Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Sven Elvers

Eine effiziente Sicherheitsarchitektur für verteilte Systeme

Fachbereich Elektrotechnik und Informatik
Department of Electrical Engineering and Computer Science

Sven Elvers

Eine effiziente Sicherheitsarchitektur für verteilte Systeme

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informatik
am Fachbereich Elektrotechnik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Martin Hübner
Zweitgutachter : Prof. Dr. Kai von Luck

Abgegeben am 01. Oktober 2004

Sven Elvers

Thema der Bachelorarbeit

Eine effiziente Sicherheitsarchitektur für verteilte Systeme

Stichworte

Sicherheit, Sicherheitsarchitektur, Verteilte Systeme, Kerberos

Kurzzusammenfassung

In der Praxis existieren IT-Systeme, bei denen die Sicherheit einen geringen Stellenwert einnimmt. Um nachträglich Sicherheit in diese Systeme integrieren zu können, wünschen sich kleine und mittlere Institutionen eine einfache und kostengünstige Lösung. In dieser Arbeit wird daher eine Sicherheitsarchitektur anhand eines entsprechenden Szenarios entworfen, die sich mit wenig Aufwand integrieren lässt und die gewünschten Sicherheitsziele erreicht. Dazu wurde untersucht, welche Faktoren für die Kosten verantwortlich sind und anhand einer Risikoanalyse wurden Sicherheitsanforderungen ermittelt. Der Entwurf besteht aus einem Maßnahmenkatalog und einer Sicherheitssoftware, die die vorhandenen Anwendungen und Dienste um ein Kerberos-ähnliches System erweitert, ohne dass Änderungen an diesen Programmen vorgenommen werden müssen.

Sven Elvers

Title of the paper

An efficient security architecture for distributed systems

Keywords

security, security architecture, distributed systems, Kerberos

Abstract

There are computer systems in practice, where security has low significance. In order to integrate security into such a computer system, small and middle institutions wish themselves a simple and economical solution. Therefore, in this bachelor-report, a security architecture is designed on the basis of an appropriate scenario, which can be integrated with less effort and which reaches the desired security goals. For this purpose it was examined, which factors are causing the costs and with the help of a risk analysis, security requirements were determined. The design consists of a measure catalog and a security software, which extends the existing applications and services with a Kerberos-like system, without modifying these programs.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Im 21. Jahrhundert sind Informations- und Kommunikationssysteme sowohl im privaten als auch im wirtschaftlichen Bereich weit verbreitet. Diese Systeme zu schützen, ist aus wirtschaftlichen aber auch gesetzlichen Gründen wichtig.

Doch häufig existieren in der Praxis bereits bestehende Systeme, bei denen die Sicherheit einen geringen Stellenwert hat, da nicht genügend Ressourcen vorhanden sind, das Budget nicht ausreicht oder einfach bisher nicht an die Sicherheit gedacht wurde.

In solche Systeme nachträglich Sicherheit zu integrieren, ist meistens mit einem hohen Aufwand verbunden, so dass sich besonders kleine und mittlere Institutionen eine einfache und kostengünstige Lösung wünschen (vgl. [BSI-L 03], S.4).

1.2 Szenario

Eine Firma hat sich bereits ein Netz mit 5 Computern aufgebaut, von denen ein Computer als Server verwendet wird. Die Sicherheit wurde beim Aufbau dieser IT-Landschaft stark vernachlässigt, denn Daten werden ungeschützt übertragen und Authentifikation findet ausschließlich lokal am PC des Angestellten statt.

Als Betriebssystem wird auf den Computern Microsoft Windows XP eingesetzt. Auf dem Server läuft ein Programm für die Personalverwaltung und eins für die Entwicklung neuer

Produkte.

Jetzt soll nachträglich eine Sicherheitsarchitektur integriert werden, um Manipulation, Spionage und andere Bedrohungen zu erschweren sowie Datenschutz zu gewährleisten. Ein wichtiger Punkt sind dabei die Kosten, da die Firma nur beschränkte finanzielle Möglichkeiten besitzt.

1.3 Aufgaben und Ziel

Ziel ist der Entwurf einer effizienten Sicherheitsarchitektur für verteilte Systeme. Dabei wurde wissentlich der Begriff effizient zum Beschreiben gewählt, da die Architektur mit geringem Aufwand die gewünschte Sicherheit im bestehenden verteilten System erreichen soll.

Die Sicherheitsarchitektur soll für das unter 1.2 beschriebene Szenario entworfen werden und dabei folgende Eigenschaften besitzen:

1. geringer Aufwand für Integration und Pflege,
2. Authentifikation - Principals, eindeutig benannte Benutzer oder Server(dienste), müssen sicher identifiziert werden,
3. Autorisation - Principals sollen auf Zugriffsrechte überprüft werden,
4. Vertraulichkeit - Informationen, die zwischen Rechnern ausgetauscht werden, dürfen nicht für andere außer den teilnehmenden Principals lesbar sein und
5. Integrität - die Informationen müssen beim Empfänger vollständig und unverändert ankommen.

Die Sicherheitsarchitektur soll aus einem Maßnahmenkatalog und einem Softwareentwurf bestehen, welcher dann als Prototyp implementiert wird. Die Architektur sollte außerdem auf andere Umgebungen übertragbar sein.

1.4 Gliederung der Arbeit

Die Aufgabenstellung wird anhand des unter 1.2 beschriebenen Szenarios erarbeitet.

Die in dieser Arbeit auftauchenden Sicherheitsprotokolle werden zum Verständnis in Kapitel 2 kurz beschrieben.

In Kapitel 3 werden die Aufgaben und Ziele aus 1.3 analysiert und spezifiziert sowie anhand einer Bedrohungsanalyse weitere Anforderungen an das System definiert.

Welche Vor- und Nachteile existierende Sicherheitssysteme in Bezug auf die Anforderungen haben, wird in Kapitel 4 untersucht.

Danach wird die Entstehung des Entwurfs einer einheitlichen Sicherheitsarchitektur in Kapitel 5 behandelt.

Das darauf folgende Kapitel 6 befasst sich mit der prototypischen Umsetzung des Entwurfs unter Laborbedingungen.

Eine Zusammenfassung der Ergebnisse und ein Ausblick, wie die Weiterentwicklung des Sicherheitssystems aussehen kann, wird in Kapitel 7 beschrieben.

Kapitel 2

Grundlagen

2.1 Public Key Infrastructure

Bei asymmetrischen kryptographischen Verfahren (”Public-Key-Verfahren”) werden Schlüsselpaare verwendet, die aus einem privaten und einem öffentlichen Schlüssel bestehen. Dabei ist der private Schlüssel nur dem Inhaber des Schlüsselpaares bekannt und der öffentliche Schlüssel ist jedem zugänglich. So können Daten von jedem mit dem öffentlichen Schlüssel kodiert werden und nur der Inhaber des Schlüsselpaares kann diese kodierten Daten mit seinem privaten Schlüssel dekodieren. Andersherum kann er mit dem privaten Schlüssel Daten signieren. Um z.B. eine Datei oder eine Nachricht zu signieren, wird mit einer kryptographischen Hashfunktion, z.B. MD5 [RFC 1321], ein Hashwert aus den Daten erstellt. Dieser wird dann mit dem privaten Schlüssel kodiert und an die Datei oder Nachricht angehängt. Um die Signatur später zu verifizieren, wird erneut ein Hashwert mit der gleichen Funktion und den gleichen Parametern erstellt. Danach wird dieser Hashwert mit dem Wert verglichen, der beim Dekodieren der Signatur erhalten wurde. Zum Dekodieren wird der öffentliche Schlüssel desjenigen benutzt, der die Daten signiert haben soll. Stimmen die Werte überein, sind die Daten nicht manipuliert worden und stammen vom Besitzer des öffentlichen Schlüssels.

Um zu gewährleisten, welcher öffentliche Schlüssel zu welchem Benutzer gehört, werden in einer Public Key Infrastructure (PKI) von einer zentralen Instanz, der Zertifizierungsstelle (Certification Authority CA), digitale Zertifikate erstellt und verwaltet. Nach [RFC 2459] enthält das X.509 Zertifikat u.a. eine Seriennummer, um das Zertifikat eindeutig zu identifizieren, den Namen des Ausstellers, ein Zeitintervall, in welchem das Zertifikat gültig

ist, den Namen des Subjekts, für den das Zertifikat ausgestellt wurde und den öffentlichen Schlüssel des Subjekts. Zusätzlich wird das Zertifikat vom Aussteller mit seinem privaten Schlüssel signiert.

Des Weiteren haben PKI eine Registrierungsinstanz (Registration Authority RA), die die Zertifikatanträge überprüft. Wenn die RA dann dem Antrag zustimmt, gibt sie der CA einen entsprechenden Auftrag das Zertifikat auszustellen. Häufig werden RA und CA auch in einem Trust Center zusammengefasst. Damit das Trust Center vor Ablauf der Gültigkeit eines Zertifikats dieses zurückziehen kann, wird eine Sperrliste (Certificate Revocation List CRL) geführt, die die Seriennummern der gesperrten Zertifikate beinhaltet. Die ausgestellten Zertifikate und die Sperrliste werden in einem Verzeichnis vom Trust Center zur Verfügung gestellt und verwaltet (vgl. [Eck 03], S.329 und [IETF 04]).

2.2 Kerberos

Kerberos ist ein Authentifikationssystem für verteilte Systeme. Benutzer und Computer melden sich einmal an Kerberos an ("single-sign-on") und werden bei jeder Kommunikation dem Partner gegenüber sicher authentifiziert, vorausgesetzt, dass die Anwendung und der Dienst Kerberos unterstützen. Für den vertraulichen Austausch von Informationen wird ein symmetrisches kryptographisches Verfahren ("Secret-Key-Verfahren") verwendet. Hierbei wird für die Kodierung und Dekodierung der Daten ein geheimer Schlüssel benötigt, den beide Kommunikationspartner kennen müssen. Der sichere Austausch des Schlüssels wird von Kerberos übernommen.

Um diese Aufgaben zu erledigen, stellt Kerberos zwei Dienste zur Verfügung, den Authentication Service (AS), an dem sich die Principals authentifizieren müssen, um Kerberos nutzen zu können, und den Ticket Granting Service (TGS), der den Principals Tickets ausstellt, damit diese eine sichere Verbindung zu einem bestimmten Dienst aufbauen können. AS und TGS laufen meistens zusammen auf einem Server, dem Key Distribution Center (KDC), dieser kennt alle Principals und ihre privaten Schlüssel. Die ausgestellten Tickets enthalten u.a. folgende Daten:

$$\text{Ticket } T_{C,S} \quad = \quad \text{S, C, authtime, endtime, } K_{C,S}$$

S ist dabei der Name des Servers, auf den zugegriffen werden soll, C ist der Name des Clients, der das Ticket angefordert hat, authtime ist der Zeitpunkt, an dem das Ticket erstellt wurde, endtime ist der Zeitpunkt, nach dem das Ticket ungültig ist, und $K_{C,S}$ ist

der Sitzungsschlüssel für die Kommunikation zwischen Client und Server. Damit nur der Server die Daten im Ticket verwenden kann, ist das Ticket vom KDC mit dem privaten Schlüssel vom Server kodiert und damit der Client sich dem Server gegenüber als C authentifiziert, schickt dieser einen Authenticator, der im Wesentlichen seinen Namen und einen Zeitstempel, wann der Authenticator erzeugt wurde, enthält:

$$\text{Authenticator } A_C \quad = \quad \text{C, timestamp}$$

Der Authenticator ist vom Client mit dem Sitzungsschlüssel kodiert, den er zusammen mit dem Ticket vom KDC erhalten hat.

Die Tabelle 2.1 beschreibt einen möglichen Nachrichtenaustausch, wenn sich ein Client anmeldet und eine sichere Verbindung zu einem Server aufbauen will. 1. und 2. sind die Nachrichten zwischen Client und AS, damit sich der Client am KDC authentifiziert und ein Ticket für den TGS bekommt. 3. und 4. beschreibt den Nachrichtenaustausch zwischen Client und TGS, um ein Ticket für den Server S zu erhalten und 5. zeigt die Nachricht vom Client an den Server, um die sichere Verbindung aufzubauen.

Von	**An**	**Nachricht**
1. Client	KDC (AS)	C, TGS, Nonce1
2. KDC (AS)	Client	$\{K_{C,TGS}, \text{Nonce1}\}^{K_C}$, $\{T_{C,TGS}\}^{K_{TGS}}$
3. Client	KDC (TGS)	$\{A_C\}^{K_{C,TGS}}$, $\{T_C\}^{K_{TGS}}$, S, Nonce2
4. KDC (TGS)	Client	$\{K_{C,S}, \text{Nonce2}\}^{K_{C,TGS}}$, $\{T_{C,S}\}^{K_S}$
5. Client	Server	$\{A_C\}^{K_{C,S}}$, $\{T_{C,S}\}^{K_S}$

Tabelle 2.1: Kerberos-Protokollablauf (vgl. [Eck 03],S.418)

Damit ein Principal angemeldet werden kann, wird von diesem der Name und das Passwort benötigt. Aus dem Passwort wird dann ein Schlüssel generiert, so dass das Passwort gleich aus dem Speicher gelöscht wird. Dem AS schickt der Principal eine Ticket-Anfrage für den TGS. Diese besteht aus seinem Namen, den Namen des TGS und einem Nonce-Wert. Wenn der Principal in der Datenbank des KDC registriert ist, sendet der AS als Antwort einen generierten Sitzungsschlüssel ($K_{C,TGS}$) und den Nonce-Wert verschlüsselt mit dem Schlüssel des Principals (K_C) sowie das Ticket für den TGS ($T_{C,TGS}$), welcher mit dem privaten Schlüssel des TGS (K_{TGS}) kodiert ist. Der Principal hat sich erfolgreich angemeldet, wenn er den Sitzungschlüssel und den Nonce-Wert mit dem aus dem Passwort generierten Schlüssel erfolgreich dekodiert hat. Der Nonce-Wert wird benötigt, um die Aktualität der Antwort vom TGS zu beweisen. Das Ticket für den TGS wird als Ticket-Granting-Ticket (TGT)

bezeichnet.

Um nun ein Ticket für einen Dienst zu erhalten, schickt der Principal das TGT, einen Authenticator, den Namen des Dienstes und einen neuen Nonce-Wert an den TGS. Wenn der KDC den Dienst kennt, schickt der TGS als Antwort das Ticket für den Dienst ($T_{C,S}$) sowie den zugehörigen Sitzungsschlüssel ($K_{C,S}$) und den Nonce-Wert verschlüsselt mit dem Sitzungsschlüssel $K_{C,TGS}$. Jetzt kann der Principal die Verbindung zum Dienst mit dem Ticket $T_{C,S}$ und einem neuen Authenticator aufbauen (vgl. [RFC 1510] und [MIT 04]).

Kapitel 3

Anforderungsanalyse

In diesem Kapitel werden anhand der gegebenen Aufgaben (siehe 1.3) die Anforderungen an die Sicherheitsarchitektur spezifiziert. 3.1 behandelt dabei, welche Faktoren den Aufwand im IT-Bereich beeinflussen und welche resultierenden Anforderungen sich für die Architektur ergeben. In 3.2 wird untersucht, welche Bedrohungen die Sicherheitsarchitektur abwenden muss.

