Passwort-Richtlinien

Password Policies
Bachelor-Thesis von Mario Schlipf
Tag der Einreichung:

1. Gutachten: Prof. Dr. Johannes Buchmann
2. Gutachten: Moritz Horsch

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Kryptographie und Computeralgebra

Passwort-Richtlinien
Password Policies

Vorgelegte Bachelor-Thesis von Mario Schlipf

1. Gutachten: Prof. Dr. Johannes Buchmann
2. Gutachten: Moritz Horsch

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 29. Oktober 2014

(Mario Schlipf)

Danksagung

Ich bedanke mich bei Prof. Dr. Johannes Buchmann und Moritz Horsch für die Bereitstellung von diesem interessanten Thema und die Möglichkeit, mich in neue Themengebiete einarbeiten zu können.

Besonders möchte ich mich bei meinem Betreuer Moritz Horsch für das große Engagement und die hervorragende Betreuung bedanken. Diese und das konstante Feedback während meiner Arbeit waren eine große Unterstützung und Motivation für mich.
Ich möchte mich auch herzlich bei Prof. Dr. med. Michael Kramer bedanken, durch den ich die letzten Jahre die Möglichkeit hatte, mich auch neben meinem Studium in diversen Themen der Informatik fortzubilden.

Schließlich möchte ich bei meiner Familie und meinen Freunden bedanken, die mich während meiner Bachelorarbeit und dem gesamten Studium stets unterstützten. Zuletzt danke ich auch allen, deren Namen ich nicht genannt habe.

Abstract

Choosing passwords to create user accounts on the Internet often confronts users with the difficulty of selecting a valid and secure password. Frequently, the password requirements are not precisely communicated to the user, so that these can only be discovered by trial and error. Even with clear communication, it is often challenging for users to choose a secure password which ideally differs from the passwords used for other accounts.

To solve these problems, a specification for password policies using the XML document format is proposed. It allows the storage of any information that is used for password related tasks such as maintaining user accounts, generating secure passwords or storing the validity period of the password. The specification allows other programs to process these policies in order to support users in many aspects, e.g. in choosing secure passwords.

Two concepts for a centralized and decentralized distribution of the password policies are introduced. In the centralized approach, a third party manages the password policies and provides an appropriate infrastructure to enable accessing a service's policies, whereas the decentralized approach enables the distribution of the policies by the service's provider itself. Both of the approaches are analyzed and compared in detail and a recommendation for a productive system is given.

The requirements for a password which can be represented in the specification have been extracted by an extensive requirement analysis based on websites of the „Alexa Top 500 Global Sites“ list as well as online banking services of various banks. Among other things, the specification allows defining the minimum and maximum length of the password, a period of validity, the allowed characters and the minimum and maximum occurrence of certain characters, such as symbols.

For demonstrating the applicability of password policies in a practical environment, an extension for the password manager KeePass has been developed, which is able to generate secure passwords by providing the URL of a service, without the need to perform other manual configurations, such as the password length or allowed characters. Generated passwords by this extension can be stored automatically with the validity period specified in the policy. The developed specifications for password policies as well as the demonstrator thus support users in the challenge of choosing secure passwords, and therefore contribute significantly to the security of authentication on the internet.

Zusammenfassung

Bei der Wahl von Passwörtern für die Erstellung von Nutzerkonten im Internet stehen Nutzer häufig vor dem Problem, ein gültiges und möglichst sicheres Passwort zu wählen. Oftmals werden die Anforderungen, die das Passwort erfüllen muss, nicht eindeutig an den Nutzer kommuniziert, sodass sich diese nur durch Ausprobieren herausfinden lassen.

Um diese Probleme zu lösen, wird eine Spezifikation vorgestellt, Passwort-Richtlinien in Form von XML-Dokumenten auszudrücken. Mit Hilfe dieser Spezifikation lassen sich alle Informationen, die mit der Verwendung von Passwörtern einhergehen, angeben. Dazu zählen Aufgaben wie die Generierung von sicheren Passwörtern, die Verwaltung von Benutzerkonten und Informationen über die Gültigkeitsdauer der Passwörter. Die Spezifikation ermöglicht es, dass andere Programme diese Richtlinien verarbeiten können, um den Nutzer in vielerlei Hinsicht, beispielsweise bei der Wahl von sicheren Passwörtern, zu unterstützen.

Bezüglich der Verfügbarkeit der Passwort-Richtlinien wurden zwei Konzepte zur zentralen und dezentralen Distribution der Richtlinien vorgestellt. Im zentralen Ansatz verwaltet eine dritte Partei die Passwort-Richtlinien und stellt eine entsprechende Infrastruktur bereit, um die Richtlinien für einen Dienst abzurufen. Der dezentrale Ansatz hingegen ermöglicht die Distribution der Richtlinien durch den Anbieter des Dienstes selbst. Die Ansätze werden ausführlich analysiert und verglichen sowie eine Empfehlung für ein Produktivsystem gegeben.

Die Anforderungen, die sich mit der Spezifikation abbilden lassen, wurden durch eine umfangreiche Anforderungsanalyse anhand von Webseiten der „Alexa Top 500 Global Sites“-Liste sowie von Online-Banking-Portalen verschiedener Banken extrahiert. Dazu zählen unter anderem die minimale und maximale Länge des Passwortes, ein Zeitraum für die Gültigkeitsdauer, die erlaubten Zeichen sowie das minimale und maximale Vorkommen bestimmter Zeichen, wie etwa Sonderzeichen.

Um die Anwendbarkeit von Passwort-Richtlinien im praktischen Umfeld zu demonstrieren, wurde ein Passwort-Generator entwickelt, der als Erweiterung für die Passwort-Verwaltung KeePass implementiert wurde und mit der Angabe der URL eines Dienstes sichere Passwörter generiert, ohne dass der Nutzer weitere Konfigurationen, wie beispielsweise die Passwortlänge oder erlaubten Zeichen manuell vornehmen muss. Die so generierten Passwörter können automatisiert mit einer von den Passwort-Richtlinien vorgegebenen Gültigkeitsdauer abgespeichert werden. Die entwickelte Spezifikation für Passwort-Richtlinien sowie der Demonstrator unterstützen Nutzer damit aktiv bei der Herausforderung der sicheren Passwortwahl und tragen dabei signifikant zur Sicherheit von Authentisierung im Internet bei.

Inhaltsverzeichnis

Abbildungsverzeichnis

Auflistungsverzeichnis

Abkürzungsverzeichnis

AES	Advanced Encryption Standard
DLL	Dynamic Linked Library
DNS	Domain Name Service
DTD	Document Type Definition
GPL	GNU General Public License
IDE	Integrated Development Environment
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
XML	Extensible Markup Language
XSD	XML Schema Definition

1 Motivation

Die derzeitige Authentisierung im Internet basiert im Wesentlichen auf Passwörtern. Durch die Vielzahl an Diensten, bei denen sich ein Benutzer registriert, ist dieser mit der Herausforderung konfrontiert, sich eine große Anzahl an Passwörtern zu merken oder diese sicher zu verwalten. Die Erfahrung zeigt, dass Nutzer dadurch dazu neigen, schwache Passwörter zu wählen und diese bei mehreren Diensten wiederzuverwenden [11]. Unterstützung bei der Wahl unterschiedlicher Passwörter können Programme zur Passwortverwaltung mit integrierten Passwort-Generatoren bieten, mit deren Hilfe Passwörter nicht nur verwaltet und in verschlüsselter Form gespeichert, sondern auch generiert werden können. Jeder Dienstanbieter hat in der Praxis jedoch eigene Anforderungen an Passwörter, wie z. B. der minimalen und maximalen Passwortlänge oder dem erlaubten Zeichensatz. Während die meisten Dienste die Wahl von Zahlen und Buchstaben erlauben, schränken andere beispielsweise die zugelassenen Sonderzeichen ein.

Die Anforderungen an das Passwort werden bei der Registrierung oftmals für den Benutzer nicht nachvollziehbar angegeben, sodass dieser mehrere Anläufe benötigt, ein vom Dienst akzeptiertes Passwort zu wählen. Insbesondere beim Einsatz von Passwort-Generatoren, um sichere Passwörter zu generieren, gestaltet sich die Konfiguration auf die jeweiligen dienstspezifischen Anforderungen als schwierig. Es bedarf daher einen standardisierten Ansatz, der es ermöglicht, diese Anforderungen in einem spezifizierten Format als Passwort-Richtlinien zu definieren und von Programmen automatisch auswerten zu lassen. Durch die Distribution der Passwort-Richtlinien kann der Aufwand für die Konfiguration von Passwort-Generatoren auf ein Minimum reduziert werden. Mit dem Auslesen der Richtlinien kann ein Passwort-Generator automatisiert dienstspezifische Passwörter generieren, ohne dass Einstellungen, wie z. B. die erlaubten Zeichen, vom Nutzer angegeben werden müssen.

Ziel dieser Arbeit

Ziel dieser Arbeit ist die Entwicklung eines Standards zur Definition von Passwort-Richtlinien. Des Weiteren sollen adäquate Methoden zur Distribution von Passwort-Richtlinien entwickelt werden. In einem ersten Schritt werden Webseiten auf ihre bisherigen Anforderungen an Passwörter überprüft. Die extrahierten Anforderungen an die Passwörter sollen genutzt werden, um eine Spezifikation für Passwort-Richtlinien zu erstellen, die in der Lage ist, diese Anforderungen zu repräsentieren. Die Passwort-Richtlinien sollen dabei in XML (Extensible Markup Language) [4] definiert werden können. Eine zu entwickelnde XML Schema Definition (XSD) [7] gibt dabei die Struktur der Passwort-Richtlinien vor, sodass diese auf Korrektheit überprüft werden können. Neben den eigentlichen Anforderungen an das Passwort sollen in diesen Dateien auch Informationen über den Dienst, wie etwa die URL für die „Passwort vergessen“-Funktion und auch die Gültigkeitsdauer des Passwortes, d. h. wann ein Benutzer

ein neues Passwort vergeben muss, abgelegt werden. Im Anschluss sollen Konzepte für die Distribution dieser Richtlinien entwickelt werden. Durch ein geeignetes Konzept zur Distribution können externe Programme wie etwa Passwort-Generatoren auf diese zugreifen. So können beispielsweise Passwörter generiert werden, welche die dienstspezifischen Anforderungen erfüllen, ohne dass der Nutzer den Passwort-Generator konfigurieren muss. Zum Abschluss soll ein Prototyp implementiert werden, um die Anwendung der Passwort-Richtlinien zu demonstrieren. Der Prototyp soll dabei ein Passwort-Generator als Plug-In für die weit verbreitete Passwortverwaltung KeePass [19] bereitstellen. Dadurch soll es möglich sein, durch die Angabe einer URL zu einem Dienst passende Passwörter zu generieren, welche die dienstspezifischen Anforderungen erfüllen.

2 Passwort-Richtlinien

Im folgenden Kapitel wird die Entwicklung einer Spezifikation für Passwort-Richtlinien beschrieben. Zuerst wird auf die Anforderungsanalyse eingegangen, um eine Spezifikation zu entwickeln, welche die Anforderungen der Dienstanbieter im Internet an Passwörter widerspiegelt. Anschließend wird die Spezifikation anhand eines Beispiels erläutert sowie Konzepte für die Distribution der Richtlinien vorgestellt.

Passwort-Richtlinien werden bereits in anderen Bereichen, wie etwa Nutzerzugängen in Unternehmensumgebungen, genutzt. Dort dienen sie dem Zweck, die Wahl von sicheren Passwörtern durch verschiedene Regeln zu erzwingen. Der Nutzer wird dabei darauf hingewiesen, welche Anforderungen sein Passwort erfüllen muss. Die Wahl eines unsicheren Passwortes soll so automatisiert vermieden werden, um Unternehmens- oder Nutzerdaten vom Zugriff Dritter zu schützen. Die Passwort-Richtlinien liegen dabei nicht in einem standardisierten Format vor, sondern werden abhängig von dem genutzten System konfiguriert und dem Nutzer auf unterschiedliche Weisen mitgeteilt.

Im Internet existieren bei der Wahl von Passwörtern in den meisten Fällen ebenfalls Anforderungen, welche die Nutzer erfüllen müssen. Diese werden allerdings nicht immer eindeutig kommuniziert, sodass der Nutzer diese erst herausfinden muss. Die hier vorgestellte Spezifikation für Passwort-Richtlinien stellt eine Möglichkeit dar, alle Informationen, die mit der Nutzung von Passwörtern im Internet zusammenhängen, in einem einheitlichen Format abzulegen. So kann sie beispielsweise genutzt werden, um Passwörter mit Hilfe von Passwort-Generatoren ohne die vorherige Konfiguration auf die dienstspezifischen Anforderungen zu generieren, oder die Anforderungen für den Nutzer einheitlich darzustellen. Durch Informationen über den jeweiligen Dienst kann der Benutzer schnell Funktionen wie Registrierungs-Formular oder die „Passwort vergessen“-Funktion wiederfinden.

2.1 Anforderungsanalyse

Um eine praxistaugliche Spezifikation für Passwort-Richtlinien zu erstellen, ist es notwendig, die Anforderungen an Passwörter von bestehenden Diensten im Internet zu untersuchen. Die so gewonnenen Informationen können genutzt werden, um geeignete Strukturen in der Spezifikation vorzusehen.

Als ein erster Anhaltspunkt wurden dafür im Rahmen dieser Bachelor-Thesis die Anforderungen an Passwörter von deutsch- und englischsprachigen Webseiten der „Alexa Top 500 Global Sites“ [1] analysiert. Dabei wurden Duplikate sowie Webseiten mit illegalen oder pornographischen Inhalten entfernt. Die so gewonnene Liste enthält 153 Webseiten, welche auf ihre Anforderungen hin überprüft wurden. Zusätzlich wurden fünf Online-Banking-Portale verschiedener deutscher Banken überprüft. Die untersuchten Portale sind in Tabelle 2.1 gelistet.

Unternehmen	Domain
Deutsche Kreditbank AG	banking.dkb.de
comdirect bank AG	comdirect.de
Commerzbank AG	commerzbanking.de
Landesbank Berlin AG	lbb.de
Sparkasse Darmstadt	sparkasse-darmstadt.de

Tabelle 2.1.: Online-Banking-Portale

Bei den untersuchten Diensten gibt es hinsichtlich der Anforderungen an ein Passwort mitunter große Unterschiede. Während bei einigen Diensten eine Minimallänge von acht Zeichen vorausgesetzt wird, dürfen bei anderen Diensten, überwiegend im Online-Banking, die Passwörter nicht länger als acht Zeichen sein. Zudem kommuniziert ein großer Teil der Anbieter die Anforderungen nicht eindeutig an die Nutzer. Abbildung 2.1 zeigt die von Google angegebenen Anforderungen an ein Passwort für die Erstellung eines neuen Nutzerkontos.

