Sebastian Hirth

Entwurf eines Sicherheitskonzepts zur Authentifikation und Autorisierung von Java Enterprise Applikationen am Beispiel der A^{3}S AdminConsole

Diplomarbeit

HOCHSCHULE MITTWEIDA (FH)

UNIVERSITY OF APPLIED SCIENCES

Mathematik/Naturwissenschaften/Informatik

Mittweida, Mai 2010

Sebastian Hirth

Entwurf eines Sicherheitskonzepts zur Authentifikation und Autorisierung von Java Enterprise Applikationen am Beispiel der A^{3}S AdminConsole

eingereicht als

Diplomarbeit

an der

HOCHSCHULE MITTWEIDA (FH)

UNIVERSITY OF APPLIED SCIENCES

Mathematik/Naturwissenschaften/Informatik

Mittweida, Mai 2010

Erstprüfer: Prof. Dr.-Ing. Andreas Ittner
Zweitprüfer: Dipl.-Inf. Holger Langner
Vorgelegte Arbeit wurde verteidigt am:

Bibliographische Beschreibung

Hirth, Sebastian:
Entwurf eines Sicherheitskonzepts zur Authentifikation und Autorisierung von Java Enterprise Applikationen am Beispiel der A^{3}S AdminConsole.-2010.-
58 Seiten.
Mittweida, Hochschule Mittweida (FH) - University of Applied Sciences, Fakultät Mathematik/Naturwissenschaften/Informatik, Diplomarbeit, 2010

Referat

Heutzutage kommt der IT-Sicherheit immmer größere Bedeutung zu. Unzureichende Sicherheitsvorkehrungen für Web-Applikationen können fatale Auswirkungen haben. Deshalb soll anhand eines konkreten Beispiels, der A^{3}S AdminConsole, eine JavaEE-Anwendung zur webgestützten Konfiguration von Recommendation Services, systematisch ein Sicherheitskonzept zur Authentifikation und Autorisierung entwickelt werden, welches definierte Sicherheitsziele erfüllt.
Das erarbeitete Sicherheitskonzept wird anhand von konkreten Anwendungsszenarien betrachtet. Für die Umsetzung werden vorhandene Verfahren und Modelle zur Authentifikation und Autorisierung untersucht und anhand ihrer Tauglichkeit für eine mögliche Realisierung des Konzeptes bewertet. Des Weiteren werden mögliche Bedrohungen betrachtet, welche die A^{3}S AdminConsole gefährden können.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACL	Access Control List
AES	Advanced Encryption Standard
CSS	Cross Site Scripting
DDI	Dynamic Domain Integration
DES	Data Encryption Standard
DoS	Denial of Service
EJB	Enterprise Java Beans
ERM	Entity Relationship Modell
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDS	Intrusion Detection System
JavaEE	Java Enterprise Edition
JAX-RS	Java API for RESTful Web Services
JMS	Java Message Service
MD	Message Digest
MITMA	Man-in-the-Middle-Attack
MVC	Model View Controller

RBAC	Role Based Access Control
REST	Representational State Transfer
RSA	Rivest Shamir Adleman
SHA	Secure Hash Algorithm
SQL	Structured Query Language
SSL	Secure Socket Layer
SSO	Single Sign-On
URI	Uniform Ressource Identifier
XML	Extensible Markup Language
XSS	Cross Site Scripting

1. Einleitung

1.1. Motivation

Da immer mehr Geschäftsprozesse über das Internet abgewickelt werden, kommt der Sicherheit eine immer größere Bedeutung zu. Dennoch spielt die Sicherheit bei vielen Web-Anwendungen eine eher untergeordnete Rolle, obwohl Schwachstellen in der Implementierung beziehungsweise die Verwendung von unsicheren Verfahren fatale Auswirkungen für Nutzer und Unternehmen haben können.

1.2. Aufgabenbeschreibung

Das Ziel der Arbeit ist die Spezifikation eines Sicherheitskonzeptes für die A^{3}S AdminConsole. Die A^{3}S AdminConsole ist eine Java Enterprise Edition (JavaEE)-Applikation zur webgestützten Konfiguration von Recommendation Services. Sie verfügt über eine Representational State Transfer (REST)- sowie über eine Spring Model View Controller (MVC)-Schnittstelle. Der Ausgangspunkt der Arbeit ist eine systematische Einführung in grundlegende Verfahren aus dem Bereich der Authentifikation und Autorisierung. In diesem Zusammenhang werden vorhandene Autorisierungskonzepte wie rollenbasierte Autorisierung und Access Control Lists, sowie Authentifkationskonzepte wie der Single Sign-On evaluiert und bewertet. Für den Entwurf des Sicherheitskonzeptes werden besonders geeignete Verfahren zur Realisierung ausgewählt, um die vordefinierten, schnittstellenspezifischen Sicherheitsziele zu erfüllen und eine Gewährleistung der Mandantenfähigkeit zu erreichen. Dabei werden auch unterschiedliche Modelle kombiniert um das gewünschte Ergebnis zu erhalten. Es erfolgen Bedrohungsanalysen um eventuelle Gefährdungen und Risiken offen zu legen, die bei der Konzeption unbedingt berücksichtigt werden müssen, um ein besonders hohes Maß an Sicherheit zu erzielen.

1.3. Gliederung der Arbeit

In Kapitel 2 wird ein Überblick über Komponenten gegeben, die in der A^{3}S AdminConsole verwendet werden. Im Anschluss daran werden in Kapitel 3 die theoretischen Grundlagen geschaffen, welche für den Entwurf eines Sicherheitskonzepts benötigt werden. Auf der Basis der erworbenen Grundlagen, werden in Kapitel 4 mögliche Bedrohungen für die A^{3}S AdminConsole anhand einer Bedrohungsanalyse herausgearbeitet. Des Weiteren werden die in Kapitel 3 vorgestellten Verfahren und Modelle zur Authentifikation und Autorisierung untersucht und entschieden welche sich für die Umsetzung des Sicherheitskonzeptes eignen. Im Anschluss daran wird in Kapitel 5 der Entwurf des Sicherheitskonzeptes durchgeführt, mit Hilfe der ausgewählten Verfahren und Modelle. In Kapitel 6 erfolgt die Zusammenfassung der Diplomarbeit, in der das erarbeitete Sicherheitskonzept eingeschätzt wird. Abschließend wird in Kapitel 7 ein Ausblick darüber gegeben, wie weiter verfahren werden kann, wenn das Sicherheitskonzept umgesetzt wurde.

2. Die A^{3}S AdminConsole

Dieses Kapitel beschreibt die Funktionsweise der A^{3}S AdminConsole und stellt die dafür genutzten Technologien, sowie deren Zusammenhang dar.

2.1. A^{3}S AdminConsole

Die A^{3}S AdminConsole ist eine JavaEE-Applikation zur web-gestützten Konfiguration von Recommendation Services. Sie verfügt über eine REST-Schnittstelle. Die A^{3}S AdminConsole gliedert sich in Scopes. Ein Scope ist ein Sichtbarkeitsbereich für eine Firma und dient als Sammelcontainer für die zu verwaltenden Objekte. In einem Scope existiert je nach Bedarf eine Anzahl von Domänen, in der Applikationen eines Aufgabengebietes verwaltet werden (Abbildung 2.1). Die A^{3}S AdminConsole setzt auf der A^{3}S Recommendation auf. Die A^{3}S Recommendation Engine stellt die benötigten Funktionalitäten für die A^{3}S AdminConsole bereit. Sie verfügt über drei Hauptkomponenten:

- **Core:** Der Core stellt die Kernfunktionalitäten für die Applikationsverwaltung bereit.
- **Dynamic Domain Integration (DDI):** Ist eine Komponente zur Integration von domänenspezifischen Daten.
- **Recommender Services:** Stellt Funktionalitäten für die Berechnung von Recommendations und die Filterung dieser zur Verfügung.

In Abbildung 2.2 werden die Komponenten der A^{3}S Recommendation Engine gezeigt.

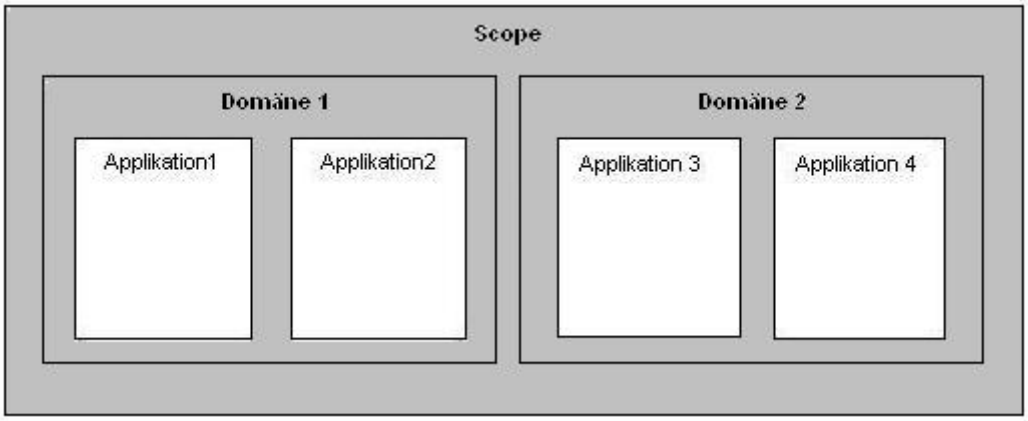

Abbildung 2.1.: Scope-Domain-Schema A³S AdminConsole

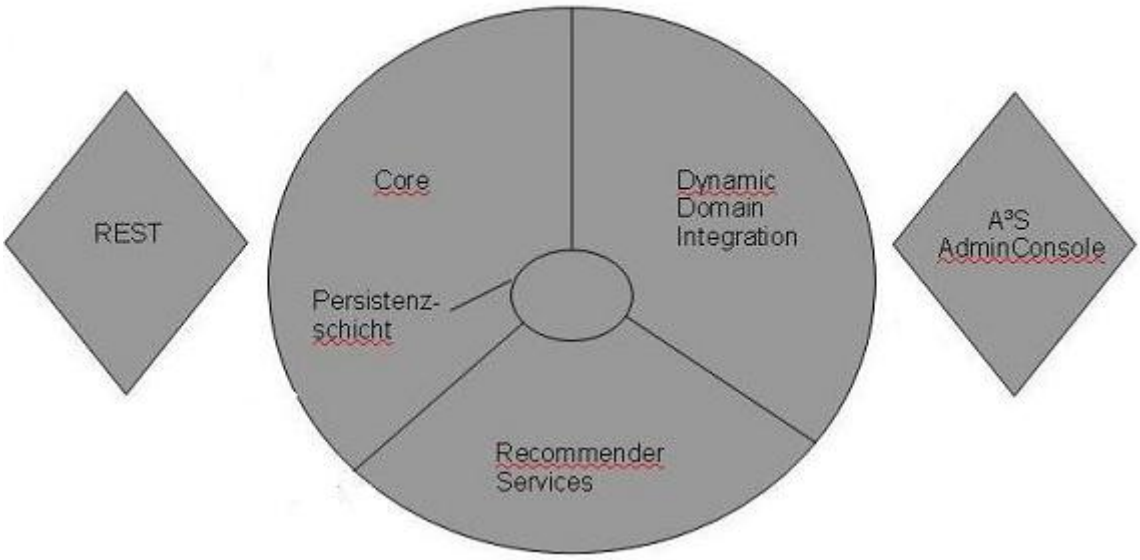

Abbildung 2.2.: Komponenten A³S

2.2. Technischer Überblick

Nachfolgend wird ein kurzer Überblick über die in der A^{3}S AdminConsole verwendeten Technologien gegeben, da diese darüber entscheiden, in welcher Art und Weise ein entwickeltes Sicherheitskonzept realisiert werden kann.

2.2.1. Die Java Enterprise Edition

Die JavaEE besteht aus einem großen Satz von Schnittstellen, die zusammen ein Modell zur Entwicklung von mehrschichtigen, verteilten Applikationen bilden.

2.2.2. Respresentational State Transfer

Der REST ist eine Sammlung von Designkriterien, die beschreiben wie Web Services beziehungsweise Web-Applikationen aufgebaut sein sollten, um das Web einfacher und interoperabler zu gestalten, auf Basis des Hypertext Transfer Protocol (HTTP), dem Uniform Ressource Identifier (URI)-Standard und Extensible Markup Language (XML) als Auszeichnungssprache . Dabei stützt sich REST auf zwei Grundpfeiler, zum Einen die Adressierbarkeit und zum Anderen die Zustandslosigkeit. Eine Anwendung ist dann adressierbar wenn alle interessanten Aspekte der Daten durch Ressourcen bereitgestellt und über URIs angesprochen werden. Zustandslosigkeit bedeutet das jeder HTTP-Request in vollständiger Isolation stattfindet. Wenn ein Client einen HTTP-Request abschickt, enthält dieser alle relevanten Informationen die für den Server notwendig sind, um die Aufgabe ordnungsgemäß zu erfüllen ohne, sich auf Daten von früheren Requests berufen zu müssen (vgl. [LR07]).

Java API for RESTful Web Services

Die Java API for RESTful Web Services (JAX-RS) ist eine Programmierschnittstelle der JavaEE um REST-basierte Web Services zu erstellen. Wie auch bei anderen Programmierschnittstellen der JavaEE, nutzt JAX-RS Annotationen, um die Entwicklung von Web Services zu vereinfachen. JAX-RS wurde dazu verwendet, die REST-Schnittstelle implementieren zu. Die REST-Schnittstelle wird dazu verwendet Recommendations aller Art abzurufen.

2.2.3. Java Enterprise Edition Server

Der JBoss Application Server

Der JBoss Application Server ist ein Anwendungsserver für Java Enterprise Applikationen, welcher eine Laufzeitumgebung für Enterprise Java Beans implementiert. Alle benötigten Dienste werden dafür bereit gestellt. Diese Dienste umfassen unter anderem Transaktionen, Sicherheit, Persistenz und Lastverteilung. Es werden auch JavaEE Web Services unterstützt.

