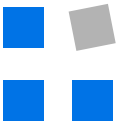

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Institute for
Internet Technologies
and Applications

Trusted Network Access Control

Semesterarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil
Institut für Internet-Technologien und -Anwendungen

Wolfgang Altenhofer, Lukas Studer

23. Dezember 2010

Experte: Prof. Dr. Andreas Steffen

Gegenleser: Walter Sprenger

Erklärung über die eigenständige Arbeit

Wir erklären hiermit,

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

Ort, Datum:	Ort, Datum:
Wolfgang Altenhofer	Lukas Studer

Abstract

Zielsetzung

Das Network Access Control (NAC) stellt ein zusätzlicher Schutz für das Netzwerk dar. Die Clients werden, bevor sie Zugriff auf die Netzwerkinfrastruktur erhalten, nach definierten Regeln geprüft und erst bei bestehen dieser Kontrolle in das Netzwerk gelassen.

Das Trusted Platform Module (TPM) ist ein im Computer festverbauter Chip, welcher dem Rechner zusätzliche Sicherheitsfunktionalitäten auf Hardwareebene zur Verfügung stellt.

Im Rahmen dieser Semesterarbeit soll sich das Team in die neue Thematik NAC und TPM einarbeiten. Es soll mit einem Proof of Concept gezeigt werden, dass sich das NAC (anhand des Microsoft Network Access Protection (NAP)) sinnvoll mit dem TPM kombinieren lässt.

Ergebnis

Als Resultat dieser Semesterarbeit hat sich das Team erfolgreich in die Themengebiete eingearbeitet. Anhand eines Prototypen ist nun bekannt, was alles benötigt wird, um erfolgreich eigene Zulassungs-Richtlinien in das NAP-System zu integrieren. Ebenfalls wurde eine Kommunikation mit dem TPM hergestellt. Aufgrund der algorithmischen Komplexität und der aufwendigen Kommunikation wurde das Handling mit dem TPM einwenig unterschätzt. Deshalb konnten nicht alle gewünschten Funktionalitäten umgesetzt und getestet werden.

Ausblick

Der jetzige Projektstand bietet eine umfassende Grundlage. Das Projekt soll deshalb als Bachelorarbeit weitergeführt werden. Es ist geplant, dass weitere Funktionalitäten des TPMs verfügbar gemacht werden. Zudem soll das TPM in die NAC-Technologie integriert werden, um das NAC manipulationssicherer zu gestalten.

Aufgabenstellung

EIN INSTITUT DER
HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Institute for
Internet Technologies
and Applications

Studienarbeit 2010

Trusted Network Access Control

Studenten: Wolfgang Altenhofer, Lukas Studer

Betreuer: Prof. Dr. Andreas Steffen

Ausgabe: Montag, 20. September 2010

Abgabe: Donnerstag, 23. Dezember 2010

Einführung

Trusted Network Access Control wurde vor ein paar Jahren von Cisco, Microsoft und anderen Firmen eingeführt, um den Gesundheitszustand eines Rechners auf Herz und Nieren zu prüfen, bevor er via WLAN oder VPN in das geschützte Heimatnetzwerk aufgenommen wird. Falls gewisse Mindestanforderungen betreffend Betriebssystem-Patches, Firewall-Einstellungen und Viren-Scanner-Updates nicht erfüllt werden, wird der Host in eine Quarantänezone verwiesen, wo er zunächst aufdatiert und entseucht werden muss.

In letzter Zeit wurden die verschiedenen proprietären Ansätze im Internet Standard "Network Endpoint Assessment" (NEA, RFC 5209) vereinigt. In dieser Studienarbeit sollen auf der Basis von Microsoft's Network Access Protection (NAP) mit Windows 7 als Client und Windows Server 2008 R2 als Network Access Authority die genauen Mechanismen untersucht werden.

Das Trusted Platform Module ist ein Chip, welcher den Computer um zusätzliche Sicherheitsfunktionalitäten erweitert. Im Rahmen dieser Arbeit soll abgeklärt werden, welche Methoden oder Mechanismen das TPM anbietet, um die Sicherheit des NAPs zu erhöhen.

Aufgabenstellung

- Einarbeitung in die Network Access Protection (NAP) Funktionalität von Microsoft.
- Aufsetzen einer NAP Testumgebung mit einem Windows 7 NAP Client und einer Windows Server 2008 R2 Infrastruktur.
- Analyse des Statement of Health (SoH) Protokolls von Microsoft im Zusammenspiel mit IPsec.
- Einarbeitung in die TPM-Architektur und Analyse der Einbindemöglichkeiten des TPMs in den NAP-Prozess.