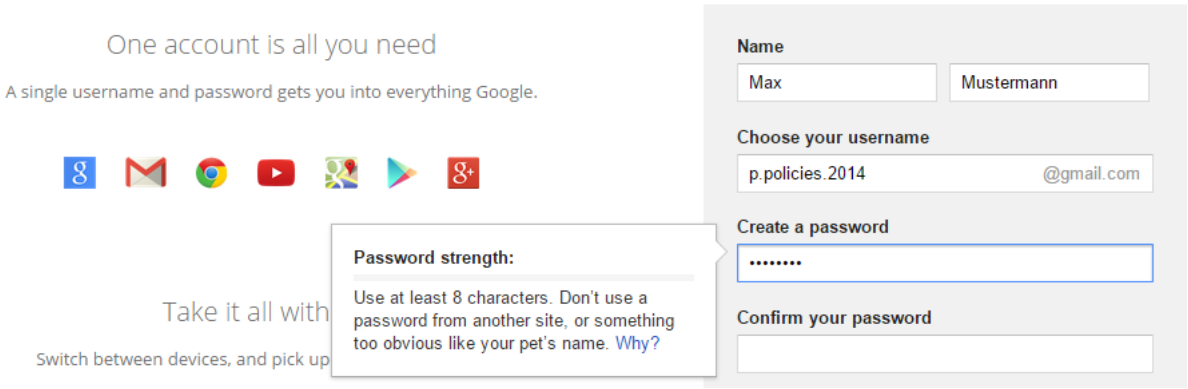

Abbildung 2.1.: Erstellung eines Nutzerkontos für Google-Dienste

Als Anforderung wird genannt, dass das Passwort mindestens aus acht Zeichen bestehen muss und nicht zu schwach sein darf. Wie genau ein „zu schwaches Passwort“ definiert ist, wird dabei offengelassen. Nachdem eine Eingabe seitens des Nutzers erfolgt ist, erhält dieser eine Rückmeldung, ob das eingegebene Passwort akzeptiert wurde. Selbst Kombinationen aus Buchstaben und Zahlen werden teilweise ohne nähere Begründung abgelehnt, wie in Abbildung 2.2 zu sehen ist.

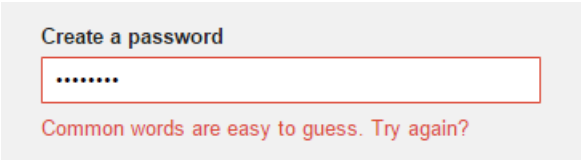

Abbildung 2.2.: Fehlermeldung bei zu schwachem Passwort

Die vollständige Analyse aller Webseiten ergab, dass Anbieter neben Groß-, Kleinbuchstaben und Zahlen unterschiedliche Zeichen für Passwörter zulassen und Bedingungen an Teile dieser Zeichen, wie etwa das Vorkommen von mindestens einem Sonderzeichen, stellen. Im Folgenden werden die aus den analysierten Webseiten gewonnenen Eigenschaften für Passwörter extrahiert, die in den Passwort-Richtlinien abgebildet werden sollen. Ein Beispiel der jeweiligen Anforderungen veranschaulicht die Verwendung bei den Diensten.

- **Minimale Länge**
  „Das Passwort muss mindestens 6 Zeichen lang sein."
- **Maximale Länge**
  „Die maximale Länge des Passwortes beträgt 20 Zeichen."
- **Gültigkeitsdauer**
  „Sie müssen Ihr Passwort spätestens alle 72 Tage erneuern."
- **Erlaubte Zeichen**
  „Ihr Passwort muss aus sechs numerischen Zeichen bestehen."
- **Minimale und maximale Vorkommnisse von Zeichen**
  „Nutzen Sie mindestens ein Sonderzeichen."

Des Weiteren wurde es ermöglicht, diese Zeichensätze für einzelne Positionen in einem Passwort zu beschränken und ein Ablaufdatum für das Passwort festzulegen. Mit der Möglichkeit, die erlaubten Zeichen für einzelne Zeichenpositionen in einem Passwort festzulegen, können selbst komplexe Regeln für Passwörter erstellt werden. Neben den genannten Eigenschaften können in dem entwickelten Dokumenten-Format weitere Informationen über den Dienst hinterlegt werden. Durch Verwendung dieser Informationen können einem Nutzer weitere Hilfestellungen, wie beispielsweise für die Registrierung bei einem Dienst, gegeben werden. Eine ausführliche Beschreibung des Dokumenten-Formats findet sich in Kapitel 2.2.

2.2 XML Schema

Bei XML handelt es sich um eine Auszeichnungssprache, die zur Beschreibung von hierarchisch strukturierten Daten genutzt werden kann. XML hat durch die einheitliche Form der hierarchisch strukturierten Daten einen Standard für den plattformunabhängigen Austausch von Daten geschaffen, welcher insbesondere im Internet zur Anwendung kommt. Die Daten eines XML-Dokumentes liegen dabei als Textdateien vor, sodass XML-Dokumente nicht nur maschinen-, sondern auch menschenlesbar sind. XML-Dokumente haben genau ein Wurzelelement, unter welchem weitere Daten gegliedert werden können.

XML Schema ist eine vom World Wide Web Consortium (W3C) entwickelte XML-basierte Alternative zur Document Type Definition (DTD) und wird zur Beschreibung der Struktur eines XML-Dokumentes genutzt. Es definiert, welche Elemente und Attribute in einem XML-Dokument vorkommen dürfen. Ebenfalls wird die hierarchische Struktur der Elemente festgelegt und somit die möglichen Kindelemente eines Elementes eingeschränkt. Dabei kann neben der Reihenfolge und Anzahl der Kindelemente auch vorgegeben werden, ob ein Element angegeben werden muss, oder nur optional verwendet werden darf. Durch das Festlegen von Standardwerten für Elemente und Attribute kann diesen ein Wert zugewiesen werden, auch wenn das entsprechende Element bzw. Attribut in dem XML-Dokument weggelassen wird. Das Schema bietet so die Möglichkeit, übertragene XML-Dokumente auf ihre Korrektheit zu überprüfen.

In dieser Arbeit wurde die Spezifikation für Passwort-Richtlinien als XML Schema entwickelt, welches das Format für XML-Schema-Instanzen, d. h. XML-Dokumente, vorgibt. Jedes XML-Dokument repräsentiert dabei die Passwort-Richtlinien für eine einzelne Domain. Das Schema ermöglicht z. B., dass Attribute und Elemente einer Passwort-Richtlinie als zwingend erforderlich, andere als optional, festgelegt werden können. Ebenfalls können damit bereits Wertebereiche für Attribute und Elemente beschränkt werden, wie zum Beispiel die Beschränkung für die Gültigkeitsdauer eines Passwortes auf ganze Zahlen. Für ein besseres Verständnis wird im Verlauf dieses Kapitels das zugrundeliegende entwickelte XML Schema anhand eines Beispiels ausführlich erläutert. Ebenfalls ist das Dokument mit Hilfe des `documentation`-Elementes ausgiebig erläutert und kann in Anhang A.1 eingesehen werden.

Auflistung 2.1 zeigt das Grundgerüst eines XML-Dokumentes, das auf dem entwickelten XML Schema basiert. Das XML Schema schreibt für die XML-Dokumente als Wurzelelement das `policies`-Element vor. Das `policies`-Element darf beliebig viele `policy`-Kindelemente beinhalten. Bei dem `policy`-Element handelt es sich um eine Repräsentation einer einzelnen Passwort-Richtlinie. Da ein Anbieter mehrere verschiedene Dienste innerhalb einer Domain zur Verfügung stellen kann, die jeweils unterschiedliche Anforderungen haben, besteht die Möglichkeit, mehrere `policy`-Elemente als Kindelemente des `policies`-Elementes anzugeben.

Die Angabe des Attributes `xmlns:xs` im Wurzelelement des Dokumentes gibt an, dass ein verarbeitendes Programm das angegebene Dokument gegen ein XML Schema validieren soll. Mit der Angabe des Attributs `xsi:noNamespaceSchemaLocation` wird spezifiziert, gegen welches Schema das Dokument validiert werden soll. Dabei kann ein Uniform Resource Identifier (URI) [14] zu dem entsprechenden Schema angegeben werden. In der gezeigten Auflistung liegt das XML Schema im gleichen Verzeichnis wie das Dokument und wird daher mit einer relativen Pfadangabe referenziert.

```
<?xml version="1.0" encoding="UTF-8"?>
<policies xmlns:vc="http://www.w3.org/2007/XMLSchema—versioning"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema—instance"
    xsi:noNamespaceSchemaLocation="./password—policies.xsd">
        <policy>
            <!-- Passwort-Richtlinie -->
        </policy>
        <policy>
            <!-- Passwort-Richtlinie -->
        </policy>
        <!-- weitere Passwort-Richtlinien -->
</policies>
```

Auflistung 2.1: Grundaufbau eines XML-Dokumentes auf Basis der Spezifikation

2.2.1 Gültigkeitsbereich

Um angeben zu können, für welchen Dienst einer Domain eine definierte Richtlinie in Form eines `policy`-Elementes gültig ist, kann für die Richtlinie das `scope`-Attribut genutzt werden. Dieses gibt an, für welchen Pfad in einer URL die aktuelle Richtlinie gültig ist. Der Pfad wird dabei relativ zu der Domain, für die das XML-Dokument gültig ist, angegeben. Die Angabe des `scope`-Attributes ist optional und hat als Standard-Wert bei Nichtabgabe den Wert „/“.

Bei der Angabe des Attributes ist zwischen zwei Methoden zu unterscheiden:

- **Referenzierung einer Datei:**
  Wird mit der Angabe des `scope` eine Datei wie in Auflistung 2.2 referenziert, d. h. das Attribut endet nicht auf einen Slash, ist die angegebene Passwort-Richtlinie allein für die referenzierte Datei gültig. Dies kann insbesondere dann genutzt werden, wenn ein Dienst etwa durch den Einsatz von JavaScript nur über eine einzige URL, die auf eine Datei verweist, abrufbar ist.

```
<policy scope="/services/mail/singleFile.php">
        <!-- ... -->
</policy>
```

Auflistung 2.2: Gültigkeitsbereich einer Passwort-Richtlinie für eine Datei

- **Referenzierung eines Ordners:**
  Endet das `scope`-Attribut mit einem Slash, verweist also auf einen Ordner, ist die Passwort-Richtlinie für den referenzierten Ordner sowie alle Unterordner und darin enthaltenen Dateien gültig. Existiert in dem XML-Dokument eine weitere Richtlinie mit einer spezifischeren Pfadangabe, ist diese der allgemeineren Richtlinie vorzuziehen. Für einen Dienst auf `http://www.example.com/services/mail/` würde wie in Auflistung 2.3 gezeigt die erste Richtlinie verwendet werden:

```
<!-- Dokument-Beginn -->
<policy scope="/services/mail/">
      <!-- Passwort-Richtlinie mit hoeherer Prioritaet -->
</policy>
<policy scope="/services/">
      <!-- Weitere Passwort-Richtlinie -->
</policy>
<!-- Dokument-Ende -->
```

Auflistung 2.3: Vergleich `scope`-Attribut zweier Passwort-Richtlinien

Für das Auflösen wird somit folgender (Pseudo-)Algorithmus angewandt:

1. Entferne Query String und relative Referenz [14] aus gegebener URL.
2. Prüfe, ob eine Passwort-Richtlinie mit exakter `scope`-Übereinstimmung zu neuer URL existiert.
3. Bilde neue URL, die übergeordneten Ordner zu gegebener URL referenziert.
4. Falls URL leer: Terminierung ohne passende Richtlinie, nutze für gesamte Domain gültige Richtlinie.
5. Springe zu Schritt 2.

Im ersten Schritt wird aus der URL die Abfrage-Zeichenkette entfernt, die von Webanwendungen für die Auswertung von benannten Parametern genutzt wird. Anschließend wird geprüft, ob eine Passwort-Richtlinie existiert, deren Gültigkeitsbereich mit der URL übereinstimmt. Ist dies nicht der Fall, findet eine entsprechende Überprüfung für den übergeordneten Ordner der URL statt. Dies wird so lange wiederholt, bis eine Richtlinie gefunden wurde oder das Wurzelverzeichnis erreicht ist. Wird schlussendlich keine Passwort-Richtlinie gefunden, wird die Standard-Richtlinie verwendet.

2.2.2 Zeichensätze

Wie sich in der Anforderungsanalyse gezeigt hat (siehe Kapitel 2.1), erlauben Dienstanbieter unterschiedliche Zeichen bei der Wahl von Passwörtern. Es muss daher möglich sein, die erlaubten Zeichen eines Passwortes in der Passwort-Richtlinie anzugeben. Für diesen Zweck kann im XML-Dokument als Kindelement des `policy`-Elementes das `characterSets`-Element angegeben werden.

```
<policy scope="/">
        <characterSets>
                <!-- ... -->
        </characterSets>
        <!-- ... -->
</policy>
```

Auflistung 2.4: `characterSets`-Element

Innerhalb dieses Elementes können beliebig viele, mindestens jedoch ein, für die Passwortwahl verfügbarer Zeichensatz angegeben werden, wie Auflistung 2.5 zeigt. Durch die in Kapitel 2 beschriebene Möglichkeit, mehrere Zeichensätze anstelle eines großen Zeichensatzes anzugeben, können komplexere Anforderungen an das Passwort gestellt werden. Die Definition der einzelnen Zeichensätze erfolgt dabei über `characterSet`-Elemente, welche Kindelemente des `characterSets`-Elementes sind. Die Festlegung eines eindeutigen Namens durch das `name`-Attribut ist dabei zwingend erforderlich, da die definierten Zeichensätze im weiteren Verlauf der Definition der Passwort-Richtlinie referenziert werden.

```
<characterSets>
        <characterSet name="firstCharacterSet">
                <!-- ... -->
        </characterSet>
        <characterSet name="secondCharacterSet">
                <!-- ... -->
        </characterSet>
</characterSets>
```

Auflistung 2.5: `characterSet`-Element

Bei der Definition besteht die Möglichkeit, die erlaubten Zeichen anzugeben oder einen neuen Zeichensatz auf Basis von einem oder mehreren vorher definierten Zeichensätzen zu erstellen. Eine Kombination der zwei Methoden, d. h. die Erweiterung bestehender Zeichensätze und die Angabe zusätzlich erlaubter Zeichen, ist ebenfalls vorgesehen.

```
<characterSets>
        <characterSet name="firstCharacterSet">
                <characters>0123456789</characters>
        </characterSet>
        <characterSet name="secondCharacterSet">
                <base characterSet="firstCharacterSet" />
                <characters>ABCDEF</characters>
        </characterSet>
        <!-- ... -->
</characterSets>
```

Auflistung 2.6: Definition von Zeichensätzen

In der beispielhaften Definition aus Auflistung 2.6 existieren die zwei Zeichensätze `firstCharacterSet` und `secondCharacterSet`. Im ersten Zeichensatz wurden die verfügbaren Zeichen explizit durch das `characters`-Elementes angegeben, sodass dieser aus der Menge der Zeichen $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ besteht. Der zweite Zeichensatz besteht durch die Nutzung des `base`-Elementes und der Angabe zusätzlicher Zeichen aus der Menge $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$.