3.1 Analyse der wirtschaftlichen Ziele

Damit sich die Sicherheitsarchitektur mit geringem Aufwand integrieren und pflegen lässt, muss erst einmal untersucht werden, von welchen Faktoren der Aufwand abhängig ist. Nach [Sal 04] gibt es für Hardware, Software, Datenmanagement, Prozesse, Netzwerk und den IT-Betrieb folgende primäre Kostenarten:

- Investitionen
- Entwicklung
- Wartung
- Betriebskosten
- Lizenzkosten
- Personalkosten

Die Kostenarten für Gebäude und Beratungsleistung wurden nicht mit aufgeführt, da diese durch Entscheidungen des Unternehmens beeinflusst werden. Aus diesem Grund können auch die Personalkosten im Weiteren vernachlässigt werden, da die Sicherheitsarchitektur diese Kostenart nur indirekt durch den Aufwand bei Investition, Wartung und Betrieb beeinflusst.

Bei der Investition fallen als erstes die Kosten für die Neuanschaffung von Produkten an, wie einmaliger Kaufpreis und Lizenzkosten. Weiterhin muss Zeit investiert werden, um diese dann in das bestehende System zu integrieren sowie einzurichten und um die Mitarbeiter einzuarbeiten. Denn diese sind während der Einarbeitungszeit weniger produktiv, da sie nicht routiniert arbeiten können, sondern den Umgang mit dem neuen Produkt erst lernen müssen. Die Entwicklung ist eine Alternative zur Neuanschaffung. Es muss das Kosten/Nutzen-Verhältnis abgewogen werden, welche Alternative für das Unternehmen in Frage kommt.

Nach der Integration der Produkte fallen Wartungs- und Pflegearbeiten sowie Betriebskosten an. Es müssen administrative Aufgaben erledigt werden, wie Datenpflege und Einstellungen. Wenn Fehler auftreten, müssen deren Ursachen gesucht und evtl. Daten wiederhergestellt werden. Betriebskosten fallen u.a. durch die Ergonomie und dem Zeitbedarf eines Produkts an, welcher für die Bedienung und das Durchlaufen seiner Routinen benötigt wird. Mit Ergonomie wird das Zusammenspiel zwischen Mensch und Produkt beschrieben, wie optimal die Handhabung ist.

Folglich müssen beim Entwurf der Sicherheitsarchitektur die nachstehenden Punkte beachtet werden, um den Aufwand einzuschränken.

- Investitionen
  - Neuanschaffungen/ Entwicklung
  - Integration/ Einrichtung
  - Einarbeitung (Seminare, Selbststudie, etc.)
- Wartung und Pflege
  - Administration
  - Fehlersuche
  - Wiederherstellung
- Betriebskosten

- Ergonomie
- Zeitaufwand

Für Neuanschaffungen vergrößert sich der Gesamtaufwand erheblich, da bei allen oben genannten Punkten weiterer Aufwand anfällt. Daraus ergibt sich die Anforderung, dass die Sicherheitsarchitektur für das bestehende System der Firma möglichst transparent ist. Das heißt, sie soll die Funktionalität der Anwendungen erweitern, ohne dass diese geändert oder in ihrem Verhalten beeinflusst werden.

3.2 Sicherheitsanalyse

Für die in Abschnitt 1.3 angegebenen Sicherheitsziele müssen beim Entwurf der zugehörigen Mechanismen die möglichen Bedrohungen bedacht werden. Dafür werden Bedrohungsbäume ([Eck 03], S.130) erstellt, da sich diese von einem möglichen Angriffsziel aus aufbauen. In diesem Fall die Unterwanderung der Sicherheitsziele.

Die Wurzel eines Bedrohungsbaums, das anvisierte Angriffsziel, kann als untergeordnete Knoten Teilziele oder Angriffsschritte haben (siehe Abbildung 3.1). Ein Teilziel kann wiederum Teilziele oder Angriffsschritte als Kindknoten besitzen und jeder Angriffsschritt ist ein Blatt des Bedrohungsbaums. Um ein Ziel zu erreichen, müssen entweder ein oder alle Kindknoten erfolgreich sein. Dabei werden die Knoten des Bedrohungsbaums mit einem UND gekennzeichnet, bei denen alle Kindknoten erfolgreich sein müssen. Bei allen anderen Knoten ist das Erreichen eines Knotens ausreichend.

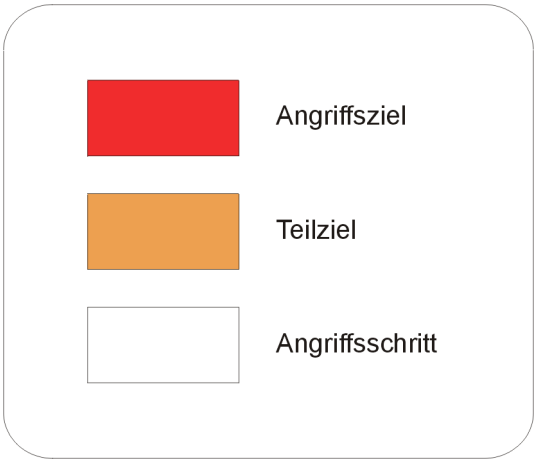


Abbildung 3.1: Legende zu den Bedrohungsbäumen

3.2.1 Maskerade

Der Bedrohungsbaum in Abbildung 3.2 zeigt die Angriffsmöglichkeiten, wie ein Angreifer eine fremde Identität vortäuschen kann. In erster Linie ist die Authentifikation bedroht. Doch auch die anderen Sicherheitsziele sind mit gefährdet, da der Angreifer Rechte erlangt, die ihm nicht zustehen. Die Identität kann durch einen korrekt ausgeführten Login oder durch die Nutzung einer bereits authentifizierten Verbindung erreicht werden.

Dem Angreifer ist es möglich, sich als ein anderer Benutzer anzumelden, indem er einem Netzwerkdienst aufgezeichnete Nachrichten zuschickt oder einen gültigen Schlüssel benutzt. Die wiedereingespielten Nachrichten sind von einer gültigen Anmeldung eines Benutzers. Die Kompromittierung eines Schlüssels kann entweder durch Social Engineering, systematisches Ausprobieren, den Einsatz eines Trojanischen Pferdes oder durch das Abhören der Leitungen erreicht werden. Bei Social Engineering wird das Opfer dazu gebracht, schützenswerte Informationen durch geschicktes Ausfragen oder durch Täuschung preiszugeben. Das systematische Ausprobieren erfolgt durch das Einsetzen aller möglichen Schlüssel (Brute-Force Angriff) oder aller in einem Wörterbuch enthaltenen Passwörter (Wörterbuch-Angriff). Um mit Hilfe eines Trojanischen Pferdes an ein Passwort oder Schlüssel zu gelangen, kann dieses zum Beispiel dazu genutzt werden Tastatureingaben oder den Speicher auszulesen. Trojanische Pferde werden Programme genannt, die Angreifern einen Zugang (Hintertür) zum befallenen Computer geben.

Eine Verbindung kann ein Angreifer nutzen, indem er den Absender seiner Nachrichten fälscht (IP-Spoofing), durch einen ”Man-in-the-Middle”-Angriff oder durch Hijacking. Bei ”Man-in-the-Middle” täuscht der Angreifer beim Verbindungsaufbau zwischen A und B seine Identität vor, gegenüber A behauptet er B zu sein und B gegenüber behauptet er A zu sein. Hijacking wird das nachträgliche Übernehmen einer bestehenden Verbindung genannt. Einer der Principals wird ausgeschaltet und seine Seite wird vom Angreifer übernommen.

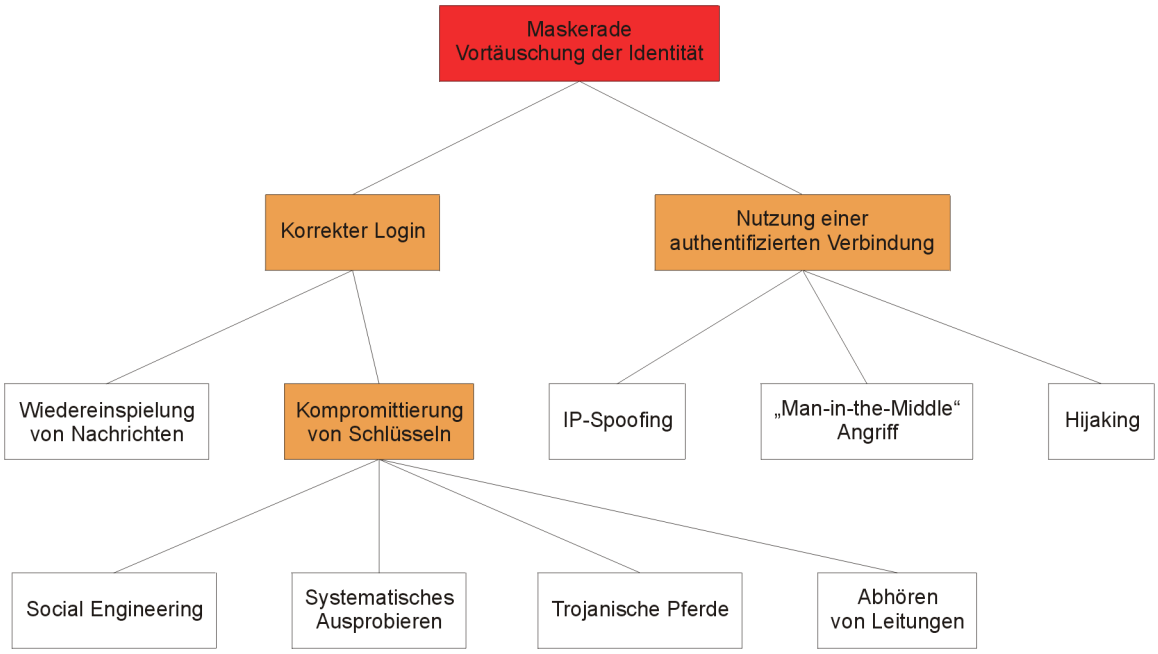


Abbildung 3.2: Bedrohungsbaum: Maskerade

Aus dem Bedrohungsbaum lassen sich folgende Anforderungen entnehmen:

- Zusätzlich zur Authentifikation muss gewährleistet sein, dass eine Verbindung von keinem Dritten genutzt werden kann, um Angriffe wie IP-Spoofing, Man-in-the-Middle, Hijacking und das Wiedereinspielen von Nachrichten zu verhindern bzw. zu erkennen.
- Damit das Abhören von Leitungen zu keinem erfolgreichen Ergebnis gelangt, dürfen keine Passwörter und Schlüssel im Klartext übertragen werden.
- Und um systematisches Ausprobieren, Social Engineering und Trojanische Pferde zu bekämpfen, müssen entsprechende Maßnahmen in der Sicherheitsarchitektur enthalten sein.

3.2.2 Manipulation/ Zerstörung von Daten und Software

Die Bedrohungen für das Sicherheitsziel Integrität ist anhand des Baums in Abbildung 3.3 dargestellt.

Daten oder Software können entweder durch das fahrlässige Handeln eines Anwenders manipuliert bzw. gelöscht werden oder wenn ein Angreifer seine Identität maskiert (siehe 3.2.1), das IT-System auf unberechtigte Weise nutzt, während der Übertragung die Daten

ändert oder Schadsoftware ins IT-System einschleust. Als Schadsoftware werden Programme, wie Viren oder Würmer, bezeichnet, die gewollt eine boshafte Absicht verfolgen, z.B. Dateien löschen oder geheime Informationen preisgeben. Eine unberechtigte IT-Nutzung liegt vor, wenn ein Anwender seine Rechte vorsätzlich missbraucht oder einen Dienst nutzt, den er nicht verwenden soll.

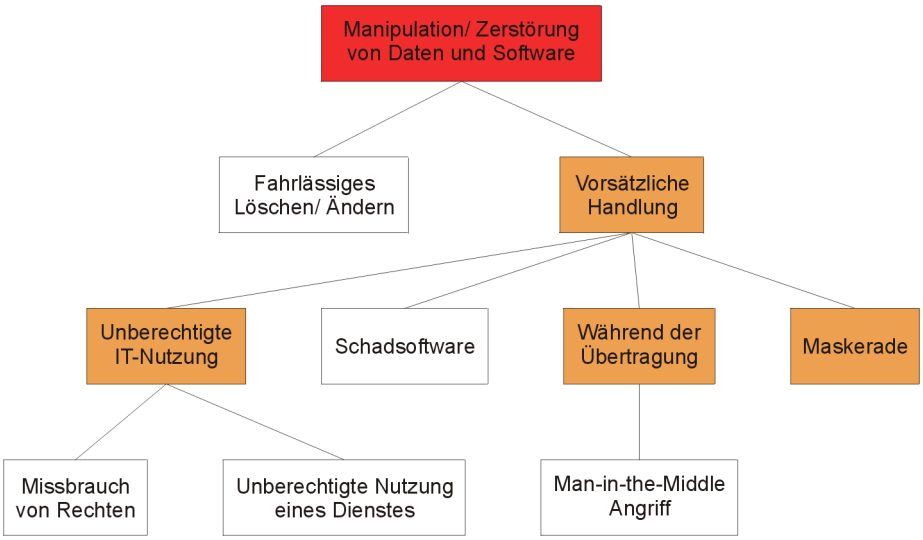


Abbildung 3.3: Bedrohungsbaum: Manipulation und Zerstören

Aus dem Bedrohungsbaum lassen sich folgende Anforderungen entnehmen:

- Wegen der Bedrohung der fahrlässigen Löschung oder Änderung von Daten müssen Maßnahmen zur Sicherung der wichtigen Daten vorgenommen werden.
- Um den Missbrauch von Rechten und die unberechtigte Nutzung von Diensten zu erschweren, müssen Zugriffskontrollen genutzt werden und den Benutzern dürfen nur Rechte vergeben werden, die sie für die Erledigung ihrer Aufgaben benötigen.
- Damit die Integrität übertragener Daten gewährleistet ist, müssen Kontrollmaßnahmen mit in die Sicherheitsarchitektur integriert werden.
- Und es müssen entsprechende Maßnahmen vorgenommen werden, um die Bedrohung durch Schadsoftware zu minimieren.

3.2.3 Vertraulichkeitsverlust wichtiger Informationen

Der Bedrohungsbaum mit dem Angriffsziel die Vertraulichkeit zu unterwandern, ist in der Abbildung 3.4 zu sehen.

Der Verlust der Vertraulichkeit kann durch einen Angreifer erreicht, aber auch durch einen Anwender verursacht werden, indem dieser unter Windows fahrlässig Informationen für Dritte freigibt. Ein Angreifer kann an vertrauliche Informationen entweder durch das Abhören von Leitungen oder einem Maskerade-Angriff (siehe 3.2.1) gelangen, aber auch durch unberechtigte IT-Nutzung oder dem Gebrauch von Schadsoftware (vgl. 3.2.2).

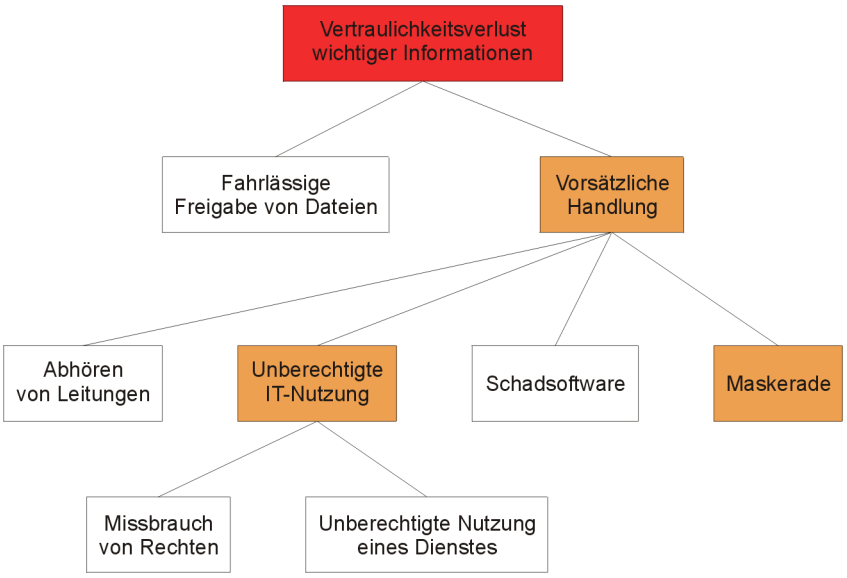


Abbildung 3.4: Bedrohungsbaum: Verlust der Vertraulichkeit

Die Bedrohungen in diesem Baum werden durch die unter 3.2.2 angegebenen und den folgenden Anforderungen abgedeckt:

- Um eine fahrlässige Freigabe von Daten zu unterbinden, müssen die Rechte entsprechend angepasst werden.
- Vertrauliche Informationen dürfen nicht in lesbarer Form übertragen werden, damit diese nicht durch Abhören erlangt werden können.