Links

- Windows Server 2008 R2 Network Access Protection (NAP)
  http://www.microsoft.com/windowsserver2008/en/us/nap-technical-resources.aspx

- TNC IF-TNCCS: Protocol Bindings for SoH
  http://security.hsr.ch/mse/TCG/IF-TNCCS-SOH_v1.0_r8.pdf
- Trusted Computing Group: Trusted Platform Module
  http://www.trustedcomputinggroup.org/developers/trusted_platform_module

Rapperswil, 20. September 2010

Prof. Dr. Andreas Steffen

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Sicherheit ist in der Informatik ein immer wichtiger werdender Aspekt. Denn durch Sicherheitslücken in aktuellen Informatiksystemen können sich Schadprogramme einnisten und weiter verbreiten. Dies kann für Unternehmen wie auch für Privatpersonen zu beträchtlichem Schaden führen. In den letzten Jahren ist eine deutliche Zunahme an finanziell motivierter Internetkriminalität zu erkennen[13].

Ein weiteres Problem sind die gezielten Angriffe auf die Unternehmensstruktur mit der Absicht, die Unternehmen zu schädigen oder nicht mehr erreichbar zu machen.

Um die möglichen Angriffspunkte zu verkleinern oder ihre Auswirkungen abzuschwächen wurden mit dem Trusted Computing neue Konzepte eingeführt, welche eine Verbesserung im Hinblick auf die Computersicherheit bieten soll.

In dieser Arbeit werden auf zwei Aspekte, namentlich das Network Access Control (NAC) sowie das Trusted Platform Module (TPM), näher eingegangen. Das Network Access Control ermöglicht, dass nur Clients mit gewissen Eckwerten auf das Netzwerk zugreifen können. In dieser Arbeit soll anhand der Microsoftlösung, dem Network Access Protection (NAP), aufgezeigt werden, wie dieser Vorgang der Statusüberprüfung abläuft. Schliesslich wird auf die Kombinierung beider Technologien eingegangen, um das NAP mit Sicherheitsfunktionen erweitern zu können.

Neben der fachlichen Auseinandersetzung sollen diese Technologien auch anhand eines Prototyps ausgetestet werden. Diese Arbeit soll als Einführung in diesen Bereich dienen. Es ist gedacht, dass diese Arbeit im Frühlingssemester 2011 als Bachelor-Arbeit fortgesetzt wird. Aus diesem Grund soll in dieser Semesterarbeit das dazu notwendige Fundament aufgebaut werden.

Kapitel 2

Vision

2.1 Einführung

In diesem Kapitel soll aufgezeigt werden, welche Ziele in dieser Arbeit in Angriff genommen wurden. Zudem soll aufgezeigt werden, mit welcher Motivation diesen Aufgaben nachgegangen wurde. Dieses Kapitel wurde im Verlaufe der Arbeit noch detaillierter formuliert, da nach einigen Wochen das nötige Know-How aufgebaut wurde und deshalb eine realistische Einschätzung, die mit der zur Verfügung stehenden Zeit im Einklang war, ermöglicht wurde.

Mit den Zielen soll festgehalten werden, welche Aspekte und Thematiken den Kernfokus der Arbeit darstellen. Die Vorgehensschwerpunkte im Bereich NAP werden detailliert aufgezeigt. Ebenfalls wird abgeklärt, welche Mechanismen das TPM zur Erhöhung der Sicherheit im Zusammenhang mit dem Network Access Protection bietet und wie sehr sich diese eignen.

2.2 Einarbeitung in Thematik Microsoft NAP

Das von Microsoft entwickelte NAP ist beiden Teammitgliedern noch nicht bekannt. Deshalb soll eine Vertiefung dieses Themengebiets stattfinden.

2.3 Aufbau einer Testumgebung

Es soll eine Testumgebung aufgebaut werden, in welcher der NAP-Prozess aktiviert ist und die erlernten NAP-Eigenschaften ausgetestet werden können.

2.4 Verbindungsablauf im NAP-Netzwerk dokumentieren

Ein Schwerpunkt soll in der Analyse des NAP-Kommunikationsprozesses gelegt werden. Denn es soll dokumentiert werden, wie bei Microsoft die Kommunikation zwischen Client (NAP-Agent) und Server (NAP-Validator) vonstatten geht und welche Protokolle für die Datenübermittlung verwendet werden.

2.5 Einarbeitung in Thematik TPM

Das sich immer weiterverbreitende TPM soll den Computer um zusätzliche Sicherheitsfunktionalitäten erweitern. Die Thematik soll durch beide Teammitglieder vertieft und dokumentarisch festgehalten werden.

2.6 Abklärung Eignung TPM in NAP-Netzwerk

Es soll untersucht werden, ob die NAP-Funktionalität ebenfalls mit dem TPM kombiniert werden kann und wie solch eine Einbindung aussehen könnte.

2.7 Programmierung

Das erworbene Wissen über das NAP und das TPM soll nun in einem Prototyp demonstriert werden. Es soll gezeigt werden, dass die Integration des TPM in den NAP-Prozess einzubinden. In Abbildung 2.1 soll dargestellt werden, wie die Kommunikation zwischen TPM und NAP zusammenhängt und gemeinsam ein *Trusted Network Access Control* bilden.