Enterprise Java Beans

Enterprise Java Beans (EJB) sind serverseitige Geschäftslogikkomponenten. Für die Arbeit mit EJBs wird eine „3-Schichten-Architektur“ verwendet. Dies ist eine spezielle Form der Client-Server-Architektur. Dabei wird das System in drei Bereiche untergliedert (Siehe Abbildung 2.3). EJBs gliedern sich in drei Typen. Entity Beans,

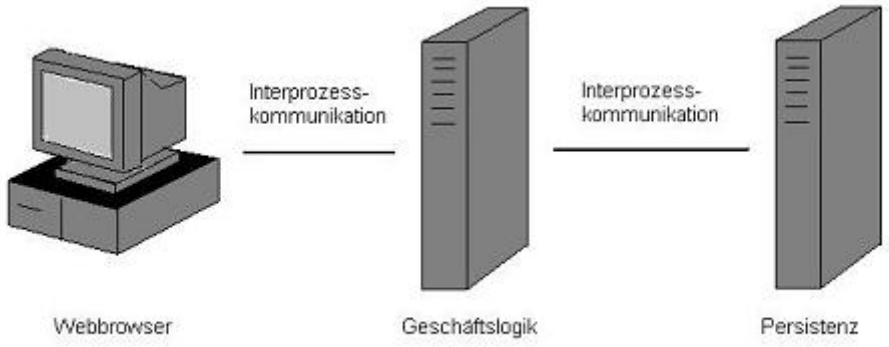

Abbildung 2.3.: 3-Schichten-Architektur

Session Beans und Message Driven Beans. Diese werden nachfolgend näher erläutert (vgl. [EL07]).

- **Entity Beans**
  Entity Beans dienen der persistenten Speicherung der Daten.
- **Session Beans**
  Session Beans repräsentieren die serverseitigen Sitzungs- beziehungsweise Servicekomponenten. Mit Hilfe von Session Beans wird die Logik von Anwendungs-

fällen oder Geschäftsprozessen abgebildet. Es gibt zwei Formen von Session Beans, zustandsbehaftete und zustandslose Session Beans. Zustandsbehaftete Session Beans werden dazu genutzt einen client-spezifischen Zustand zu behalten. Das heißt, das über mehrere Aufrufe hinweg ein Clientzustand gespeichert wird.

- **Message-Driven Beans**
  Message-Driven Beans sind zustandslose, asynchrone Nachrichtenempfänger. Um eine asynchrone Kommunikation herzustellen wird Java Message Service (JMS) genutzt. Der JMS stellt eine Programmierschnittstelle zum Senden und Empfangen von Nachrichten aus einem Client heraus dar.

2.2.4. Spring MVC Web Framework

Das Spring MVC Web Framework ist ein Teil der Spring Distribution. Derzeit wird Version 3.0.2 Spring Distribution verwendet. Das Spring MVC ist ein Framework zum Erstellen von Web-Applikationen (vgl. [RB08]). Mit Hilfe von Spring MVC wurde das Graphical User Interface (GUI) der A^{3}S AdminConsole entwickelt.

3. Theoretische Grundlagen

In diesem Kapitel werden Verfahren, Algorithmen, Modelle vorgestellt, welche im Internet zur Authentifikation und Autorisierung eingesetzt werden. Desweiteren wird ein Überblick über verschiedene Angriffe und Angriffstechniken gegeben, die im Web-Umfeld häufig eingesetzt werden, um besser einschätzen zu können, welchen Bedrohungen eine Web-Anwendung ausgesetzt sein kann.

3.1. Die Grundlagen der Authentifikation

3.1.1. Begriffsdefintion

Die Authentifikation ist im Allgemeinen der Nachweis der Identität einer Person. Im Web-Umfeld ist die Authentifikation eine Sicherheitsmaßnahme zur Überpfrüfung der Identität von Subjekten beziehungsweise von Objekten.

3.1.2. Basiselemente der Authentifikation

Man unterscheidet grundsätzlich drei Authentifikationsverfahren (vgl. [Sch01]).

- **Wissen**
  Die Authentifikation durch Wissen ist die am häufigsten eingesetzte Variante im Webumfeld. In der Regel muss der Benutzer im Besitz von einer oder mehreren geheimen Informationen sein, um sich authentifizieren zu können. Solche Informationen umfassen Geheimnummern, geheime Schlüssel, persönliche Daten oder Passwörter. Letztere werden am Häufigsten eingesetzt.
- **Besitz**
  Bei dieser Form der Authentifikation muss der Besitz eines schwer zu fälschendem Gegenstands nachgewiesen werden, um sich erfolgreich authentifizieren zu

können. Beispiele hierfür könnten ein Ausweis oder die Chipkarte einer Bank sein. Der Besitz eines schwer fälschbaren Gegenstands lässt sich über das Netz nicht direkt feststellen, aber spielt trotzdem eine Rolle. Es ist beispielsweise möglich, einen geheimen Schlüssel auf einer Chipkarte unlesbar zu speichern.

- **Biometrie**
  Biometrische Verfahren stellen eine weitere Methode zur Auhtentifikation dar. Dabei werden eindeutige, unverwechselbare persönliche Merkmale wie Fingerabdrücke, Unterschriften oder auch die Stimme zum Identitätsnachweis genutzt. Das Problem bei dieser Methode besteht darin, dass wenn ein Messwert über das Netz geschickt wird, ein möglicher Angreifer diesen wiederverwerten kann. Ein Service, der diese Form der Authentifikation nutzt, hat keine Möglichkeit zu erkennen, ob der Wert von einem Messgerät stammt oder ob er auf andere Weise eingespielt wurde.

3.1.3. Kryptografische Grundlagen

Es wird zwischen zwei Arten von Verschlüsselungsalgorithmen unterschieden. Zum einen existieren symmetrische und zum anderen asymmetrische Algorithmen. Bei symmetrischen Algorithmen wird für die Verschlüsselung ein Schlüssel generiert. Mit diesem Schlüssel wird eine Nachricht ver- und entschlüsselt. Das bedeutet, das alle Kommunikationspartner über den gleichen Schlüssel verfügen müssen. Das heißt der Schlüssel wird von einem Kommunikationspartner generiert und dieser versendet den Schlüssel an alle Partner die an der Kommunikation teilnehmen sollen. Dies birgt das Risiko das ein Angreifer den Schlüssel abfängt und von da an die Kommunikation verfolgen kann. Um die Sicherheit zu erhöhen gibt es die Möglichkeit asymmetrische Verahren einzusetzen. Bei diesen Verfahren wird ein Schlüsselpaar generiert. Ein öffentlich zugänglicher Schlüssel und ein geheimer, privater Schlüssel, den nur der Anwender kennt. Mit Hilfe des öffentlichen Schlüssels werden Texte zur Kommunikation verschlüsselt und mit dem privaten Schlüssel wieder entschlüsselt. Jeder Kommunikationspartner generiert ein Schlüsselpaar und versendet den öffentlichen Schlüssel an den Kommunikationspartner. Bei einer Kommunikation wird der Text mit dem erhaltenen geheimen Schlüssel verschlüsselt und an den jeweiligen Teilnehmer übertragen. Dieser entschlüsselt den Text mit seinem privaten Schlüssel. Diese

Verfahren sind erheblich sicherer als symmetrische Verfahren, da ein Angreifer unter Kenntnis des öffentlichen Schlüssels Nachrichten nicht entschlüsseln kann, weil dieser nur zum Verschlüsseln verwendet wird. Aber die Verwendung von asymmetrischen Algorithmen birgt ein großes Problem. Asymmetrische Algorithmen sind wesentlich langsamer als symmetrische Algorithmen. Um den Geschwindigkeitsunterschied zu demonstrieren, wurde ein Java-Programm entwickelt, welches unter Verwendung des Security-Application Programming Interface von Java die Geschwindigkeit von verschiedenen Algorithmen testet. Dazu werden für symmetrische Algorithmen Schlüssel und für asymmetrische Schlüsselpaare generiert. Diese werden dazu genutzt, um Daten zu ver- beziehungsweise zu entschlüsseln. Die Ver- und Entschlüsselungszeiten werden einzeln berechnet. Außerdem werden für die Messung unterschiedliche Schlüssellängen verwendet, da diese für benötigte Zeit mitentscheidend sind. In Abbildung 3.1 sind die Ergebnisse der Geschwindigkeitsmessung verschiedener Algorithmen für die Verschlüsselung und in Abbildung 3.2 sowie in Abbildung 3.3 für die Entschlüsselung von Daten dargestellt. Im Anschluss daran werden einige Merkmale von verschiedenen weitverbreiteten symmetrischen Algorithmen und dem asymmetrischen RSA-Algorithmus vorgestellt (vgl. [Sch96]).

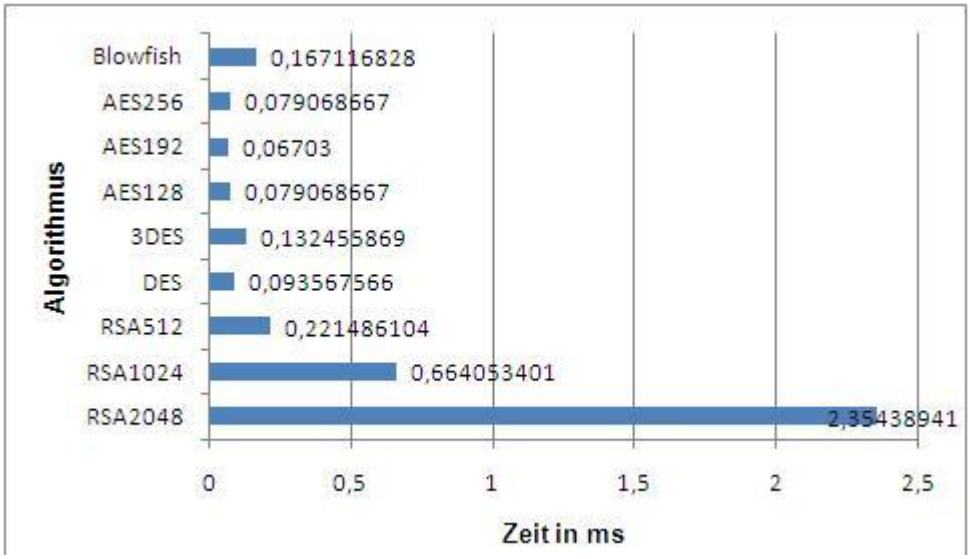

Abbildung 3.1.: Benötigte Verschlüsselungszeit verschiedener kryptografischer Algorithmen

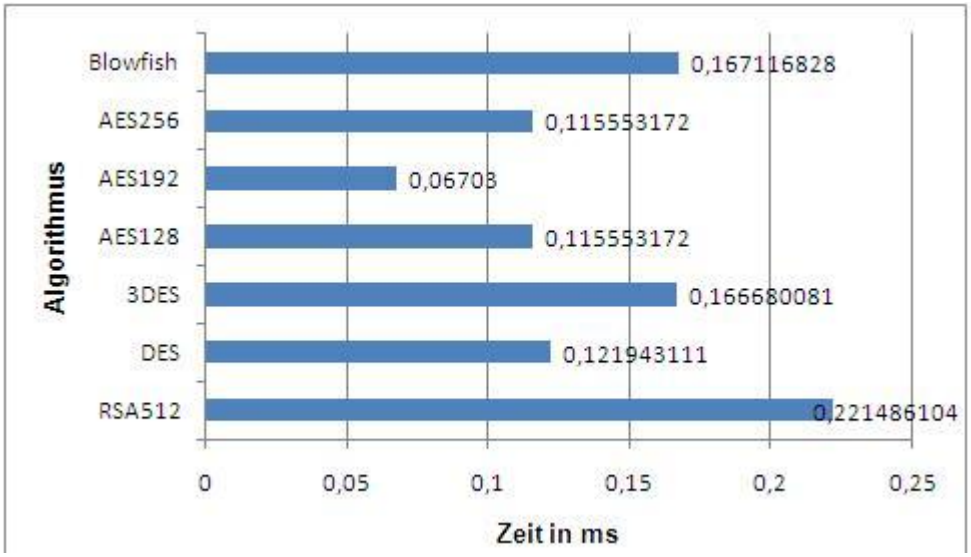

Abbildung 3.2.: Benötigte Entschlüsselungszeit verschiedener kryptografischer Algorithmen

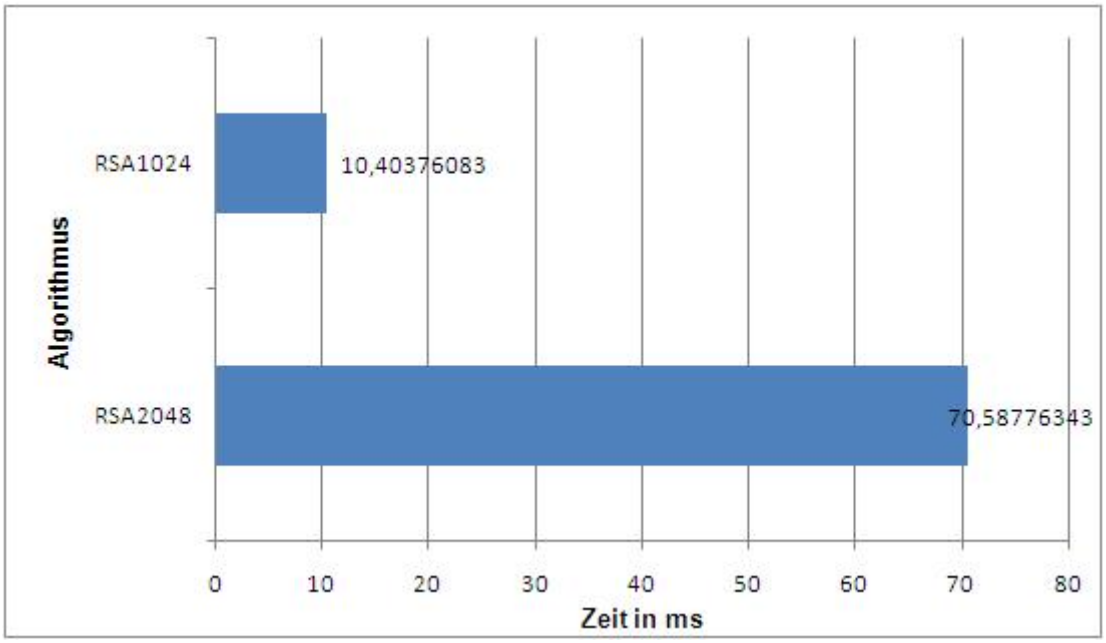

Abbildung 3.3.: Benötigte Entschlüsselungszeit verschiedener kryptografischer Algorithmen

Asymmetrische Algorithmen

Rivest Shamir Adleman

Rivest Shamir Adleman (RSA) ist das bekannteste asymmetrisches Kryptosystem, welches von Ronald L. Rivest, Adi Shamir und Leonard Adleman 1978 entwickelt wurde. Dieses System kann sowohl für die Verschlüsselung als auch für digitale Signaturen genutzt. Das Verfahren basiert auf dem Faktorisierungsproblem für sehr große natürliche Zahlen. Es werden Schlüssellängen bis 2048 Bit unterstützt.