2.2.3 Passwort-Eigenschaften

Anforderungen an die Eigenschaften der Passwörter können im `properties`-Element einer Passwort-Richtlinie festgelegt werden. Das Element befindet sich im `policy`-Element als direkter Nachfolger des `characterSets`-Elementes (siehe Auflistung 2.7).

```
<policy scope="/">
        <!-- ... -->
        </characterSets>
        <properties>
                <!-- ... -->
        </properties>
        <!-- ... -->
</policy>
```

Auflistung 2.7: `properties`-Element

Die über die verfügbaren Kindelemente des `properties`-Elementes (siehe Auflistung 2.8) zu definierenden Eigenschaften des Passwortes werden im Folgenden erläutert:

```
<properties>
        <minLength>5</minLength>
        <maxLength>50</maxLength>
        <expires>72</expires>
        <characterSettings>
                <!-- ... -->
        </characterSettings>
</properties>
```

Auflistung 2.8: Kindelemente des `properties`-Elementes

Die minimale bzw. maximale Länge eines Passwortes lässt sich über das `minLength`- bzw. `maxLength`-Element definieren. Die Gültigkeitsdauer eines Passwortes in Tagen kann durch Angabe einer nichtnegativen ganzen Zahl im `expires`-Element angegeben werden. Ein Wert von 0 bzw. ein Weglassen des Elementes wird als nicht ablaufendes Passwort interpretiert. Die Anforderungen an die Zeichen werden im folgenden Kapitel 2.2.4 erläutert.

2.2.4 Zeichensatz-Anforderungen

Um Anforderungen bezüglich der möglichen Zeichen, wie etwa ein minimales Vorkommen bestimmter Zeichensätze, abzubilden, werden im `characterSettings`-Element die vorher definierten Zeichensätze referenziert.

Zeichensätze, die im gesamten Passwort verwendet werden dürfen, werden durch das Kindelement `availableCharacterSet` mit Angabe des `characterSet`-Attributes referenziert. Jeder referenzierte Zeichensatz muss zuvor im `characterSets`-Element entsprechend definiert worden sein, wie in Kapitel 2.2.2 erläutert wurde. Die Attribute `minQuantity` und `maxQuantity` geben für den jeweiligen Zeichensatz ein minimales bzw. maximales Vorkommen im Passwort an.

```
<characterSettings>
        <availableCharacterSet characterSet="firstCharacterSet" minQuantity="1"
            maxQuantity="5" />
        <availableCharacterSet characterSet="secondCharacterSet" minQuantity="1"
            maxQuantity="10" />
</characterSettings>
```

Auflistung 2.9: Beispielhafte Definition für die Zeichensatz-Anforderungen

Das Beispiel aus Auflistung 2.9 schreibt somit vor, dass die Zeichensätze `firstCharacterSet` und `secondCharacterSet` jeweils mindestens ein Mal und maximal fünf bzw. zehn Mal vorkommen dürfen. Da die Passwörter in den meisten Fällen keine feste Länge haben, sondern durch Angabe der `minLength` und `maxLength` (vgl. Auflistung 2.8) verschiedene Passwortlängen zulassen, können für die `minQuantity`- bzw. `maxQuantity`-Attribute nicht nur ganzzahlige, sondern auch reelle Werte im Bereich $(0, 1)$ angegeben werden. Dadurch wird ein Vorkommen relativ zur tatsächlichen Passwortlänge angegeben. Auflistung 2.10 zeigt eine beispielhafte Definition, bei der das `firstCharacterSet` mindestens 50 Prozent aller Zeichen im Passwort einnehmen muss. Ein Auslassen der Attribute oder die Angabe von 0 sagt aus, dass keine vorhandenen Beschränkungen für das minimale oder maximale Vorkommen eines Zeichensatzes existieren.

```
<characterSettings>
        <availableCharacterSet characterSet="firstCharacterSet" minQuantity="0.5" />
        <availableCharacterSet characterSet="secondCharacterSet" />
</characterSettings>
```

Auflistung 2.10: Relative Angabe für minimales Vorkommen eines Zeichensatzes

Neben dem `availableCharacterSet`-Element besteht die Möglichkeit, die genannten Einschränkungen für Zeichensätze nur für angegebene Zeichenpositionen im Passwort vorzunehmen. Im `rescrictions`-Element kann dafür für jeden zu konfigurierenden Zeichensatz ein `restriction`-Element angegeben werden.

```
<characterSettings>
        <availableCharacterSet characterSet="firstCharacterSet" minQuantity="0.5" />
        <availableCharacterSet characterSet="secondCharacterSet" />
        <restrictions>
                <restriction characterSet="secondCharacterSet" position="3" />
        </restrictions>
</characterSettings>
```

Auflistung 2.11: `characterSettings` mit `restrictions`

Ist für eine Zeichenposition mindestens eine `restriction` angegeben, gelten für keine der in den `availableCharacterSet`-Elementen angegebenen Eigenschaften für Zeichensätze. Dies bedeutet, dass, wie in Auflistung 2.11 zu sehen, für die Position 3 des zu generierenden Passwortes nur das `secondCharacterSet` verwendet werden darf und nicht das `firstCharacterSet`, obwohl dieses im `availableCharacterSet`-Element der Passwort-Richtlinie als erlaubter Zeichensatz angegeben wurde. Gleichermaßen wie bei der Angabe der `availableCharacterSet`-Elementen, können mehrere `restriction`-Elemente für eine Zeichenposition angegeben, um weitere Zeichensätze zuzulassen. Die Positionsangaben werden dazu als Komma-separierte Liste im `position`-Attribut des angegeben. Zu beachten ist, dass die Angabe der Positionen das erste Zeichen mit Position 0 referenziert wird.

Für die Angabe einer Position gibt es dabei drei Möglichkeiten:

- **Angabe als nicht-negative ganze Zahl:**
  Angabe als tatsächliche Position, beginnend mit Position 0 für das erste Zeichen.
- **Angabe als negative ganze Zahl:**
  Positionsangabe vom Ende des Passwortes beginnend. Die Angabe von -1 referenziert somit das letzte Zeichen des Passwortes. Die Zeichenposition kann berechnet werden als $i = l + p$, wobei l der Länge des generierten Passwortes entspricht und p die Angabe der Position aus dem `position`-Attribut ist.
- **Relative Positionsangabe:**
  Wie bei der Angabe des `minQuantity` bzw. `maxQuantity`-Attributes für Zeichensätze (vgl. Kapitel 2.2.2) kann eine Positionsangabe ebenfalls als reeller Wert im Intervall $(0, 1)$ geschehen. Die tatsächliche Position berechnet sich dann als $i = round((l-1) * p)$

Die verschiedenen Möglichkeiten zur Angabe von Zeichenpositionen können dabei gemischt verwendet werden. Ebenfalls ist es möglich, wie im `availableCharacterSet`-Element (vgl. Auflistung 2.9) durch die Angabe einer `minQuantity` bzw. `maxQuantity` die minimale und maximale Anzahl an Vorkommnissen des im `restriction`-Element angegebenen Zeichensatzes für die angegebenen Zeichenpositionen zu steuern.

```
<characterSettings>
        <availableCharacterSet characterSet="firstCharacterSet" minQuantity="0.5" />
        <availableCharacterSet characterSet="secondCharacterSet" />
        <restrictions>
                <restriction characterSet="firstCharacterSet" position="0,-1,0.5" />
                <restriction characterSet="secondCharacterSet" position="0.5,1,2"
                    minQuantity="2" />
        </restrictions>
</characterSettings>
```

Auflistung 2.12: Vollständige Definition eines `characterSettings`-Elementes

Auflistung 2.12 zeigt eine beispielhafte Definition eines vollständigen `characterSettings`-Elementes mit unterschiedlichen Referenzierungen der Zeichenpositionen. Sie schreibt vor, dass das Passwort sowohl aus dem `firstCharacterSet`, als auch dem `secondCharacterSet` bestehen darf. Mindestens die Hälfte der Zeichen des gesamten Passwortes muss aus dem `firstCharacterSet` stammen. Das erste und letzte Zeichen des Passwortes darf nur aus dem `firstCharacterSet` bestehen. Das zweite und dritte Zeichen darf nur mit Zeichen des `secondCharacterSet` gefüllt werden. Das mittlere Zeichen des Passwortes wird von beiden `restriction`-Elementen referenziert, und darf damit, wie auch die restlichen Zeichen des Passwortes, Zeichen aus beiden Zeichensätzen enthalten. Es gilt allerdings, dass in der

Gruppe bestehend aus dem ersten, letzten und mittleren Zeichen des Passwortes mindestens zwei dieser Zeichen aus dem `secondCharacterSet` stammen müssen, was durch die zweite `restriction` bewirkt wird.

Durch die in diesem Kapitel erläuterten Möglichkeiten, Anforderungen an die Zeichen in einem Passwort auszudrücken, können selbst sehr komplexe Anforderungen an diese vorgegeben werden. Die Angabe von relativen Positionen erlaubt es zudem, die Anforderungen in Abhängigkeit der Passwortlänge zu stellen. So ist es beispielsweise möglich, eine Verwendung von Sonderzeichen für mindestens zehn Prozent aller Zeichen in einem Passwort zu fordern.

2.2.5 Dienst-Informationen

Weitere Informationen über den Dienst werden im `service`-Element abgelegt, welches im `policy`-Element nach dem `characterSettings`-Element anzugeben ist, wie in Auflistung 2.13 veranschaulicht. Die in diesem Element angegebenen Informationen können genutzt werden, um dem Nutzer Daten über den jeweiligen Dienst zur Verfügung zu stellen. Dies ermöglicht ein schnelles Wiederfinden gängiger Funktionen zur Verwaltung eines Nutzerkontos bei einem Dienst.

```
<policy scope="/">
        <!-- ... -->
        </characterSettings>
        <service>
                <!-- ... -->
        </service>
</policy>
```

Auflistung 2.13: `service`-Element

Die verfügbaren Kindelemente des `service`-Elementes werden im Folgenden erläutert:

- **registerURL:**
  URL zur Registrierungs-Funktion des Dienstes.
- **passwordForgottenURL:**
  URL zur „Passwort ändern“-Funktion des jeweiligen Dienstes.
- **passwordChangeURL:**
  URL zur „Passwort vergessen“-Funktion des Dienstes.
- **passwordMaxRetries:**
  Maximale Anzahl an fehlgeschlagenen Login-Versuchen, bevor ein Benutzerkonto gesperrt wird. Angabe als nicht-negative ganze Zahl. Eine Angabe von 0 bedeutet eine unbegrenzte Anzahl an Versuchen.

Auflistung 2.14 zeigt eine beispielhafte Definition eines `service`-Elementes, in dem alle zur Verfügung stehenden Kindelemente angegeben wurden.

```
<service>
        <registerURL>
                http://www.example.com/services/mail/register
        </registerURL>
        <passwordChangeURL>
                http://www.example.com/services/mail/myProfile
        </passwordChangeURL>
        <passwordForgottenURL>
                http://www.example.com/services/mail/forgotPassword
        </passwordForgottenURL>
        <passwordMaxRetries>3</passwordMaxRetries>
</service>
```

Auflistung 2.14: Beispielhafte Definition für Service-Informationen

2.2.6 Vollständiges Beispiel

Als abschließende Übersicht zeigt Anhang B.1 das vollständige in den vorherigen Kapiteln entwickelte Beispiel. Zusätzlich zu dieser Passwort-Richtlinie zeigt Anhang B.2 eine minimale Richtlinie bestehend aus einem einzigen Zeichensatz. In dieser Richtlinie wurden alle Elemente und Attribute des XML-Dokumentes ausgelassen, die in dem XML Schema nicht durch das `required`-Attribut vorausgesetzt sind.

2.3 Erstellung von Passwort-Richtlinien

Um Passwort-Richtlinien im Internet effektiv nutzen zu können, müssen diese für neue sowie bereits bestehende Dienste erstellt werden. Die Erstellung der Passwort-Richtlinien kann dabei auf verschiedene Arten erfolgen. Als einen ersten Ansatz bietet sich dabei die Erstellung der Richtlinien durch die Anbieter der Dienste selbst an. Der Anbieter arbeitet dabei die Richtlinien für alle auf einer Domain befindlichen Dienste aus und stellt diese gemeinsam in einem XML-Dokument auf Basis des vorgestellten Standards zur Verfügung. Auf diese Weise hat der Anbieter die Kontrolle über die Richtlinien und kann sie mit den Anforderungen seiner Dienste bei Aktualisierungen abgleichen.

Eine weitere Möglichkeit ist die Erstellung der Passwort-Richtlinien durch eine dritte Partei. Dabei müssen die Richtlinien nicht durch den Anbieter selbst erstellt werden, sondern können von externen Anbietern oder Nutzern durch andere Methoden ausgearbeitet werden. Als Ansatz hierfür bietet sich ein automatisierter Webcrawler [8] an, der die Passwort-Anforderungen extrahiert. Der Webcraw-

ler durchsucht das Internet nach möglichen Diensten, welche die Registrierung eines Nutzers bzw. die Verwendung eines Passwortes erfordern. Wird ein solcher Dienst gefunden, kann dieser automatisiert die vom Anbieter in Textform kommunizierten Anforderungen an das Passwort auslesen, um sie in einer Passwort-Richtlinie abzubilden. Da die Passwort-Anforderungen von den Anbietern oftmals nicht oder nicht eindeutig an den Nutzer kommuniziert werden, sondern beispielsweise erst bei einer Fehleingabe ein entsprechender Hinweis ausgegeben wird, muss der Webcrawler hierfür unter Umständen eigenständig versuchen, eine Registrierung bei einem Dienst vorzunehmen. Eine Hürde stellt dabei das Auslesen und Interpretieren der in Textform bereitgestellten Informationen dar. Diese sind bei verschiedenen Diensten in unterschiedlichen Sprachen vorhanden und müssen mit möglichst hoher Präzision verarbeitet werden.

