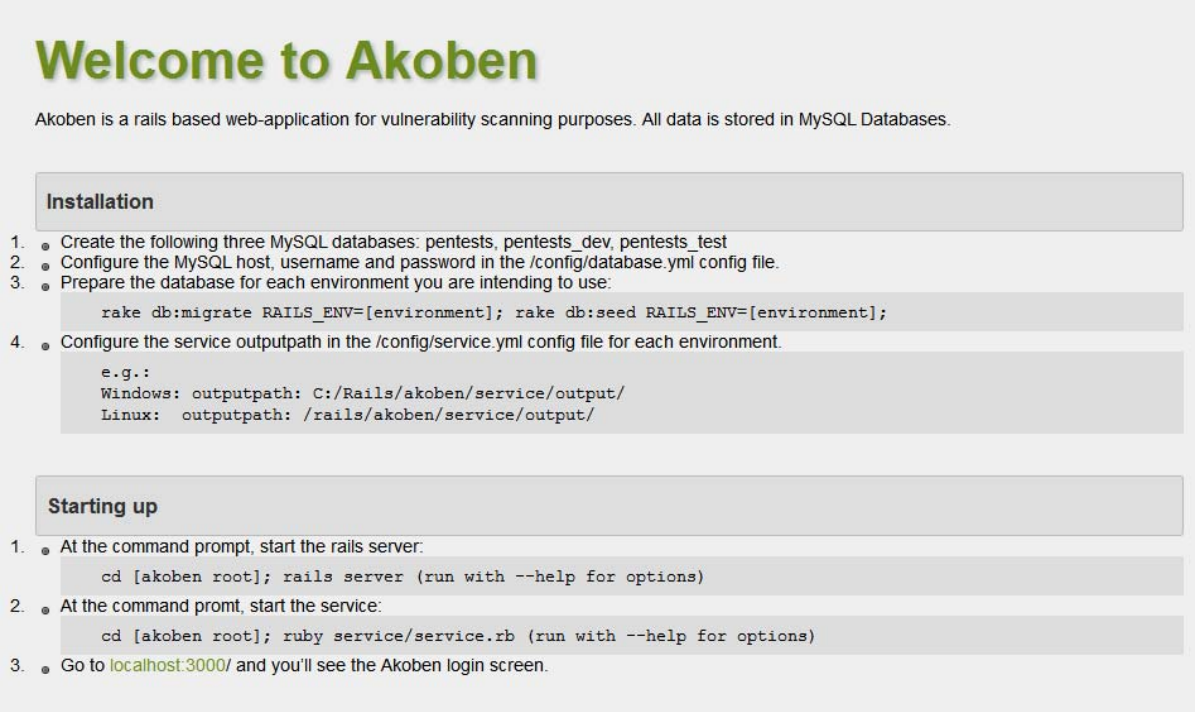
Falls ein Scan nicht über Akoben ausgeführt wurde, hat man die Möglichkeit, die Rohdaten hoch zuladen. Dabei wird der „running“ Status übersprungen und es wird gleich beim „parsing“ eingestiegen.

Abbildung 49: Job Upload

13 Anleitung

Die vollständige Anleitung findet sich direkt in folgendem Quellcode-Verzeichnis:
[Akoben root]/doc/README_FOR_APP

Alternativ kann die Anleitung auch über die Rails Applikation Dokumentation eingesehen werden:
[Akoben root]/doc/app/doc/README_FOR_APP.html



Welcome to Akoben

Akoben is a rails based web-application for vulnerability scanning purposes. All data is stored in MySQL Databases.

Installation

1. Create the following three MySQL databases: pentests, pentests_dev, pentests_test
2. Configure the MySQL host, username and password in the /config/database.yml config file.
3. Prepare the database for each environment you are intending to use:

```
rake db:migrate RAILS_ENV=[environment]; rake db:seed RAILS_ENV=[environment];
```

4. Configure the service outputpath in the /config/service.yml config file for each environment.

```
e.g.:
Windows: outputpath: C:/Rails/akoben/service/output/
Linux:   outputpath: /rails/akoben/service/output/
```

Starting up

1. At the command prompt, start the rails server:

```
cd [akoben root]; rails server (run with --help for options)
```

2. At the command promt, start the service:

```
cd [akoben root]; ruby service/service.rb (run with --help for options)
```

3. Go to localhost:3000/ and you'll see the Akoben login screen.

Abbildung 50: Installationsanleitung in der Codedokumentation

14 Ausblick

14.1 Vision

Akoben soll für den Pentester ein One-Stop-Shop für die ersten Arbeitsschritte einer Security Analyse sein. Zu diesem Zweck muss Akoben die Interaktion mit einer Grosszahl der etablierter Penetration- und Vulnerability-Testing-Software ermöglichen. Unsere Software soll zugleich standardisierte Abläufe vereinfachen, wie auch flexibel auf spezielle, detailliertere Anforderungen des Benutzers eingehen können.