Symmetrische Algorithmen

Man unterscheidet zwei Arten von symmetrischen Algorithmen, Block- und Stromchiffrierungen. Bei Blockchiffrierungen wird der zu bearbeitende Klartext und Chiffretext in Blöcken bearbeitet. Stromchiffrierungen bearbeiten immer ein Bit oder ein Byte von Chiffretext und Klartext([Sch96]). Nachfolgend sind Beispiele für symmetrische Algorithmen gegeben.

- **Data Encryption Standard**
  Der Data Encryption Standard (DES) ist eineBlockchiffre mit 56 Bit effektiver Schlüssellänge. Aufgrund der geringen Schlüssellänge ist der DES nicht mehr sicher.
- **Triple DES**
  Durch eine dreifache Komposition des DES, wird eine Schlüssellänge von 168 Bit erreicht.
- **Blowfish**
  Blowfish ist Blockchiffre, die Schlüssel bis zu einer Bitlänge von 446 Bit unterstützt.
- **Advanced Encryption Standard**
  Der Advanced Encryption Standard (AES) ist eine Blockchiffre und ist der offizielle Nachfolger des DES. Er unterstützt Schlüssellängen von 128, 192 und 256 Bit.

Hashfunktionen

Hashfunktionen sind integraler Bestandteil kryptografischer Infrastrukturen. Mittels starker Hashfunktionen werden digitale Fingerabdrücke von Datenobjekten berechnet. Diese werden zusammen mit dem Objekt versandt. Anhand der Überprüfung des Hashwertes kann sichergestellt werden, das die Authenzität eines Objektes gewährleistet ist . Eine wichtige Anforderung an Hashfuntkionen ist, das sie kollisionsresistent ein müssen. Das bedeuted, dass es praktisch unmöglich sein muss zwei Nachrichten zu finden, welche den gleichen Hash-Wert besitzen (vgl. [Eck09]). Bekannte Vertreter starker Hashfunktionen sind der Secure Hash Algorithm Secure Hash Algorithm (SHA)-2 und der Message Digest Message Digest (MD)-5.

3.1.4. Authentifikationsverfahren im Internet

Einfache Passwortabfrage

Bei der einfachen Passwortabfrage wird ein Passwort zum Verbindungsaufbau übermittelt. Danach wird die gesamte Kommunikation ohne weitere Sicherheitsabfragen durchgeführt. Ein sehr bekanntes Beispiel hierfür ist die **HTTP Basic Authentication**.

Einmal-Passwörter

Für die Authentifikation durch Einmal-Passwörter werden im Vorfeld der Kommunikation mehrere Passwörter vereinbart die jeweils nur einmal verwendet werden dürfen.

Mathematischer Ablauf für die Erzeugung und Verwendung von Einmalpasswörtern nach [Sch01]:

- p_0 - Zufallszahl, kh - kryptografische Hashfunktion, n - Anzahl der generierten Werte

1. Erzeugen von p_0,
2. p_1=kh(p_0), durch Anwendung der kryptografischen Hashfunktion wird ein neuer Wert erzeugt,

3. ...
4. p_n=kh(p_{n-1}), letzer erzeugter Wert,
5. Übergabe von p_0 bis p_{n-1} und Speicherung von p_n,
6. p_{n-1} ist erster zu verwendeter Schlüssel für Authentifikation,
7. Nutzer übergibt zur Authentifikation p_{n-1},
8. Wenn $p_n = kh(p_{n-1}) \hookrightarrow$ Authentifikation erfolgreich und Speicherung von p_{n-1}
9. Nachfolgende Authentifikation mit p_{n-2}

Digitale Signaturen

Eine digitale Signatur wird unter Einsatz von asymmetrischen, kryptografischen Verfahren mit Hilfe eines privaten Schlüssels erzeugt. Mit Hilfe des dazugehörigen öffentlichen Schlüssels kann die Signatur jederzeit überprüft und damit der Inhaber und die Unverfälschtheit der Daten festgestellt werden. Es werden zwei Arten von digitalen Signaturen unterschieden. Zum einen deterministische digitale Signaturen und zum anderen nicht-deterministische digitale Signaturen. Bei deterministischen digitalen Signaturverfahren ist die digitale Signatur durch die Nachricht und den Schlüssel eindeutig festgelegt, bei nicht-deterministischen digitalen Signaturverfahren gehen Zufallswerte in die Signaturberechnung ein, so dass die digitale Signatur zu einer Nachricht und einem Schlüssel viele verschiedene Werte annehmen kann.

Mathematische Formalierung nach [Sch01]:

- m - Nachricht, a - öffentlicher Schlüssel, x - geheimer Schlüssel, s - digitale Signatur, u - Funktion zum Erzeugen der digitalen Signatur, v - Funktion zum verifizieren der Signatur

1. s=u(m,x), Erzeugen der digitalen Signatur,
2. m'=v(a,s), Verifizieren der Signatur,
3. m=m', wenn dieser Sachverhalt gilt ist die Signatur echt.

Kombination von Authentifikationsverfahren

Hybride Verschlüsselung

Dieses Verfahren ist eine Kombination aus der symmetrischen und asymmetrischen Verschlüsselung. Vor der Kommunikation generieren beide Partner einen geheimen und einen öffentlichen Schlüssel. Außerdem generiert einer von beiden einen symmetrischen Schlüssel. Der öffentliche Schlüssel wird wie bei der Erstellung von digitalen Signaturen an den jeweiligen Kommunikationspartner gesendet. Mit Hilfe des erhaltenen öffentlichen Schlüssels wird der symmetrische Schlüssel verschlüsselt und an den Partner versendet. Mit diesem symmetrischen Schlüssel werden die Nachrichten verschlüsselt. Das Public-Key-Verfahren wird also nur eingesetzt, um den Schlüsseldiebstahl zu erschweren und es bleibt der Vorteil der schnellen Datenverschlüsselung durch Nutzung des symmetrischen Verfahrens. Diese Methode ist auf jeden Fall sicherer als die Symmetrische. Dennoch könnte sich ein Angreifer unbemerkt in die Kommunikation einschalten, indem er die öffentlichen Schlüssel der beiden Teilnehmer abfängt. Sich selber zwei Schlüsselpaare generiert und die öffentlichen Schlüssel den Gesprächsteilnehmern zusendet. Damit kann er beim Schlüsselaustausch seinen eigenen privaten Schlüssel verwenden, um an den symmetrischen Schlüssel zu kommen. Damit kann er dann jede Nachricht abfangen und entschlüsseln (vgl. [Sch01]).

Single Sign-On

Das große Problem in der heutigen Zeit ist, dass Nutzer meist für jeden Dienst unterschiedliche Nutzernamen beziehungsweise Passwörter besitzen und sich deshalb unter Umständen viele Authentifikationsdaten merken müssen. Mit Hilfe des Single Sign-On (SSO) soll Abhilfe geschafft werden. Der Nutzer authentifiziert sich gegenüber einem System einmalig und hat danach Zugriff auf verschiedenste Dienste, ohne sich bei jedem Dienst einzeln anmelden zu müssen. Dies ist sehr positiv aus Gründen der Benutzerfreundlichkeit, besitzt aber den entscheidenden Nachteil, das wenn ein Angreifer an die Authentifikationsdaten gelangt, er Zugriff auf alle Dienste erhält. Deshalb ist der SSO für den netzweiten Betrieb eher ungeeignet und wird eher nur unternehmens- beziehungsweise universitätsweit eingesetzt (vgl. [Sch01]).

Plattformübergreifendes Berechtigungsmanagement

Der Einsatz des Plattformübergreifenden Berechtigungsmanagemant findet sich in Systemen mit großer Serverzahl. Damit nicht jeder Nutzer an einem Serversystem neu eingetragen werden muss, wird ein zentraler Server aufgesetzt der Schnittstellen zu allen Servern besitzt. Somit kann der Administrator zentral neue Nutzer hinzufügen und diese auf das entsprechende System beziehungsweise Systeme übertragen lassen. Dadurch, dass Nutzer nicht mehr einzeln auf jedem System eingetragen werden müssen, wird der Administrationsaufwand erheblich reduziert. Der Nachteil bei diesem Verfahren ist die hohe Komplexität, da unterschiedlichste Serversysteme unter einen Hut gebracht werden müssen(vgl.[Sch01]).

Company Cards

Eine weitere Möglichkeit verschiedene Authentifikationsverfahren miteinander zu kombinieren, stellen die Company Cards dar. Sie finden Verbreitung in Unternehmen oder Universitäten. Mit Hilfe solcher Company Cards ist es möglich, sich am Betriebssystem einzuloggen und gleichzeitig ist es möglich, dass der Nutzer durch SSO authentisiert wird. Auf der Karte könnte ein Foto des Nutzers enthalten sein. Somit kann sie auch als Mitarbeiter- oder Studentenausweis dienen(vgl. [Sch01]).

3.2. Die Grundlagen der Autorisierung

3.2.1. Begriffsdefinition

Die Autorisierung ist der Nachweis, eines im Besitz befindlichen Rechtes zu sein, um eine bestimmte Aktion durchführen zu können.

3.2.2. Autorisierungsstrategien

Nach [Eck09] werden Sicherheitsstrategien in zwei Klassen unterteilt. Zum einen existieren Zugriffskontrollstrategien und zum anderen Informationsfluss-Strategien.

Zugriffskontrollstrategien

Discretionary Access Control

Dieses Strategie basiert auf der Grundlage, dass Benutzer selber darüber entscheiden können, wer auf die selbst erzeugten Daten zugreifen darf. Die Umsetzung dieses Konzeptes erfolgt durch Access Control List (ACL)s. Sie werden unter anderem zur Rechteverwaltung in Unix-Systemen eingesetzt. Die Discretionary Access Control ist ein sehr flexibles Autorisierungskonzept.

Access Control Lists

In ACLs stehen nicht immer die Nutzer im Vordergrund die auf bestimmte Ressourcen zugreifen wollen, sondern es können beliebige Objekte sein, die an beliebige andere Objekte Anfragen stellen. Objekte die Anfragen stellen, werden als Access Request Objects bezeichnet und Objekte die Anfragen erhalten als Access Control Objects. Eine ACL muss Zugriffe der Access Request Objects auf die Access Control Objects registrieren, überprüfen und anschließend darüber entscheiden ob der Zugriff gewährt oder gesperrt wird (vgl. [HMDf04]). Tabelle 3.1 zeigt ein Beispiel für eine ACL.

Tabelle 3.1.: Beispiel einer ACL-Matrix (vgl. [Eck09])

	Datei 1	**Datei 2**	**...**	**...**
Nutzer 1	owner,r,w,x	-		
Nutzer 2	r,x	w		
Nutzer 3	-	owner,r,x		

Mandatory Access Control

Bei diesem Autorisierungsmodell werden systemglobale Eigenschaften definiert. Anders als bei der Discretionary Access Control hat der Benutzer kein Entscheidungsrecht darüber, wer auf die von ihm erzeugten Objekte Zugriff hat.

Bell-LaPadula Modell

Beim Bell-LaPadula Modell erfolgt eine Einordnung von Subjekten und Objekten zu Sicherheitsklassen. Damit ist es möglich, Subjekten verschiedene Vertraulichkeitsstu-

fen zuzuordnen, um Zugriffe auf Objekte zu beschränken. Jedem Subjekt wird eine Sicherheitsklasse und jedem Objekt eine Sicherheitsklassifikation zugeordnet. Bei der Authentifikation muss das Subjekt seine Sicherheitsklasse angeben. Diese darf die maximale Sicherheitsstufe nicht überschreiten. Der Zugriff auf Objekte wird durch zwei durch das System festgelegte Regeln bestimmt (vgl. [Eck09]):

- **Simple-Security**
  Diese Regel wird auch als no-read-up Regel bezeichnet, was bedeutet das ein Lesezugriff auf ein Objekt nur gestattet ist, wenn das Subjekt das benötigte Recht besitzt und die Sicherheitsklasse des Subjekts größer oder gleich der Subjektklassifikation ist.

- ***-Eigenschaft**
  Auch bekannt als no-write-down Regel besagt, dass ein schreibender Zugriff auf ein Objekt durch ein Subjekt nur dann zulässig ist, wenn die Sicherheitsklassifikation des Objekts mindestens genauso hoch ist wie die Sicherheitsklasse des Subjekts. Bei einem Append Zugriff muss die Klasssifikation des Objekts mindestens so hoch sein wie die Sicherheitsklasse des Subjekts. Abbildung 3.4 zeigt ein Beispiel für ein Bell-LaPadula Modell.

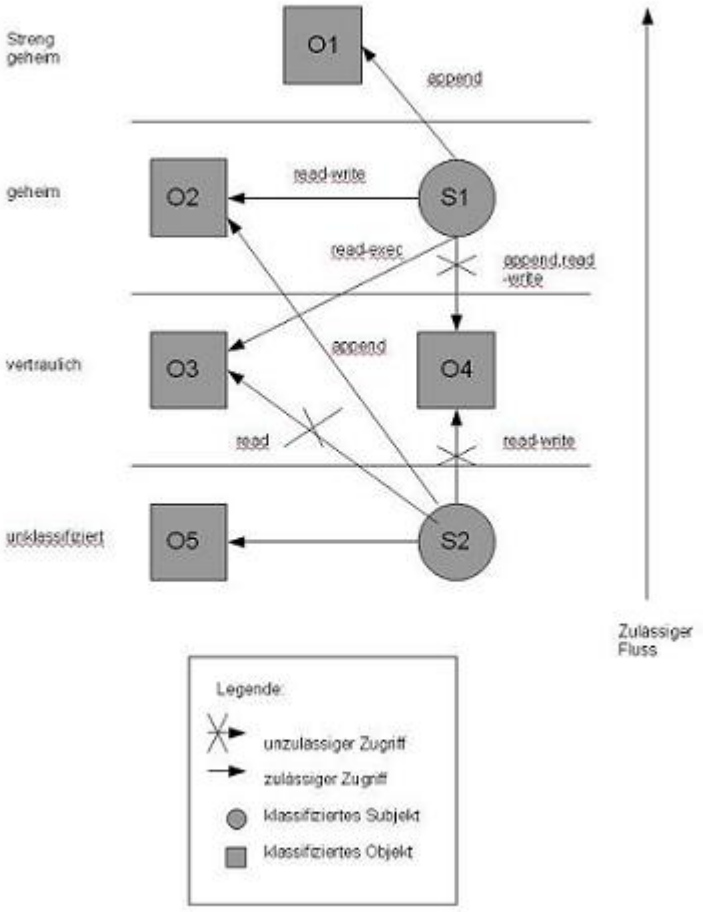

Abbildung 3.4.: Beispiel eines Bell-LaPadula Modells

Role Based Access Control

Das Konzept der rollenbasierten Zugriffskontrolle basiert auf der Zuordnung von Rechten zu Rollen und Benutzer werden Mitglieder geeigneter Rollen. Damit erlangen die Benutzer die der Rolle zugeordneten Rechte. Das hat den entscheidenden Vorteil, das Rechte nicht mehr einzeln jedem Nutzer zugeordnet werden müssen und damit der Verwaltungsaufwand erheblich sinkt.