Als Alternative zu einem Webcrawler können in einem gemeinschaftsbasiertem (eng. „community-based“) Modell Passwort-Richtlinien erstellt werden, die von einem oder mehreren Benutzern erstellt und überprüft werden. Durch die redundante Erstellung beziehungsweise Überprüfung der Richtlinien von mehreren Benutzern kann die Korrektheit dieser sichergestellt werden. Hierfür muss eine breite Nutzerbasis bestehen, um die Richtlinien entsprechend vieler Dienste im Internet abbilden zu können. Ebenfalls wird diese Nutzerbasis benötigt, um eine Korrektheit der Richtlinien sicherzustellen, da ansonsten falsche Richtlinien verbreitet werden können, die möglicherweise Sicherheitsrisiken für deren Nutzer bergen, aufgrund der Tatsache, dass diese Richtlinien beim Generieren von Passwörtern möglichst schwache Passwörter erzwingen.

2.4 Distribution

Um die vorhandenen Passwort-Richtlinien für Nutzer und Anwendungen zugänglich zu machen, ist ein Konzept zur Distribution der Passwort-Richtlinien notwendig. Die Richtlinien können dabei, abhängig von der Art der Distribution, über eine „well-known location“ gemäß RFC 5785 [17] oder eine Schnittstelle, wie etwa der Web Services Description Language (WSDL) [5] oder SOAP [3], ausgetauscht werden. Im Folgenden werden zwei Methoden für die zentrale und dezentrale Distribution erläutert. Bei der zentralen Distribution stellen ein oder mehrere vertrauenswürdige Richtlinien-Dienste die Passwort-Richtlinien für Dienste im Internet bereit. Im dezentralen Ansatz stellen die Dienst-Anbieter die Richtlinien für die eigenen Dienste selbst bereit, wodurch keine Notwendigkeit einer dritten Partei besteht.

2.4.1 Zentrale Distribution

Bei der zentralen Distribution werden die Daten von einem bekannten vertrauenswürdigen Richtlinien-Dienst über eine Schnittstelle bereitgestellt. Wie in Abbildung 2.3 dargestellt ist, werden dabei Anfragen unabhängig von der vom Nutzer besuchten Domain an den selben Dienst gestellt.

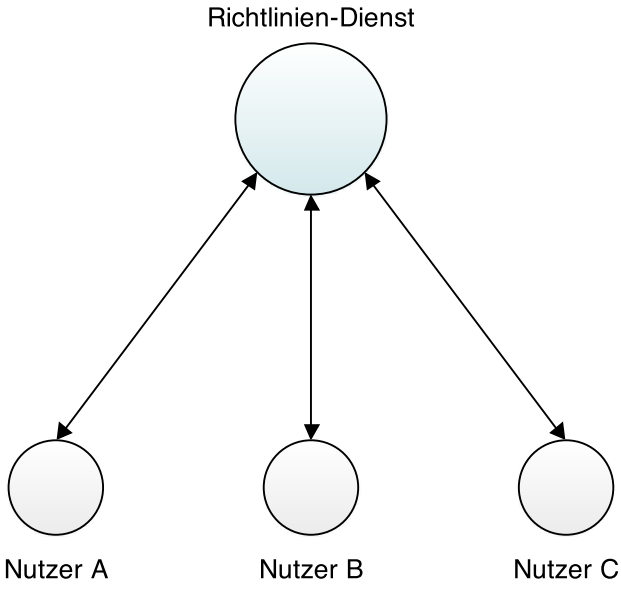

Abbildung 2.3.: Netzstruktur für zentrale Distribution

Wird bei der Kommunikation mit einem Dienst im Internet eine Richtlinie benötigt, wird diese nicht vom Anbieter des Dienstes angefragt, sondern dem Richtlinien-Dienst stattdessen die aufgerufene Domain oder URL übermittelt, sodass dieser die passende Richtlinie an den Nutzer übermitteln kann.

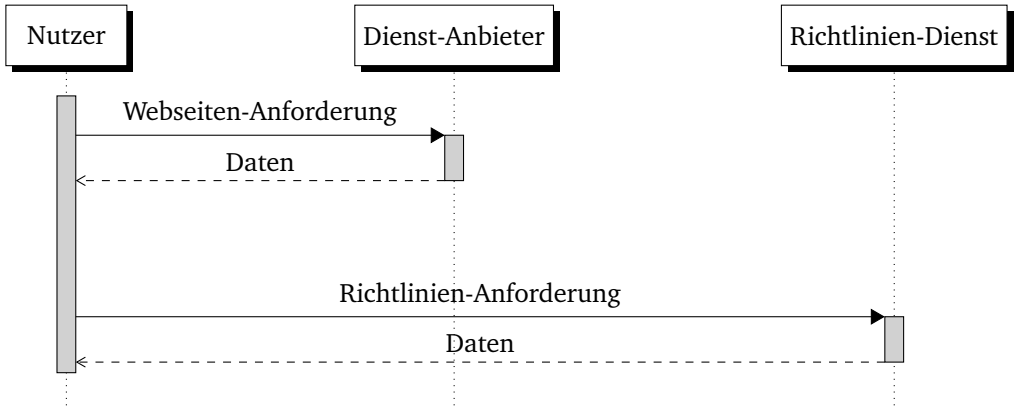

Abbildung 2.4.: Sequenzdiagramm zentralisierte Distribution

Abbildung 2.4 zeigt die Kommunikation zwischen dem Nutzer, dem Anbieter des Dienstes sowie dem Richtlinien-Dienst. Im ersten Schritt geschieht die Kommunikation zwischen Nutzer und Dienst-Anbieter. Dabei handelt es sich um die gewohnte Kommunikation beim Abruf einer URL über HTTP [6] oder HTTPS. Wird für den aufgerufenen Dienst eine Passwort-Richtlinie benötigt, wird in einem zweiten Schritt dem Richtlinien-Dienst die URL des aufgerufenen Dienstes übermittelt und die entsprechende Richtlinie angefragt. Die abgerufenen Richtlinien können dabei auf verschiedenen Wegen erstellt worden sein, wie in Kapitel 2.3 beschrieben. Werden die Richtlinien von den Anbietern der jeweiligen Dienste

selbst erstellt, muss der Richtlinien-Dienst eine Schnittstelle bereitstellen, damit die Anbieter ihre Richtlinien registrieren und aktualisieren können. Dabei muss sichergestellt werden, dass die Richtlinien nicht von unautorisierten Personen eingestellt oder bearbeitet werden können, da dies zu erheblichen Sicherheitsrisiken führen kann. Ist eine unautorisierte Person imstande, Richtlinien für einen Dienst in dem zentralen System zu registrieren bzw. die Anforderungen dieser zu verändern, so kann sie etwa durch Veränderung der verfügbaren Zeichensätze erzielen, dass Passwort-Generatoren nur noch schwache Passwörter generieren. Durch die Einschränkung von Zeichensätzen auf bestimmten Positionen mit Hilfe der `restrictions` (vgl. Kapitel 2.2) kann erreicht werden, dass Passwörter bei der Generierung scheinbar zufällig aussehen, sich allerdings bei erneuten Generierungen nicht unterscheiden, sodass jeder Nutzer das gleiche Passwort erhält. Die Überprüfung der Authentizität der Richtlinien ist bei einer zentralen Distribution somit essenziell. Nicht nur die Überprüfung der Richtlinien beim Eintragen über die bereitgestellten Schnittstellen ist dabei von Bedeutung, sondern auch die Sicherheit des Gesamtsystems. Kann das System kompromittiert werden oder ein Angreifer falsche Richtlinien, etwa durch DNS-Spoofing [10] oder einen „man in the middle“-Angriff [18], übermitteln, birgt dies die selben Sicherheitsrisiken.

Für eine zentrale Distribution spricht, dass hierbei nicht die Mitarbeit der Dienst-Anbieter erforderlich ist, da die Richtlinien von Außenstehenden anhand der tatsächlichen Anforderungen an die Passwörter erstellt werden können. Anbieter von Programmen zur Passwortverwaltung können beispielsweise die Adresse eines eigenen Richtlinien-Dienstes in ihre Programme integrieren, sodass die Passwort-Generatoren automatisiert von diesem Dienst die Anforderungen an die Passwörter abrufen. Der Abruf der Richtlinien kann dabei entweder als einzelne Richtlinie für einen bestimmten Dienst oder als ganzes XML-Dokument für eine Domain geschehen, sodass die verarbeitenden Programme die passende Richtlinie auswählen müssen.

Ein Ausfall des Systems würde bedeuten, dass die Verfügbarkeit der Passwort-Richtlinien für alle Dienste ausfällt, da sie von einem einzelnen Anbieter zur Verfügung gestellt werden und somit ein „Single Point of Failure“ besteht. Ebenfalls wird durch diese Art der Distribution das Anfertigen von Bewegungsprofilen der Nutzer möglich, da die Nutzer die aufgerufenen URLs oder zumindest die Domains übermitteln müssen.

2.4.2 Dezentrale Distribution

Die dezentrale Distribution ermöglicht es, dass die Richtlinien auf der selben Domain abgelegt werden können, auf der ein Dienst zur Verfügung gestellt wird. Ähnlich wie bei dem Robots Exclusion Standard [12] kann das XML-Dokument über eine „well-known location“ abgefragt werden, in welcher Richtlinien für alle Dienste der aktuell besuchten Domain enthalten sind. Eine „well-known location“ definiert, unter welcher URL bestimmte Informationen für eine gegebene Domain gesucht werden. Für Passwort-Richtlinien kann dies beispielsweise unter dem relativen Pfad `./.well-known/password-policies.xml` geschehen. Für die Domain `example.com` würden die Richtlinien beispielsweise unter `example.com/.well-known/password-policies.xml` gesucht werden.

Die Dienst-Anbieter erstellen bei der dezentralen Distribution die für ihre Dienste gültigen Passwort-Richtlinien selbst. Wie in Abbildung 2.5 veranschaulicht, kommunizieren Nutzer so nicht mit einem einzigen Richtlinien-Dienst, sondern mit dem gleichen Anbieter, der den Dienst bereitstellt.

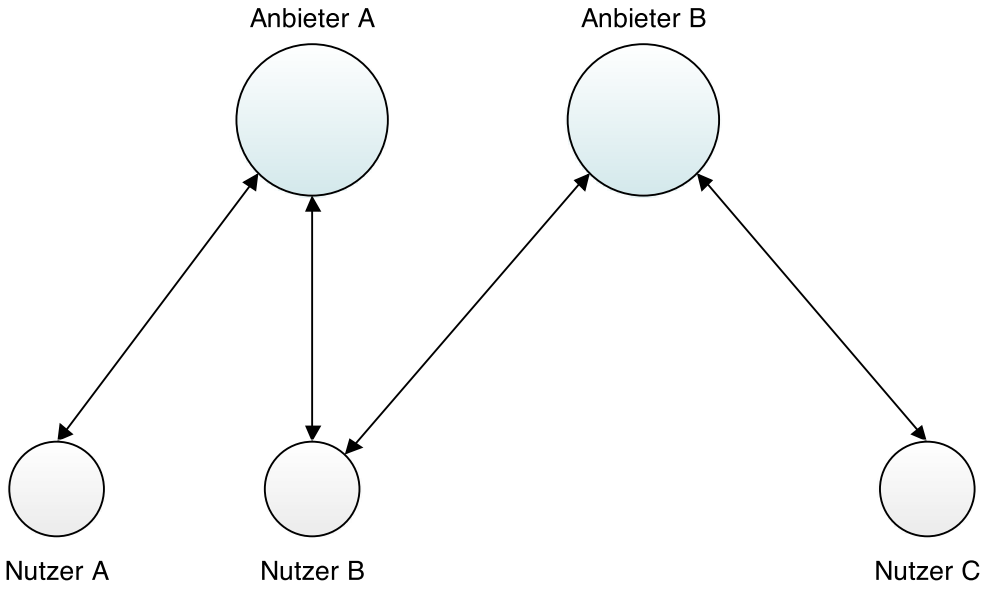

Abbildung 2.5.: Netzstruktur für dezentrale Distribution

Da in dem Dokument alle Richtlinien für die gesamte Domain enthalten sind, kann die Datei nach dem Abrufen für einen festgelegten Zeitraum zwischengespeichert werden, sodass ein erneutes Abrufen nicht bei jedem Seitenaufruf nötig wird. Dadurch wird die Last für den bereitstellenden Anbieter minimiert. Abbildung 2.6 zeigt den Ablauf für das Aufrufen einer Webseite eines Dienstes. Dabei wird zuerst die URL von dem Anbieter, der den Dienst bereitstellt, abgerufen. Nachdem die Webseite an den Nutzer übertragen wurde, werden vom selben Anbieter die Passwort-Richtlinien abgefragt, sollten diese nicht bereits zwischengespeichert sein.

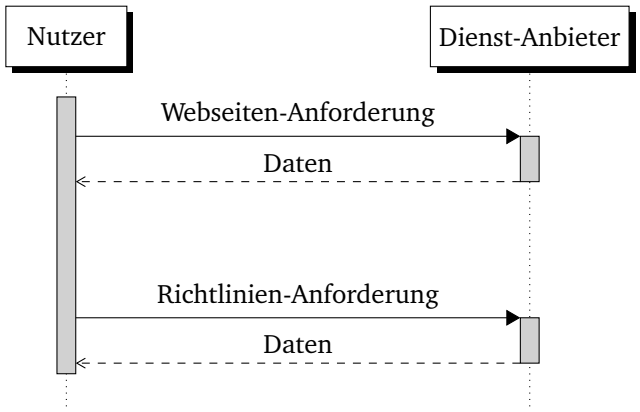

Abbildung 2.6.: Sequenzdiagramm dezentralisierte Distribution

Für die dezentrale Distribution muss somit keine neue Infrastruktur aufgebaut werden, denn es können die bestehenden Strukturen genutzt werden. Ebenfalls gibt es keinen Anbieter, der durch die Bereitstellung von Passwort-Richtlinien Nutzerprofile erstellen kann. Da der Abruf der Richtlinien von dem gleichen Anbieter erfolgt, auf dem die Webseite abgerufen wurde, gewinnt der Anbieter keine weiteren Informationen über den Nutzer.