14.2 „Future Features“

Folgende Funktionen und Erweiterungen könnten in einer zukünftigen Version implementiert sein (zusätzlich zu den Punkten im Abschnitt „Ausblick“ im Management Summary):

- Dynamisches Update der States im GUI
  Damit die Jobübersicht nicht andauernd neu geladen werden muss soll sich der Status über AJAX dynamisch aktualisieren.
- Watchdog / Lifecheck Konzept
  Es soll ein eigenständiger Service-Teil eingeführt werden, welcher in regelmässigen Abständen überprüft, ob der Zustand des Servers und der Internetverbindung in Ordnung ist. Damit soll verhindert werden, dass bei Verbindungsfehlern falsche Ergebnisse in der Datenbank gespeichert werden.
- Kontrolle des Job-Services über GUI
  Der Administrator sollte die Möglichkeit haben, den Service von der Web-Oberfläche starten/stoppen/neustarten zu können.
- Kontrolle der Plugin-Rückgabewerte
  Momentan wird angenommen, dass der Code der Plugins von einer vertrauenswürdigen Quelle stammt und kein Schadcode ausgeführt wird. Es empfiehlt sich, dies in einer zukünftigen Version entsprechend anzupassen.
- State-Model in Enums umwandeln
  Um eine möglichst konsistente Verwendung des State-Models zu garantieren, sollten diese in einer konstanten Art im Programm hinterlegt sein und nur vom Entwickler angepasst werden können.
- Veröffentlichung einer Community Edition
  Die grosse Menge an verfügbarer Sicherheitssoftware verunmöglicht die Entwicklung von Plugins für jedes beliebige Tool. Würde man eine Community Edition von Akoben veröffentlichen, wäre es Möglich, dass auch andere Entwickler sich daran beteiligen und die Qualität der Software um einiges steigern liesse.
- Veröffentlichung als Dienstleistung
  Da das Interesse der Öffentlichkeit im Bereich der IT-Sicherheit in letzter Zeit gestiegen ist, könnte man Akoben dazu nutzen, der Bevölkerung eine Überprüfung der eigenen IT-Infrastruktur anzubieten. Dies würde zum Beispiel so funktionieren, dass ein anonymer Benutzer berechtigt ist, eine bestimmte Art von Scans auf sein Ausgangssystem (das System,

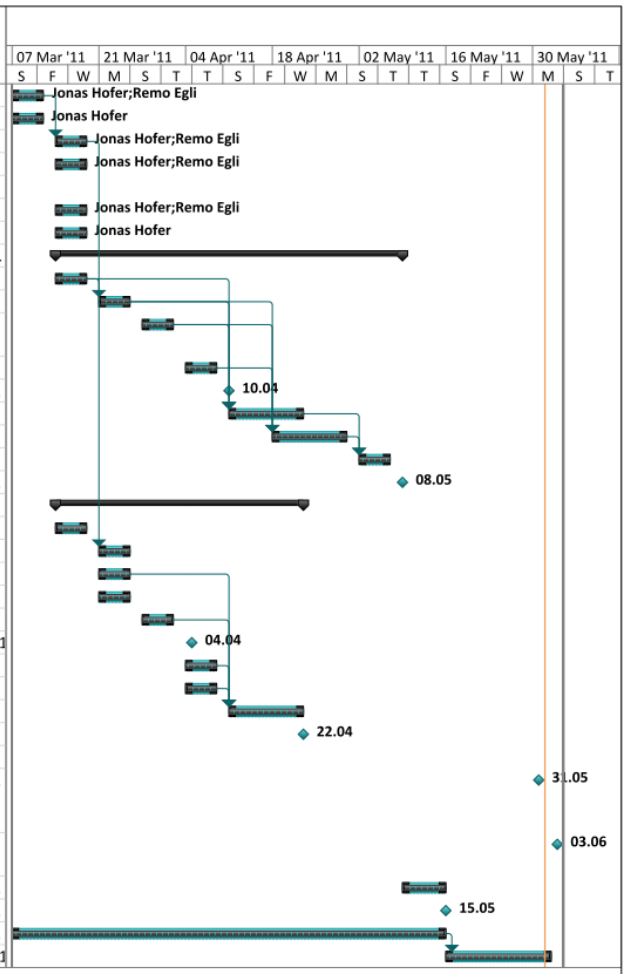
von welchem er auf unseren Service zugreift) durchzuführen. Zugriff auf das Resultat erfolgt über einen One-Time-URL.