[SCFY96] beschreibt die rollenbasierte Zugriffskontrolle als ein vierstufiges Konzept (Abbildung 3.5).

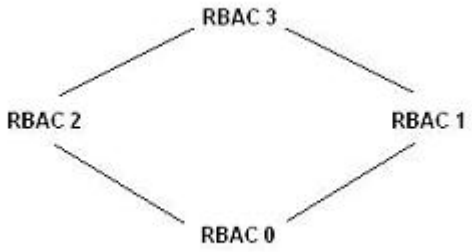

Abbildung 3.5.: RBAC-Modell

$RBAC_0$-Basiskonzept

Nach [SCFY96] hat das $RBAC_0$-Modell folgende Komponenten:

- U, R, P und S (Benutzer, Rollen, Rechte und Sitzungen),
- $PA \subseteq PxR$, eine viele-zu-viele Beziehung zwischen Rolle und Recht,
- $UA \subseteq UxR$, eine viele-zu-viele Beziehung zwischen Benutzer und Rolle
- $Benutzer: S \rightarrow U$, Zuordung von einer Sitzung S_i zu genau einem Benutzer(S_i), und
- $Benutzer: S \rightarrow 2^R$, eine Funktion in der jeder Sitzung S_i ein Set von $Rollen(S_i) \subseteq \{r|(Benutzer(S_i), r) \in UA\}$ zugewiesen wird und die Sitzung S_i hat die Rechte $U_{r \in roles(S_i)}\{p|(p, r) \in PA\}$.

$RBAC_1$ - Rollenhierarchie Die Definition für das $RBAC_1$-Modell lautet nach [SCFY96]:

- U, R, P, S, PA, UA bleiben unverändert,
- $RH \subseteq RxR$, ist die Rollenhierarchie,

- *Benutzer*: $S \rightarrow 2^R$, ist eine Modifizierung des $RBAC_0$-Modells um die Erforderlichkeit von $Rollen(S_i) \subseteq \{\exists r\prime|(Benutzer(S_i), r\prime) \in UA\}$ und die Sitzung S_i besitzt die Rechte $U_{r \in roles(S_i)}\{p|(\exists r\prime\prime \leq r)[(p, r\prime\prime) \in PA]\}$.

$RBAC_2$ - Einschränkungen Nach [SCFY96] lautet die Definition des Role Based Access Control $(RBAC)_2$-Modells: $RBAC_2$ besitzt gegenüber $RBAC_0$ keine Veränderungen, außer das Einschränkungen bezüglich der Rollen gegeben sind, die entscheiden welche Werte eine Rolle annehmen kann.

$RBAC_3$ - Kombination

$RBAC_3$ vereint $RBAC_1$ und $RBAC_2$ miteinander (Vgl.[SCFY96]).

Informationsfluss-Strategien

Bei Informationsfluss-Strategien steht die Informationssicherheit im Vordergrund. Sie haben den Zweck darüber zu entscheiden, in welcher Art und Weise Daten, auf die ein Subjekt Zugriff erlangt hat, weiterverarbeitet werden dürfen. Außerdem werden gültige und verbotene Informationskanäle zwischen Subjekten und Subjektgruppen festgelegt und systemglobale Eigenschaften werden verallgemeinert.

3.3. Angriffsanalyse

3.3.1. Allgemeines Angriffsmuster

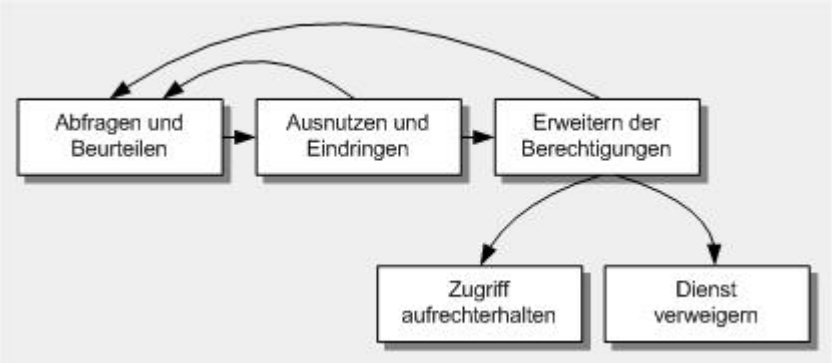

Abbildung 3.6.: Die Anatomie eines Angriffs [msd04]

- **Abfragen und Beurteilen**
  Die Abfrage eines Ziels ist der erste Schritt eines Angreifers. Der Angreifer versucht damit verschiedene Merkmale wie Protokolle, Dienste und Schwachstellen zu analysieren und zu beurteilen, um so einen Zugangspunkt zum System zu finden.

- **Ausnutzen und Eindringen**
  Nachdem die Analyse erfolgt ist, wird versucht, durch die gefundene Schwachstelle in das System einzudringen. Der einfachste Weg für das Eindringen in eine Anwendung besteht für den Angreifer darin, zu versuchen, in die Anwendung einzudringen über den auch berechtigte Subjekte gelangen. Dies könnte eine Anmeldungsseite sein.

- **Erweitern der Berechtigungen**
  Wenn es einem Angreifer gelungen ist in das System einzudringen, besteht der nächste Schritt darin die eigenen Berechtigungen zu erweitern. Das Ziel ist das Erlangen von administrativen Berechtigungen um so ungehinderten Zugriff auf sensible Daten zu erhalten.

- **Zugriff aufrechterhalten**
  Wenn der Zugriff auf das System oder die Anwendung erfolgreich war, stellt der Angreifer sicher das zukünftige Zugriffe für ihn einfacher zu gestalten sind. Außerdem muss der Angreifer versuchen seine Spuren zu verwischen.

- **Dienst verweigern**
  Wenn einem Angreifer der Zugriff versagt bleibt kann er trotzdem noch eine Bedrohung für das System beziehungsweise die Anwendung darstellen. Er könnte versuchen, andere Nutzer an der Nutzung von Diensten zu hindern.

3.3.2. Angreifer-Typen

Angreifer für eine Anwendung oder ein System lassen sich folgenden Kategorien zuordnen (vgl. [Eck09]):

- **Mitarbeiter**
  Bei der Entwicklung eines Sicherheitskonzeptes ist nicht nur zu betrachten das

Angriffe durch externe Personen durchgeführt werden, sondern oftmals durch Mitarbeiter einer Firma durchgeführt werden.

- **Hacker**
  Der Begriff Hacker beschreibt einen technisch sehr versierten Angreifer. Der Hacker versucht Schwachstellen im System aufzudecken un diese auszunutzen. Sie verfolgen meist nicht das Ziel diese für ihren persönlichen Nutzen zu verwenden um beispielsweise finanzielle Gewinne zu erzielen, sondern wenden sich in Regel an die Öffentlichkeit um auf die Verwundbarkeiten aufmerksam zu machen.

- **Cracker**
  Auch der Cracker ist ein technisch sehr versierter Angreifer. Im Gegensatz zu einem Hacker versucht der Cracker die Schwachstellen eines Systems für seinen eigenen persönlichen Nutzen zu verwenden oder auch um gezielt Schäden am System oder bei Personen hervorzurufen.

- **Skript Kiddie**
  Skript-Kiddies verfügen meist nicht über umfangreiches technisches Wissen. Diese Art der Angreifer verwenden frei verfügbare Programme um ihre Angriffe durchzuführen. Das Motiv für einen Angriff ist des Öfteren der Spieltrieb oder auch Neugierde.

- **Wirtschaftsspione**
  Wirtschaftsspione versuchen durch das Abhören von Datenleitungen an geheime beziehungsweise sicherheitskritische Informationen zu gelangen.

3.4. Gefährdungsfaktoren

Nach [Eck09] lassen sich die Gefährdungsfaktoren fünf Kategorien zuordnen. Diese sind Tabelle 3.2 zu entnehmen. Da ein Sicherheitskonzept zur Authentifikation und Autorisierung entwickelt werden soll, sind hauptsächlich die Gefährdungsfaktoren Vorsatz und organisatorische Mängel von Bedeutung.

Tabelle 3.2.: Gefährdungsfaktoren

Höhere Gewalt	**Fahrlässigkeit**	**Technisches Versagen**
Blitzschlag	Irrtum	Stromausfall
Feuer	Fehlbedienung	Hardware-Ausfall
Überschwemmung	unsachgemäße Behandlung	Fehlfunktionen
Erdbeben		
Demonstration		
Streik		
Organisatorische Mängel	**Vorsatz**	
unberechtigter Zugriff	Manipulation	
Raubkopie	Einbruch	
ungeschultes Personal	Hacking	
	Vandalismus	
	Spionage	
	Sabotage	

3.5. Angriffsverfahren im Überblick

Im folgenden werden Angriffsverfahren beschrieben, die im Hinblick für den Entwurf des Sicherheitskonzeptes von großer Bedeutung sind.

Cross Site Scripting

Cross Site Scripting wird auch als Cross Site Scripting (XSS) oder Cross Site Scripting (CSS) bezeichnet. XSS ist das Ausführen von JavaScript oder HTML auf Domain A aber das Laden des Codes erfolgt in Domain B. Der Angreifer versucht von anderen Ressource stammende Skripte nachzuladen. XSS existiert in verschiedensten Ausprägungen (Vgl. [HMDf04]:

- **Reflektives XSS:** Man spricht von reflektivem XSS wenn fremdes JavaScript auf einer Domain ausgeführt wird, wenn der Angreifer bestimmte Parameter manipuliert und diese lediglich in der dem Request folgenden Response des Servers auftauchen (vgl. [HMDf04]).

- **Persistentes XSS:** Wenn es einem Angreifer gelingt nicht nur dann JavaScript auf fremden Domains auszuführen, wenn Parameter manipuliert oder der User auf einen präparierten Link klickt sondern auf anderem Wege den entsprechenden Request feuert, ist von persistentem XSS die Rede (vgl. [HMDf04]).

- **Lazy XSS:** Sind Angriffe auf das Backend. Beispiel Ein Angreifer schleust JavaScript in die Anwendung, erhält aber zunächst kein Feedback, ob das JavaScript ausgeführt wurde oder nicht. Erst wenn zu einem späteren Zeitpunkt der Seitenbetreiber im Backend verschiedene Aktionen durchführt, wird im schlimmsten Fall der Exploit ausgeführt und der Angreifer kann ab dieser Stelle weitermachen. Besonders gut geeignet für die Nutzung von Lazy XSS sind Kontaktformulare, Newsletter und Nutzerstatistiken. Beispielsweise kann ein Angreifer den User-Agent String modifizieren und ersetzt den üblichen Text durch eine Script-Injection, ist es möglich das diese Seiten im Backend gesammelt dargestellt werden und dass die Injection beim nächsten Besuch der Statistiken dargestellt werden. Dies geschieht im Rechtekontext des angemeldeten User (vgl. [HMDf04]).

Hijacking

Beim Hijacking werden Pakete in das Netz eingeschleust, die einen Client blockieren. Der Server erkennt nicht, dass anstelle des Original-Clients ein anderes Prgramm ausgeführt wird. Nach der Übernahme kann ein Angreifer Aktionen im Namen eines Nutzers ausführen(vgl. [Inf]).

Man-In-The-Middle-Angriff

Bei Man-in-the-Middle-Attack (MITMA) steht ein Angreifer zwischen Kommunikatonspartnern. Dabei hört er die Kommunikationsleitungen ab und kann den Datenverkehr manipulieren oder auch Daten komplett abblocken (vgl. [Sch01]).

SQL-Injections

Wird als das Einschleusen von schädlichen Datenbankbefehlen , in Zusammenhang mit Structured Query Language (SQL)-Datenbanken bezeichnet. Diese Technik wird ermöglicht durch eine mangelhafte Eingabeüberprüfung (vgl. [Inf]).

Spoofing

Als Spoofing werden Täuschungsversuche zur Verschleierung der eigenen Identität bezeichnet. Beispielsweise könnte ein Angreifer Nachrichten versenden und diese mit einer falschen IP-Adresse versehen, um seine Identität zu verbergen. (vgl. [Sch01]).

Verkehrsflussanalyse

Mit Hilfe der Verkehrsflussanalyse führt ein Angreifer Statistiken darüber wann und wie eine Kommunikation zwischen zwei Kommunikationspartnern stattfindet. Damit kann der Angreifer feststellen welche Daten für ihn nützlich sein könnten (vgl. [Sch01]).

Denial of Service

Denial of Service Denial of Service (DoS)-Angriffe haben das Ziel die Kommunikation lahmzulegen. Es gibt zwei weitverbreitete Herangehensweisen für diese Art von Angriff. Zum einen könnte der Angreifer Nachrichten abfangen oder so verändern das der Besendete damit nichts mehr anfangen kann. Zum anderen besteht die Möglichkeit, das der Angreifer einen Kommunikationspartner mit gefälschten Nachrichten überschüttet, damit dieser die Echten nicht mehr von den Gefälschten unterscheiden kann (vgl. [Sch01]).

Replay-Angriff

Bei einem Replay-Angriff sammelt der Angreifer mitgehörte Nachrichten und sendet diese zu einem späteren Zeitpunkt selbst an den Empfänger. Ein Angreifer könnte Authentifkationsdaten aufzeichnen und diese verwenden um Zugriff auf das System zu erhalten (vgl. [Inf]).