3.2.4 Zusammenfassung

Durch die Sicherheitsanalyse sind zusätzlich zu den Sicherheitszielen Authentifikation, Autorisation, Vertraulichkeit und Integrität nachstehende Anforderungen bestimmt worden.

1. Vertrauliche Informationen, wie Passwörter, dürfen nicht in für Angreifer lesbarer Form über die Netzwerkleitungen übertragen werden.
2. Nachrichten dürfen nicht von einem Angreifer stammen, indem dieser IP-Spoofing, Man-in-the-Middle, Hijacking oder Wiedereinspielung von Nachrichten verwendet.
3. Dem Verlust der Integrität und Vertraulichkeit von wichtigen Daten durch fahrlässige Handlung und Schadsoftware muss vorgebeugt werden.
4. Der Schaden durch Missbrauch von Rechten muss eingeschränkt werden.
5. Die Wahrscheinlichkeit an ein korrektes Passwort durch systematisches Ausprobieren zu gelangen, muss minimiert werden.
6. Damit durch Social Engineering keine vertrauliche Informationen erlangt werden können, müssen entsprechende Maßnahmen bestimmt werden.

Kapitel 4

Analyse existierender Sicherheitssysteme

Bevor ein neues Protokoll entwickelt wird, soll in diesem Kapitel untersucht werden, welche Vor- und Nachteile der Einsatz von Public Key Infrastructure (PKI) (Abschnitt 4.1) oder Kerberos (Abschnitt 4.2) in Bezug auf die Aufgabenstellung haben.

4.1 Public Key Infrastructure

Da sich die Sicherheitsziele mit Hilfe einer Public Key Infrastructure erreichen lassen, ist der Einsatz einer PKI interessant. Jeder Principal wird durch sein Zertifikat und dem zugehörigen privaten Schlüssel eindeutig authentifiziert, so dass die Überprüfung der Autorisation auf jedem Computer vorgenommen werden kann. Beim Verbindungsaufbau lässt sich sicher ein geheimer Schlüssel austauschen, um die zu übermittelnden Daten vor Dritten zu schützen. Um die Integrität der Daten zu gewährleisten, kann zum Beispiel eine Prüfsumme eingesetzt werden. Die Tatsache, dass es keine zentrale Stelle gibt, bietet einen weiteren Vorteil, da eine PKI nicht von einer Komponente abhängig ist.

Von den Anforderungen aus der Sicherheitsanalyse (siehe 3.2.4) wird der erste Punkt unterstützt, da durch das asymmetrische Verschlüsselungsverfahren keine Passwörter und Schlüssel ungeschützt über die Netzwerkleitungen gehen. Die zweite Anforderung wird teilweise unterstützt. Durch den Gebrauch von Sitzungsschlüsseln sind erfolgreiche IP-Spoofing- und Hijacking-Angriffe nahezu unwahrscheinlich. Man-in-the-Middle ist ebenso

unwahrscheinlich, wenn sich die Principals vergewissern, dass der Verbindungspartner der Richtige ist und nicht nur die Gültigkeit des Zertifikats kontrollieren. Um die Wiedereinspielung von Nachrichten zu erkennen, muss eine zusätzliche Kontrolle eingeführt werden.

Gegen PKI spricht der hohe Aufwand bei Wartung und Pflege, da die Daten auf den Rechnern verteilt sind und in regelmäßigen Abständen neue Zertifikate verteilt werden müssen. Ein weiterer Nachteil ist das Aufbewahren des privaten Schlüssels. Dieser muss sicher aufbewahrt und beim Wechsel des Computers mitgeführt werden.

4.2 Kerberos

Auch mit Kerberos lassen sich die Sicherheitsziele erreichen. Die Principals authentifizieren sich mit Namen und Passwort am KDC (Key Distribution Center) und untereinander mit Tickets, die der KDC bei Anfrage ausstellt. Der Key Distribution Center kann die Zugriffskontrolle übernehmen, welcher Principal eine Verbindung zu einem Dienst aufbauen darf. Die Vertraulichkeit wird durch den Gebrauch von Sitzungsschlüsseln sichergestellt und die Integrität lässt sich wie bei PKI durch eine Prüfsumme realisieren. Ein weiterer Vorteil von Kerberos ist, der überschaubare Aufwand von Wartung und Pflege, der sich durch den zentralen KDC realisieren lässt. Des Weiteren spricht für Kerberos, dass sich beide Seiten beim Verbindungsaufbau als die richtigen Principals authentifizieren müssen.

Kerberos unterstützt den ersten und zweiten Punkt der Sicherheitsanforderungen von 3.2.4. Aus den Passwörtern werden Schlüssel generiert, mit denen die empfangenen Tickets oder KDC-Antworten entschlüsselt werden können. In den Tickets sind Sitzungsschlüssel enthalten, um die weitere Kommunikation verschlüsseln zu können, ohne den privaten Schlüssel seines Gegenübers zu kennen. Wie bei PKI sind IP-Spoofing und Hijacking wegen der Sitzungsschlüssel unwahrscheinlich, ebenso wie Man-in-the-Middle, da durch die Tickets sichergestellt wird, zwischen welchen beiden Principals die Verbindung aufgebaut wird. Der Angreifer müsste entweder den Sitzungsschlüssel zwischen A und dem TGS kennen oder den privaten Schlüssel von B, wenn A eine Verbindung zu B aufbauen will. Durch den Gebrauch von Nonce-Werten und Timestamps wird die Aktualität der Nachrichten gewährleistet, so dass Angriffe durch Wiedereinspielung erkannt werden.

Durch den KDC ist das System von einer Komponente abhängig. Denn der Ausfall des KDC würde das ganze Kerberos-System beeinflussen. Der schwerwiegendere Nachteil ist der Aufwand für das Anpassen der Anwendungen, da diese auf Kerberos ausgerichtet sein müssen. Denn diese müssen den Namen des Principals übergeben, zu dem eine Verbindung aufgebaut werden soll.

Kapitel 5

Entwurf der Sicherheitsarchitektur

5.1 Architektur

Zuerst wird entschieden, wie sich die Sicherheitssoftware in das bestehende Firmennetzwerk integriert, da diese Entscheidungen den gesamten Aufbau beeinflusst. Danach wird abgewogen, ob eine der unter 4 analysierten Systeme verwendet wird.

Damit die Architektur transparent für die Anwendungen ist, wie es in der Anforderungsanalyse im Abschnitt 3.1 gefordert wird, bietet es sich an, eine Middleware zu entwerfen. Folglich ergeben sich Einschränkungen bei der Realisierung der Sicherheitsziele. Die Autorisation hat maximal eine Genauigkeit, welche Verbindungen erlaubt sind, so dass die Benutzer sich an den Diensten wie zuvor explizit anmelden müssen, da diese den authentifizierten Benutzernamen nicht übernehmen. Aus diesen Tatsachen ergibt sich, dass die Sicherheitssoftware nach dem ISO/OSI-Modell[1] (Abbildung 5.1) vor der Transportschicht sitzen muss, da diese für die Verbindung zwischen Prozessen zuständig ist.

Anhand der beschriebenen Vor- und Nachteile von PKI und Kerberos in Kapitel 4 müssen bei PKI einige Maßnahmen vorgenommen werden, wie eine zentrale Stelle für die Pflege, um den Aufwand gering zu halten und Kerberos lässt sich nicht einsetzen, da dann Anwendungen benötigt werden, die Kerberos unterstützen. Trotzdem bietet sich ein Kerberosähnliches System an, da weniger Anpassungen als bei dem Einsatz einer PKI vorgenommen werden müssen. Sowohl bei PKI als auch bei Kerberos muss eine Zugriffs- und Integritätskontrolle eingeführt werden. Bei der Zugriffskontrolle hat Kerberos den Vorteil, dass diese

[1] Für ausführliche Erläuterungen der Schichten des ISO/OSI-Modell siehe [Eck 03] S.59-62 und [KR 02]

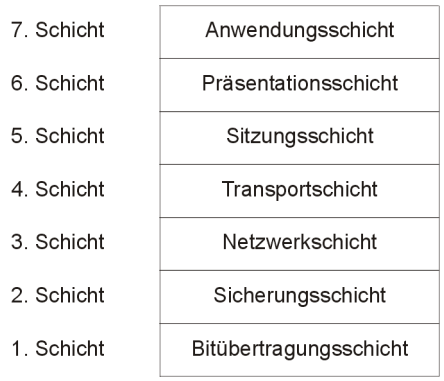


Abbildung 5.1: ISO/OSI Modell

durch den KDC übernommen werden kann und folglich der Aufwand für Wartung und Pflege geringer ist. Ein weiterer für Kerberos sprechender Punkt ist, dass im Gegensatz zu PKI die zweite Sicherheitsanforderung von 3.2.4 komplett erfüllt wird. Die weiteren Punkte drei bis sechs können weder von Kerberos als auch von einer PKI erfüllt werden, so dass im Maßnahmenkatalog entsprechende Maßnahmen getroffen werden müssen. Beim Kerberos-ähnlichen System kann die Anpassung der Anwendungen umgangen werden, indem dieses nicht wie bei Kerberos den Benutzernamen des Principals B von der Anwendung erfährt, wenn A eine Verbindung zu B aufbauen will, sondern den Namen anders erhält.

5.2 Anwendungsfälle

Die Anwendungsfälle entstehen durch die Sicherheitsziele, den möglichen Kommunikationsarten der Anwendungen und durch die Entscheidung ein Kerberos-ähnliches System aufzubauen. Eine Anwendung bestimmt die Art der Kommunikation durch die Verwendung von TCP oder UDP und die Angabe einer IP-Adresse.

Je nach IP-Adresse wird ein Unicast, Multicast oder Broadcast ausgeführt. Bei Unicast steht die IP-Adresse für einen Host und die Nachrichten werden ausschließlich an diesen gesendet. Bei Multicast steht die IP-Adresse für eine Gruppe von Hosts und die Nachrichten werden an alle gesendet, die sich als Gruppenmitglieder registriert haben. Und bei Broadcast werden die Nachrichten an alle Hosts gesendet, die zu dem zugehörigen Netzwerk gehören (vgl. [Cis 04]). Im Weiteren wird nur Unicast, die Kommunikation zwischen genau zwei Hosts, beachtet, da Broadcast und Multicast für das Szenario nicht unterstützt werden müssen.

Somit ergeben sich die folgenden Anwendungsfälle. Für die Authentifikation, muss sich ein Principal am System An- bzw. Abmelden können. Durch TCP und UDP ergeben sich Anwendungsfälle für verbindungsorientierte und verbindungslose Kommunikation. Und durch die Verwendung von Passwörtern muss es dem Anwender möglich sein, dieses zu ändern. Diese Anwendungsfälle werden in den folgenden Abschnitten durch Szenarios beschrieben [Khb 01].

5.2.1 Anmelden

1. Erfolgreiche Anmeldung: Ein Principal gibt seinen Usernamen und Passwort ein. Dadurch erhält er ein Ticket, das Ticket-Granting-Ticket (TGT), um sich mit dem Ticket Granting Service (TGS) des KDC zu verbinden.
2. Anmeldung gescheitert: Der Principal kann das empfangene Packet nicht richtig entschlüsseln und erhält so das Ticket für den TGS nicht.

5.2.2 Verbindungsorientierte Kommunikation (TCP)

Wenn der Dienst keine Authentifikation benötigt gibt es folgende Szenarios:

1. Normale Verbindung: Ein Principal baut eine Verbindung zu einem anderen auf.
2. Mit Authentifikation: Ein Principal hat sich vom TGS ein Ticket besorgt und baut damit die Verbindung auf.

Bei Diensten, die eine Authentifikation verlangen, existieren folgende Szenarios:

1. Normale Verbindung: Ein Principal holt sich ein entsprechendes Ticket vom TGS und baut eine Verbindung zu einem anderen auf.
2. Fehlende Autorisation: Der Principal hat keine Zugriffsrechte für den Dienst.
3. Keine Authentifikation: Der Principal versucht die Verbindung ohne Ticket aufzubauen.

5.2.3 Verbindungslose Kommunikation (UDP)

1. Normale Kommunikation: Die Nachrichten werden an den Empfänger geschickt, ohne zu kontrollieren, ob dieser existiert und die Daten empfängt.
2. Authentifizierte Kommunikation: Dem Empfänger wird das Ticket zugesandt und die Kommunikationsdaten werden mit dem Sitzungsschlüssel verschlüsselt.

5.2.4 Abmelden

1. Netzwerkabmeldung: Es werden die Tickets und Schlüssel aus dem Speicher gelöscht.
2. Lokal abmelden: Ein Benutzer meldet sich lokal am Rechner ab. Es wird die Netzwerkabmeldung ausgeführt.

5.2.5 Passwort ändern

1. Erfolgreich Ändern: Ein Benutzer meldet sich erfolgreich am KDC an und ändert sein Passwort.

5.3 Realisierung der Sicherheitsziele

5.3.1 Authentifikation

Bei Kerberos kann jeder Dienst einen eigenen Namen und Passwort haben. Da aber die Sicherheitssoftware anwendungsunabhängig ist und einen Dienst anhand des Zielports bestimmt, können Name und Passwort nur für Verbindungen genutzt werden, die ein Dienst annimmt, nicht aber wenn der Dienst eine Verbindung aufbaut. Deshalb sollte die Sicherheitsarchitektur als Principals nur Benutzer und Server unterscheiden.

Damit ein Principal Verbindungen zu anderen Principals aufbauen kann, muss dieser sich zuerst am KDC anmelden. Dazu muss der Benutzer seinen Username, das Passwort und die IP-Adresse des KDC angeben. Name und Passwort könnten von der Anmeldung ans Betriebssystem übernommen werden, um die Sicherheitsarchitektur vollkommen transparent für den Anwender zu halten. Doch bei Ablauf des TGT müsste dieser sich neu am

Betriebssystem anmelden und vorher die Anwendungen schließen, evtl. ohne Änderungen speichern zu können, so dass diese Variante nicht praktikabel ist. Statt dessen sollte die Sicherheitssoftware die Möglichkeit integrieren, dass explizit eine Anmelderoutine vom Benutzer oder automatisiert nach dem Anmelden ans Betriebssystem gestartet wird. Zwar muss der Anwender sich ein zweites Mal anmelden, doch bietet dies den Vorteil, wenn eine Verbindung aufgebaut und ein neues TGT dafür benötigt wird, kann der Prozess diese Routine nutzen und muss nicht unterbrochen werden. Die IP-Adresse des KDC vom Benutzer zu erfragen, ist nicht sinnvoll, da dieser sie nicht kennen muss. Statt dessen sollte die Sicherheitssoftware eine Option zur Konfiguration bieten, um die IP-Adresse angeben zu können.