- Anbindung anderer Informationsquellen
  Es bestehen bereits viele Datenbanken, deren Informationen sich mit unserem System verknüpfen liessen. Um nur ein paar davon zu nennen:
  - SHODAN
  - Metasploit
  - OWASP
  - WHOIS
  - Google Maps
- Auswertung/Aufbereitung der Informationen
  Wenn mehrere Scans von einem bestimmten Ziel-Host gemacht wurden sollte man diese Informationen vergleichen können, so dass Änderungen und Abweichungen beispielsweise farbig markiert sind.

15 Projektplanung

15.1 Grobplanung

ID	Task Name	Duration	Start	Finish
1	Abklärung Anforderungen	5 days	Mon 07.03.11	Fri 11.03.11
2	Abklärung Technologie	5 days	Mon 07.03.11	Fri 11.03.11
3	Erfassung UseCases	5 days	Mon 14.03.11	Fri 18.03.11
4	Projektplan	5 days	Mon 14.03.11	Fri 18.03.11
5				
6	Design DB-Schema	5 days	Mon 14.03.11	Fri 18.03.11
7	Umsetzung: DB-Schema	5 days	Mon 14.03.11	Fri 18.03.11
8	**ServiceApplication**	**40 days**	**Mon 14.03.11**	**Sun 08.05.11**
9	Design ServiceApplication	5 days	Mon 14.03.11	Fri 18.03.11
10	Prototype: Threading	5 days	Mon 21.03.11	Fri 25.03.11
11	Prototype: Plugin Mechanismus	5 days	Mon 28.03.11	Fri 01.04.11
12	Prototype DB Enititymanger	5 days	Mon 04.04.11	Fri 08.04.11
13	Prototyp Release	1 day	Sun 10.04.11	Sun 10.04.11
14	Prototyp erstes Plugin: NMAP	10 days	Mon 11.04.11	Fri 22.04.11
15	Protypes Zusammen führen	10 days	Mon 18.04.11	Fri 29.04.11
16	Prototypes Verfeinern	5 days	Mon 02.05.11	Fri 06.05.11
17	PreRelease ServiceApp	1 day	Sun 08.05.11	Sun 08.05.11
18	**WebApplication**	**30 days**	**Mon 14.03.11**	**Fri 22.04.11**
19	Domain Entitys in RoR	5 days	Mon 14.03.11	Fri 18.03.11
20	Paperprototype Userinterface	5 days	Mon 21.03.11	Fri 25.03.11
21	RoR: Login	5 days	Mon 21.03.11	Fri 25.03.11
22	RoR: Admin Backend	5 days	Mon 21.03.11	Fri 25.03.11
23	RoR: Job erfassungs Tool	5 days	Mon 28.03.11	Fri 01.04.11
24	PenTeser kan Job erfassen	0 days	Mon 04.04.11	Mon 04.04.11
25	RoR: Pen. Tester View	5 days	Mon 04.04.11	Fri 08.04.11
26	RoR: Anonymous View	5 days	Mon 04.04.11	Fri 08.04.11
27	RoR: LiveSearch	10 days	Mon 11.04.11	Fri 22.04.11
28	PreRelease WebApp	1 day	Fri 22.04.11	Fri 22.04.11
29				
30	Abgabe an Betreuer: Abstract/Kurzfassung + A0 Poster	0 days	Tue 31.05.11	Tue 31.05.11
31	Abgabe als WordDokument ans Abteilsungssekretariat	0 days	Fri 03.06.11	Fri 03.06.11
32	Testing & Debugging	6 days	Mon 09.05.11	Sun 15.05.11
33	FinalRelease Software	1 day	Sun 15.05.11	Sun 15.05.11
34	Laufende Dokumentation	51 days	Mon 07.03.11	Sun 15.05.11
35	Überarbeitung Dokumentation	13 days	Mon 16.05.11	Wed 01.06.11



15.2 Sprints

Sprint 1 **Von: 28. 02. 2011** **Bis: 02. 03. 2011**

Task 1: Remo, Jonas	Schema der Vulnerability-Datenbank analysieren
Task 2: Remo, Jonas	Entwerfen des Datenbank-Schemas
Task 3: Remo, Jonas	Konzept für Verwaltungsinformationen (Benutzer, Zugriffsmodell, etc.)