Ausprobieren von Passwörtern

Man unterscheidet zwei verschiedene Angriffe um Passwörter durch systematisches Ausprobieren herauszufinden (vgl. [Inf]).

- **Brute-Force-Angriff:** Ist das systematische Ausprobieren aller möglichen Zeichenkombinationen bis zu einer bestimmten Länge.
- **Wörterbuch-Angriff:** Darunter versteht man das systematische Ausprobieren anhand einer Liste mit Zeichenkombinationen.

Schad-Software

Es gibt verschiedenste Typen von Schad-Software. Zur Schad-Software gehören unter anderem trojanische Pferde und Viren.

- **Trojanische Pferde:** Ein Trojanisches Pferd ist ein Programm mit einer verdeckten, nicht dokumentierten Funktion oder Wirkung. Nutzer können auf die Ausführung der Funktion keinen Einfluss nehmen (vgl. [Inf]).
- **Viren:** Ein Virus ist eine Befehlsfolge, welches ein Wirtsprogramm zur Ausführung braucht. Viren enthalten meistens einen Schadensteil. Dieser kann durch einen Auslöser aktiviert werden. Des Weiteren ist ein Virus reproduktionsfähig. Bei der Ausführung eines Virus werden mit Hilfe der Kopie eines Virus Speicherbereiche infiziert, die diese Befehlssequenz noch nicht besitzen (vgl. [Eck09]).

Social Engineering

Social Engineering ist eine Angriffsmethode um sich einen Zugang zu unberechtigten Informationen eines IT-Systems zu verschaffen. Es werden menschliche Eigenschaften ausgenutzt. Diese umfassen Vertrauen, Angst oder Respekt vor Autorität (vgl. [Inf]).

4. Analyse

In diesem Kapitel werden die in Kapitel 3 vorgestellten Verfahren und Modelle anhand ihrer Tauglichkeit für die A^3 AdminConsole untersucht und bewertet, um bestmöglichen Schutz für die Authentifikation und Autorisierung zu erreichen. Bevor die Absicherung einer Anwendung beginnen kann, muss im Vorfeld geklärt werden welchen Risiken sie ausgesetzt ist. Dies können bösartige Attacken von Hackern sein oder auch Probleme die durch die Software beziehungsweise Hardware ausgelöst werden.

4.1. Mandantenfähigkeit

Für den Entwurf des Sicherheitskonzepts muss beachtet werden, dass die Mandantenfähigkeit gewährleistet werden soll. „Ein IT-System, welches mehrere Mandanten (zum Beispiel Kunden, Unternehmen oder Auftraggeber) abbilden kann, ohne dass ein gegenseitiger Einbilck in die jeweiligen Daten, Benutzerverwaltung oder Ähnliches möglich ist, wird in der Informationstechnik allgemein als mandantenfähig bezeichnet“ [HG07]. Die Mandantenfähigkeit muss durch das zu entwickelnde Autorisierungskonzept erreicht werden.

4.2. Schutzziele

Zu Anfang muss definiert werden, welche Anforderungen an das Sicherheitskonzept gestellt werden. Dazu werden Schutzziele herausgearbeitet, die vom zu entwerfenden Sicherheitskonzept erfüllt werden müssen (vgl. [Eck09]).

- **Datenintegrität:**

  Die Datenintegrität soll gewährleisten, dass Subjekte keine Daten unbemerkt und unautorisiert manipulieren können. Dies ist zu erreichen durch die Vergabe von Rechten.

- **Vertraulichkeit der Daten**

  Die Anwendung soll gewährleisten, dass keine unautorisierte Datengewinnung möglich ist.

- **Verbindlichkeit:**

  Mit Hilfe der Verbindlichkeit soll sichergestellt werden, das von einem Subjekt durchgeführte Aktionen im Nachhinein nicht von diesem abgestritten werden können. Zugriffe auf sicherheitskritische Objekte und Daten sollten protokolliert werden.

- **Verfügbarkeit:**

  Die Verfügbarkeit soll sicherstellen das authentifizierte und autorisierte Subjekte in der Wahrnehmung ihrer Berechtigungen nicht beeinträchtigt werden dürfen.

- **Authenzität:**

  Unter Authenzität versteht man die Echtheit und Glaubwürdigkeit eines Subjekts beziehungsweise Objekts, die anhand einer eindeutigen Identität und charakteristischen Eigenschaften überprüfbar ist.

4.3. Typische Nutzer

Um die für die A^{3}S AdminConsole erforderlichen Rechte zu bestimmen, werden Definitionen von möglichen Nutzern benötigt, um herauszufinden, welche Rechte benötigt werden.

Scope-Administrator

Der Scope-Administrator verfügt über den größten Satz an Rechten, da er in der Lage sein muss, den Scope, die einzelnen im Scope enthaltenen Domänen, sowie die

den Domänen zugordneten Applikationen zu verwalten. Ein weiteres Aufgabengebiet des Scope-Administrators ist die Nutzerverwaltung. Das beinhaltet das Hinzufügen, Bearbeiten und Löschen von Nutzern. Außerdem soll der Scope-Administrator die Möglichkeit besitzen, Nutzern zu Domänen-Adminstratoren beziehungsweise zu Applikations-Adminstratoren zu machen um den Verwaltungsaufwand zu reduzieren.

Domänen-Administrator

Der Domänen-Administrator verfügt über weniger Rechte als der Scope-Administrator. Zu seinem Aufgabengebiet gehört die Verwaltung einer Domäne, sowie der darin enthaltenen Applikationen. Zu den weiteren Aufgaben gehören das Verwalten von der Domäne zugeordneten Nutzern, der darin enthaltenen Applikationen, sowie das Ernennen von Applikationsadministratoren.

Applikations-Administrator

Zu den Aufgaben eines Applikations-Administrator gehört das Verwalten einer Applikation, sowie der dazugehörigen Nutzer.

Einfache Nutzer

Auf der untersten Hierarchieebene befinden sich die einfachen Nutzer, die somit über die geringsten Rechte verfügen. Zwei Arten von Nutzern werden unterschieden:

- **Applikations-Nutzer:**
  Applikations-Nutzer sind einer Applikation zugeordnet und können diese sowie die dafür vorgesehenen Regeln nutzen.
- **Domänen-Nutzer:**
  Domänen-Nutzer sind einer Domäne zugeordnet und können domänenspezifische Details abrufen.

4.4. Anwendungsszenarien

Um das zu entwerfende Konzept zu veranschaulichen, werden zwei Anwendungsszenarien betrachtet. Für jedes Szenario werden eine Reihe von Anwendungsfällen betrachtet, um anhand dieser das Konzept zu erläutern.

1. Für das erste Szenario wird davon ausgegangen, dass in einem Scope eine Domäne existiert, die viele Applikationen enthält.
2. Beim zweiten Fall existieren in einem Scope viele unterschiedliche Domänen, welche jeweils ein bis zwei Applikationen enthalten.

4.4.1. Anwendungsfall: Domänen-Administrator

Ein Beauftragter eines Unternehmens registriert sich an der Adminconsole. Er erzeugt eine Domäne für seine Firma. Nachdem der Beauftragte registriert ist, wird er zum Domänen-Administrator. Der Domänen-Administrator ist dazu berechtigt Applikationen anzumelden und diese zu verwalten. Außerdem ist er dazu berechtigt neue Nutzer hinzuzufügen und diesen Applikationen zuzuweisen die sie nutzen können. Desweiteren hat er die Möglichkeit, die Administration der von ihm angelegten Applikationen an registrierte Nutzer der A^{3}S AdminConsole zu übertragen. Für das Szenario, dass eine Domain nur ein bis zwei Applikationen besitzt, ist die Übertragung von Administrationsrechten kaum relevant, da der Verwaltungsaufwand überschaubar bleibt. Erst wenn eine Domain viele Applikationen besitzt, wird dieser Fall interessant, weil der Verwaltungsaufwand für eine Person viel zu hoch wäre. Wenn der Domänen-Aministrator einen Applikations-Administrator festlegt, muss die Möglichkeit gegeben sein, dass ein Nutzer mehrere Applikationen verwalten kann. Applikations-Administratoren können zwischen den Applikationen beliebig getauscht und ersetzt werden, aber nur innerhalb einer Domäne. Die Administratoren der Applikationen beziehungsweise der Domain-Administrator besitzt spezielles Insiderwissen. Deshalb muss vor der Registrierung eines neuen Nutzers geprüft werden, ob dieser Nutzer bereits in einem Scope als Administrator tätig war. Wenn dies der Fall sein sollte wird die Registrierung untersagt.

4.4.2. Anwendungsfall: Nutzer

Ein Nutzer registriert sich an der A^{3}S AdminConsole. Dabei übermittelt er seine persönlichen Daten und wählt ein Passwort zur Anmeldung an der A^{3}S AdminConsole, welches entsprechenden Sicherheitsanforderungen genügt. Außerdem wählt er die Domäne in der die Applikation, die er nutzen möchte, angemeldet ist. Nach erfolgreichem Abschluss des Registrierungsvorganges kann sich der Nutzer mit Nutzername und Passwort anmelden. Wenn die Anmeldung erfolgreich war, bekommt der Nutzer einen Sichtbarkeitsbereich in der A^{3}S AdminConsole zugewiesen, in dem er agieren kann. Zu Beginn verfügt der Nutzer über minimale Rechte für seinen Account. Diese sind in einer Standardrolle festgelegt. Der Domänen-Administrator hat die Möglichkeit, die Rechte der Nutzer zu erweitern, durch die Vergabe von zusätzlichen Rollen. Wenn dies der Fall sein sollte, wird der Sichtbarkeitsbereich des Nutzers erweitert um zusätzliche Funktionalitäten. Desweiteren kann der Nutzer zusätzliche Rollen erhalten, indem er bei der Registrierung eine Art Premium-Account wählt oder nachträglich diesen beantragt. Dieser ist zwar kostenpflichtig, aber erweitert die zur Verfügung stehende Funktionalität.

4.4.3. Anwendungsfall: Neue Funktionalität

Ein weiterer zu betrachtender Anwendungsfall stellt das Hinzufügen einer neuen Funktionalität durch einen Entwickler dar. Für eine neue Funktionaliät werden durch die Entwickler bereitgestellte Rechte an die Domänen-Administratoren ausgeliefert. Die Domänen-Administratoren sind dazu berechtigt, diese Rechte zu nutzen beziehungsweise die Rechte oder einen Teil der Rechte den Applikations-Administratoren oder den Nutzern zur Verfügung stellen. Wenn Grundfunktionalitäten hinzugefügt werden, welche die Bedienbarkeit der A^{3}S AdminConsole im Allgemeinen verbessern, sollte auch die Möglichkeit bestehen, die Rechte für diese Funktionaliät sofort allen Nutzern bereitzustellen. Die Entscheidung über die Verteilung der Rechte für eine neue Fuktionalität liegt im Ermessen der Entwickler. Dabei sollte beachtet werden, das alle Personen nur mit den unbedingt notwendigen Rechten versehen werden, die für die Durchführung von Aufgaben nötig sind.

4.5. Durchführen der Bedrohungsanalyse

Anhand einer Bedrohungsanalyse werden Gefahren ermittelt, die dazu führen können, dass das System beschädigt wird. Dabei gibt es zwei verschiedene Ansätze wie eine Bedrohungsanalyse durchgeführt werden kann [Eck09].

4.5.1. Bedrohungsmatrix

Mit Hilfe einer Bedrohungsmatrix werden die Gefährdungsbereiche klassifiziert. Diese bilden die Zeilen der Matrix. In [Eck09] werden fünf Kategorien beschrieben die die wichtigsten Problembereiche abdecken sollen:

1. **Bedrohungen durch externe Angriffe**
   Das sind Aktionen eines Angreifers, mit dem Ziel das bedrohte System physisch zu beschädigen beziehungsweise Systemkomponenten zu entwenden.

2. **Bedrohungen der Datenintegrität und der Informationsvertraulichkeit**
   Diese Art von Angriff zielt darauf ab, Schutzmechanismen des Systems unwirksam zu machen.

3. **Bedrohungen der der Verfügbarkeit und der Ressourcennutzung**
   Darunter versteht man die illegale Nutzung von Ressourcen, sowie die Ressourcenbelegung übersteigende Nutzung von Ressourcen.

4. **Abstreiten durchgeführter Aktionen**
   Bedrohungen dieser Klasse umfassen die Abrechnung von genutzten Diensten und Ressourcen.

5. **Missbrauch erteilter Berechtigungen**
   Diese Bedrohung entsteht wenn ein autorisierter Nutzer, die ihm zugeteilten Rechte missbraucht.

Tabelle 4.1 zeigt ein Beispiel für eine Bedrohungsmatrix dargestellt.

Tabelle 4.1.: Beispiel einer Bedrohungsmatrix

	Programmierer	**interner Benutzer**	**externer Benutzer**	**mobiler Code**
externe Angriffe	Vandalismus	Beobachten der Passworteingabe	-	-
interne Angriffe	direkter Speicherzugriff	logische Bomben	Passwort knacken	Viren
Verfügbarkeit	Speicher belegen	Prozesse erzeugen	Netzlast erzeugen	Monopolisieren der CPU

4.5.2. Bedrohungsbäume

Ein weiteres Verfahren zum Erfassen von Bedrohungen bilden die Bedrohungsbäume. Die Wurzel eines Bedrohungsbaumes stellt das Angriffsziel dar. In den Unterknoten werden Teilziele definiert die erreicht werden müssen, um das Gesamtziel zu erfüllen. Bei Abhängigkeiten von Subzielen werden UND- beziehungsweise ODER-Knoten verwendet. Mit Hilfe von Baumblättern werden einzelne Angriffsschritte beschrieben [Eck09]. Bedrohungsbäume existieren in zwei verschiedenen Ausprägungen, grafische Uml-artige und textuelle Darstellung. Die Bedrohungsanalyse für die A^3S Admin-Console wird mit Hilfe von Bedrohungsbäumen realisiert, da es so besser möglich ist, konkrete Angriffszenarien nachvollziehbar abzubilden. Abbildung 4.2 zeigt ein Beispiel für einen grafischen und Abbildung 4.3 für einen textuellen Bedrohungsbaum. Desweiteren wird daraus schlüssig wann eine bestimmte Art von Angriff stattfinden kann. Es nicht immer möglich einen Schutz gegen alle möglichen Angriffe zu erreichen. Unter Umständen ist dies auch garnicht nötig. Denn wenn zum Erreichen eines Zieles mehrere Teilziele erreicht werden müssen, ist es durchaus möglich nur ein Teilziel zu vereiteln um den den gesamten Angriff zu verhindern.