Bei einem Server ist es nicht anwendbar, dass der Administrator Name und Passwort eingeben muss, wenn ein TGT benötigt wird. Denn es müsste jemand den Server regelmäßig kontrollieren. Aus diesem Grund muss es bei der Konfiguration eine Option geben, dass es sich um einen Server handelt, damit Name und Schlüssel wieder verwendet, bzw. aus einer Datei oder anderen Quelle gelesen werden, wenn sie das erste Mal benötigt werden. Die Rechte für diese Quelle müssen entsprechend angepasst werden, um den Zugriff vor unberechtigten Usern und Prozessen zu schützen (vgl. [Eck 03], S.338).

Die an einer Verbindung teilnehmenden Principals müssen sich jeweils dem Anderen gegenüber authentifizieren, indem der verbindungsaufbauende Principal sich vom KDC ein Ticket besorgt, in dem u.a. ein Sitzungsschlüssel sowie der Name des Zielprincipals und sein eigener steht. Dieses Ticket entschlüsselt der Empfänger und kontrolliert die Angaben und den Sendernamen kontrolliert er mit dem Namen, der mit dem Sitzungsschlüssel kodiert ist. Da die Anwendungen nicht den Empfängernamen übergeben, kann der Sender dem KDC nur die IP-Adresse und den Port mitteilen. Um den Benutzernamen zu erfahren, könnte der Zielprincipal gefragt werden, dies ist aber nicht sinnvoll, da der Empfänger ein Angreifer und nicht der Zieldienst sein könnte. Dadurch ergibt sich, dass der KDC eine Abbildungstabelle benötigt. Diese kann statisch oder dynamisch sein. Bei der statischen Variante werden die Informationen z.B. aus einer Datei oder anderen Quelle gelesen. Dadurch sind die Adressen sicher zuzuordnen, doch sind die Principals an ihre Computer gebunden. Die dynamische Liste kann der KDC durch die Information beim Anmelden aufbauen. Dadurch sind die Principals nicht an ihre Computer gebunden, doch kann ein authentifizierter Benutzer die IP-Adresse eines Servers annehmen und so an Passwörter und andere vertrauliche Daten gelangen. Folglich sollte der KDC eine statische Liste für die Server benutzen, da die Dienste an einen Rechner gebunden sind und ihnen eine feste IP zugeordnet werden kann. Zusätzlich sollte auch die dynamische Liste verwendet werden,

um die Benutzer nicht an ihren Computer zu binden. Bei einem Platzwechsel zum Beispiel, müsste sonst der Computer mitgenommen werden oder entsprechende Konfigurationen vorgenommen werden.

Bei einer Kommunikation über UDP können Pakete verloren gehen und empfangene Nachrichten können von verschiedenen Absendern stammen. Der Zielprincipal müsste deshalb beim Empfangen eines Tickets eine Bestätigung zurückschicken, damit der Client weiß, dass er die weiteren Nachrichten mit dem Sitzungsschlüssel kodieren kann. Der Zielprincipal müsste dann diese Sitzungsschlüssel abhängig von der IP-Adresse des Client speichern, um zum Dekodieren den richtigen Schlüssel zu nutzen. Da aber keine Verbindung zwischen den Principals besteht, kann der Serverdienst nicht entscheiden, wie lange der Sitzungsschlüssel genutzt werden muss. Statt dessen sollte die Sicherheitssoftware bei UDP die einzelnen Nachrichten authentifizieren. Die Anwendungsdaten einer Nachricht werden mit dem Sitzungsschlüssel kodiert und zusammen mit dem Ticket sowie dem Authenticator in einem UDP-Paket übertragen. Dadurch werden die Vorteile von UDP unterstützt. Nachrichten werden versandt, ohne kontrolliert zu werden, ob diese ankommen. Zusätzlich kann der Serverdienst den Absender sicher bestimmen und muss die Sitzungsschlüssel nicht verwalten.

Durch die in Abschnitt 5.2 beschriebenen Szenarios für eine verbindungsorientierte und verbindungslose Kommunikation können Dienste eine Authentifikation benötigen oder nicht. Um den Aufwand bei Wartung und Pflege gering zu halten, sollte dieses Wissen vom KDC verwaltet werden, indem dieser eine Liste mit den Diensten kennt, an denen eine Authentifikation stattfinden muss. Je nach Dienst gibt der KDC ein Ticket heraus oder eine Nachricht, dass kein Ticket benötigt wird. Dieses Verfahren alleine würde den KDC stark belasten, denn bei Arbeiten, wie Internetrecherche, würde dieser häufig kontaktiert. Daher sollte jeder Principal wissen bei welchen IP-Adressen er den KDC kontaktieren muss, z.B. durch die Angabe eines Adressbereiches, in dem die Adressen der Server liegen, wie 192.168.0.1 - 192.168.0.100. Dies hat aber den Nachteil, dass Änderungen an allen Computern vorgenommen werden müssen. Die Sicherheitssoftware sollte statt dessen die Variante implementieren, dass der KDC diese Information verwaltet und bei der Anmeldung oder evtl. auch wenn ein Ticket erfragt wird, die Daten auf den Clients aktualisiert. Da diese Daten selten geändert werden, reicht ein Austausch der Informationen beim Anmelden am KDC. Zusätzlich muss jeder Principal seine zu schützenden Dienste kennen, damit keine ungeschützten Verbindungen aufgebaut werden können. Diese verteilte Konfiguration ist akzeptabel, da Änderungen nur am entsprechenden Computer vorgenommen werden müssen.

5.3.2 Autorisation

Da für jede Verbindung ein Ticket benötigt wird, kann der KDC die Zugriffskontrolle übernehmen, indem dieser nur Tickets herausgibt, wenn ein Principal den gewünschten Dienst nutzen darf. Außerdem bietet dieses Verfahren den Vorteil, dass Änderungen nur am KDC vorgenommen und nicht auf verschiedenen Computern nachgezogen werden müssen, z.B. wenn ein neuer Benutzer hinzugefügt wird.

Daraus folgt, dass der KDC eine Liste mit den Zugriffsrechten benötigt. Hierfür kann eine Access Control List (ACL) oder Capabilityliste (C-List) verwendet werden ([Eck 03], S.453 u. S.460). Bei einer ACL wird den Objekten, hier die angebotenen Dienste, eine Liste der Subjekte, hier den Principals, mit den entsprechenden Rechten zugeteilt. Eine C-List ist andersherum aufgebaut. Den Subjekten werden die Rechte für Objekte zugeordnet. Da der KDC Tickets für die Subjekte ausstellt, sollte eine C-List genutzt werden. Doch diese erfordert einen hohen Aufwand bei Wartung und Pflege, da Änderungen meist redundant vorgenommen werden müssen, z.B. beim Hinzufügen oder Entfernen eines Objekts müssen die Rechte von mehreren Subjekten angepasst werden. Dieses lässt sich umgehen, indem Principals mit gleichen Rechten die gleiche Rolle erhalten und die Rolle anstelle der Principals in der C-List aufgeführt wird. Genauso sollten die Dienste in Gruppen zusammengefasst werden. Da diese anhand des Principals bzw. der IP-Adresse und der Portnummer im Netzwerk identifiziert werden, kann die Anzahl der Dienste unübersichtlich groß und dadurch die C-List unwartbar werden. Den Diensten Gruppen zuzuteilen, sollte mit der Liste kombiniert werden, die der KDC nach Abschnitt 5.3.1 braucht, um zu wissen für wen er Tickets ausstellen muss.

Da alle Dienste eines Servers bzw. eines Benutzers den gleichen Schlüssel haben, müssen die Tickets angeglichen werden. Diese müssen den Name des Principals und den Dienstport enthalten, damit ein Angreifer sich nicht für einen Dienst des Servers A ein Ticket ausstellen lässt, sich aber an einem anderen Dienst von A mit dem Ticket anmeldet.

5.3.3 Vertraulichkeit

Für die Vertraulichkeit übertragener Daten müssen keine Anpassungen vorgenommen werden. Wenn ein Principal sich am KDC anmeldet, erhält dieser ein mit seinem privaten Schlüssel kodiertes Paket, in dem u.a. ein Sitzungsschlüssel und das TGT enthalten ist. Dieses enthält den gleichen Sitzungsschlüssel und ist mit dem privaten Schlüssel des KDC

kodiert. Beim Verbindungsaufbau erfährt der TGS durch das Ticket den Sitzungsschlüssel, so können alle Nachrichten zwischen Principal und KDC ausgetauscht werden, ohne dass Dritte sie lesen können. Für die Kommunikation zwischen zwei Principals wird das gleiche Verfahren verwendet. Ein Principal holt sich für den Zielprincipal ein Ticket und ein Sitzungsschlüssel vom TGS. Das Ticket ist wiederum nur von dem Zielprincipal lesbar und die weitere Kommunikation wird mit dem Sitzungsschlüssel vor Dritten geschützt.

5.3.4 Integrität

Nach [RFC 1510] Abschnitt 3.4 gibt es einen Nachrichtentyp KRB_SAFE. Diesen muss die Sicherheitssoftware nutzen, um die Integrität der Nachrichten zu gewährleisten. Denn KRB_SAFE verwendet eine mit einem Schlüssel erstellte Prüfsumme, um mögliche Modifikationen zu erkennen. Um die Prüfsumme zu erstellen, wird eine kryptographische Hashfunktion, z.B. MD5 [RFC 1321], als Message Authentication Code (MAC) verwendet. Dies wird erreicht, indem der Schlüssel i.A. vor Anwendung der Hashfunktion an die Nachricht angehängt wird und dadurch mit in die Berechnung eingeht. Für dieses Verfahren kann der Sitzungsschlüssel verwendet werden. Zusätzlich muss bei dem Nachrichtentyp KRB_SAFE mindestens eine der optionalen Verfahren ”timestamp” oder ”sequence number” genutzt werden, um die Aktualität der Nachrichten sicher zu stellen. Da bei der Verwendung von UDP Nachrichten verloren gehen können, sollten Timestamps genutzt werden.

5.4 Realisierung der Anwendungsfälle

Durch die Realisierung der Sicherheitsziele (Abschnitt 5.3) werden die Anwendungsfälle Anmelden, Verbindungsorientierte Kommunikation (TCP) und Verbindungslose Kommunikation (UDP) von der Sicherheitssoftware bisher unterstützt. Es fehlen noch Abmelden und Passwort ändern im Entwurf.

5.4.1 Abmelden

Ein Benutzer oder Server ist angemeldet, wenn er ein gültiges TGT besitzt. Dieses bekommt er vom KDC zusammen mit einem Sitzungsschlüssel, die der Principal mit seinem privaten Schlüssel entschlüsseln muss. Um den Principal erfolgreich abzumelden, müssen folglich der private Schlüssel und das TGT entfernt werden. Damit durch eine Speicheranalyse diese

Daten nicht zu einem späteren Zeitpunkt ausgelesen werden können, reicht ein einfaches Verwerfen nicht. Die Sicherheitssoftware muss das TGT und den privaten Schlüssel mit anderen Daten überschreiben. Auch die anderen Tickets müssen überschrieben werden, da sich ein Principal einem anderen gegenüber mit diesen authentifiziert.

5.4.2 Passwort ändern

In [RFC 3244] ist ein weiterer Dienst für den KDC beschrieben, um Passwörter ändern zu können. Für diesen muss sich der Principal vom TGS ein Ticket ausstellen lassen und baut dann eine sichere Verbindung zum Dienst auf. Um nun das Passwort zu ändern, wird vom Principal eine mit dem Sitzungsschlüssel kodierte Nachricht versandt, die das neue Passwort und optional u.a. den Namen des Principals enthält. Es wird das Passwort und nicht der Schlüssel übertragen, damit der KDC dieses auf seine Gültigkeit überprüfen kann. Wenn UDP anstelle von TCP genutzt wird, müssen alle Daten in einem Paket übertragen werden. Der Dienst schickt als Antwort eine entsprechende Nachricht, ob die Änderung erfolgreich war oder aus welchem Grund die Anfrage nicht ausgeführt wurde.

5.5 Protokollierung

Um den Aufwand bei Wartung und Pflege weiter zu minimieren, ist es sinnvoll Informationen zu protokollieren. Denn dadurch wird bei der Fehlerbehebung die Suche nach der Ursache unterstützt.

Jeder Eintrag sollte Datum und Uhrzeit beinhalten, damit zu einem späteren Zeitpunkt erkennbar ist, welche Informationen relevant sind. Da das Protokollieren die Fehlersuche unterstützen soll, muss die Sicherheitssoftware jede Fehlernachricht mit Principalnamen und Portnummer der beteiligten Hosts aufzeichnen. Da aber diese Nachrichten ausbleiben können, ist es sinnvoll die Anfragen und positiven Antworten mit zu protokollieren. So lassen sich nicht beantwortete Anfragen erkennen. Des Weiteren könnte der Nachrichtenaustausch zwischen Anwendung und Dienst mit aufgezeichnet werden. Dies sollte aber nicht implementiert werden, da durch die Menge der Informationen das Auswerten der Protokolldatei erschwert wird. Um Fehler, die durch ”falsche” Daten verursacht werden, zu finden, könnten die Daten mitprotokolliert werden. Davon muss aber abgeraten werden, da ein Verstoß gegen die Vertraulichkeit entsteht. Z.B. würde die Protokolldatei eine Sammlung an Tickets beinhalten.

5.6 Sicherheitsanforderung

Bei einem Angriff z.B. eines Trojanischen Pferdes kann es vorkommen, dass der Speicher ausgelesen wird. Deshalb sollten sensitive Daten überschrieben werden, wenn diese nicht mehr benötigt werden. Das Passwort muss nach dem generieren des privaten Schlüssels überschrieben werden, Tickets und Sitzungsschlüssel, wenn die Verbindung geschlossen wird oder spätestens zusammen mit dem TGT, wenn sich der Benutzer abmeldet ([Eck 03], S.338).

5.7 Module

Bei der Sicherheitssoftware brauchen der KDC und die Principal-Software ein Modul, um auf die Konfigurationsdaten zugreifen zu können und ein Modul, um den Nachrichtenaustausch, wie unter 5.5 beschrieben in einer Datei zu speichern.

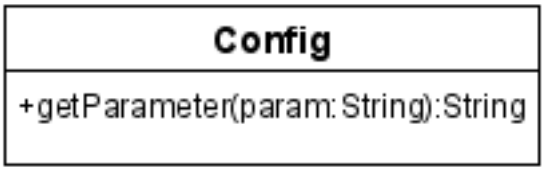


Abbildung 5.2: Modul - Config

Das Modul *Config*, das für die Konfigurationsdatei zuständig ist, bietet eine Methode getParameter() an, um Einstellungen aus einer Datei zu lesen. Dazu wird der Methode der Name des Parameters übergeben und das Modul gibt dann den Wert für den Parameter zurück. So kann die Principal-Software die IP-Adresse des KDC auslesen und wenn es sich um einen Server als Principal handelt, Namen und Passwort des Servers. Der KDC nutzt das Modul, um u.a. die Werte für ein Ticket zu erhalten, z.B. wie lange ein Ticket gültig ist.

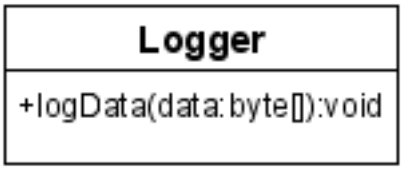


Abbildung 5.3: Modul - Logger

Das Modul *Logger* verwaltet die Protokolldatei. Es bietet eine Funktion logData() an, welcher Daten übergeben werden, die dann mit einem Zeitstempel in die Protokolldatei geschrieben werden. Wenn keine Datei vorhanden ist, wird eine neue Datei erstellt.