Sprint 2	Von: 02. 03. 2011	Bis: 18. 03. 2011
	Task 4: Jonas	Update des Schemas gemäss Besprechung
	Task 5: Remo	Überarbeitetes Schema Herr Oesch zustellen
	Task 6: Jonas	DB Schema auf Ruby umsetzen
	Task 7: Remo	Konzept Service
	Task 8: Jonas	Design Rails Applikation
	Task 9: Remo	Code Style Guide Definieren

Sprint 3	Von: 18. 03. 2011	Bis: 25. 03. 2011
	Task 10: Remo	Design Job-Service Applikation
	Task 11: Jonas	Anpassung Domain Entities
	Task 12 :Remo, Jonas	Use Cases definieren

Sprint 4	Von: 25. 03. 2011	Bis: 01. 04. 2011
	Task 13: Jonas	DB-Schema Final
	Task 14: Jonas	VM Vorbereitung
	Task 15: Jonas, Remo	Milestones definieren
	Task 16: Remo	Threading Model definieren

Sprint 5	Von: 01. 04. 2011	Bis: 11. 04. 2011
	Task 17: Remo, Jonas	Jobs aus DB Lesen und in Queue Schreiben
	Task 18: Jonas, Remo	Job Durchführung Multithreading
	Task 19: Remo, Jonas	Ablagestruktur für Rawoutput

Task 20: Remo	Abklärung Tooloutputs
Task 21: Jonas	Nmap Plugin

Sprint 6 Von: 29. 04. 2011 Bis: 09. 05. 2011

Task 22: Remo, Jonas	Bereinigung Prototype Probleme
Task 23: Remo, Jonas	Nmap parallele Ausführung
Task 24: Jonas	GUI Nice & Smooth
Task 25: Remo	Netsparker Plugin
Task 26: Jonas	Link zu den Rohdaten
Task 27: Remo	Curl Plugin
Task28: Jonas, Remo	Vertikaler Durchstich
Task 29: Jonas	Suche: Alle Host mit einer IP
Task 30: Jonas	Job Upload-Funktion
Task 31: Jonas	Userrechte Verwaltung
Task 32: Remo	Status Handling
Task 33: Remo	Outputfile Check
Task 34: Jonas	SQL Injection beheben
Task 35: Remo	Threading Probleme beheben

Sprint 7 Von: 09. 05. 2011 Bis: 22. 05. 2011

Task 36: Jonas	Indexieren der Datenbankfelder für bessre Performance
Task 37: Remo	Curl Plugin abschliessen
Task 38: Remo, Jonas	Installationsanleitung
Task 39: Jonas	Job Copy

Task 40: Remo	Config – File Mechanismus
Task 41: Jonas	Suche Nach Vulnerability und Banner
Task 42: Jonas	GUI Überarbeitung

Sprint 8 **Von: 22. 05. 2011** **Bis: 29. 05. 2011**

Task 43: Jonas, Remo	Refactoring
Task 44: Remo, Jonas	Dokumentation
Task45: Remo, Jonas	Poster
Task46: Remo	Job-Service Startup mit Parameter und Hilfe
Task 47: Remo	Config – File Mechanismus
Task 48: Jonas	Suche Nach Vulnerability und Banner
Task 49: Jonas	GUI Überarbeitung
Task 50: Remo	Ruby Doc
Task 51: Jonas, Remo	Abgabe!

16 Persönlicher Bericht: Jonas Hofer

16.1 Generell

Die Arbeit hat mir gut gefallen und ich konnte viel profitieren.
Im Verlauf der Arbeit wurde ich ein richtiger Ruby on Rails Fan und werde in zukünftigen Webprojekten wahrscheinlich wieder diese Technik in Betracht ziehen.
Auch mit dem Endprodukt bin ich grösstenteils zufrieden.