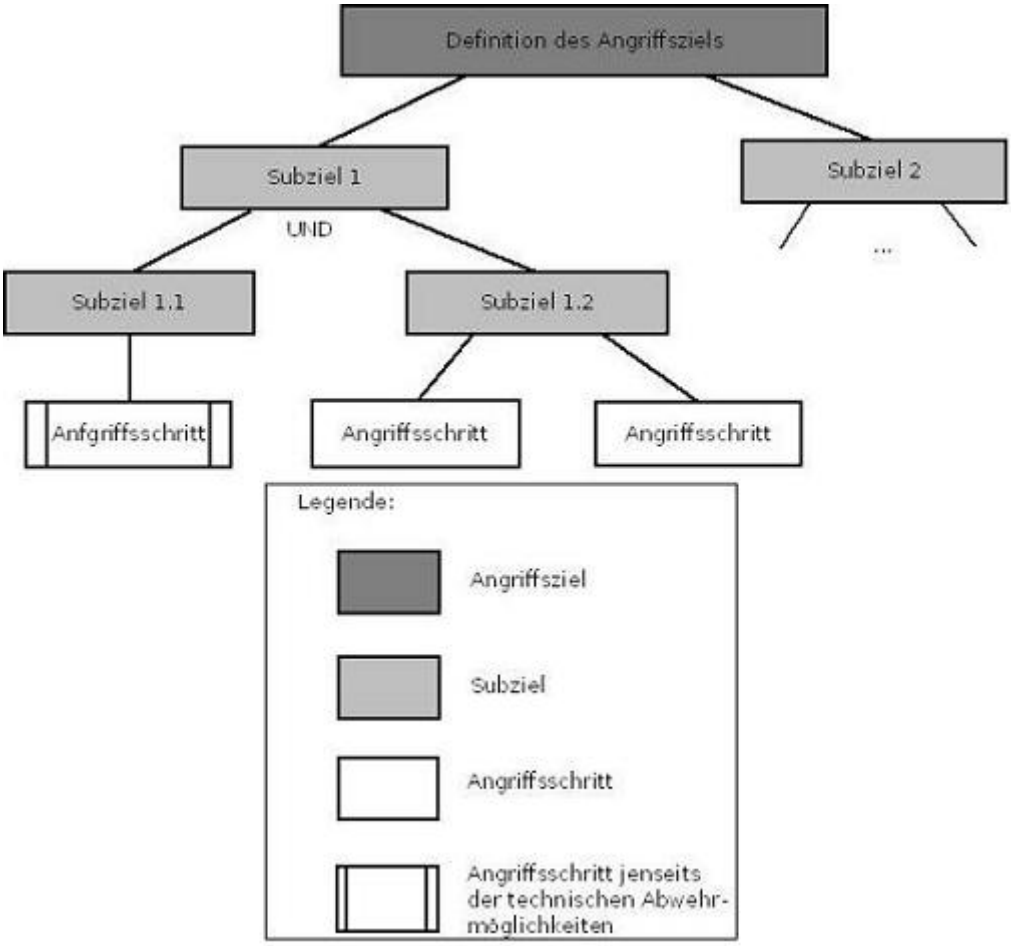

Abbildung 4.1.: Bedrohungsbaum allgemein

In dieser Arbeit werden nur Angriffsszenarien beschrieben, die im Hinblick die Authentifikation und Autorisierung. Nachfolgend sind relevante Angriffsszenarien aufgeführt. Die typischen Angriffsszenarien werden nachfolgend vorgestellt (Abbildung 4.4 bis 4.6).

Definition des Angriffsziels
ODER **Subziel 1**
UND **Subziel 1.1**
ODER **Subziel 1.1.1**
Subziel 1.1.2
UND **Subziel 1.1.2.1**
Subziel 1.1.2.2
Subziel 1.2
ODER Angriffschritt
Angriffschritt
Angriffschritt
Subziel 1.2.1
ODER Angriffsschritt
Angriffsschritt

Abbildung 4.2.: Bedrohungsbaum textuell

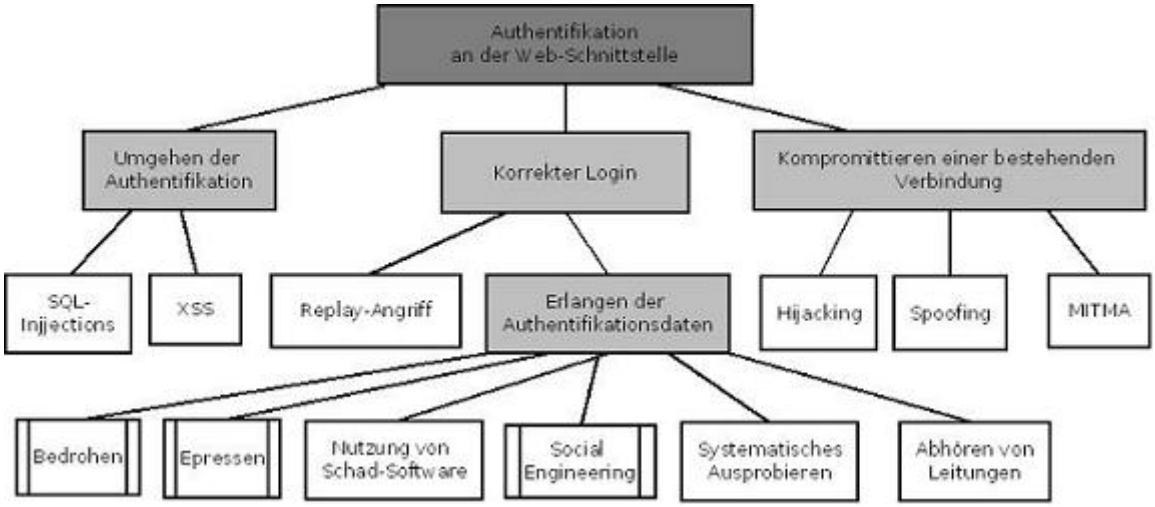

Abbildung 4.3.: Bedrohungsbaum: Authentifikation Web-Schnittstelle

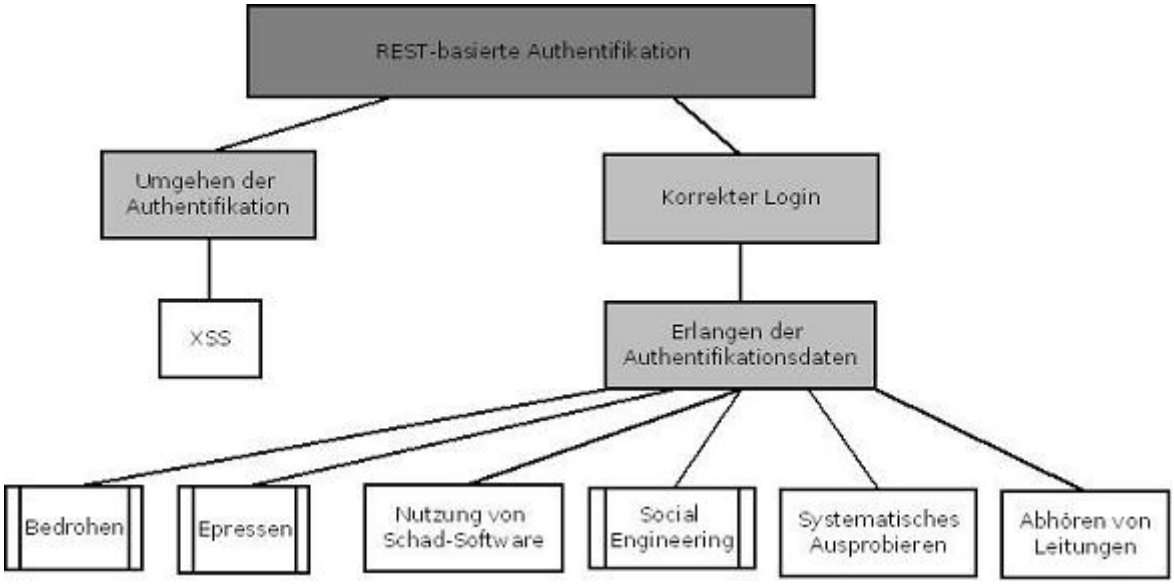

Abbildung 4.4.: Bedrohungsbaum: Authentifikation REST-Schnittstelle

Im Unterschied zur Authentifikation an der Web-Schnittstelle, spielt das Kompromittieren einer bestehenden Sitzung bei der REST-Schnittstelle keine Rolle. Da alle Übertragungen über HTTP übertragen werden, denn HTTP ist ein zustandsloses Protokoll. Das heißt HTTP besitzt kein „Gedächtnis“. Bei jeder Anfrage müssen die Auhtentifikationsinformationen erneut gesendet werden, um diese zu erfüllen (vgl. [LR07]).

Durch die definierten Bedrohungsbäume wurde festgestellt, welche Bedrohungen wann auftreten können und welche Schutzziele beeinträchtigt werden. Nun müssen Gegenmaßnahmen gefunden werden, die die möglichen Bedrohungen verhindert.

4.6. Bewertung von Authentifikationsverfahren

Bei der Erstellung des Authentifikationskonzeptes für die A^{3}S AdminConsole ist zwischen der Authentifikation über die Web-Schnittstelle und der Authentifikation über die REST-Schnittstelle zu unterscheiden. Da sich diese beiden Schnittstellen sehr voneinander unterscheiden, ist es durchaus möglich, dass zwei unterschiedliche Authentifikationsverfahren gefunden werden müssen, damit ausreichende Sicherheit gewährleistet wird.

4.6.1. Bewertung der einfachen Passwortabfrage

Dieses Verfahren ist sehr unsicher, da die Daten unverschlüsselt übertragen werden und somit ein Angreifer die Möglichkeit hat das übermittelte Passwort abzufangen und dann jederzeit wieder einzusetzen. Die einfache Passwortabfrage sollte auf keinen Fall für die Authentifikation verwendet werden. Es ist nicht möglich, die definierten Schutzziele mit diesem Verfahren zu erreichen. Es kann weder festgestellt werden, von wem ein Request stammt, noch kann sichergestellt werden, dass die Integrität der Daten gewährleistet.

4.6.2. Bewertung von Einmal-Passwörtern

Durch die Änderung des Passwortes bei jedem neuen Login, wird es einem Angreifer unmöglich gemacht, ein Passwort ein zweites Mal zu verwenden. Selbst wenn einem Angreifer ein erfolgreicher Angriff in die Passwortdatenbank gelingen sollte, hat er keine für ihn verwertbaren Passwörter. Der Schwachpunkt dieses Systems besteht in der Übertragung der Passwörter an einen Nutzer. Wenn es einem Angreifer gelingen sollte, die übertragenen Passwörter abzufangen, besitzt es so Zugriff zum Account eines Nutzers. Deshalb muss für die Übertragung der Passwortliste der Übertragungskanal gesichert werden. Dieses Verfahren eignet sich durchaus für die Authentifikation an der Web-Schnittstelle. Für die Authentifikation über die REST-Schnittstelle ist dieses Verfahren aber eher ungeeignet, da eine Authentifikation mit jeder gesendeten Anfrage stattfindet und daher jedesmal ein neues Passwort verwendet werden muss.

4.6.3. Bewertung von digitalen Signaturen

Durch eine digitale Signatur lassen sich mehrere Schutzziele erfüllen. Die Authenzität des Inhabers und der Daten wird eindeutig nachgewiesen. Außerdem wird die Datenintegrität gewährleistet. Es lassen sich Bedrohungen wie Replay-, Man-in-the-Middle- und Hijacking-Angriffe ausschließen, wenn jede Nachricht signiert wird. Trotz all dieser Vorzüge werden in der Praxis häufig andere Verfahren bevorzugt. Denn die Erzeugung einer digitalen Signatur kostet relativ viel Zeit. Eine digitale Signatur

eignet sich dadurch ausgezeichnet für Authentifikation über die REST-Schnittstelle. Die Authentifikation dauert länger, aber Sicherheit ist gewährleistet.

4.6.4. Bewertung des SSO

Der SSO stellt sicher eine interesante Möglichkeit dar eine Authentifikation durchzuführen. Der größte Nachteil bei der Verwendung des SSO besteht darin, das wenn ein Angreifer erfolgreich die Login-Informationen eines Nutzers ausspähen konnte, er Zugriff auf alle dem Nutzer zur Verfügung stehenden Anwendungen erhält. Dies ist besonders kritisch für den netzweiten Betrieb. SSO eignet sich eher für einen unternehmensweiten Einsatz. Ein weiteres Problem besteht in der Verfügbarkeit. Die Verfügbarkeit der Anwendung hängt nun auch von der Verfügbarkeit des genutzten SSO-Systems ab. Das bei Problemen des SSO-Systems ist die gesamte Anwendung lahmgelegt, ohne Einfluss darauf nehmen zu können.

4.6.5. Bewertung der biometrischen Authentifikation

Die Biometrische Authentifikation stellt sicherlich eine interessante Möglichkeit dar, aber da die benötigte Hardware nicht zur Verfügung steht, spielt sie für die Verwendung in der A^{3}S AdminConsole keine Rolle.

4.7. Bewertung von Autorisierungsmodellen

Nachfolgend werden die im vorigen Kapitel beschriebenen Autorisierungsmodelle hinsichtlich ihrer Einsetzbarkeit in der A^{3}S AdminConsole untersucht, um so ein passende Autorisierugsmöglichkeit herauszuarbeiten.

4.7.1. Bewertung von Access Control Lists

Ein großes Problem bei diesem Konzept ist die schwierige Handhabung und das jeder Nutzer die Rechteanpassung selbst durchführen muss. Jedes Objekt muss über Zugriffsliste verfügen, in denen die Berechtigungen der einzelnen Nutzer auf dieses

Objekt festgelegt ist. Dadurch können riesige Listen entstehen, was ressourcentechnisch irgendwann zu einem Engpass führen könnte. Jedesmal wenn ein Zugriff auf das zu schützende Objekt durchgeführt wird, muss die gesamte Liste auf die Berechtigung untersucht werden, was unter Umständen bei großen Listen sehr viel Zeit kostet. Wenn neue Objekte beziehungsweise Subjekte dem System hinzugefügt werden, muss jedes Subjekt einzeln in die entsprechenden Listen eingetragen beziehungsweise bei dem Hinzufügen von Objekten muss ein neue ACL erzeugt werden.