5.7.1 KDC

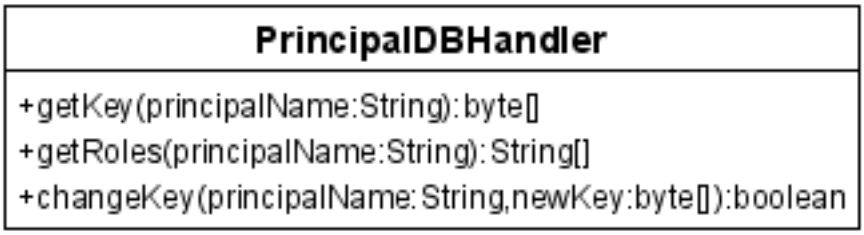


Abbildung 5.4: Modul - PrincipalDBHandler

Der KDC muss alle Principals mit ihren Schlüsseln und Rollen verwalten. Das Modul *PrincipalDBHandler* ist für den Zugriff auf diese Informationen zuständig und muss drei Methoden getKey(), getRoles() und changeKey() anbieten. Den Methoden getKey() und getRoles() wird der Name eines Principals als Parameter übergeben, dabei wird von getKey() der Schlüssel dieses Principals zurückgegeben und von getRoles() eine Liste mit den Rollen des Principals. Mit changeKey() kann einem Principal ein neuer Schlüssel zugeordnet werden. Dazu wird der Methode der Name des Principals übergeben und der neue Schlüssel.

Abbildung 5.5: Modul - ServiceDBHandler

Da der KDC zusätzlich zu den Principals eine Liste mit Diensten verwaltet, an denen sich die Principals authentifizieren müssen, wird ein Modul *ServiceDBHandler* benötigt. Dieses muss zwei Methoden isTicketNeededFor() und getGroups() bereitstellen. Mit isTicketNeededFor() wird überprüft, ob für den übergebenen Dienst ein Ticket benötigt wird und getGroups() hat als Rückgabe eine Liste mit den Gruppennamen, die dem übergebenen Dienst zugeordnet sind.

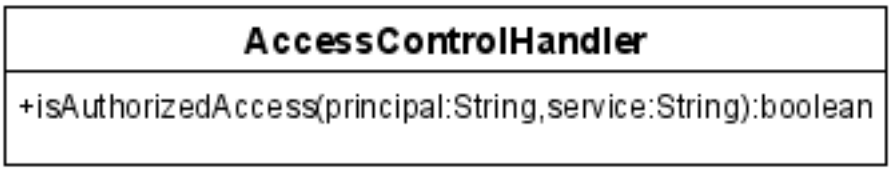


Abbildung 5.6: Modul - AccessControlHandler

Der KDC der Sicherheitssoftware stellt den Principals lediglich Tickets aus, wenn diese berechtigt sind, den angegebenen Dienst zu nutzen. Dafür verwaltet das Modul *AccessControlHandler* die C-List und bietet die Methode isAuthorizedAccess() an. Mit dem übergebenen Principalnamen holt sich das Modul vom *PrincipalDBHandler* die Rollen des Principals und mit dem übergebenen Dienst werden vom *ServiceDBHandler* die Gruppen geholt. Jetzt iteriert das Modul über die Rollen und schaut in der C-List nach, ob eine der Rollen auf eine der Gruppen zugreifen darf. Wenn eine der Kombinationen erfolgreich ist, ist der Rückgabewert von isAuthorizedAccess() *true*, anderenfalls ist der Wert *false*.

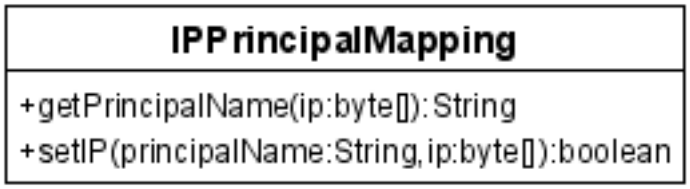


Abbildung 5.7: Modul - IPPrincipalMapping

Das Modul *IPPrincipalMapping* verwaltet eine statische und eine dynamische Liste, um IP-Adressen einer Ticketanfrage auf Principals abbilden zu können. Die statische Liste mit den Serveradressen liest das Modul aus einer Datei und um die dynamische Liste zu füllen, muss das Modul eine Methode setIP() anbieten. Dieser wird der Name eines Principals und eine IP-Adresse übergeben. Die IP-Adresse wird dann dem Principal zugeordnet, wenn keine der beiden Angaben in der statischen Liste enthalten sind. Des Weiteren muss das Modul eine Methode getPrincipalName() anbieten. Dieser Methode wird eine IP-Adresse übergeben und gibt als Rückgabewert den Namen des assoziierten Principals.

Abbildung 5.8: Modul - AuthenticationService

Das Modul *AuthenticationService* ist für die TGT-Anfrage eines Principals zuständig. Dazu holt es vom PrincipalDBHandler die privaten Schlüssel vom TGS und vom Principal, der in der Anfrage angegeben ist. Als nächstes generiert er einen Sitzungsschlüssel und erstellt dann die Antwortnachricht (siehe Tabelle 2.1). Zusätzlich holt das Modul von *Config* die Liste mit den IP-Adressen, bei denen die Principals den KDC nach einem Ticket fragen sollen. Diese wird dann in einer KRB_SAFE Nachricht gesendet, um ihre Integrität zu gewährleisten. Bei einem Fehler sendet das Modul dem Principal anstelle der Antwort eine Fehlernachricht.

Abbildung 5.9: Modul - TicketGrantingService

Das Modul *TicketGrantingService* ist für die weiteren Ticket-Anfragen eines Principals zuständig. Nachdem das TGT und der Authenticator erfolgreich überprüft wurden, wird die IP-Adresse, von der die Anfrage gesendet wurde, und der Principalname des Anfragenden dem Modul *IPPrincipalMapping* übergeben, um das Mapping zu aktualisieren. Danach wird *IPPrincipalMapping* nach dem Principalnamen gefragt, dem die IP-Adresse aus der Anfrage zugeordnet ist. Aus diesem Namen und der Portnummer aus der Ticketanfrage generiert das Modul *TicketGrantingService* dann den Dienstnamen und ruft die Methode isTicketNeededFor() vom *ServiceDBHandler* auf, ob für diesen Dienst ein Ticket benötigt wird. Wenn ein Ticket benötigt wird, wird dem *AccessControlHandler* der Principalname und der Dienstname übergeben. Ist der Principal berechtigt den Dienst zu nutzen, wird vom *PrincipalDBHandler* der private Schlüssel des Zielprincipals geholt und ein Sitzungsschlüssel generiert. Dann wird die Antwort mit Ticket erstellt (siehe Tabelle 2.1). Im Gegensatz zu Kerberos enthält dieses Ticket anstelle des Namens vom Zielprincipal den generierten Dienstnamen. Bei einem Fehler, keiner Berechtigung oder wenn kein Ticket für den Dienst benötigt wird, schickt *TicketGrantingService* eine entsprechende Nachricht.

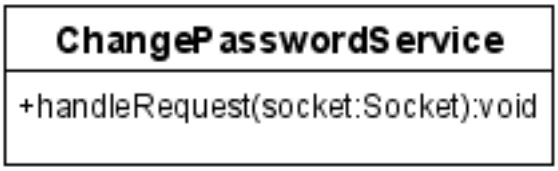


Abbildung 5.10: Modul - ChangePasswordService

Um die Anfrage für eine Passwortänderung bearbeiten zu können, wird das Modul *ChangePasswordService* benötigt. Nachdem das Ticket und der Authenticator erfolgreich überprüft wurden, wird vom Modul *Config* die Passwortbedingungen geholt und das Passwort aus der Anfrage überprüft. Wenn das Passwort den Bedingungen genügt, wird aus diesem der neue Schlüssel generiert. Danach wird vom Modul *PrincipalDBHandler* die Methode changeKey() aufgerufen und der Name des Principals sowie der neue Schlüssel übergeben.

5.7.2 Principal-Software

Abbildung 5.11: Modul - IPAddressList

Die Principal-Software entscheidet anhand einer Liste mit IP-Adressen, ob sie eine Ticketanfrage an den KDC schickt. Das Modul *IPAddressList* verwaltet diese Liste und bietet die Methoden setIPList() und involveKDC() an. Mit setIPList() kann dem Modul eine Liste mit IP-Adressen übergeben. Der Methode involveKDC() wird beim Aufruf eine IP-Adresse übergeben. Wenn diese Adresse in der Liste des Moduls enthalten ist, ist der Rückgabewert *true*, anderenfalls ist der Wert *false*.

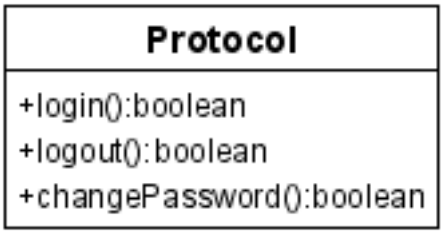


Abbildung 5.12: Modul - Protocol

Das Modul *Protocol* ist für die Netzwerkverbindungen und die Kommunikation mit dem KDC verantwortlich. Es muss die Methoden login(), logout() und changePassword(). Die Methode login() nutzt das Modul *Config* oder öffnet einen Dialog, um Name und Passwort zu erhalten. Aus dem Passwort wird der private Schlüssel generiert. Dann wird eine TGT-Anfrage an den KDC gesendet und die Antwort wird mit dem Schlüssel dekodiert, um das TGT und den Sitzungsschlüssel zu erhalten. Mit dem Sitzungsschlüssel wird die Nachricht mit den IP-Adressen dechiffriert und diese dem Modul *IPAddressList* übergeben. Die Methode logout() überschreibt das TGT und den zugehörigen Sitzungschlüssel und die Methode changePassword() fragt den Benutzer nach seinem aktuellen Passwort und dem neuen Passwort. Anhand des aktuellen Passworts wird kontrolliert, ob die Angaben nicht

von einem Angreifer sind. Dann wird vom KDC ein Ticket für den Change Password Service besorgt und eine sichere Verbindung zu dem Dienst aufgebaut. Nun wird das neue Passwort an den Dienst gesendet und die Antwort ausgewertet.

Sobald eine Anwendung eine Verbindung aufbaut oder auf eine eingehende Verbindung wartet und der Principal sich noch nicht angemeldet hat, ruft das Modul *Protocol* die Methode login() auf. Bei einer eingehenden Verbindung werden von *Config* die Portnummer geholt und überprüft, ob die Verbindung mit einem Ticket aufgebaut werden muss. Sofern die Verbindung ein Ticket benötigt, wird sie erst an die Anwendung weiter gereicht, wenn in einer angemessenen Zeit ein gültiges Ticket empfangen wurde. Andernfalls wird die Verbindung geschlossen und auf die nächste gewartet. Wenn bei einem Verbindungsaufbau die IP-Adresse des Zieldienstes in der Liste vom Modul *IPAddressList* ist, wird an dem KDC eine Ticketanfrage gesendet, die die IP-Adresse und den Port des Dienstes enthält. Sofern die Antwort eine Fehlermeldung ist oder der Prinzipal nicht berechtigt ist den Dienst zu nutzen, wird der Verbindungsaufbau abgebrochen. Besagt die Antwort, dass der Dienst kein Ticket benötigt, wird der Verbindungsaufbau ohne Ticket aufgebaut. Andernfalls ist in der Antwort ein Ticket und ein Sitzungsschlüssel enthalten, mit denen eine sichere Verbindung zum Dienst aufgebaut wird. Besitzt eine Verbindung einen Sitzungsschlüssel, werden die Nachrichten der Anwendung in einer KRB_SAFE-Nachricht versendet und empfangene Nachrichten entpackt. Wenn eine Verbindung geschlossen wird, überschreibt das Modul *Protocol* den Sitzungsschlüssel.

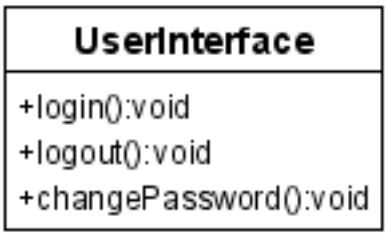


Abbildung 5.13: Modul - UserInterface

Das Modul *UserInterface* bietet dem Principal die Methoden login(), logout() und changePassword(), die er z.B. über eine Anwendung direkt aufrufen kann. Diese Funktionen rufen die gleichnamigen Methoden des Moduls *Protocol* auf. Außerdem ist das Modul *UserInterface* verantwortlich, die Methode logout() aufzurufen, wenn der Principal sich am Betriebssystem abmeldet.

5.8 Maßnahmenkatalog

Um den Aufwand für die Integration und Einrichtung gering zu halten, sollte der Maßnahmenkatalog übersichtlich bleiben und nicht jede Möglichkeit zur Verbesserung der Sicherheit beinhalten, sondern sich auf die wichtigen beschränken. Auf weiterführende Maßnahmen sollte aber trotzdem hingewiesen werden, damit diese später noch bei Bedarf integriert werden können.

5.8.1 Allgemein

Das Bundesamt für Sicherheit in der Informatik (BSI) hat einen Leitfaden IT-Sicherheit [BSI-L 03] mit den wichtigsten Maßnahmen herausgegeben, um kleinen und mittleren Institutionen mit beschränkten finanziellen und personellen Möglichkeiten einen überschaubaren Einstieg zu ermöglichen.

Nach [BSI-L 03] sind die Maßnahmen in Systematisches Herangehen an IT-Sicherheit, Sicherheit von IT-Systemen, Vernetzung und Internet-Anbindung, Faktor Mensch, Wartung von IT-Systemen, Verwendung von Sicherheitsmechanismen und Schutz vor Katastrophen und Elementarschäden gruppiert. Von diesen Maßnahmen müssen die aufgeführten Maßnahmen ausgeführt werden, um die ausstehenden Anforderungen 3 bis 6 aus Abschnitt 3.2.4 zu realisieren. Die Nummerierung dieser Maßnahmen wurde vom Leitfaden IT-Sicherheit übernommen.

Systematisches Herangehen an IT-Sicherheit

7. Die Zuständigkeit für Aufgaben muss festgelegt sein und jeder Verantwortliche braucht einen Stellvertreter, der im Notfall einspringen kann. Wichtig ist hierbei, dass dieser auch in der Lage ist, die Aufgaben von der vertretenen Person zu erfüllen.

8. Die Mitarbeiter müssen über bestehende Richtlinien und Zuständigkeiten informiert werden. Damit sie wissen, wie sie sich beim Auftreten eines Problems verhalten und an welchen Mitarbeiter sie sich wenden müssen. Aber vor allem soll mit dieser Maßnahme erreicht werden, dass keine vertraulichen Informationen wie Passwörter an Unberechtigte herausgegeben werden.

Sicherheit von IT-Systemen

13. Die Nutzung der vorhanden Sicherheitsmechanismen

14. Der Einsatz von Virenschutzprogrammen
15. Die Vergabe der Zugriffsrechte sollte nach dem "Need-to-Know Prinzip" vergeben werden. Jeder Benutzer sollte nur die Zugriffsrechte bekommen, die er für die Ausführung seiner Arbeit benötigt.
16. Den Benutzern sollten Rollen und Profile zugeordnet werden, damit die Rechteverwaltung überschaubar ist, wer welche Rechte hat.
17. Administrationsrechte sollten wie bei den Anwendern auf ein Mindestmaß beschränkt sein, um einen möglichen Missbrauch der Rechte einzuschränken.
18. Programme sollten wie Benutzer nur die benötigten Rechte haben, damit ein Angreifer diese nicht nutzen kann, um an mehr Rechte zu gelangen.
19. Die Standardeinstellung von IT-Systemen sollte geeignet angepasst werden, da diese auf den reibungslosen Ablauf eingestellt sind und weniger den Aspekt der Sicherheit berücksichtigen.
20. Handbücher und Dokumentationen sollten frühzeitig gelesen werden, damit wichtige Hinweise des Herstellers bekannt sind.
21. Es müssen ausführliche Installations- und Systemdokumentationen erstellt und regelmäßig aktualisiert werden. Bei Personalausfall oder wenn eine Tätigkeit wiederholt werden muss, kann auf dieses Wissen zurückgegriffen werden.