16.2 Projektverlauf

Die Aufgabenstellung und die Anforderungen der Arbeit waren sehr offen definiert.
Dies hatte sowohl Vorteile, wie Nachteile. Einerseits war es schön, dass wir das Produkt massgebend mit gestallten konnten und unser Betreuer Ivan Bütler immer ein offenes Ohr für unsere Ideen hatte. Andererseits haben wir aber dadurch alleine für die Bestimmung der Domäne ca. 4 Wochen verbraucht und an den Sitzungen wurde oftmals zu viel Zeit damit verbracht, sich Features zu überlegen, die interessant sein könnten, wobei aber niemand daran gedacht hatte, wie sie umzusetzen sind.

16.3 Planung

Anfangs haben wir einen Projektplan erstellt und versucht die Arbeit grob zu schätzen und zeitlich einzuteilen. Was uns aber nicht wirklich gelungen ist, da für uns beide Ruby und Ruby on Rails eine mehr oder weniger komplett neue Technologie war und wir sehr viel Recherchearbeit betreiben mussten. In Zukunft sind wir dann agiler verfahren, indem wir Ein- bis Zweiwochen Sprints hatten und jeweils an den Sitzungen in Jona definiert haben, welche Features wir bis zum nächsten Meeting implementieren wollen. Dies hat bis auf ein paar Ausnahmen sehr gut funktionierte.

16.4 Resultat

Mit dem Resultat der Applikation bin ich zufrieden. Ich habe viel gelernt und die Arbeit hat mir Grösstenteils viel Freude bereitet. Remo wird diese Arbeit evtl. als Bachelor Arbeit weiterführen und wir überlegen uns, je nach Feedback der Compass Security AG ein OpenSource Projekt daraus zu machen und es auch in Zukunft weiterentwickeln.

17 Persönlicher Bericht: Remo Egli

17.1 Generell

Ich habe während dieser Arbeit viel gelernt, nicht nur im Bezug auf die Technologie, sondern auch den Umgang mit einem Auftraggeber. Im Verlauf der Arbeit konnte ich wieder einiges an Programmier-Erfahrung gewinnen, was mir zwischenzeitlich auch viel Spass macht. Mein ursprüngliches Hauptinteresse an der Arbeit lag eher beim Netzwerk/Security Aspekt, hat sich aber nun auch in den Bereich Ruby und Ruby On Rails verschoben. Mich in diese neue Technologie einzuarbeiten hat mir anfangs grosse Mühe bereitet, da ich meist sehr hohe Erwartungen an mich stelle, dies jedoch teilweise zu Blockaden führt. Durch die Zusammenarbeit mit Jonas habe ich gelernt, dass der erste Entwurf/Prototyp durchaus ein „Gebastel“ sein kann, welches sich dann mit der Zeit entwickelt und schlussendlich doch zu einer stabilen Software führt.
Die Zusammenarbeit mit Ivan Bütler und Philipp Oesch war stets sehr freundlich und respektvoll. Beide waren sehr hilfsbereit und haben sich jeweils viel Zeit für unsere Fragen genommen, was mich beeindruckt hat und wofür ich sehr dankbar bin.

17.2 Projektverlauf

Zu Beginn des Projekts war ich etwas unsicher, da die Aufgabenstellung sehr viel Interpretationsraum offen liess. Da ich mich aber persönlich sehr für Security interessiere, konnte ich im Verlauf der Arbeit ein gutes Verständnis für die Konzepte und Ideen entwickeln. Einiges schwieriger war es dann, diese Gedanken in die Realität umzusetzen und es erforderte viel Arbeit das „Big Picture“ zu analysieren und entsprechend in die Software zu integrieren. Aufgrund dieses Aufwandes haben wir auch viel Zeit mit dem Konzept und der Architektur der Software verbraucht, wodurch sich unser Zeitplan manchmal verschoben hat, oder der Arbeitsbelastung höher war.
Mit zunehmender Dauer der Arbeit wurde meine Vorstellung vom Endprodukt immer genauer und detaillierter, was mich teilweise wieder etwas blockierte. Meine Vision und der damit verbundene Aufwand übersteigt mittlerweile die für die Semester- und Bachelorarbeit zur Verfügung stehende Zeit. Ich überlege mir dieses Projekt, zusammen mit Jonas oder auch alleine, weiterzuentwickeln und eventuell auch zu veröffentlichen in der Hoffnung, Feedback von erfahrenen Entwicklern zu erhalten.