4.7.2. Des Bell-LaPadula Modell

In diesem Modell werden die Zugriffsrechte als universelle Rechte festgelegt, was dazu führt, dass bestimmte Zugriffe auf Objekte von Vornherein verboten werden. Ein großes Problem beim Bell-LaPadula Modell besteht darin, dass Subjekte auf höher klassifizierte Objekte schreibend zugreifen dürfen. Damit scheidet dieses Modell aus. Denn der Zugriff von niederen Subjekten auf höher klassifizierte Objekte darf nicht stattfinden.

4.7.3. Bewertung des RBAC-Modells

Ein großer Vorteil der RBAC besteht darin das Berechtigungen für den Zugriff auf Objekte aufgabenorientiert vergeben werden können. Damit ist dieses Konzept sehr flexibel. Durch die RBAC ist es möglich den Verwaltungsaufwand erheblich zu reduzieren. Durch die Definition von Rollen, müssen Rechte nicht mehr einzeln, immer wieder an gleichartige Subjekte vergeben werden sondern es reicht die Übertragung einer Rolle. Bei sehr vielen verschiedenen Nutzerprofilen ist der Aufwand für die Umsetzung viel zu groß, da jedes Profil über einen anderen Satz an Rollen verfügen muss und damit der Vorteil der Zusammenfassung von Rechten zu Rollen verloren geht. Für den Einsatz in der A^{3}S AdminConsole ist dieses Modell absolut geeignet, da die Anzahl unterschiedlicher Nutzertypen bergrenzt ist (Siehe 4.3).

4.8. Fazit

In diesem Kapitel wurden Schutzziele ermittelt, welche durch den Entwurf eines Sicherheitskonzeptes erfüllt werden müssen. Desweiteren wurde ermittelt, zu welchen Zeitpunkten mögliche Bedrohungen auftreten können. Dies ist beim Entwurf zu bedenken. Außerdem wurden die in Kapitel 3 vorgestellten Modelle und Verfahren für die Authentifikation und Autorisierung hinsichtlich ihrer Tauglichkeit untersucht und bewertet. Für die Realisierung der Authentifikation an der Web-Schnittstelle wird das Einmal-Passwort-Verfahren verwendet. Um eine Authentifikation über die REST-Schnittstelle zu entwerfen wird eine digitale Signatur verwendet. Für die Realisierung der Autorisierung wird das RBAC-Modell benutzt.

5. Konzeptentwurf

Dieses Kapitel beschreibt den Entwurf eines Sicherheitskonzeptes auf Basis der für tauglich befundenen Verfahren zur Authentifikation und Autorisierung

5.1. Entwurf des Authentifikationskonzeptes

5.1.1. Authentifikation über die REST-Schnittstelle

Anforderungen an eine digitale Signatur

Um eine Realisierung der Authentifikation mit Hilfe von digitalen Signaturen zu erreichen, werden folgende Anforderungen an die Signatur gestellt(vgl. [Eck09]):

1. **Zweifelsfreie Identität:** Sie bestätigt dem Empfänger des Dokuments, dass der Unterzeichner das Dokument persönlich unterzeichnet hat.
2. **Fälschungssicher:** Es ist bewiesen, dass nur der Sender und kein anderer das Dokument unterzeichnet haben kann.
3. **Nicht Wiederverwendbarkeit:** Die Signatur ist Teil des Dokuments und kann nicht in anderen Dokumenten wiederverwendet werden.
4. **Unveränderbarkeit:** Nachdem das Dokument signiert wurde, können keine Änderungen an diesem vorgenommen werden.
5. **Verbindlichkeit:** Der Sender kann nicht leugnen, dass er das Dokument signiert hat.

Faktoren für die Realisierung

Es sind mehrere Faktoren zu berücksichtigen, damit eine digitale Signatur ausreichende Sicherheit gewährleistet.

Richtlinien für Schlüssel

Bei der Übertragung kann es vorkommen, dass Schlüssel beschädigt werden, und somit Teile des Chiffretextes oder sogar der gesamte Chiffretext nicht mehr entschlüsselt werden kann. Deshalb sollten Schlüssel mit Bits zur Fehlererkennung und Fehlerkorrektur übertragen werden(vgl. [Sch96]). Die Gültigkeitsdauer von Schlüsseln sollte auf jeden Fall begrenzt sein. Es wird eine Mindestlänge für Schlüssel gefordert. Für den Einsatz von RSA wird eine Schlüssellänge von 1024 Bit gewählt. Eine Verschlüsselung mittels RSA bei der gewählten Schlüssellänge konnte bisher noch nicht geknackt werden.

Auswahl der Hashfunktion

Die SHA-2 Familie

Zur Verwendung für die digitale Signatur wird ein Algorithmus der SHA-2 Familie verwendet. Das [Inf] empfiehlt eine Verwendung des SHA-2 Algorithmus. Die SHA-2-Familie umfasst SHA-224, SHA-256, SHA-384, SHA-512 . Die zur SHA-2-Familie gehörenden Algorithmen unterscheiden sich hinsichtlich der Länge der Hash-Werte. Je länger der Hash-Wert ist, desto sicherer ist der Algorithmus. Mit größerer Hash-Länge steigt aber der Aufwand zur Berechnung.

Parameter für die Signatur

Im Anschluss daran werden Parameter definiert, die in die Signierung einer Nachricht einfließen. dazu gehören der URI, ApplicationKey der für die eindeutige Zuordnung zu einer Applikation verwendet wird Applikation wird und der Hashwert der Daten die Übertragen werden.

Entwurf der Signatur

Für die Nutzer A^{3}S AdminConsole wird ein Programm zur Verfügung gestellt, welches RSA Schlüsselpaare generiert. Ein Nutzer kann dieses Prgramm downloaden. Danach lässt er sich ein Schlüsselpaar, bestehend aus privatem und öffentlichem

Schlüssel, generieren. Den privaten Schlüssel speichert er ab und den öffentlichen Schlüssel sendet er an die A^{3}S AdminConsole. Wenn der Nutzer nun Anfragen an die REST-Schnittstelle stellt, benötigt er einen dringend erforderlichen HTTP-Header, den Authorization-Header. Dieser enthält die erzeugte Signatur. Wenn Requests an die A^{3}S AdminConsole geschickt werden, wird die im Authorization-Header enthaltene Signatur überprüft. Wenn die Signatur allen Anforderungen entspricht wird der Zugriff gestattet auf die entsprechende Ressource gewährt. Für den Fall, dass der Auhtorization-Header fehlen sollte beziehungsweise die im Authorization-header enthaltene Signatur falsch sein sollte, wird der Statuscode 403 Forbidden zurückgegeben.

5.1.2. Authentifikation über die Web-Schnittstelle

Entwurf des Einmal-Passwort-Verfahrens

Für die Realisierung einer Authentifikation über die Web-Schnittstelle werden Einmal-Passwörter verwendet. Der Ablauf für Erzeugung und Übergabe der Passwortliste ist nachfolgend dargestellt.

1. Erzeugen der Zufallszahl p_0,
2. p_1=SHA-512(p_0), durch Anwendung SHA-512 wird ein neuer Wert erzeugt,
3. ...
4. p_n=SHA-512(p_{n-1}), letzer erzeugter Wert,
5. Übergabe der Passwortliste and den Kunden von p_0 bis p_{n-1} und Speicherung von p_n,
6. p_{n-1} ist erster zu verwendeter Schlüssel für Authentifikation,
7. Nutzer übergibt zur Authentifikation p_{n-1},
8. Wenn $p_n = SHA - 512(p_{n-1}) \hookrightarrow$ Authentifikation erfolgreich und Speicherung von p_{n-1}
9. Nachfolgende Authentifikation mit p_{n-2}

Kanalverschlüsselung

Für die Übergabe der Passwortliste sollten die Daten unbedingt abhörsicher mittels Secure Socket Layer Secure Socket Layer (SSL) übertragen werden. Für diese Form der Kanalabsicherung existiert die HTTP-Erweiterung Hypertext Transfer Protocol Secure (HTTPS), in dem SSL implementiert ist. Dabei wird eine eine sichere SSL Verbindung zum Server aufgebaut, über die dann die Daten übertragen werden.

Fehlermeldungen bei Falscheingaben

Ein wichtiger Punkt, der ebensfalls auf keinen Fall außer Acht gelassen werden sollte, ist die angemessene Reaktion auf fehlerhafte Eingaben. Das Problem vieler Web-Applikationen besteht darin, dass sie potenziellen Angreifern zu detailierte Meldungen liefern bei Falscheingaben. Ein Beispiel dafür wäre ein gescheiterter Login-Versuch. Es sollte auf keinen Fall genau spezifiziert werden, ob Nutzername oder Passwort falsch war. Sondern die Meldung sollte allgemein gehalten werden wie zum Beispiel „Login Failed!“. Damit ist es möglich, von Vornherein Angriffe wie das systematische Ausprobieren von Anmeldungsinformationen zu erschweren.

5.2. Entwurf des Autorisierungskonzeptes

5.2.1. Das Entity Relationship Modell

Auf der Basis des RBAC-Modells wird ein Entity Relationship Modell (ERM) erstellt in dem der Zusammenhang zwischen allen, für den Entwurf des Autorisierungsmodells benötigten Komponenten gezeigt wird (Siehe Abbildung). Das ERM besteht aus folgenden Komponenten:

- **Session:** Session repräsentiert die Sitzung eines Nutzers. In dieser Sitzung werden, die dem Nutzer zugeordneten Rollen aktiviert
- **Role:** Stellt eine Sammlung von Rechten dar.
- **User:** Umfasst die Daten von Nutzern

- **Application:** Respräsentiert die durch die A^{3}S AdminConsole zu verwaltenden Applikationen
- **Permission:** Repräsentiert die zum Durchführen von Aktionen benötigten Rechte.
- **Scope:** Stellt den Sichtbarkeitsbereich eines Unternehmens dar.
- **Domain:** Repräsentiert einen dem Scope zugeordneten Container für Applikationen eines Themengebietes.

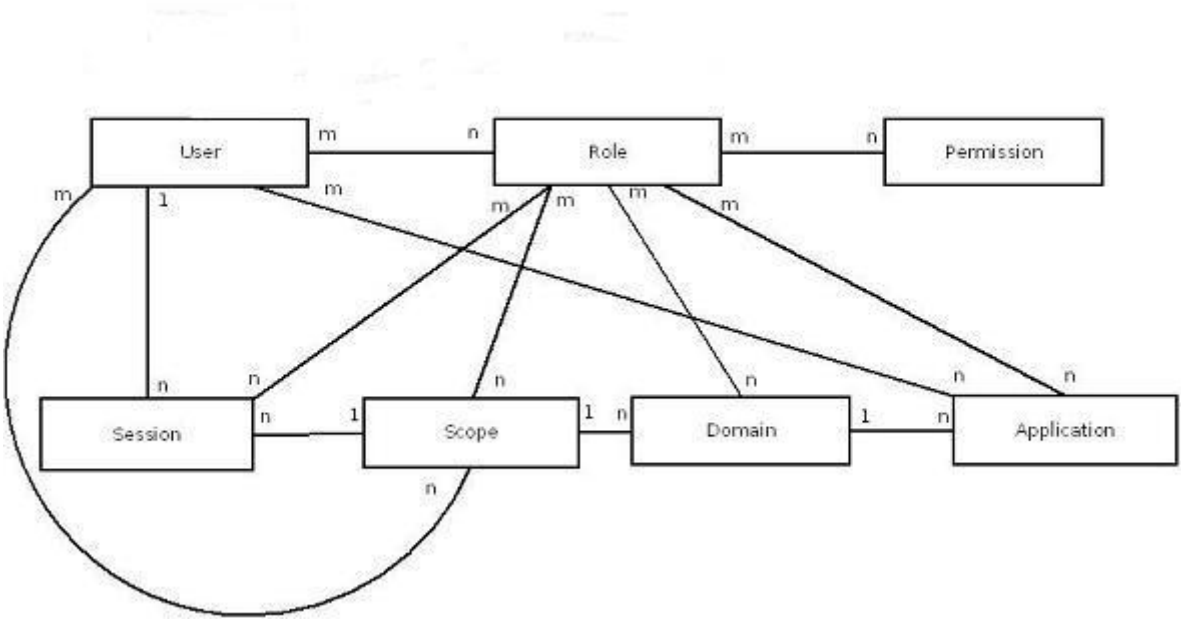

Abbildung 5.1.: Entity Relationship Grobmodell

5.2.2. Rechteentwurf

Anhand der funktionalen Anforderungen wird nun eine Liste der benötigten Rechte zusammengestellt(Siehe A.2.1). Für den Rechteentwurf werden verschiedene Hierarchiestufen eingeführt, um eine Sicherheitsstufe der einzelnen Rechte zu erreichen. In Abbildung 5.2 ist der Zusammenhang zwischen den einzelnen Hierarchiestufen dargestellt. Hierarchiestufe 1 ist die höchste Hierarchiestufe und Hierarchiesutfe 4 die niedrigste Hierarchiestufe.

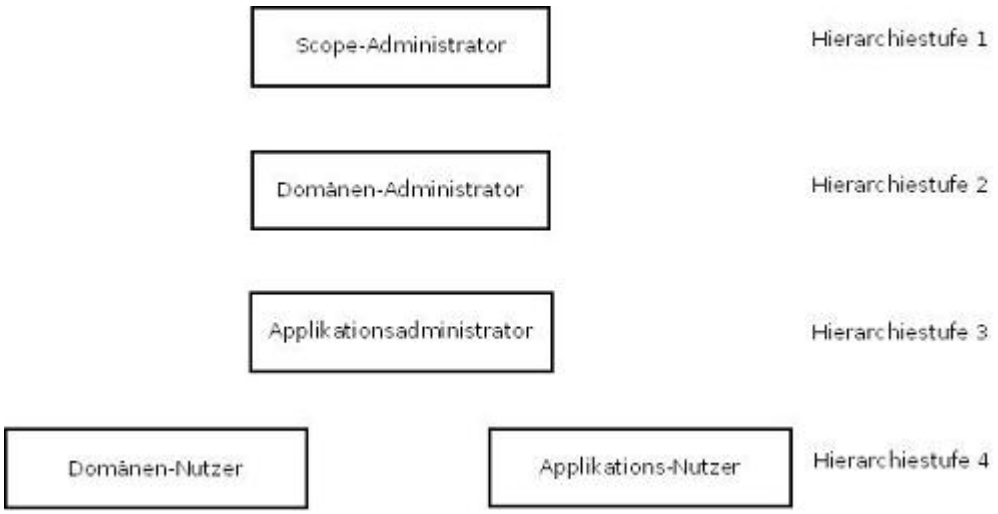

Abbildung 5.2.: Hierarchiestufen der A^{3}S AdminConsole

Die definierten Rechte werden den einzelnen Nutzertypen zugeordnet.