Faktor Mensch

29. Die Sicherheitsrichtlinien und -anforderungen müssen beachtet werden, denn sie helfen bei Nichtbeachtung nicht.
30. Schützenswerte Informationen sollten nicht am Arbeitsplatz frei zugänglich sein.
31. Bei Wartungs- und Reparaturarbeiten sind besondere Vorsichtsmaßnahmen zu beachten. Servicetechniker sollten nie ohne Aufsicht an IT-Systemen arbeiten und Dateien auf Datenträger, die das Haus verlassen, müssen sicher mit speziellen Tools gelöscht werden, denn sonst sind sie rekonstruierbar.
32. Um das Sicherheitsbewusstsein der Mitarbeiter zu erhöhen, sollten diese regelmäßig geschult werden, z.B. durch Rundschreiben oder Vorträge.
33. Eine ehrliche Selbsteinschätzung ist wichtig, damit keine Überschätzung der eigenen Fähigkeiten oder falsche Sparsamkeit auftreten. Es ist in manchen Fällen wegen

mangelndem Wissen oder mangelnder Zeit ratsam, einen Experten zu beauftragen.

34. Es sollten Kontrollmechanismen aufgebaut werden, um das Einhalten der Sicherheitsvorgaben zu gewährleisten.

Wartung von IT-System

37. Regelmäßiges einspielen von Sicherheits-Updates

38. Um über die neusten Sicherheitserkenntnisse informiert zu sein, sollten regelmäßig Recherchen durchgeführt werden.

39. Für das Einspielen von Sicherheits-Updates sollten Aktionspläne erstellt werden, damit diese als Prozess verankert sind und diszipliniert durchgeführt werden.

40. Softwareänderungen sollten in einer Testumgebung kontrolliert werden, um ein reibungsloses Funktionieren des Systems nach der Änderung zu gewährleisten.

Verwendung von Sicherheitsmechanismen

41. Sicherheitsmechanismen sollten sorgfältig nach der Qualität ausgesucht werden.

42. Es müssen sichere Passwörter eingesetzt werden. Es sollte länger als sieben Zeichen lang sein und Sonderzeichen und Ziffern enthalten. Sie sollten nicht in einem Wörterbuch zu finden sein und nicht aus einem Wort bestehen, dem Ziffern oder Sonderzeichen vor- bzw. nachgestellt sind.

43. Voreingestellte und leere Passwörter sollten geändert werden. Denn diese sind einfache Angriffsziele für Hacker, da sie meistens bekannt sind.

44. Arbeitsplatzrechner sollten beim Verlassen mit Bildschirmschoner und Kennwort geschützt werden, um ein Missbrauch durch Dritte zu verhindern.

45. Sensitive Daten und Systeme müssen geschützt werden.

Schutz vor Katastrophen und Elementarschäden

46. Notfallchecklisten sollten jedem Mitarbeiter bekannt sein, um beim Auftreten eines Notfalls den Schaden zu begrenzen und gegebenenfalls zu beheben.

47. Damit alle wichtigen Daten wiederhergestellt werden können, müssen diese regelmäßig gesichert werden (Backup).

48. Gegen Feuer, Überhitzung, Wasserschäden und Stromausfälle müssen IT-Systeme angemessen geschützt werden.

49. Maßnahmen zum Schutz vor Einbrechern und dem unbefugten Betreten müssen umgesetzt werden.

50. Inventarlisten über Hard- und Software sollten erstellt und regelmäßig aktualisiert werden, nicht zuletzt aus versicherungstechnischen Gründen.

Da dieses nur ein Auszug aus dem Leitfaden IT-Sicherheit [BSI-L 03] in Bezug auf die Anforderungen ist, ist es den Verantwortlichen des Unternehmens nahe zu legen, die hier nicht aufgeführten Maßnahmen nachzulesen, um weiterführende Maßnahmen vornehmen zu können. Eine weitere ausführlichere Quelle ist das vom BSI herausgegebene IT-Grundschutzhandbuch [BSI-G 03].

5.8.2 Sicherheitssoftware

Da die Sicherheitssoftware keinen Schutz vor Manipulation und Auslesen vertraulicher Informationen bietet, müssen entsprechende Maßnahmen vorgenommen werden. Um die Software vor Manipulation zu schützen, dürfen keine Schreibrechte auf die Dateien vergeben werden, mit Ausnahme der Dateien für das Protokollieren und der Konfigurationsdateien, die der Anwender ändern darf. Server müssen bei Abwesenheit eines Administrators stets gesperrt sein, da sich sonst eine unberechtigte Person an diesen setzen und die Passwortdatei auslesen kann.

Kapitel 6

Implementierung

In diesem Kapitel soll anhand eines Prototyps gezeigt werden, dass sich Anwendungen erweitern lassen. Dazu wird eine Testumgebung, in der der Prototyp getestet wird, die Implementierung des Prototyps und der Ablauf des Tests beschrieben sowie die Ergebnisse aufgeführt.

6.1 Testumgebung

Als Testumgebung werden zwei Computer genutzt, auf denen J2SE[1] (Java 2 Platform, Standard Edition) und eine Standardinstallation von Windows XP installiert sind. Einer der beiden Computer wird als Server genutzt und der zweite als Client. Zusätzlich wird auf einem der beiden PC's der "network protocol analyzer" Ethereal[2] installiert und gestartet, um die Daten sehen zu können, die zwischen Client und Server ausgetauscht werden.

Auf dem Server läuft als Dienst ein Testprogramm (siehe Abbildung 6.1) auf Port 1234 und als Test-TGS auf Port 1000, um den TGS zu simulieren. Bei diesem Programm wird im Edit-Feld "Port" eine Portnummer angegeben, für die der Testserver ein Serversocket öffnen soll und mit "initSocket" wird dann die Eingabe bestätigt. Bei einem erneuten Drücken des Buttons werden alle Sockets geschlossen und ein neues Serversocket erstellt. Nachrichten die er empfängt und versendet werden im Textfeld ausgegeben. Wenn z.B. "hello" empfangen wird, sieht die Ausgabe wie folgt aus "19:53:58 Received >> hello". Als Antwort sendet

[1] <http://java.sun.com/j2se/>
[2] <http://www.ethereal.com/>

der Testserver ”21 - 13 = 8”, ”Hallo Welt”, ”3 + 4 = 7” oder ”Test Test Test”. Im Textfeld wird dann z.B. ”19:53:58 Send >> 21 - 13 = 8” ausgegeben.

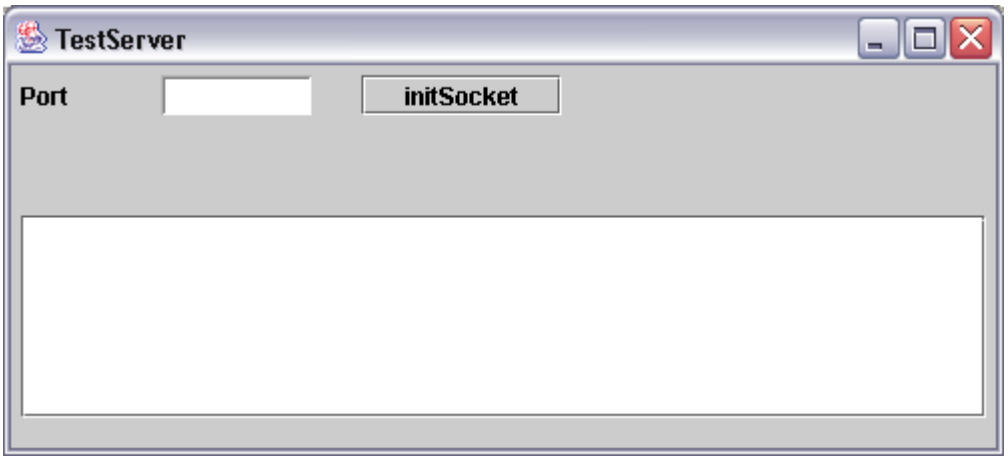


Abbildung 6.1: TestServer

Das Testprogramm auf dem Client (siehe Abb. 6.2) baut eine Verbindung zu dem im Editfeld ”IP:Port” angegebenen Serverdienst auf, wenn auf den Button ”connect” geklickt wird. Bei erneutem Klicken, wird die vorherige Verbindung geschlossen und eine neue Verbindung aufgebaut. Die Angabe in ”IP:Port” muss als ”<host>:<port>” angegeben werden. Dabei ist <host> entweder die IP-Adresse oder der Hostname und <port> die Portnummer. Jetzt können die im Editfeld ”Message” angegebenen Nachrichten mit dem Button ”send” an den Zieldienst geschickt werden. Wie bei dem Programm Testserver werden die gesendeten und empfangenen Nachrichten im Textfeld protokolliert.

Abbildung 6.2: TestClient

6.2 Prototyp

Von Microsoft wird seit der Einführung von Winsock 2, einer Schnittstelle von Windows für Anwendungen, um über ein Netzwerk kommunizieren zu können, die Möglichkeit angeboten, diese durch Layered Service Provider (LSP) in der Funktionalität zu erweitern [HOB 99]. Von diesen LSP können mehrere installiert sein und werden deshalb in einer LSP-Verkettung organisiert. Jeder dieser Provider kennt seinen nachfolgenden Provider, um Aufrufe von Funktionen weiterleiten zu können.

Um die Entwicklung eines LSP zu unterstützen, stellt Microsoft eine Vorlage[3] als Makefile-Projekt zur Verfügung. Beim Kompilieren des Projekts wird eine DLL, der Service Provider, und ein Installationsprogramm erstellt. Die DLL muss in den system32-Ordner des Windows-Verzeichnisses kopiert werden, um sie allen Programmen zugänglich zu machen. Das Installationsprogramm fügt dann den LSP an oberster Stelle der LSP-Verkettung hinzu. Wenn das Programm ein zweites Mal ausgeführt wird, entfernt es den Provider wieder.

Für die Implementierung der Sicherheitssoftware können die in Tabelle 6.1 beschriebenen Funktionen genutzt werden. In den Funktionen WSPBind und WSPConnect kann die Anmelderoutine aufgerufen werden, wenn der Benutzer noch nicht angemeldet ist. In WSPConnect kann ein Ticket vom TGS geholt werden, um dann eine sichere Verbindung aufzubauen. Hierfür muss in WSPAccept das empfangene Ticket überprüft werden, um den Sitzungsschlüssel zu erhalten oder gegebenenfalls die Verbindung sofort zu schließen und auf die nächste zu warten. In den Funktionen WSPRecv und WSPSend kann dann das Ver-

Funktion	**Beschreibung**
WSPAccept	Nimmt bedingt eine Verbindung an.
WSPBind	Bindet ein Socket an einen lokalen Port.
WSPCloseSocket	Schließt ein Socket.
WSPConnect	Baut eine Verbindung auf.
WSPGetPeerName	Gibt die IP-Adresse und den Port des Dienstes zurück, mit dem der Socket verbunden ist.
WSPGetSockName	Gibt den lokalen Port des Sockets zurück.
WSPRecv	Empfängt Daten.
WSPSend	Versendet Daten.

Tabelle 6.1: Auszug des Service Provider Interface (SPI) [MSDN 04]

[3] <ftp://ftp.microsoft.com/bussys/Winsock/Winsock2/>

und Entschlüsseln der Kommunikationsdaten stattfinden. Und in WSPCloseSocket können die sensitiven Daten, wie der Sitzungschlüssel, überschrieben werden.

Um zu zeigen, dass die Kommunikationsdaten manipuliert werden können, kodiert der Prototyp die Nachrichten, die an den Serverdienst (Port 1234) gesendet werden und die dieser sendet, mit einem einfachen XOR-Stromchiffre. Als Schlüssel wird 0x0F benutzt, um bei ASCII-Code die Buchstaben durch andere Buchstaben zu ersetzen. Beispiel:

Verschlüsselung:	0110 0011 (c)	xor	0000 1111 (0x0F)	=	0110 1100 (l)
Entschlüsselung:	0110 1100 (l)	xor	0000 1111 (0x0F)	=	0110 0011 (c)

Da vom TGS ein Ticket für eine sichere Verbindung benötigt wird, baut der Prototyp bei einem Verbindungsaufbau zum Serverdienst an Port 1234 zuerst eine Verbindung zum Test-TGS auf und sendet diesem die Nachricht "TGS Ticket-Anfrage". Danach wartet er auf die Antwort des Dienstes, bevor die eigentliche Verbindung aufgebaut wird. Dadurch zeigt der Prototyp, dass er nicht an ein Socket gebunden ist, sondern weitere erstellen kann, um mit dem KDC kommunizieren zu können. Da bei den in Tabelle 6.1 aufgelisteten Funktionen die Sockets übergeben werden, zeigt dieses auch, dass der Prototyp nach belieben Nachrichten senden und empfangen sowie die Sockets schließen kann, ohne dass die eigentliche Anwendung dieses übernehmen muss. Und um die Anmeldung zu simulieren, erscheint ein einfacher Dialog (Abbildung 6.3), wenn zum ersten Mal eine Verbindung zum Server Port 1234 aufgebaut oder ein Serversocket für Port 1234 erstellt wird. Die Anmeldung bleibt beim Prototyp so lange bestehen, bis alle Prozesse beendet werden, die die DLL einbinden. Ein ungewolltes Abmelden wird aber bei einer kompletten Implementierung nicht auftreten, wenn beim Login ein Programm gestartet wird, welches die DLL einbindet. Dieses Programm kann z.B. als Tray Icon[4] in der Taskleiste erscheinen und dem Anwender die Funktionen Anmelden, Abmelden und Passwort ändern anbieten.Außerdem ist es sinnvoll, wenn das Programm beim Beenden ein Logout durchführt. So wird sicher

Abbildung 6.3: Prototyp-Dialog: Login

[4]Eric Gunnerson: *Creating a System Tray Application*
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncscol/html/csharp06102002.asp>

gestellt, dass keine Tickets und Schlüssel im Speicher bleiben, wenn sich der Benutzer lokal am Betriebssystem abmeldet.

6.3 Testergebnisse

Mit der Clientsoftware wurde eine Verbindung zum Dienst an Port 1234 des Servers aufgebaut und dann zuerst die Nachricht ”Dies ist eine Testnachricht!” gesendet. Als Antwort wurde dann ”21 - 13 = 8” empfangen. Als nächstes wurde die Nachricht ”Zur Kontrolle eine zweite Nachricht” an den Serverdienst geschickt und ”Hallo Welt” als Antwort erhalten. In den Abbildungen 6.4 und 6.5 ist zu sehen, dass die vom Server bzw. vom Client gesendeten Nachrichten jeweils vom Anderen in lesbarer Form erhalten wurden.