17.3 Planung

Die ursprüngliche Planung erwies sich aufgrund des konzeptionellen Arbeitsteiles als etwas schwerfällig, weshalb wir uns für eine agilere Variante entschieden. Die Scrum-ähnliche Methode erwies sich insofern auch als praktisch, da Jonas als Teilzeitstudent nicht immer an der HSR anwesend war, ich aber trotzdem wusste welche Arbeiten ich zu erledigen hatte.

17.4 Resultat

Obwohl ich viel Zeit und Energie in dieses Projekt investiert habe bin ich noch nicht ganz zufrieden damit. Zu viele Ideen von „coolen Features“ und Ausbaumöglichkeiten haben den Weg in meine Gedanken gefunden und bis die Software den Reifegrad erreicht, welchen ich anstrebe, wird noch viel Aufwand nötig sein. Ich kann diesbezüglich auch nur schwer beurteilen, ob das Ziel im Rahmen einer Semesterarbeit erreicht wurde, da das Endprodukt nicht meiner Vorstellung entspricht.

Ich bin froh, dass der Prototyp mittlerweile eine, nach meinem Ermessen, akzeptable Stabilität erreicht hat und bin gespannt, wie die Testphase bei der Compass Security AG verlaufen wird.

18 Glossar

Begriff	Erklärung
parsen	unter parsen versteht man das abfüllen der Rohdaten in der abgelegten Datei in die Domain-Entities der Applikation.
Paperprototype	Sind Prototypes, die nur auf Papier bestehen. Sie eigenen sich besonders in der Anfangsphase für Abklärungen und Usability Tests.
Meta-Informationen	Sind Informationen zu Informationen.
Job-Service	Im Kontext dieser Arbeit wird unter Job-Service ein Service, der im Hintergrund Aufträge abarbeitet, verstanden.
Plugin-In	Ein Plugin-In ist eine Komponente die sich einfach in ein bestehendes System einfügen lässt.
Search-Engine	Im Kontext dieser Arbeit wird unter Search-Engine ein Mechanismus, welcher es ermöglicht angenehm über die erfassten Daten zu suchen, verstanden.
Akoben	Akoben. Ist der Name der Applikation. Er kommt aus dem Afrikanischen und heisst übersetzt Kriegshorn.
Vulnerability	Vulnerability ist englisch für Verletzlichkeit. Unter einer Verletzlichkeit, versteht man eine Schwachstelle in einem IT-System.
Framework	Framework ist englisch für Rahmenstruktur, darunter wird ein Grundgerüst verstanden, welches die Arbeit mit einer Technologie vereinfacht.
Workflow	Workflow ist englisch für Arbeitsfluss. Ein Workflow ist der generelle Ablauf eines Prozesses.
Job	Job ist englisch für Auftrag. Ein Job beinhaltet die Eckdaten eines Auftrags.
Pluginexecution	Pluginexecution ist englisch für „Plugin“-Ausführung. Im Rahmen dieser Arbeit wird darunter die Ausführung eines Plugin-In der Applikation verstanden. Was meist im Hintergrund ein Penetration-Tool ist.
Scope	Scope ist englisch für Bereich. Mit Scope ist in unserem Fall eine bestimmten Typ einer Meta-Information gemeint.
Geolocation	Geolocation ist die Geographische Ortsangabe mit Längen und Breitengrad.

CWE	Common Weakness Enumeration. Nummern für typische Sicherheitslücken in Software.
OWASP	Open Web Application Security Project Nummer für die Klassifizierung einer Verletzlichkeit.
WASC	Web Application Security Consortium: Nummer für die Klassifizierung einer Verletzlichkeit.
CAPEC	Common Attack Pattern Enumeration and Classification: Nummer für die Klassifizierung einer Verletzlichkeit.
Ipv4	IPv4 steht für Internet Protokoll Version 4.
ipv6	IPv6 steht für Internet Protokoll Version 6.
Banner	Informationen über einen Service auf einem Host
Domain	Eine Adresse zu einem Host.
Pentester	Leute die mit diversen Tools Sicherheitstests auf Netzwerke machen.
Host	Ein Einheit die sich in einem Netzwerkbefindet.
Commandline	Englisch für Kommandozeile. Ein Eingabefester über welches textuelle Befehle abgesetzt werden können.