5.2.3. Rollenentwurf

Für den Rollentwurf werden zwei Arten von Rollentypen eingeführt. Zum Einen Steuerungsrollen um die allgemeine Rollenverwaltung zu handhaben und zum Anderen Rollen, die für dieNutzung in der A^{3}S AdminConsole verwendet werden.

Meta-Rollen

Meta-Rollen werden benötigt zur Steuerung der Autorisierung. Durch diese Rollen soll die Vergabe von Rollen und Rechten an Nutzer geregelt werden.

Rolleneinschränkungen

Für die Rollen und die darin enthaltenen Rechte gelten folgende Einschränkungen:

1. Es können beliebige Rechte zu einer Rolle zusammengefasst werden. Die Hierarchiestufe der daraus folgenden Rolle ist gleich der Hierarchiestufe der Rechte.
2. Rollen können in der selben oder in höheren Hierarchiestufen genutzt werden.

Alle Rollen unterliegen einer Hierarchiestufe um die unberechtigte Vererbung von Rollen zu verhindern und damit das Erlangen sicherheitskritischen Rechten.

Fallbeispiel: Rollenvergabe
Für das Beispiel werden die in definierten Rollen verwendet. Scope-Administrator ernennt Nutzer, der bisher über die DefaultUserRole verfügt, zum Domänen-Administrator einer Domäne die zwei Applikationen enthält(App1 und App2). Der Domänenadministrator möchte einem Nutzer der über die DefaultUserRole verfügt das Recht uploadAppKey entziehen für App1 entziehen.

Scope-Admin Rollen:
Der Scope-Administrator besitzt alle definierten Rechte und Rollen. Er stattet den Domänen-Administrator mit den benötigten Rollen aus, die er laut Definition zum erfüllen seiner Aufgaben benötigt. Gestattet sind zwei Vorgehensweisen.

1. Der Scope-Administrator verwendet die vorgegebenen Rollen. Er übergibt die DomainManagementRole, die ApplicationManagementRole, die UserModifyingRole, die RuleManagementRole sowie die Meta-Rollen RoleManagementRole und die RoleToUserRole . Er kann Rollen Nutzern zuweisen, da er über die Meta-Rolle RoleToUserRole verfügt. Der Domänen-Administrator besitzt nun alle in den Rollen enthaltenen Rechte. Diese kann er je nach Bedarf zu neuen Rollen zusammenfassen. Der Domänen-Administrator kann das Entziehen des uploadAppkey Rechtes durch zwei Arten erreichen. Die erste Möglichkeit besteht darin, für den Nutzer eine neue Rolle zu erstellen in der das uploadAppKey-Recht nicht vorhanden ist. Diese Rolle weist er dem Nutzer zu und entzieht ihm die DefaultUserRole in der das Recht vorhanden ist. Die zweite Möglichkeit besteht darin das uploadAppKey-Recht aus der DefaultUserRole zu entfernen. Er darf dies weil er über die RoleManagementRole verfügt in der das Recht removePermission vorhanden ist.

2. Der Scope-Administrator erzeugt aus dem ihm zur Verfügung stehenden Rechten neue Rollen, die er dem Domänenadministrator übergibt.

6. Zusammenfassung

Ziel dieser Diplomarbeit war es, ein Sicherheitskonzept für die Authentifikation und Autorisierung zu der A^{3}S AdminConsole entwickeln.
Auf der Basis von vorhandenen Modellen, Verfahren und Algorithmen ist ein Sicherheitsmodell entwickelt wurden, welches die in Kapitel 4 definierten Sicherheitsziele erfüllt.

Für die Authentifikation über die REST-Schnittstelle wird eine digitale Signatur genutzt, welche starke kryptografische Verfahren nutzt, um ein Höchstmaß an Sicherheit zu gewährleisten. Zur Authentifikation über die Web-Schnittstelle werden Einmal-Passwörter eingesetzt. Diese werden durch eine kollisionsresistente Hashfunktionen erzeugt.

Durch die Verwendung von starken kryptografischen Verfahren und der Nutzung von großen Schlüssellängen sowie der Nutzung von SSL ist ein hohes Maß an Sicherheit gewährleistet, aber auf Kosten der Geschwindigkeit. Er ist eine prototypische Umsetzung wird zeigen, wie es sich in der Praxis mit der Geschwindigkeit verhält.

Das Autorisierungsmodell wurde auf Basis des RBAC-Modells entworfen. Es wurden Rechte definiert, welche den Zugriff auf Methoden und Objekte regeln. Durch den Einsatz von Hierarchiestufen für Rechte wird der Zugriff für Nutzer beschränkt. Danach wurden gleichartige Rechte zu sinnvollen Rollen zusammengefasst.

7. Ausblick

Nach der Umsetzung des Sicherheitskonzepts, sollten im nächsten Schritt Penetrationstests durchgeführt werden, um mögliche Schwachstellen in der Implementierung festzustellen. Penetrationstests umfassen unter anderem das Aufzeichnen und Manipulieren des Netzverkehrs oder auch das Ausnutzen bekannter Software-Schwachstellen (vgl. [Eck09]). Da es oftmals nicht möglich ist, festzustellen ob ein Angriff stattgefunden hat, empfiehlt es sich nach der Umsetzung des Sicherheitskonzepts eine Implementierung eines Intrusion Detection System (IDS) zu verwenden. Man könnte ein solches System als eine Art Alarmanlage für Web-Anwendungen betrachten.

A. Anhang

A.1. Rechtedefinitionen

- **acceptUser:** Akzeptieren einer Nutzerregistrierung. Hierarchieebene:1
- **createUser:** Anlegen eines Nutzers. Hierarchieebene:1
- **modifyUsers:** Bearbeiten der persönlichen Daten eines beliebigen Nutzers. Hierarchieebene:3
- **modifyUserSelf:** Bearbeiten der eigenen persönlichen Daten. Hierarchieebene:4
- **deleteUser:** Löschen eines Nutzers. Hierarchieebene:1
- **showUserSelf:** Anzeigen der persönlichen Daten. Hierarchieebene:4
- **showUsers:** Anzeigen aller Nutzer. Hierarchieebene:1
- **lockUser:** Sperren eines Nutzers. Hierarchieebene:3
- **unlockUser:** Entsperren eines Nutzers. Hierarchieebene:3
- **activateRule:** Aktivieren einer Regel für eine Applikation. Hierarchieebene:3
- **deactivateRule:** Deaktivieren einer Regel für eine Applikation. Hierarchieebene:3
- **createRule:** Anlegen einer neuen Regel für eine Applikation. Hierarchieebene:3
- **modifyRule:** Bearbeiten einer vorhandenen Regel. Hierarchieebene:3
- **deleteRule:** Löschen einer Regel. Hierarchieebene:3
- **showRuleDetails:** Anzeigen der Details einer Applikationsregel. Hierarchieebene:4
- **showAllApps:** Anzeigen aller vorhandenen Applikationen. Hierarchieebene:1
- **showDomainApps:** Anzeigen aller in einer Domäne vorhandenen Applikationen. Hierarchieebene:2

- **showAppDetails:** Anzeigen der Details für eine Applikation. Hierarchieebene:4
- **createDomain:** Erzeugen einer neuen Domäne. Hierarchieebene:1
- **modifyDomain:** Bearbeiten einer Domäne. Hierarchieebene:2
- **deleteDomain:** Löschen einer Domäne. Hierarchieebene:
- **showDomains:** Anzeigen aller im Scope enthaltenen Domänen. Hierarchieebene:1
- **showDomainDetails:** Anzeigen der Daten einer Domäne. Hierarchieebene:4
- **dataUploadDomain:** Hochladen von Domänen-Daten. Hierarchieebene:4
- **showScopeDetails:** Anzeigen der Scope-Daten. Hierarchieebene:1
- **modifyScope:** Ändern von Scope-Daten. Hierarchieebene:1
- **uploadAppKey:** Hochladen eines Applikationsschlüssels. Hierarchieebene:4
- **createApp** Anmelden einer neuen Applikation. Hierarchieebene:3
- **changeAppConfig** Ändern von Applikationsdaten. Hierarchieebene:3
- **deleteApp** Entfernen einer Applikation. Hierarchieebene:3
- **createRole** Erstellen einer neuen Rolle. Hierarchieebene:3
- **addPermissionToRole** Hinzufügen eines Rechtes zu einer Rolle. Hierarchieebene:3
- **removePermissionFromRole** Entfernen eines Rechtes von einer Rolle. Hierarchieebene:3
- **deleteRole** entfernen einer Rolle. Hierarchieebene:3
- **grantRoleToUser** ändern von Rolle. Hierarchieebene:3
- **revokeRoleToUser** entfernen einer Rolle. Hierarchieebene:3

A.2. Rollendefinitionen

A.2.1. Nutzer-Rechte-Zuordnung

Tabelle A.1.: Zuordnung von Rechten zu möglichen Nutzertypen

Rechte	**Scope-Admin**	**Domänen-Admin**	**App-Admin**	**Domänen-Nutzer**	**App-Nutzer**
acceptUser	√	X	X	X	X
createUser	√	X	X	X	X
modifyUser	√	X	X	√	X
modifyUserSelf	√	√	√	√	√
deleteUser	√	X	X	X	X
showUsers	√	X	X	X	X
showUserSelf	√	√	√	√	√
lockUser	√	√	√	X	X
unlockUser	√	√	√	X	X
activateRule	√	√	√	X	X
deactivateRule	√	√	√	X	X
createRule	√	√	√	X	X
modifyRule	√	√	√	X	X
deleteRule	√	√	√	X	X
showRule-Details	√	√	√	√	√
showAllApps	√	X	X	X	X
showDomain-Apps	√	√	X	X	X
showApp-Details	√	√	√	X	X
showDomain-Details	√	√	X	X	X
createDomain	√	X	X	X	X
modifyDomain	√	√	X	X	X
deleteDomain	√	X	X	X	X
dataUpload-Domain	√	√	√	√	√
showDomains	√	X	X	X	X
showScope-Details	√	X	X	X	X
modifyScope	√	X	X	X	X
uploadAppKey	√	√	√	X	√
createApp	√	√	√	X	X
changeApp-Config	√	√	√	X	X
deleteApp	√	√	X	X	X
createRole	√	√	√	X	X

addPermission-ToRole	√	√	√	X	X
removePer-missionFromRole	√	√	√	X	X
deleteRole	√	√	√	X	X
grantRole-ToUser	√	√	√	X	X
revokeRole-FromUser	√	√	√	X	X

A.2.2. Standardrollen

UserModifyingRole: Stufe 1

- lockUser
- unlockUser

DomainManagementRole: Stufe 2

- modifyDomain
- showDomainDetails

ScopeManagementRole: Stufe 1

- createDomain
- modifyScope
- showScopeDetails
- createUser
- acceptUser
- modifyUser
- deleteUser

ApplicationManagementRole: Stufe 3

- createApp
- changeAppConfig
- deleteApp

showAllRole: Stufe 1

- showAllApps
- showAllDomains
- showUsers

DefaultUserRole: Stufe 4

- uploadAppKey
- dataUploadDomain
- showUserSelf
- modifyUserSelf
- showRuleDetails
- showAppDetails

RuleManagementRole: Stufe 3

- activateRule
- deactivateRule
- createRule
- modifyRule
- deleteRule

A.2.3. Meta-Rollen

Meta-Rollen werden benötigt zur Steuerung der Autorisierung. Durch diese Rollen soll die Vergabe von Rollen und Rechten an Nutzer geregelt werden.

RoleManagementRole: Stufe 3

- createRole
- addPermissionToRole
- removePermissionFromRole
- deleteRole

RoleToUserRole: Stufe 3

- grantRoleToUser
- revokeRoleToUser

Literaturverzeichnis

[Eck09] Eckert, Claudia: *IT-Sicherheit: Konzepte-Verfahren-Protokolle*. Bd. 5. Auflage. Oldenbourg, 2009. – ISBN 978–3–486–58270–3

[EL07] Eberling, Werner ; Lessner, Jan: *ENTERPRISE JavaBeans 3: DAS EJB3-PRAXISBUCH FÜR EIN- UND UMSTEIGER*. Hanser, 2007. – ISBN 978–3–446–41085–5

[HG07] Hesseler, Martin ; Görtz, Marcus: *Basiswissen ERP-Systeme: Auswahl, Einführung & Einsatz betriebswirtschaftlicher Standardsoftware*. W3L GmbH, 2007. – ISBN 978–3–937137–38–4

[HMDf04] Heiderich, Mario ; Matthies, Christian ; Dahse, Johannes ; fukami: *Sichere Webanwendungen: Das Praxishandbuch*. Galileo-Computing, 2004

[Inf] Informationssicherheit, Bundesamt für Sicherheit in d.: *IT-Grundschutzhandbuch*. `http://www.bsi.de/gshb/deutsch/menue.htm`

[LR07] Leonard Richardson, Sam R.: *Web Services mit REST*. Bd. 1.Auflage. O'REILLY, 2007. – ISBN 978–3–89721–727–0

[msd04] msdn: *Bedrohungen und Gegenmaßnahmen*. Website. `http://msdn.microsoft.com/de-de/library/aa302418.aspx`. Version: 2004. – aufgerufen am 1.4.2010

[RB08] Ryan Breidenbach, Craig W.: *Spring IN ACTION*. Bd. 2. MANNING, 2008. – ISBN 1–933988–13–4

[SCFY96] Sandhu, Ravi S. ; Coynek, Edward J. ; Feinsteink, Hal L. ; Youmank, Charles E.: Role-Based Access Control Models. In: *IEEE Computer* (1996)

[Sch96] Schneier, Bruce: *Angewandte Kryptographie: Protokolle, Algorithmen und Sourcecode in C*. Addison-Wesley, 1996. – ISBN 3–89319–854–7

[Sch01] Schmeh, Klaus: *Kryptografie und Public-Key-Infrastrukturen im Internet*. 2. dpunkt.verlag, 2001. – ISBN 3–93258–90–8

Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Mittweida, 31. Mai 2010

Sebastian Hirth