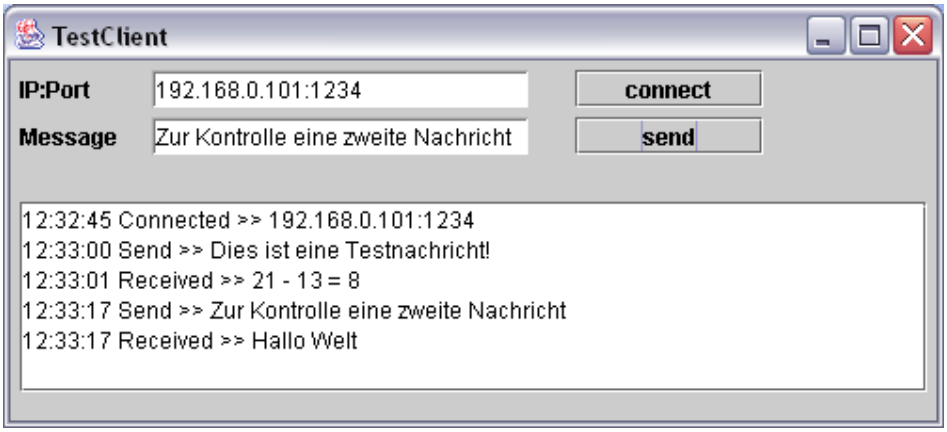


Abbildung 6.4: Screenshoot des TestClient

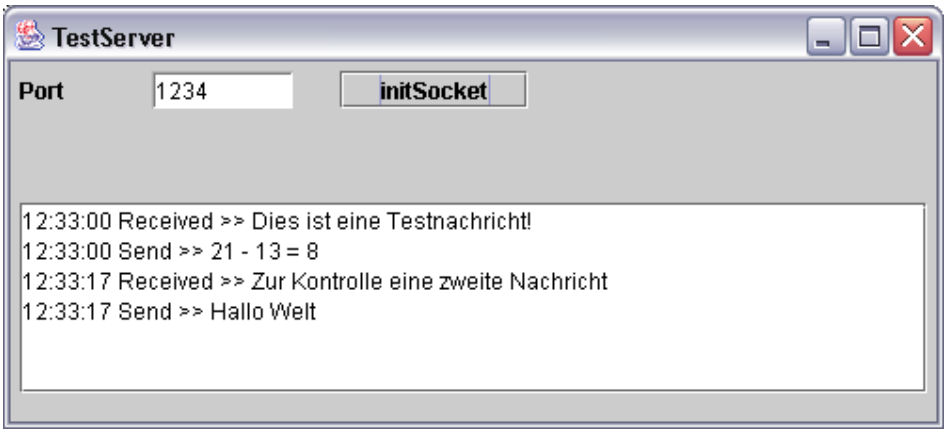
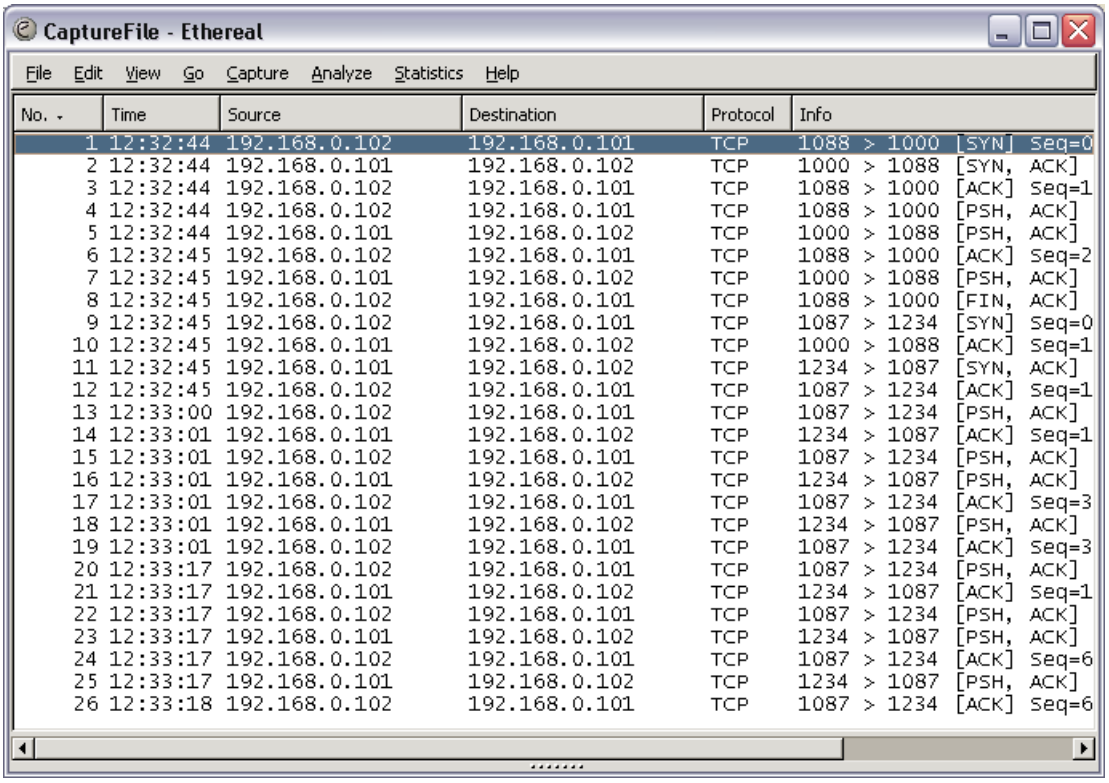


Abbildung 6.5: Screenshoot des TestServer



No.	Time	Source	Destination	Protocol	Info
1	12:32:44	192.168.0.102	192.168.0.101	TCP	1088 > 1000 [SYN] Seq=0
2	12:32:44	192.168.0.101	192.168.0.102	TCP	1000 > 1088 [SYN, ACK]
3	12:32:44	192.168.0.102	192.168.0.101	TCP	1088 > 1000 [ACK] Seq=1
4	12:32:44	192.168.0.102	192.168.0.101	TCP	1088 > 1000 [PSH, ACK]
5	12:32:44	192.168.0.101	192.168.0.102	TCP	1000 > 1088 [PSH, ACK]
6	12:32:45	192.168.0.102	192.168.0.101	TCP	1088 > 1000 [ACK] Seq=2
7	12:32:45	192.168.0.101	192.168.0.102	TCP	1000 > 1088 [PSH, ACK]
8	12:32:45	192.168.0.102	192.168.0.101	TCP	1088 > 1000 [FIN, ACK]
9	12:32:45	192.168.0.102	192.168.0.101	TCP	1087 > 1234 [SYN] Seq=0
10	12:32:45	192.168.0.101	192.168.0.102	TCP	1000 > 1088 [ACK] Seq=1
11	12:32:45	192.168.0.101	192.168.0.102	TCP	1234 > 1087 [SYN, ACK]
12	12:32:45	192.168.0.102	192.168.0.101	TCP	1087 > 1234 [ACK] Seq=1
13	12:33:00	192.168.0.102	192.168.0.101	TCP	1087 > 1234 [PSH, ACK]
14	12:33:01	192.168.0.101	192.168.0.102	TCP	1234 > 1087 [ACK] Seq=1
15	12:33:01	192.168.0.102	192.168.0.101	TCP	1087 > 1234 [PSH, ACK]
16	12:33:01	192.168.0.101	192.168.0.102	TCP	1234 > 1087 [PSH, ACK]
17	12:33:01	192.168.0.102	192.168.0.101	TCP	1087 > 1234 [ACK] Seq=3
18	12:33:01	192.168.0.101	192.168.0.102	TCP	1234 > 1087 [PSH, ACK]
19	12:33:01	192.168.0.102	192.168.0.101	TCP	1087 > 1234 [ACK] Seq=3
20	12:33:17	192.168.0.102	192.168.0.101	TCP	1087 > 1234 [PSH, ACK]
21	12:33:17	192.168.0.101	192.168.0.102	TCP	1234 > 1087 [ACK] Seq=1
22	12:33:17	192.168.0.102	192.168.0.101	TCP	1087 > 1234 [PSH, ACK]
23	12:33:17	192.168.0.101	192.168.0.102	TCP	1234 > 1087 [PSH, ACK]
24	12:33:17	192.168.0.102	192.168.0.101	TCP	1087 > 1234 [ACK] Seq=6
25	12:33:17	192.168.0.101	192.168.0.102	TCP	1234 > 1087 [PSH, ACK]
26	12:33:18	192.168.0.102	192.168.0.101	TCP	1087 > 1234 [ACK] Seq=6

Abbildung 6.6: Ablaufverfolgung der Kommunikation

Die Abbildung 6.6 zeigt den aufgezeichneten Nachrichtenaustausch. Anhand der TCP-Segmente 1 bis 3 ist zu sehen, dass der Client zuerst eine Verbindung zum Test-TGS (Port 1000) aufgebaut hat. Mit dem TCP-Segment 4 (Abbildung 6.8) sendete er dann die Ticket-Anfrage und erhielt in 5 und 7 (Abbildung 6.9) die Antwort. Mit den TCP-Segmenten 8 und 10 wurde diese Verbindung danach geschlossen. Abbildung 6.7 zeigt die empfangene und die gesendete Nachricht des Test-TGS.

Erst mit den TCP-Segmenten 9, 11 und 12 wurde eine Verbindung zwischen dem Client und dem Serverdienst an Port 1234 aufgebaut. ”Dies ist eine Testnachricht!” wurde in den Segmenten 13 und 15 an den Server gesendet. Die Antwort ”21 - 13 = 8” wurde in 16 und 18 übertragen. Der Client sendete dann in den TCP-Segmenten 20 und 22 ”Zur Kontrolle eine zweite Nachricht” und ”Hallo Welt” wurde dann in den Segmenten 23 und 25 vom Server übertragen. Die Abbildungen 6.10 und 6.11 zeigen als Beispiel den Inhalt der TCP-Segmente 22 und 25, um vorzuweisen, dass der Datenaustausch zwischen Client und Server (Port 1234) nicht lesbar, sondern verschlüsselt ist.

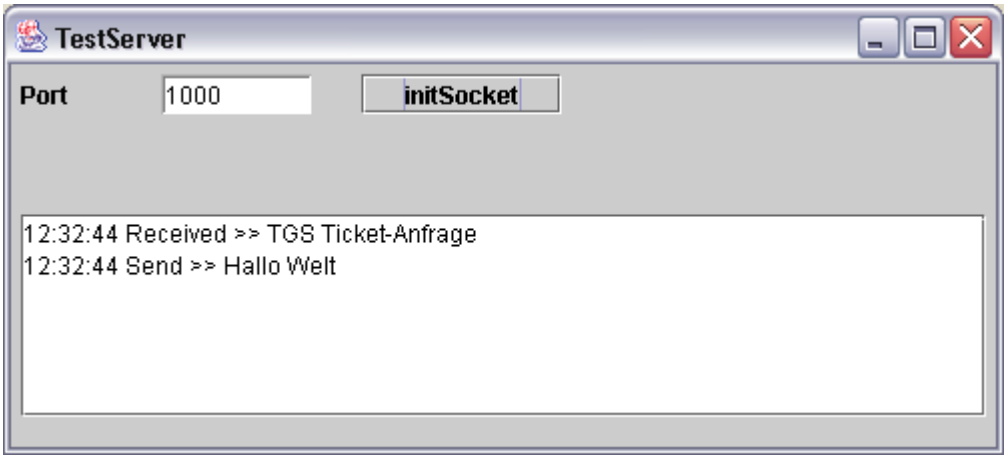


Abbildung 6.7: Screenshoot des TestServer (Test-TGS)