Ebenfalls erleichtert die dezentrale Distribution die Überprüfung der Authentizität der Passwort-Richtlinien. Dadurch, dass sowohl Webseite als auch Richtlinien vom selben Anbieter abgerufen werden, kann die Überprüfung der Authentizität der Richtlinien mit der Überprüfung der Identität des Anbieters geschehen, sodass keine separate Implementation zur Überprüfung der Richtlinien erforderlich ist. Im Internet wird die Authentifizierung und Identifikation der Kommunikationspartner in der Regel mit der Transport Layer Security (TSL) realisiert. Mit Hilfe des TLS Handshake Protocol [2] authentifiziert sich der Anbieter dabei durch asymmetrische Verschlüsselungsverfahren und Public-Key-Kryptographie bei dem Nutzer. Durch die Übermittlung eines X-509-v3-Zertifikats [20] kann der Nutzer die Identität des Anbieters überprüfen. Der Einsatz von TLS bei der Kommunikation im World Wide Web hängt dabei davon ab, ob HTTPS zur Kommunikation mit dem Anbieter verwendet wird. In Bereichen, in denen sicherheitskritische Daten wie Nutzerpasswörter oder -Daten übermittelt werden, ist dies üblicherweise der Fall. Für den Abruf der Passwort-Richtlinien sollte daher wenn möglich ebenfalls HTTPS genutzt werden.

Eine Kompromittierung eines Dienstes ist ebenso als weniger kritisch anzusehen als die eines zentralen Richtlinien-Dienstes: Zum einen, da nur ein einziger Dienst davon betroffen ist, das heißt, dass auch nur für diesen Dienst die Richtlinien verändert werden können. Zum anderen können bei einer Kompromittierung andere, sicherheitskritischere Veränderungen am Dienst vorgenommen werden, sodass dieser Aspekt zu vernachlässigen ist.

2.4.3 Vergleich

Der Vorteil der zentralen Distribution besteht darin, dass sie nicht von der Akzeptanz der Dienst-Anbieter abhängt. Sie kann ohne Zutun der Anbieter erfolgen, da prinzipiell jeder die Anforderungen an Passwörter in einer Datenbank sammeln und zur Verfügung stellen kann. Während bei der dezentralen Distribution das Engagement der Dienst-Anbieter vonnöten ist, kann bei einem zentralen Ansatz jeder einen solchen Dienst bereitstellen und in seine Anwendung, wie etwa einem Programm zur Passwortverwaltung, integrieren. Die Implementation der Schnittstellen für den Abruf der Richtlinien bleibt dadurch ebenfalls jedem Anbieter eines solchen Dienstes überlassen.

Diesem Vorteil gegenüber stellt sich die Frage nach der Umsetzbarkeit eines solchen Dienstes in einem größeren Maßstab. Geht man von einem bei den Nutzern etablierten Dienst für den Abruf von Passwort-Richtlinien aus, muss eine entsprechend starke Infrastruktur aufgebaut werden, die eine große Menge an Anfragen bewältigen kann. Da die Richtlinien idealerweise nicht erst abgerufen werden, wenn sie benötigt werden, um eine Verzögerung etwa bei der Registrierung eines Nutzerkontos zu vermeiden, werden somit viele Anfragen an den Dienst gestellt. Im Falle einer Implementation eines Caches, bei der die jeweiligen Richtlinien pro Benutzer und Domain nur ein einziges Mal am Tag abgerufen werden, stellt dies alleine für die Domain facebook.com bei einer täglichen Nutzerzahl von 829 Millionen[1] eine große infrastrukturelle Herausforderung dar. Weiterhin müsste ein Weg gefunden werden, die entstehende Infrastruktur zu finanzieren oder die Kosten für Betrieb und Wartung dieser an die Dienstanbieter weiterzugeben, was in der Praxis schwer umsetzbar sein wird. Neben der infrastrukturellen Herausforderung besteht ebenfalls das Hindernis, die Passwort-Richtlinien ohne Hilfe der Dienst-Anbieter auf einem aktuellen Stand zu halten. Sind die Richtlinien nicht konform mit den tatsächlichen Anforderungen des Dienstes, können dadurch unter Umständen nicht akzeptierte Passwörter generiert werden.

Bei der dezentralen Distribution bestehen diese Hindernisse nicht. Die Richtlinien werden vom Anbieter selbst zur Verfügung gestellt, welcher sie bei der Neuerstellung oder Aktualisierung eines Dienstes auf dem aktuellen Stand halten kann. Die Richtlinien müssen somit nicht manuell durch Dritte oder durch möglicherweise fehleranfällige automatisierte Prozesse gesammelt werden. Ebenfalls bestehen hier nicht die Sicherheitsprobleme, wie es bei der zentralen Distribution der Fall ist. Um eine sinnvolle Nutzung der Richtlinien über die dezentrale Distribution zu ermöglichen, ist es jedoch essentiell, eine breite Akzeptanz der Dienst-Anbieter zu haben, um eine ausreichende Abdeckung zu ermöglichen. Ist diese Akzeptanz gegeben, lässt sich festhalten, dass in diesem Fall eine dezentrale Distribution einer zentralen Lösung vorzuziehen ist.

[1] `http://newsroom.fb.com/company-info/`, Stand Juni 2014

3 Prototyp

Nachdem die Anforderungen an das Dateiformat für Passwort-Richtlinien analysiert, festgelegt und ein entsprechendes XML Schema als Grundlage für XML-Dokumente erstellt wurde, wurde ein Prototyp entwickelt, um die Durchführbarkeit für die Nutzung der Passwort-Richtlinien zu zeigen. Der Prototyp stellt dabei einen Passwort-Generator zur Verfügung, der vom Benutzer nicht mehr eigenhändig konfiguriert werden muss. Stattdessen können allein durch die Eingabe einer URL des Dienstes, für den ein Passwort erstellt werden soll, gültige Passwörter generiert werden. Der Prototyp ist als Plug-In für die Passwortverwaltung KeePass [19] implementiert. Dadurch kann er in der aktuellen Version der KeePass Professional Edition eingesetzt werden, sodass er auch mit bereits bestehenden Datenbanken der Passwortverwaltung kompatibel ist.

Im Folgenden wird die Plattform vorgestellt, die für die Entwicklung des Prototyps genutzt wurde, sowie ihre Funktion erläutert. Ebenfalls wird auf die Entwicklungsumgebung und die verwendete Programmiersprache eingegangen.

3.1 KeePass

Als Zielplattform für den Prototyp wurde KeePass gewählt. Dabei handelt es sich um ein weit verbreitetes Programm zur Passwortverwaltung [21], welches unter der GNU General Public License (GPL) von Dominik Reichl veröffentlicht ist. Mit dem Programm ist es möglich, Passwörter inklusive weiterer Informationen wie Benutzername, URL und Ablaufdatum sicher abzuspeichern.

Die Datenbank der Passworteinträge kann dabei standardmäßig mit Twofish oder dem Advanced Encryption Standard (AES) verschlüsselt werden. AES gilt als ein sicheres Verschlüsselungsverfahren, das ebenfalls von Regierungen zum Schutz von Dokumenten mit Geheimhaltungsstufe eingesetzt wird [9]. Die Verschlüsselung der Datenbank beschränkt sich dabei nicht nur auf die Passwörter, sondern auch alle anderen Informationen, die mit diesen abgelegt sind, werden verschlüsselt. Mit einem gewählten Master-Passwort kann die Datenbank bei Programmstart geöffnet werden. Alternativ zu einem Master-Passwort bietet KeePass die Möglichkeit, ein „Key File“ (Schlüsseldatei) zu verwenden, um die Sicherheit der Datenbank zu erhöhen. Ebenfalls kann ein Nutzer diese zwei Methoden kombinieren oder die Datenbank durch die Login-Informationen des Windows-Nutzerkontos schützen. Durch einen integrieren Speicherschutz sind die Passwörter auch im laufenden Betrieb des Programmes geschützt. Dieser verhindert, dass die geöffnete Datenbank von externen Programmen aus dem Arbeitsspeicher ausgelesen werden kann. Der Schutz beschränkt sich dabei nicht nur auf die Datenbank, sondern auch auf andere Bereiche, in denen die Passwörter angezeigt oder verarbeitet werden, wie beispielsweise bei dem Bearbeiten eines bestehenden Passwort-Eintrages. Im Standardumfang unterstützt das Programm sowohl den

Import als auch den Export der Datenbank in gängige Dateiformate, um etwa Excel-Listen zu erstellen oder Passwörter aus anderen Programmen zur Passwortverwaltung zu importieren. Gespeicherte Login-Daten können durch die „Auto-Type“-Funktion mittels einer Tastenkombination direkt in Anwendungen wie einem Browser eingetragen werden. Durch die mögliche Kategorisierung der Login-Informationen sollen die gesuchten Informationen schneller gefunden werden. Ebenfalls bietet das Programm dem Nutzer die Möglichkeit, nach gespeicherten Einträgen anhand verschiedener Attribute, z. B. dem Titel, zu suchen (siehe Abbildung 3.1).

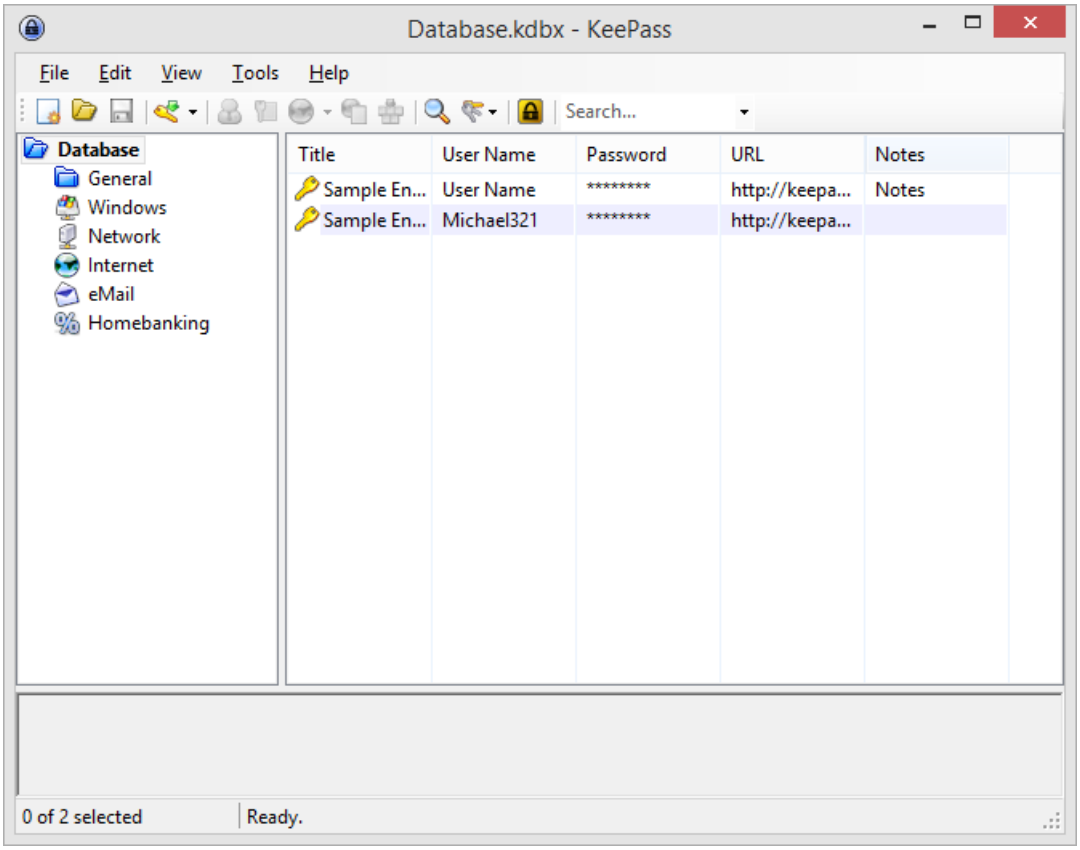

Abbildung 3.1.: Benutzeroberfläche von KeePass

KeePass ist in der Programmiersprache C# geschrieben und bietet die Möglichkeit, den Funktionsumfang durch Erweiterungen zu ergänzen. Erweiterungen können so beispielsweise andere Verschlüsselungsverfahren für die Datenbank sowie Import- und Export-Funktionen für andere Dateiformate oder auch Passwort-Generatoren bereitstellen. Sie müssen dafür ein vorgegebenes „Managed Interface“ zur Verfügung stellen, um mit dem Hauptprogramm interagieren zu können. Managed Interfaces sind Programmschnittstellen, welche in Form von „Managed Code“ [15] vorliegen. Unter Managed Code versteht man Quellcode, welcher zur Ausführung eine „Common Language Runtime“, die Laufzeitumgebung des .NET Frameworks, benötigt. Quellcode für das .NET Framework kann unter anderem in den Programmiersprachen C#, J# oder C++ erstellt werden. Erweiterungen können entweder als kompilierte Dynamic Linked Library (DLL) oder in dem KeePass-eigenen Dateiformat PLGX vorliegen. Der Prototyp wird als kompilierte DLL bereitgestellt.

3.2 Entwicklungsumgebung

Aufgrund der Verfügbarkeit des KeePass-Quellcodes in der Programmiersprache C# und einiger anderer Vorteile gegenüber anderen Programmiersprachen, wie etwa einer automatisierten „garbage collection“ [13], wurde zur Programmierung des Prototyps ebenfalls C# gewählt.

Als integrierte Entwicklungsumgebung (Integrated Development Environment, IDE) wurde Visual Studio von Microsoft eingesetzt. Die verwendeten Werkzeuge zur Entwicklung inklusive ihrer Versionen sind in Tabelle 3.1 aufgeführt.

Werkzeug	Version
Programmiersprache C#	3.0
.NET Framework	3.5
KeePass Professional Edition	2.27
Microsoft Visual Studio	Professional 2013

Tabelle 3.1.: Genutzte Entwicklungswerkzeuge

3.3 Modulübersicht

Der Prototyp besteht aus drei Hauptklassen, welche die Grundfunktionalität der Erweiterung bereitstellen. Im Folgenden werden diese Klassen vorgestellt und ihre Funktionalität erläutert. Abschließend folgt eine Übersicht über die verfügbaren Hilfsklassen.