19 Literaturverzeichnis

JQuery. (22. Mai 2011). Abgerufen am 22. Mai 2011 von Wikipedia: http://de.wikipedia.org/wiki/JQuery

Mysql. (Mai. 22 2011). Abgerufen am 22. Mai 2011 von Wikipedia: http://de.wikipedia.org/wiki/Mysql

Ruby (Programmiersprache). (20. Mai 2011). Abgerufen am 20. Mai 2011 von Wikipedia: http://de.wikipedia.org/wiki/Ruby_(Programmiersprache)

Ruby On Rails. (22. Mai 2011). Abgerufen am 22. Mai 2011 von Wikipedia: http://de.wikipedia.org/wiki/Ruby_On_Rails

20 Abbildungsverzeichnis

21 Anhang

21.1 Plakat

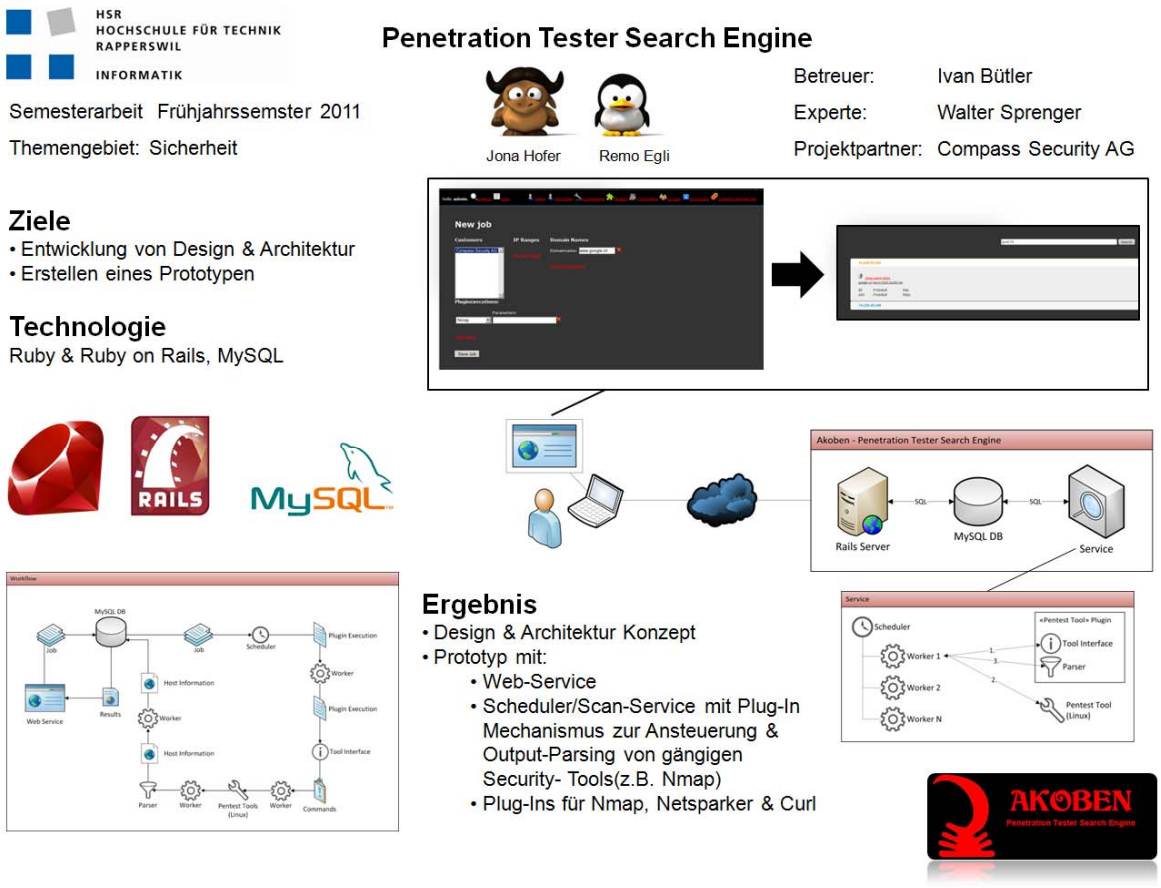


Abbildung 51: Plakat