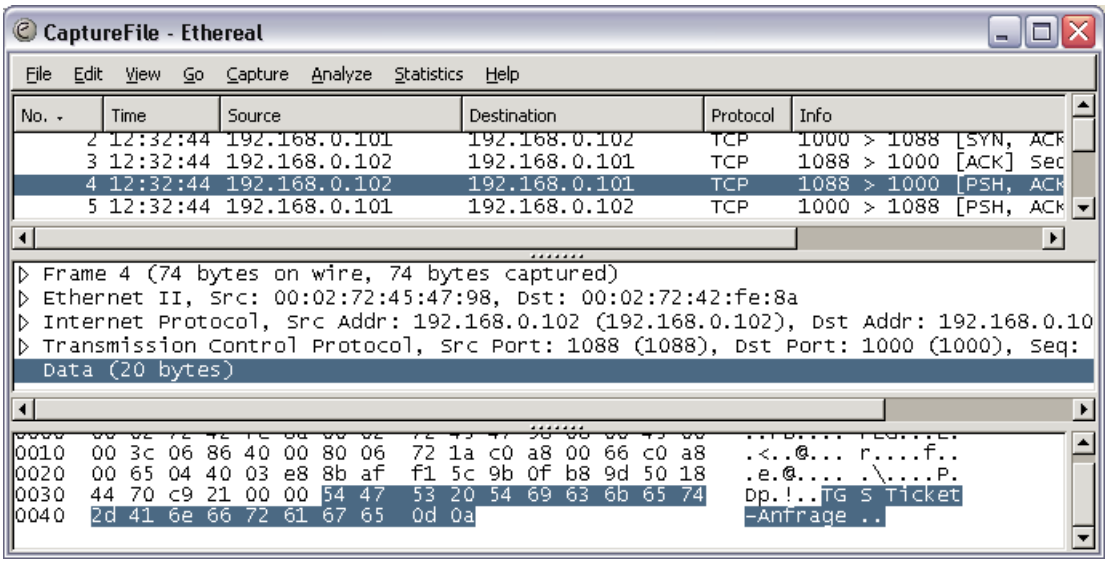


Abbildung 6.8: Übertragene Daten an den Test-TGS

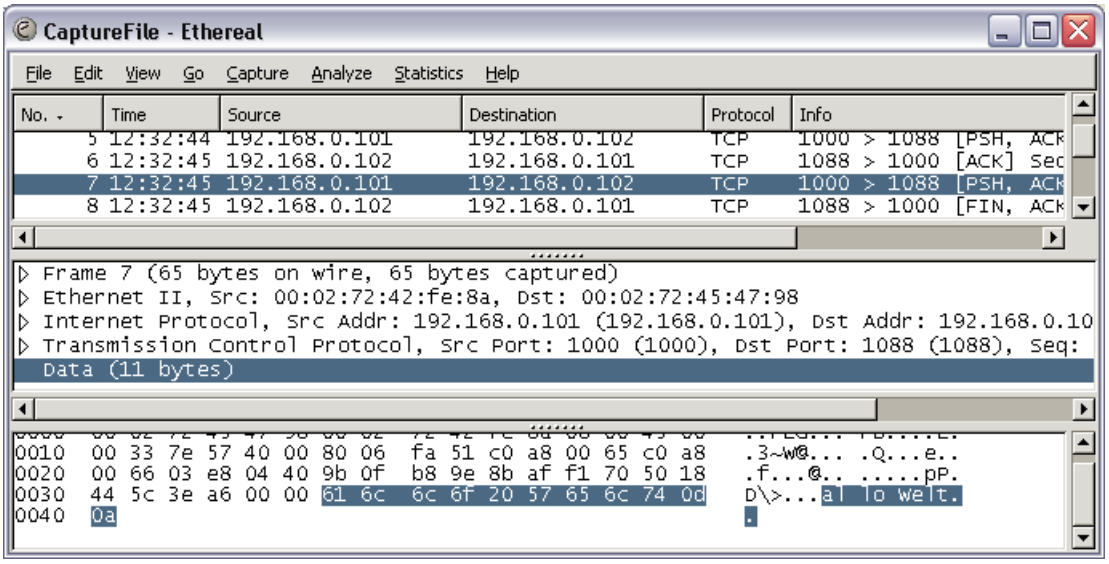


Abbildung 6.9: Übertragene Daten vom Test-TGS

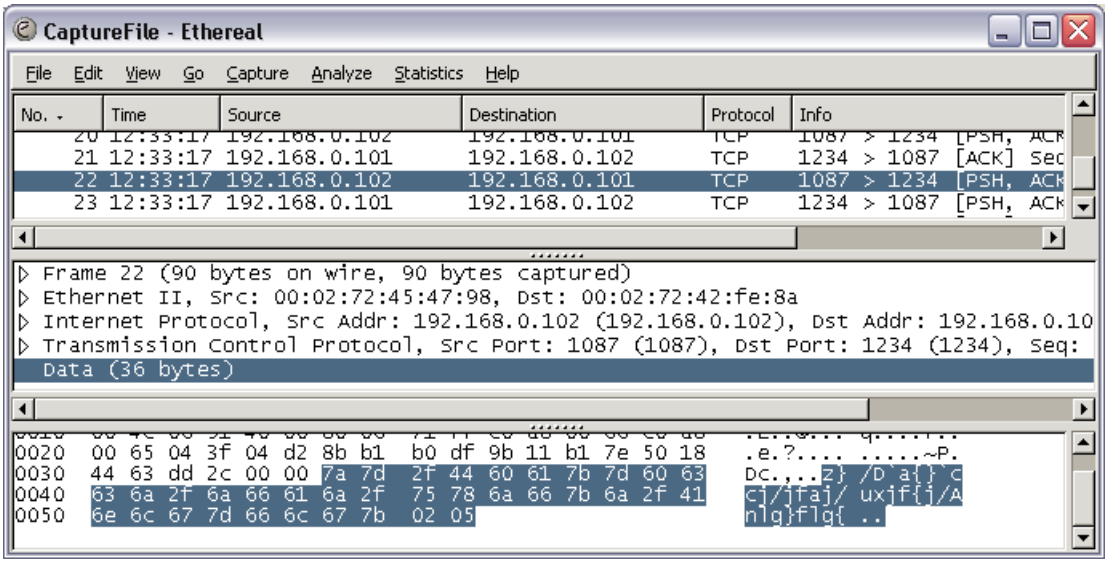


Abbildung 6.10: Übertragene Daten vom Client

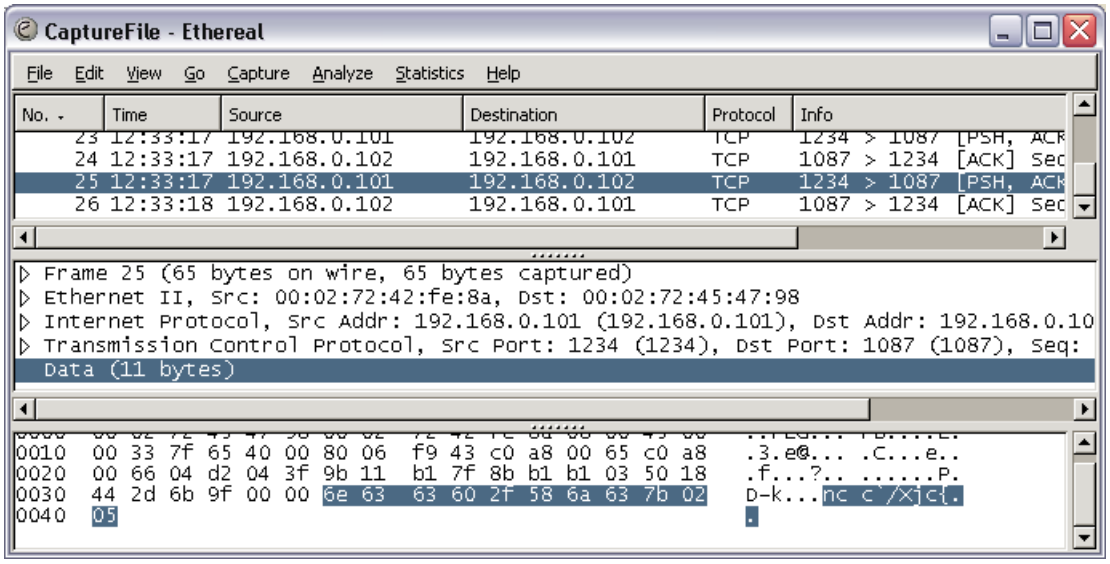


Abbildung 6.11: Übertragene Daten vom Server

Kapitel 7

Zusammenfassung und Ausblick

In dieser Bachelorarbeit ging es um den Entwurf einer Sicherheitsarchitektur für das unter 1.2 beschriebene verteilte System. Die Besonderheit dieses Entwurfs liegt in dem geringen Aufwand, der für die nachträgliche Integration in das bestehende System benötigt wird. Denn kleine und mittlere Institutionen haben nur ein begrenztes Budget zur Verfügung und wünschen sich deshalb eine kostengünstige Lösung.

Als erstes wurden die Ziele analysiert, um alle Anforderungen an die Sicherheitsarchitektur zu bestimmen. Es wurde untersucht, von welchen Faktoren der Aufwand abhängig ist bzw. welche Faktoren Kosten verursachen. Um die möglichen Gefahren für die Sicherheitsziele zu erkennen, wurden Bedrohungsbäume erstellt. Auf Basis dieser Anforderungen wurden dann PKI und Kerberos untersucht, welche Vor- und Nachteile sie im Bezug auf die Aufgabenstellung haben. Als nächstes wurde dann eine Sicherheitssoftware, die die bestehenden verteilten Anwendungen erweitert und ein Maßnahmenkatalog entworfen, um die Ziele mit Ausschluss der Bedrohungen zu erreichen. Danach wurde ein Prototyp implementiert und die Testergebnisse aufgeführt, um zu zeigen, dass die Sicherheitssoftware realisierbar ist.

Der Entwurf gewährleistet die Sicherheitsziele, indem die Sicherheitssoftware ein Kerberos-ähnliches System integriert. Es mussten Anpassungen gegenüber Kerberos vorgenommen werden, damit die Anwendungen nicht angepasst werden müssen. Denn dieses würde einen hohen Aufwand verursachen, den die Institutionen nicht möchten. Dafür musste aber der Kompromiss eingegangen werden, dass die Authentifikation nicht von den Diensten übernommen werden, sondern der Anwender sich weiterhin an diesen Anmelden muss. Daraus folgt, dass die Autorisation darauf eingeschränkt ist, welcher Principal einen Dienst nutzen darf. Dieses wird vom KDC übernommen, indem er ein Ticket nur herausgibt, wenn die

Berechtigung vorhanden ist. Zusätzlich wurde der Maßnahmenkatalog (siehe 5.8) erstellt, um den Gefahren vorzubeugen, die die Sicherheitssoftware nicht berücksichtigt.

Der Einsatz der Sicherheitsarchitektur ist nicht nur für kleine und mittlere Institutionen interessant, die in ihre IT-Systeme nachträglich Sicherheit integrieren wollen, sondern für jeden, der mit geringem Aufwand sein System schützen möchte. Aber auch wenn eine bestimmte verteilte Anwendung genutzt werden soll, die keine oder kaum Sicherheit bietet, ist der Einsatz des Entwurfs interessant. Wichtig ist, dass die IT-Systeme aus Rechnern bestehen, die eine Windows-Version mit Winsock 2 als Betriebssystem nutzen. Auch Windows 95 ließe sich mit einem entsprechenden Add-On Paket aufrüsten (vgl. [HOB 99]). Server, die keinen durch die Sicherheitssoftware zu schützenden Dienst anbieten, können auch andere Betriebssysteme nutzen, da zu diesen eine normale Verbindung aufgebaut wird.

Die Sicherheitssoftware ist sehr speziell, da in der Arbeit die Integration einer Sicherheitsarchitektur in ein bestehendes System vorrangig behandelt wurde. Hier ist es interessant den Entwurf weiter zu entwickeln, um zukünftigen Anwendungen eine Schnittstelle zu bieten, die Authentifikation übernehmen zu können und um verteilte Anwendungen zu unterstützen, die Kerberos benutzen.

Da häufig auch andere Betriebssysteme genutzt werden, z.B. Linux oder Mac OS und Netzwerke auch aus heterogenen Systemen bestehen können, wäre es interessant zu untersuchen, ob und wie sich die Sicherheitsarchitektur auf diese Systeme migrieren lässt.

Glossar

AS Authentication Service

Brute-Force Bei diesem Angriff werden systematisch alle möglichen Schlüssel/ Passwörter ausprobiert.

BSI Bundesamt für Sicherheit in der Informationstechnik

Hijaking Dies ist ein Netzwerkangriff, bei dem der Angreifer eine bereits bestehende Verbindung übernimmt.

IP-Spoofing Dies ist ein Netzwerkangriff, bei dem der Angreifer die Absenderadresse der IP-Packete fälscht.

KDC Key Distribution Center

LSP Layered Service Provider
Diese erweitern unter Windows die Funktionalität der Sockets. (siehe auch *SPI*)

MAC Massage Authentication Code

Man-in-the-Middle Dies ist ein Netzwerkangriff, bei dem der Angreifer sich A gegenüber als B ausgibt und B gegenüber als A ausgibt.

PKI Public Key Infrastructure

Principal ist ein eindeutig benannter Benutzer oder Server(dienst), der an einer Netzwerkkommunikation teilnimmt

Schadsoftware ist ein Sammelbegriff für Programme, die gewollt eine boshafte Absicht verfolgen, wie z.B. Dateien löschen oder geheime Informationen preisgeben.

Sniffer ist ein Programm, um den Netzwerkverkehr aufzuzeichnen

Social Engineering Bei diesen Angriffen wird das Opfer dazu gebracht schützenswerte Informationen preiszugeben, z.B. durch geschicktes Ausfragen oder Täuschung.

SPI Service Provider Interface
Schnittstelle von Windows, die die *LSP* implementieren müssen

TCP ist ein Protokoll, das eine Verbindung zwischen zwei Hosts aufbaut und die zuverlässige Übertragung von Paketen gewährleistet. (siehe auch *UDP*)

TGS Ticket Granting Service

TGT Ticket Granting Ticket

Tray Icon ist ein Symbol, das bei Windows unten rechts neben der Uhr erscheint. Dieses kann als User-Interface für eine Anwendung genutzt werden, die kaum mit dem Benutzer interagiert, aber nebenbei laufen muss, um ihre Aufgaben zu bewältigen. Dies kann z.B. ein Virenschutzprogrammen sein.

Trojanische Pferde sind *Schadprogramme*, die Angreifern einen Zugang (Hintertür) zum befallenen Computer geben.

UDP ist ein verbindungsloses Protokoll, das Pakete zum Zielhost versendet, ohne zu gewährleisten, dass diese ankommen. (siehe auch *TCP*)

Viren werden im IT-Bereich *Schadprogramme* bezeichnet, die sich selbst reproduzieren können. Aber um ausgeführt zu werden, brauchen sie Wirtsprogramme.

Winsock ist eine von Windows bereitgestellte Schnittstelle für Anwendungen, um über ein Netzwerk kommunizieren zu können.

Wörterbuch-Angriff Bei diesem Angriff werden alle Wörter aus einem gegebenen Wörterbuch als Passwort ausprobiert.

Würmer werden im IT-Bereich *Schadprogramme* bezeichnet, die sich selbst reproduzieren können und im Gegensatz zu *Viren* selber ausführbare Programme sind.

Literaturverzeichnis

[BSI-G 03] Bundesamt für Sicherheit in der Informationstechnik: *IT-Grundschutzhandbuch, 5. EL Stand Oktober 2003*. Im Internet zu finden unter: <http://www.bsi.de/gshb/deutsch/download/GSHB2003.pdf> (23.Juli 2004)

[BSI-L 03] Bundesamt für Sicherheit in der Informationstechnik: *Leitfaden IT-Sicherheit, Stand Oktober 2003*. Im Internet zu finden unter: <http://www.bsi.de/gshb/Leitfaden/GS-Leitfaden.pdf> (14.Juli 2004)

[Cis 04] Cisco Systems, Inc: *Internet Protocols (IP)*. Im Internet zu finden unter: <http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ip.htm> (8.September 2004)

[Eck 03] Claudia Eckert: *IT-Sicherheit, 2. Aufl.* Oldenbourg, 2003. - ISBN 3-486-27205-5

[HOB 99] Wei Hua, Jim Ohlund, Barry Butterklee: *Unraveling the Mysteries of Writing a Winsock 2 Layered Service Provider*, Mai 1999. Im Internet zu finden unter: <http://www.microsoft.com/msj/0599/LayeredService/LayeredService.aspx> (13.September 2004)

[IETF 04] The Internet Engineering Task Force: *Public-Key Infrastructure (X.509) (pkix)*, 7.September 2004. Im Internet zu finden unter: <http://www.ietf.org/html.charters/pkix-charter.html> (25.September 2004)

[Khb 01] Bernd Kahlbrandt: *Software-Engineering mit der Unified Modeling Language, 2. Aufl.* Springer, 2001. - ISBN 3-540-41600-5

[KR 02] James F. Kurose, Keith W. Ross: *Computernetze, 1. Aufl.* Pearson Studium, 2002. - ISBN 3-8273-7017-5

[MIT 04] Massachusetts Institute of Technology: *Kerberos: The Network Authentication Protocol*. Im Internet zu finden unter:
<http://web.mit.edu/kerberos/> (12.September 2004)

[MSDN 04] Microsoft Corporation: *Winsock Reference*. Im Internet zu finden unter:
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/winsock/winsock_reference.asp> (13.September 2004)

[RFC 1321] R. Rivest: *The MD5 Message-Digest Algorithm*, RFC 1321, April 1992. Im Internet zu finden unter:
<ftp://ftp.rfc-editor.org/in-notes/rfc1321.txt> (25.September 2004)

[RFC 1510] J. Kohl, C.Neuman: *The Kerberos Network Authentication Service (V5)*, RFC 1510, September 1993. Im Internet zu finden unter:
<ftp://ftp.rfc-editor.org/in-notes/rfc1510.txt> (30.Juli 2004)

[RFC 2459] R. Housley, W. Ford, W. Polk, D. Solo: *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, RFC 2459, Januar 1999. Im Internet zu finden unter:
<ftp://ftp.rfc-editor.org/in-notes/rfc2459.txt> (12.September 2004)

[RFC 3244] M. Swift, J. Trostle, J. Brezak: *Microsoft Windows 2000 Kerberos Change Password and Set Password Protocols*, RFC 3244, Februar 2002. Im Internet zu finden unter:
<ftp://ftp.rfc-editor.org/in-notes/rfc3244.txt> (15.September 2004)

[Sal 04] Theo Saleck: *Chefsache IT-Kosten, 1. Aufl.* Vieweg, 2004. -
ISBN 3-528-05872-2

Anhang A

Die zugehörige CD-ROM

Inhalt

Im Wurzelverzeichnis befinden sich die Verzeichnisse ”Projects”, ”Binaries”, ”Testresults” und ”Documents”.

Das Verzeichnis ”Projects” enthält zwei weitere Verzeichnisse. Das Verzeichnis ”LSP”, in dem das C++ Projekt für den Prototyp enthalten ist, welches mit der Entwicklungsumgebung Microsoft Visual C++ 6.0 erstellt wurde. Das zweite Verzeichnis ”Testapplication” enthält die Java-Projekte für den TestClient und den TestServer. Diese wurden mit der Entwicklungsumgebung Eclipse[1] Version 2.1.0 erstellt.

Das Verzeichnis ”Binaries” enthält die kompilierten Programme. Der TestClient und der TestServer sind als ausführbare JAR-Dateien, der Layered Socket Provider als *BecomeSecure.dll* und das Installationsprogramm als *inst_BecomeSecure.exe* enthalten. Zusätzlich gibt es im Verzeichnis die DLL *sporder.dll*, die von dem Installationsprogramm benötigt wird.

Im Verzeichnis ”Testresults” sind die Screenshoots vom Test enthalten und die Datei *CaptureFile*, die sich mit Ethereal[2] einsehen lässt und die alle Kommunikationsdaten enthält.

Das Verzeichnis ”Documents” enthält diese Bachelorarbeit als PDF-Datei und weitere Textdateien mit Informationen zu den Projekten und zu den erstellten Programmen.

[1] <http://www.eclipse.org>
[2] <http://www.ethereal.com/>

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den 01.10.2004

Sven Elvers