3.3.1 PasswordPoliciesExt

Die Klasse `PasswordPolicies.PasswordPoliciesExt` stellt die Schnittstelle zwischen dem KeePass-Hauptprogramm und der entwickelten Erweiterung bereit. Sie initialisiert die Erweiterung und führt Veränderungen am Hauptprogramm durch, darunter das Registrieren des Passwort-Generators als „custom algorithm“ und das Editieren der Menüs des Hauptfensters. Die Instanziierung der Klasse erfolgt automatisch durch das Hauptprogramm, sobald die Erweiterung geladen worden ist. Bei der Instanziierung wird eine Referenz des Hauptfensters an die Erweiterung übergeben, sodass diese das Haupt- sowie Kontextmenü des Hauptfensters bearbeiten kann und der Nutzer die Möglichkeit bekommt, neben dem klassischen „Add Entry“ die Funktion „Add Entry using Password Policies“ zu wählen, um dienstspezifische Passwörter zu generieren (siehe Abbildung 3.2).

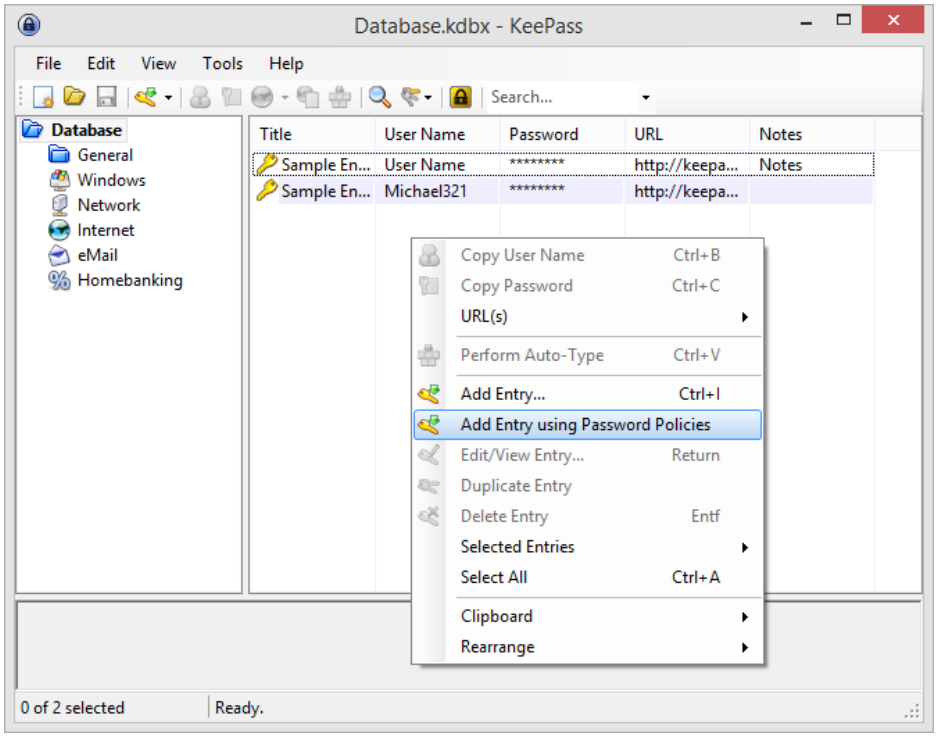

Abbildung 3.2.: KeePass Kontextmenü - Richtlinienabhängiges Eintragen von Datensätzen

3.3.2 PwGenerator

Die KeePass-interne Klasse `KeePassLib.Cryptography.PasswordGenerator.CustomPwGenerator` wird von der Klasse `PasswordPolicies.PwGenerator` erweitert und stellt Funktionen zur Generierung von Passwörtern bereit. Mit einer ersten Funktion können Passwörter generiert werden, ohne dass Informationen über die dienstspezifische URL zur Verfügung stehen. Da die abstrakte Elternklasse eine entsprechende Funktion zur Generierung von Passwörtern vorschreibt, nutzt der Passwort-Generator in diesem Fall eine hinterlegte Standard-Richtlinie. Diese Funktion wird beispielsweise genutzt, wenn ein Nutzer eine Liste von mehreren Passwörtern generieren möchte.

Die zweite Funktion bietet die Möglichkeit einer Generierung von Passwörtern anhand einer übergebenen Passwort-Richtlinie. Diese Passwort-Richtlinie wurde vorher durch das Abfragen der dienstspezifischen URL vom Benutzer in der internen Datenbank abgerufen, sodass gültige Passwörter für diesen Dienst generiert werden können.

3.3.3 PasswordPolicyFactory

Die Klasse `PasswordPolicyFactory` stellt die Schnittstelle zur Datenbank aller Passwort-Richtlinien dar. Wird für eine gegebene URL eine Richtlinie abgefragt, wird das Dokument für die Domain der URL nach einer passenden Richtlinie durchsucht. Wird keine Richtlinie gefunden, wird die Standard-Richtlinie zurückgegeben.

Die Suche nach einer entsprechenden Passwort-Richtlinie geschieht dabei über das Dateisystem. In einem Unterverzeichnis des KeePass-Hauptprogramms wird nach einer XML-Datei gesucht, die als Namen die Domain der angegebenen URL zum Dienst hat. Die gefundene Datei wird ausgelesen und die jeweils gültige Richtlinie, wie in Kapitel 2.2.1 erklärt, ausgewählt. Wird keine passende Richtlinie gefunden, wird die hinterlegte Standard-Richtlinie genommen, sodass immer ein Passwort generiert werden kann.

3.3.4 Hilfsklassen

Im Folgenden wird ein Überblick über die vorhandenen Hilfsklassen und deren Funktion gegeben:

- **CharacterSetProfile:**
  Repräsentation eines im gesamten Passwort oder für bestimmte Positionen verfügbaren Zeichensatzes.
- **PasswordPolicy:**
  Repräsentiert eine einzelne Passwort-Richtlinie für einen bestimmten Gültigkeitsbereich.
- **XmlPolicyParser:**
  XML-Parser zum Auslesen der im Dateisystem verfügbaren Passwort-Richtlinien.
- **EntryAddGeneratorForm:**
  Darstellung des neu hinzugefügten Formulars, welches in Kapitel 3.5 beschrieben ist.

3.4 Generierung von Passwörtern

Die Generierung von Passwörtern anhand einer gegebenen Richtlinie wurde im Wesentlichen mit dem KeePass-internen Passwort-Generator umgesetzt. Der interne Generator beinhaltet bereits viele Sicherheitsfunktionen, wie beispielsweise einem Schutz vor dem Auslesen der generierten Passwörter aus dem Arbeitsspeicher oder der sicheren Generierung von Zufallszahlen, die für die zufällige Auswahl aus den verfügbaren Zeichen unerlässlich ist. Da es durch die Richtlinien möglich ist, für einzelne Zeichenpositionen in einem Passwort unterschiedliche Zeichensätze vorzugeben, geschieht die Generierung der Passwörter im Gegensatz zum internen Prozess in mehreren Stufen.

In einem ersten Schritt wird für jede Zeichenposition, für die eine Einschränkung gilt, d. h. es steht ein zum Rest des Passwortes eingeschränkter bzw. unterschiedlicher Zeichensatz zur Verfügung, ein zufälliges Zeichen mit dem KeePass Passwort-Generator generiert. Nachdem für alle Positionen mit Einschränkungen entsprechende Zeichen zufällig generiert sind, findet eine Überprüfung statt, ob das Passwort bis zu diesem Punkt noch den Anforderungen der Richtlinie entspricht, oder ob bei der Generierung beispielsweise das `maxQuantity`-Attribut einer `restriction` (vgl. Auflistung 2.9) überschritten wurde. In diesem Fall wird die Generierung abgebrochen und von Neuem gestartet.

Im zweiten Schritt werden für alle restlichen Positionen, d. h. alle, für die die `availability`-Elemente im XML-Dokument gelten, Zeichen generiert. Zeichensätze, welche bereits im ersten Schritt das `maxQuantity`-Limit erreicht haben, werden bei der Generierung nicht ausgelassen. Würde dies geschehen, wäre die zufällige Auswahl verschiedener Zeichen bzw. Zeichensätze unausgewogen (eng. „biased“), was einem Angreifer helfen könnte, Passwörter zu erraten. Stattdessen wird das Passwort nach diesem Schritt erneut auf Konformität überprüft und das Passwort gegebenenfalls verworfen und die Generierung neu gestartet.

Schlägt die Generierung eines Passwortes mehrmals fehl, wird der Benutzer, wie in Abbildung 3.3 illustriert, darüber informiert und kann entscheiden, ob die Generierung abgebrochen werden soll oder weitere Versuche unternommen werden sollen. Das mehrmalige Fehlschlagen einer Generierung eines gültigen Passwortes kann der Fall sein, wenn es kein Passwort gibt, das die Anforderungen erfüllen kann.

Abbildung 3.3.: Fehlgeschlagene dienstspezifische Generierung eines Passwortes

3.5 Benutzeroberfläche

Im folgenden Kapitel wird die Benutzerführung von KeePass für die bisherige Generierung von Passwörtern in KeePass erläutert und im Weiteren auf die neu hinzugefügten Funktionen, die sich den bereits bestehenden Funktionen angliedern, eingegangen.

3.5.1 Generierung von Passwörtern in KeePass

Um ein neues Passwort in KeePass generieren und speichern zu können, wird das „Add Entry“-Formular, wie in Abbildung 3.4 zu sehen ist, aufgerufen. KeePass bietet in diesem Formular die Möglichkeit, neben

dem Passwort selbst einige weitere Informationen wie einem Benutzernamen, einer URL und Bemerkungen zu speichern. Außerdem ist ein Feld für das Ablaufdatum des Passwortes vorgesehen.

Der Benutzer kann ein selbst gewähltes Passwort eingeben oder durch einen Klick auf den in Abbildung 3.4 markierten Knopf ein Passwort generieren lassen. Dabei wird auf in KeePass registrierte Passwort-Generatoren zurückgegriffen, die von diesem Formular aus nicht weiter konfiguriert werden können. Über den Reiter „Advanced“ lassen sich zusätzliche Informationen als „Key-Value“-Paar und Dateianhänge dem Passwort zuordnen. In den „Properties“ lassen sich Darstellungseinstellungen für die Passwortliste im Hauptfenster und Tags für die in KeePass integrierte Suchfunktion festlegen. Ebenfalls lässt sich dort der Browser wählen, mit dem die für das Passwort gespeicherte URL bei Bedarf aufgerufen werden soll. Im Reiter „Auto-Type“ lassen sich individuelle auf den Dienst angepasste Einstellungen für das automatische Ausfüllen von Login-Formularen im Browser speichern, sodass durch eine Tastenkombination ohne Wechsel des Programmfensters die Login-Daten auf einer Webseite eingetragen werden können. Die „History“ zeigt eine Liste von Revisionen des Passwort-Eintrages.

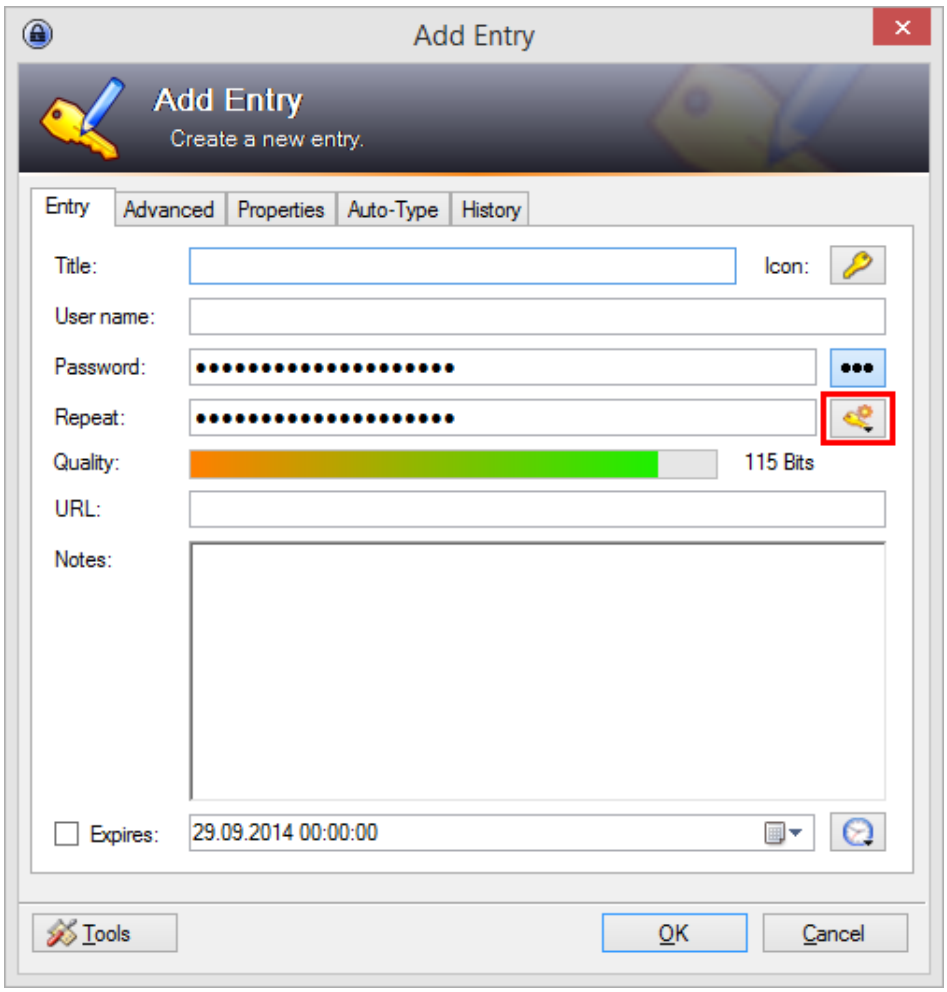

Abbildung 3.4.: Hinzufügen eines neuen Eintrags in KeePass

3.5.2 Dienstspezifische Generierung von Passwörtern

Das vom Programm bereitgestellte „Add Entry“-Formular zum Hinzufügen von neuen Passwort-Einträgen kann nicht ohne Eingriff in den Quellcode angepasst werden. Um die Kompatibilität für zukünftige Programmversionen zu erhalten, wurde daher ein anderer Ansatz verfolgt. Bei der Nutzung des hinzugefügten Menü-Elements „Add Entry using Password Policies“ wird daher ein weiteres Formular geöffnet, welches vor dem eigentlichen Formular zum Hinzufügen eines Eintrags angezeigt wird. Das geöffnete Formular fragt vom Benutzer die URL des Dienstes ab, für die ein dienstspezifisches Passwort erstellt werden soll.

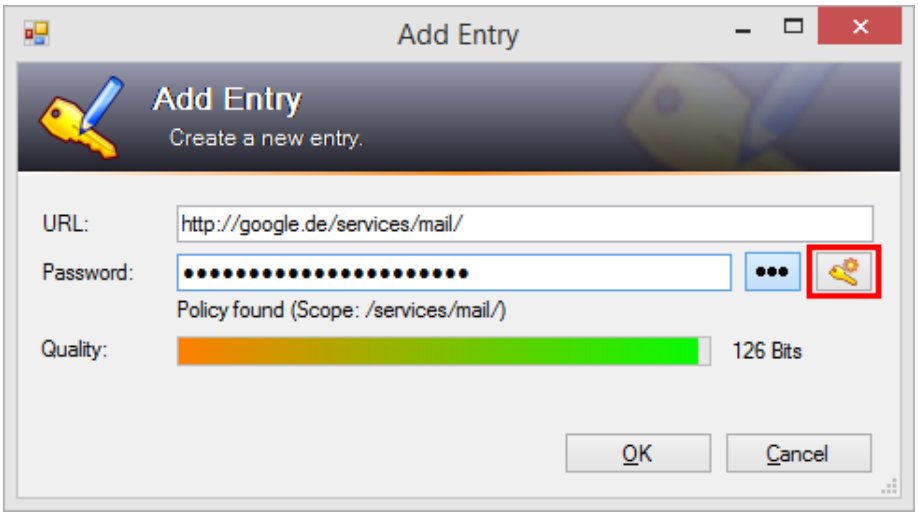

Abbildung 3.5.: Hinzufügen eines neuen Eintrags mit Hilfe der Passwort-Richtlinien

Die spezifischen Password-Richtlinien werden während der Eingabe einer URL im Hintergrund in der internen Datenbank gesucht. Mit einem Klick auf den in Abbildung 3.5 markierten Knopf wird ein neues Passwort generiert. Der `Quality`-Balken ist ebenso wie im KeePass-internen „Add Entry“-Formular ein Indikator für die Stärke des generierten Passwortes, welche anhand der berechneten Entropie bestimmt wird.

Wird eine fehlerhafte URL eingegeben oder kann die Richtlinie des zugehörigen Dienstes in der Datenbank nicht gefunden werden, bekommt der Nutzer in dem Formular die Rückmeldung, dass keine entsprechende Passwort-Richtlinie gefunden werden konnte und stattdessen die Standard-Richtlinie zur Generierung von Passwörtern genutzt wird (siehe Abbildung 3.6). Bei der Standard-Richtlinie handelt es sich um eine im Programm hinterlegte Richtlinie, welche nicht auf einen bestimmten Dienst abgestimmt ist. Sie soll es ermöglichen, auch für Dienste, die nicht bekannt sind, mit hoher Wahrscheinlichkeit gültige Passwörter zu liefern. Durch die Verwendung dieser Richtlinie kann es allerdings vorkommen, dass die generierten Passwörter für den präferierten Dienst nicht gültig sind und der Nutzer die Anforderungen manuell konfigurieren muss.

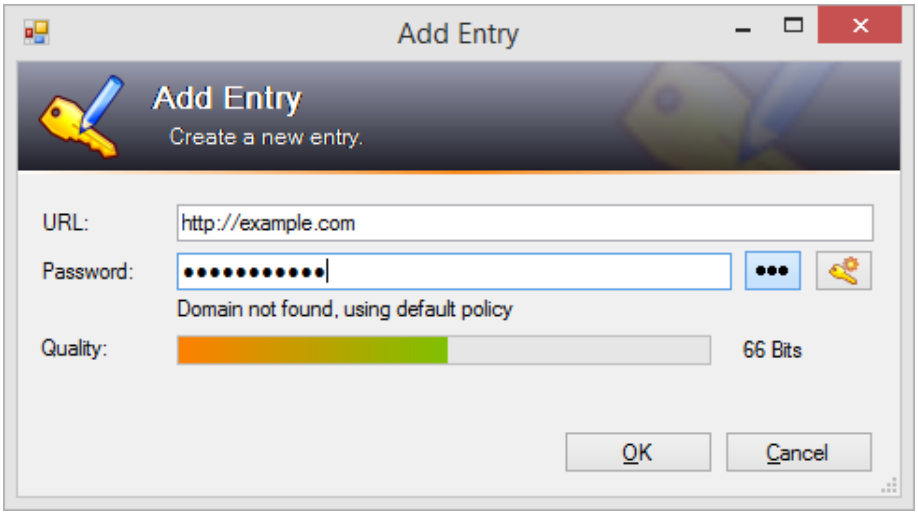

Abbildung 3.6.: Rückgriff auf die Standard-Richtlinie

Durch Klicken auf `OK` wird das Formular geschlossen und die Informationen an das integrierte Formular (siehe Abbildung 3.7) übertragen. So hat der Benutzer die Möglichkeit, nach der Generierung eines Passwortes wie gewohnt weitere Informationen einzutragen.

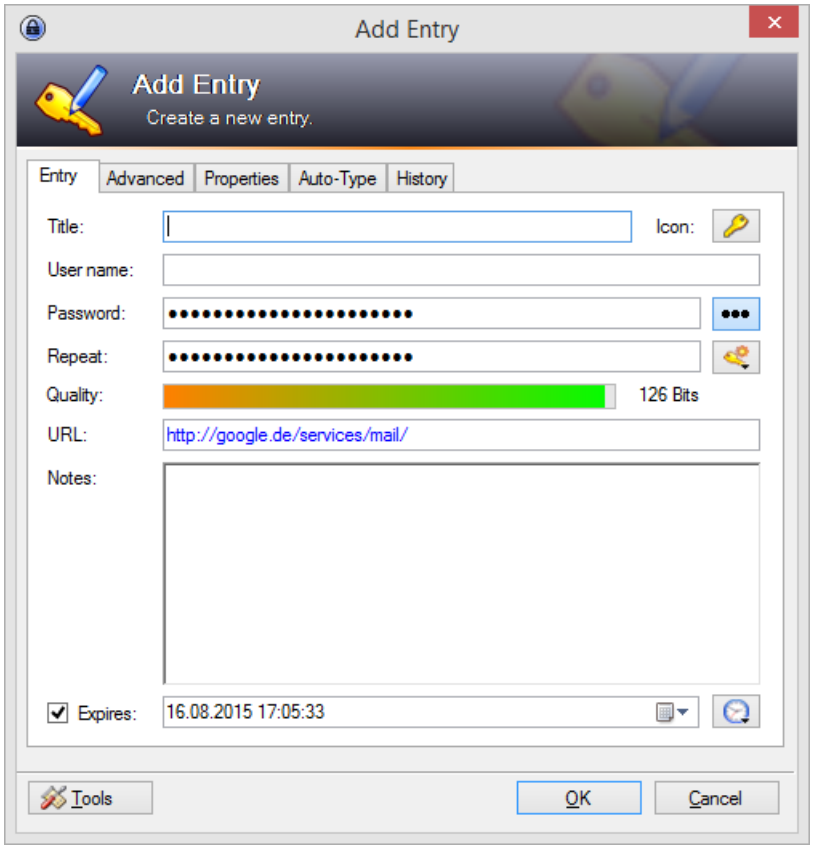

Abbildung 3.7.: An das Hauptprogramm weitergegebene Passwort-Informationen

3.5.3 Integration als benutzerdefinierter Algorithmus

KeePass bietet neben der gezeigten „Add Entry“-Funktion ebenfalls die Möglichkeit, wie in Abbildung 3.8 gezeigt, mehrere Passwörter gleichzeitig zu generieren. Dabei hat der Nutzer die Möglichkeit, die Parameter des Generators manuell zu konfigurieren. Es besteht die Möglichkeit, die Länge des Passwortes festzulegen und aus einer gegebenen Liste auszuwählen, welche Zeichen in den zu generierenden Passwörtern vorkommen dürfen. Ebenfalls kann der Nutzer eine eigene Liste an Zeichen angeben oder mit Hilfe eines regulären Ausdrucks komplexere Bedingungen stellen.

Als dritte Möglichkeit besteht die Generierung von Passwörtern anhand eines aus einer Liste auswählbaren benutzerdefinierten Algorithmus (eng. „custom algorithm“). Dabei handelt es sich nicht, wie der Name „custom algorithm“ suggeriert, um einen reinen Algorithmus, welcher vom KeePass Passwort-Generator ausgeführt wird. Vielmehr besteht hier ebenfalls die Möglichkeit, eigenen Programmcode auszuführen und so die entwickelte mehrstufige Prozedur für die Generierung von Passwörtern anhand der entwickelten Passwort-Richtlinien durchzuführen. Das entwickelte KeePass Plug-In ist in dieser Liste ebenfalls registriert, sodass auf die Generierung anhand von XML-Passwort-Richtlinien zurückgegriffen werden kann.

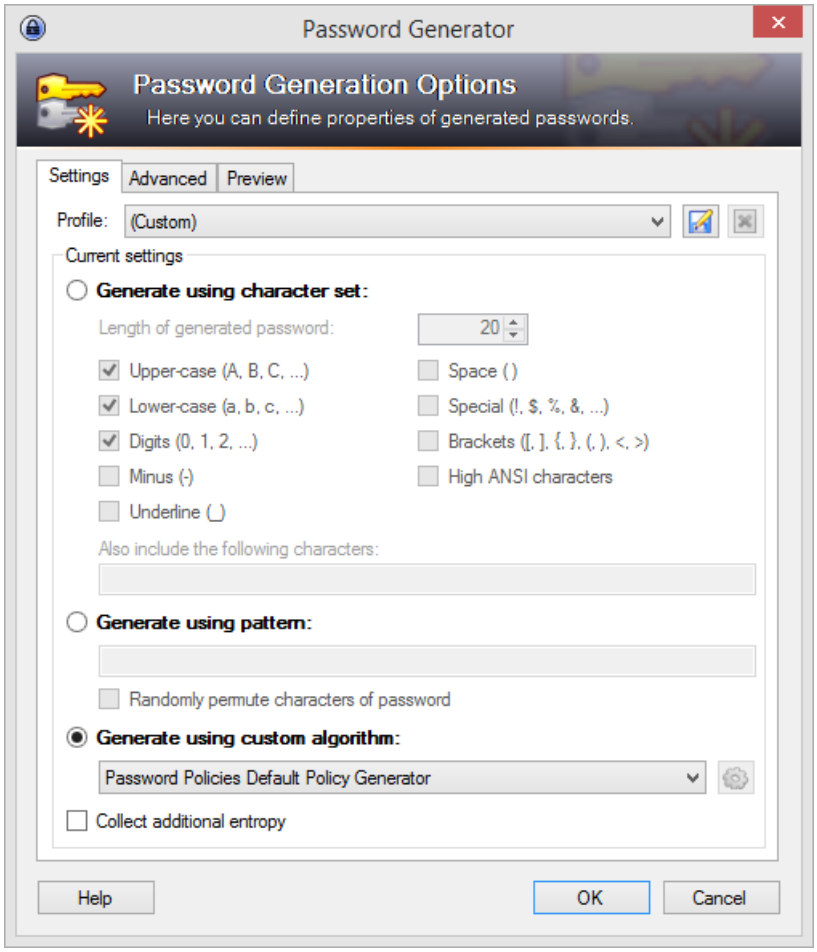

Abbildung 3.8.: Generieren von mehreren Passwörtern in KeePass

Da die Konfiguration solcher benutzerdefinierter Algorithmen seitens KeePass nur sehr bedingt möglich ist, wurde auf eine Möglichkeit zur Konfiguration des Generators in diesem Formular gänzlich verzichtet. Somit wird für die Generierung von Passwörtern über den von der Erweiterung registrierten „custom algorithm“ ausschließlich die hinterlegte Standard-Richtlinie genutzt. Auf diese Weise generierte Passwörter sind also nicht dienstspezifisch und werden so folglich nicht von allen Diensten akzeptiert.

4 Fazit und Ausblick

Die vorgestellte Spezifikation von Passwort-Richtlinien ermöglicht es, die Anforderungen an ein Passwort präzise in einem Dokument festzuhalten, damit andere Programme diese verarbeiten können, um den Nutzer in verschiedenen Aspekten zu unterstützen. Die Angabe der Eigenschaften, die ein Passwort erfüllen muss, erlaubt beispielsweise die Entwicklung von Passwort-Generatoren, welche möglichst sichere Passwörter generieren, ohne dass der Nutzer den Generator zuvor auf die dienstspezifischen Anforderungen konfigurieren muss. Durch die weitreichenden Konfigurationsmöglichkeiten, die bei der Angabe von erlaubten Zeichen in einem Passwort bestehen, sind selbst sehr komplexe Regeln möglich, die von der Länge des gewählten Passwortes abhängen. Die Spezifikation erleichtert nicht nur die Wahl von sicheren Passwörtern, sondern kann dem Nutzer auch bei bestehenden Nutzerkonten dabei helfen, Funktionen wie z. B. die „Passwort vergessen“-Funktion eines Dienstes zu finden.

Mit der Wahl von XML als Dokumentenformat für die Passwort-Richtlinien liegen diese in einem menschen- und maschinenlesbaren Format vor, welches im Internet insbesondere beim plattformunabhängigen Austausch von Daten Anwendung findet. Die breite Verfügbarkeit von XML-Parsern in den gängigen Programmiersprachen ermöglicht die schnelle und einfache Implementation eines Programmes, welches die Passwort-Richtlinien verarbeiten kann.

Bei der Distribution der Richtlinien hat sich gezeigt, dass der dezentrale Ansatz vorzuziehen ist. Bei diesem Ansatz ist keine neue Infrastruktur notwendig und die Dienst-Anbieter haben die volle Kontrolle über ihre Richtlinien, da diese auf den eigenen Servern zur Verfügung gestellt werden. Dies minimiert Sicherheitsrisiken und erfordert nicht den Aufbau und die Finanzierung weiterer Infrastruktur. Bis ein solches Konzept umgesetzt werden kann, bei dem vor allem die Mitarbeit der Anbieter gefragt ist, kann die zentrale Distribution genutzt werden, um einen entsprechenden Dienst aufzubauen. Eine Kombination dieser Ansätze, bei dem zuerst durch dezentrale Distribution eine entsprechende Richtlinie abgefragt wird und danach auf einen zentralen Dienst zurückgegriffen wird, ist ebenfalls denkbar.

Der entwickelte Prototyp zeigt, dass die Nutzung von Passwort-Richtlinien einen Mehrwert für den Nutzer bieten kann. Dadurch ist es möglich, nicht nur möglichst sichere und zufällige Passwörter zu generieren, die den Kriterien eines Dienstes entsprechen, sondern auch weitere Informationen wie etwa die Gültigkeitsdauer des Passwortes können automatisiert abgespeichert werden.

Abschließend betrachtet gibt es über den Prototypen hinaus weitere Möglichkeiten, die Nutzung der Passwort-Richtlinien auf verschiedene Weisen zu optimieren. Durch die Analyse einer großen Zahl an Webseiten kann eine Passwort-Richtlinie gesucht werden, die für einen Großteil der Dienste im Internet gültige Passwörter generieren lässt. So kann trotz fehlender Verbreitung der Richtlinien im Internet

eine Standard-Richtlinie in einem entsprechenden Generator hinterlegt werden, welche mit hoher Wahrscheinlichkeit ein gültiges Passwort liefert und genutzt wird, wenn der angefragte Dienst bisher keine eigene Richtlinie hinterlegt haben sollte oder im zentralen Ansatz dieser Dienst nicht bekannt ist.

Programme wie KeePass bieten bereits eine Möglichkeit, die gespeicherten Login-Informationen per Tastenkombination („Auto-Type“) an den Browser zu übertragen, um ein Programmwechsel beim Login zu vermeiden. Die entwickelte Erweiterung kann dahingehend angepasst werden, ähnlich zu dieser Funktion für den Nutzer bereits vorherzusehen, für welchen Dienst er ein Passwort generieren möchte. Dafür kann aus dem geöffnetem Browser die aktuell besuchte URL ausgelesen werden, und bei der Abfrage der dienstspezifischen URL bereits standardmäßig eingetragen werden. Der Nutzer kann diese daraufhin ändern, oder die Voreinstellung beibehalten und ein Passwort erstellen lassen.

Auch in Browsern integrierte Passwortverwaltungen erfreuen sich immer größerer Beliebtheit. Dabei fragt der Browser nach dem Absenden eines Formulars, in dem ein Passwort-Feld ausgefüllt wurde, ob das Passwort für die gegebene Webseite gespeichert werden soll. In den meisten Fällen werden diese Passwörter in einem lokalen Profil oder in der Registrierungsdatenbank des Betriebssystems abgelegt. Andere Browser bieten eine Synchronisierung über einen Dienst im Internet an, um die gespeicherten Passwörter über verschiedene Geräte hinweg nutzen zu können. Es bietet sich an, diese Funktionalität zu erweitern, dass auch hier Passwörter mit einem Klick nicht nur gespeichert, sondern auch generiert werden können. Dadurch entfällt der Programmwechsel zu einem externen Programm vollständig. Externe Programme zur Passwortverwaltung gelten allerdings durch ihre Sicherheitsfunktionen im Allgemeinen als sicherer. Um einen ähnlichen Komfort bei höherer Sicherheit zu bieten, kann eine entsprechende Erweiterung auch dahingehend genutzt werden, mit diesen externen Programmen zu kommunizieren, um sowohl generierte Passwörter abzurufen, als auch die entsprechenden Informationen direkt in diesem Programm speichern zu können. So ist zum einen die sichere Aufbewahrung der Daten gewährleistet und zum anderen die Nutzerfreundlichkeit erhöht.

Schlussendlich lässt sich festhalten, dass für eine Nutzung der Passwort-Richtlinien als erstes die Distribution der Richtlinien umgesetzt werden muss. Ist die entsprechende Infrastruktur vorhanden und für einen Großteil der Dienste nutzbar, können die genannten Optimierungen in der Nutzung dieser Richtlinien umgesetzt werden, um eine breite Nutzerbasis anzusprechen. Ein Ausblick für das Erstellen der Richtlinien ohne das Zutun der Anbieter bieten hier Ansätze des Machine Learning, bei welchen automatisiert versucht werden kann, die Anforderungen an ein Passwort auszulesen sowie weitere Informationen für die entsprechenden Passwort-Richtlinien zusammenzutragen.

Literaturverzeichnis

[1] Alexa Top 500 Global Sites, 2014. http://www.alexa.com/topsites. (4)

[2] C. Allen and T. Dierks. The TLS protocol version 1.0. 1999. (21)

[3] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. Simple object access protocol (SOAP) 1.1, 2000. (17)

[4] T. Bray, J. Paoli, and C. Sperberg-McQueen. Extensible Markup Language (XML), W3C (World Wide Web Consortium), 1998. (2)

[5] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, et al. Web services description language (WSDL) 1.1, 2001. (17)

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol–HTTP/1.1, 1999. (18)

[7] S. Gao, C. M. Sperberg-McQueen, H. S. Thompson, N. Mendelsohn, D. Beech, and M. Maloney. W3C XML schema definition language (XSD) 1.1 part 1: Structures. *W3C Candidate Recommendation*, 30, 2009. (2)

[8] R. Harbich. Webcrawling–Die Erschließung des Webs, 2008. (16)

[9] L. Hathaway. National policy on the use of the advanced encryption standard (AES) to protect national security systems and national security information. *National Security Agency*, 2003. (23)

[10] U. Helmbrecht. DNS Spoofing. *Bundesamt für Sicherheit in der Informationstechnik (Hrsg.): IT-Grundschutzhandbuch*, page 675, 2004. (19)

[11] B. Ives, K. R. Walsh, and H. Schneider. The domino effect of password reuse. *Communications of the ACM*, 47(4):75–78, 2004. (2)

[12] M. Koster. *A standard for robot exclusion*. NEXOR., 1994. (19)

[13] R. D. Lins. Garbage collection: algorithms for automatic dynamic memory management. 1996. (25)

[14] L. Masinter, T. Berners-Lee, and R. T. Fielding. Uniform resource identifier (URI): Generic syntax. 2005. (7, 9)

[15] MSDN. What is Managed Code?, 2014. http://msdn.microsoft.com/en-us/library/windows/desktop/bb318664(v=vs.85).aspx. (24)

[16] D. Y. Na and B. C. DeRocher. Html5: What's different for user experience design and the web? *Connectivity and the User Experience*, page 45, 2011. ()

[17] M. Nottingham and E. Hammer-Lahav. Defining well-known uniform resource identifiers (URIs). 2010. (17)

[18] A. Ornaghi and M. Valleri. Man in the middle attacks Demos. *Blackhat [Online Document]*, 2003. (19)

[19] D. Reichl. KeePass Password Safe, 2014. `http://www.keepass.info`. (3, 23)

[20] D. Solo, R. Housley, and W. Ford. Internet X. 509 public key infrastructure certificate and CRL profile. 1999. (21)

[21] J. Völker. Passwörter sicher verwalten. *Datenschutz und Datensicherheit-DuD*, 38(5):314–317, 2014. (23)

A XML Schema Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning" elementFormDefault="qualified"
    vc:minVersion="1.1">
    <xs:element name="policies">
        <xs:complexType>
            <xs:sequence>
                <xs:element maxOccurs="unbounded" minOccurs="0" ref="policy"/>
            </xs:sequence>
            <xs:attribute name="version" type="xs:string">
                <xs:annotation>
                    <xs:documentation>The current version of the policies
                        file.</xs:documentation>
                </xs:annotation>
            </xs:attribute>
            <xs:attribute name="versionTimestamp" type="xs:integer">
                <xs:annotation>
                    <xs:documentation>Unix timestamp when this document was last
                        updated.</xs:documentation>
                </xs:annotation>
            </xs:attribute>
        </xs:complexType>
    </xs:element>
    <xs:element name="policy">
        <xs:annotation>
            <xs:documentation>The policy element represents a password policy for a
                given scope.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="characterSets">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element maxOccurs="unbounded" ref="characterSet"/>
```

```
                </xs:sequence>
            </xs:complexType>
            <xs:unique name="characterSetUniqueName">
                <xs:selector xpath=".//characterSet"/>
                <xs:field xpath="@name"/>
            </xs:unique>
        </xs:element>
        <xs:element name="properties">
            <xs:annotation>
                <xs:documentation>Required. Represents the properties of the
                    password.</xs:documentation>
            </xs:annotation>
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="minLength" type="xs:positiveInteger"
                        minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>Minimum length of the password.
Omitted if minimum length is not present.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="maxLength" type="xs:positiveInteger"
                        minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>Maximum length of the password.
Omitted for no maximum length.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="expires" type="xs:positiveInteger"
                        minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>Password expiry in days.
Omitted for no expiry.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="characterSettings">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="availableCharacterSet"
                                    maxOccurs="unbounded" type="characterSetSetting">
```

```
                    <xs:annotation>
                        <xs:documentation>Specifies a character set
                            that can be used in any character
                            position of the
                            password.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element name="restrictions" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="restriction"
                            maxOccurs="unbounded" minOccurs="0">
                            <xs:annotation>
                            <xs:documentation>A restriction specifies
                                a character set that can be used for
                                the given character
                                positions.</xs:documentation>
                            </xs:annotation>
                            <xs:complexType>
                            <xs:complexContent>
                            <xs:extension base="characterSetSetting">
                            <xs:attribute name="position" type="xs:string"
                            use="required">
                            <xs:annotation>
                            <xs:documentation>Comma separated list of
                                positions where the defined character
                                set can occur.</xs:documentation>
                            </xs:annotation>
                            </xs:attribute>
                            </xs:extension>
                            </xs:complexContent>
                            </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:sequence>
```

```
                </xs:complexType>
            </xs:element>
            <xs:element name="service" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Holds information related to the service the
                        password is used for.</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="registerURL" type="xs:string" minOccurs="0"/>
                        <xs:element name="passwordChangeURL" type="xs:string"
                            minOccurs="0"/>
                        <xs:element name="passwordForgottenURL" type="xs:string"
                            minOccurs="0"/>
                        <xs:element minOccurs="0" name="passwordMaxRetries"
                            type="xs:positiveInteger"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="scope" type="xs:string" use="optional" default="/">
            <xs:annotation>
                <xs:documentation>Relative path of the webpage where this policy
                    will be used.
If the scope attribute references a file (i.e. does not end with a slash), the policy
    is only valid for the referenced file. A policy with a scope that ends with a
    slash is valid for the referenced folder, all subfiles and folders.
Policies with a more specific scope are chosen over ones with more general scopes.
The default value of the scope is "/", which is valid for all files and
    folders.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:complexType>
    <xs:unique name="uniqueScope">
        <xs:selector xpath="policy"/>
        <xs:field xpath="scope"/>
    </xs:unique>
    <xs:keyref name="availableCharacterSetKeyRef" refer="characterSetKey">
        <xs:selector xpath="properties/characterSettings/availability"/>
        <xs:field xpath="@characterSet"/>
```

```
        </xs:keyref>
        <xs:keyref name="restrictedCharacterSetKeyRef" refer="characterSetKey">
            <xs:selector xpath="properties/characterSettings/restrictions/restriction"/>
            <xs:field xpath="@characterSet"/>
        </xs:keyref>
        <xs:key name="characterSetKey">
            <xs:selector xpath=".//availableCharacterSets/characterSet"/>
            <xs:field xpath="@name"/>
        </xs:key>
    </xs:element>
    <xs:element name="characterSet">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="characters" type="xs:string" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>String containing all characters that will be
                            added to the character set.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element maxOccurs="unbounded" minOccurs="0" name="base"
                    type="xs:string">
                    <xs:annotation>
                        <xs:documentation>Reference to an already defined character
                            set. All characters from the character set referenced by this
                            element will be added to the character set this element is a
                            child of.</xs:documentation>
                    </xs:annotation>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="name" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="characterSetSetting">
        <xs:attribute name="characterSet" type="xs:string" use="required">
            <xs:annotation>
                <xs:documentation>Reference to a character set name that is specified
                    in the "characterSets" element.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="minQuantity" type="xs:positiveInteger"/>
```

```
        <xs:attribute name="maxQuantity" type="xs:positiveInteger"/>
    </xs:complexType>
</xs:schema>
```

Auflistung A.1: Vollständiges XML Schema

B Beispiele

```
<?xml version="1.0" encoding="UTF-8"?>
<policies xmlns:vc="http://www.w3.org/2007/XMLSchema—versioning"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema—instance"
    xsi:noNamespaceSchemaLocation="schema.xsd">
        <policy scope="/services/mail/">
                <characterSets>
                        <characterSet name="firstCharacterSet">
                                <characters>0123456789</characters>
                        </characterSet>
                        <characterSet name="secondCharacterSet">
                                <base characterSet="firstCharacterSet" />
                                <characters>ABCDEF</characters>
                        </characterSet>
                </characterSets>
                <characterSettings>
                        <availableCharacterSet characterSet="firstCharacterSet"
                            minQuantity="0.5" />
                        <availableCharacterSet characterSet="secondCharacterSet" />
                        <restrictions>
                                <restriction characterSet="firstCharacterSet" position="0,—1,0.5"
                                    />
                                <restriction characterSet="secondCharacterSet" position="0.5,1,2"
                                    minQuantity="2" />
                        </restrictions>
                </characterSettings>
                <service>
                        <registerURL>
                                http://www.example.com/services/mail/register
                        </registerURL>
                        <passwordChangeURL>
                                http://www.example.com/services/mail/myProfile
                        </passwordChangeURL>
                        <passwordForgottenURL>
                                http://www.example.com/services/mail/forgotPassword
```

```
                    </passwordForgottenURL>
                <passwordMaxRetries>3</passwordMaxRetries>
            </service>
        </policy>
        <policy>
            <characterSets>
                <characterSet name="singleCharacterSet">
                    <characters>0123456789ABCDEFabcdef</characters>
                </characterSet>
            </characterSets>
            <characterSettings>
                <availableCharacterSet characterSet="singleCharacterSet" />
            </characterSettings>
        </policy>
</policies>
```

Auflistung B.1: Vollständiges Beispiel eines XML-Dokumentes für Passwort-Richtlinien

```
<policy>
    <characterSets>
        <characterSet name="singleCharacterSet">
            <characters>0123456789ABCDEFabcdef</characters>
        </characterSet>
    </characterSets>
    <characterSettings>
        <availableCharacterSet characterSet="singleCharacterSet" />
    </characterSettings>
</policy>
```

Auflistung B.2: Minimales Beispiel einer Passwort-Richtlinie