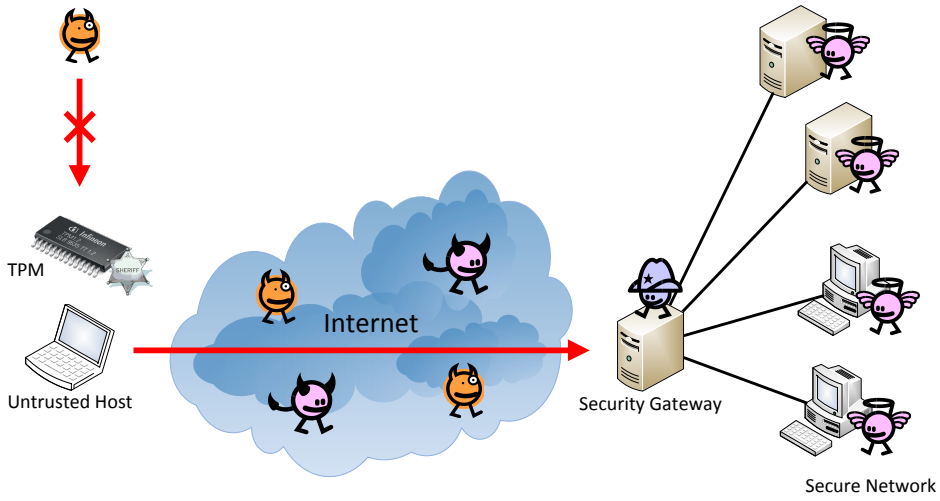

Abbildung 2.1: Übersicht über die Vision des Trusted Network Access Control

Kapitel 3

Grundlagen

3.1 Trusted Computing

3.1.1 Beschreibung

Trusted Computing beschreibt eine Technologie, die Endcomputer sicherer gestalten will, indem die Computerkonfiguration (Software und Hardware) kontrolliert werden kann. Das System soll eine korrekte Arbeitsweise ermöglichen. Es soll manipulationssicher und in der Funktionsweise verifizierbar sein.

Eine Möglichkeit dieser Umsetzung wird durch das Ziel verfolgt, den Computer mit einem zusätzlichen Chip auszurüsten, welcher die Integrität der Software wie auch der Hardware messen kann. Somit kann geprüft werden, ob Manipulationen an der Anwendersoftware, am Betriebssystem oder am Computer stattfanden.

Ein anderer Aspekt des Trusted Computing ist die Bewertung der Vertrauenswürdigkeit und das Nachverfolgen von Änderungen eines Systems aus der Ferne (Remote Attestation). Beispielsweise kann so festgestellt werden, ob auf dem Client ein Mindestmass an Sicherheit aktiviert hat ist.

In den weiteren Unterkapiteln wird vertieft auf diese zwei Aspekte des Trusted Computing eingegangen.

3.2 Trusted Platform Module (TPM)

Beim Trusted Platform Module, kurz TPM, handelt es sich um einen im Computer eingebauten Chip, der zusätzliche Sicherheitsfunktionen anbietet.

Er kann mit einer nicht portablen Smartcard verglichen werden, welche aber nicht an den Benutzer, sondern an den Computer gebunden ist.

Im Vergleich zu gängigen Chipfunktionalitäten bietet das TPM spezielle sicherheitstechnische Erweiterungen. Es kann erzeugte Prüfsummen sicher speichern und stellt kryptographische Funktionen bereit. Die angebotenen Sicherheitsfunktionen ermöglichen, dass ein Computer zusammen mit einem vertrauenswürdigen Betriebssystem eine Trusted Computing Plattform darstellt.

3.2.1 Aufbau des TPM

Das TPM besteht aus drei Hauptbereichen: der funktionellen Einheiten *(Functional Units)*, welche die kryptographischen Funktionalitäten zur Verfügung stellen, dem nichtflüchtigen Speicherbereich, welcher für die permanente Hinterlegung von Informationen (beispielsweise Schlüsseln) dient, sowie dem flüchtigen Speicher, der als temporärer Speicher genutzt wird.

In Abbildung 3.1 ist ersichtlich, mit welchen Komponenten das TPM ausgerüstet ist. Folgend soll auf die wichtigsten Komponenten eingegangen werden:

Symmetrische Verschlüsselung

Die symmetrische Verschlüsselung wird für die Verschlüsselung von TPM-intern verwendeten Authentisierungsdaten, wie beispielsweise Passwörtern oder Zahlencodes benötigt. Zudem wird es für die Kommunikation zwischen System und TPM oder Software und TPM und für die Verschlüsselung der Datenblöcke, welche ausserhalb des TPM abgelegt werden, eingesetzt.

Die symmetrische Verschlüsslung ist nur TPM-intern und nicht für externe Komponenten nutzbar.

Keyed-Hash Message Authentication Code (HMAC)-Komponente

Diese Komponente ermöglicht die Erzeugung des Message Authentication Code (MAC). Ein MAC basiert auf einer kryptographischen Hash-Funktion und einem geheimen symmetrischen Schlüssel und wird vorallem für den Nachweis der Integrität bei Nachrichten verwendet.

Die HMAC-Funktion wird nur TPM-intern angeboten. Für die Generierung von geschützten Nachrichten im System oder der Software muss auf eine Dritt-Komponente ausgewichen werden.

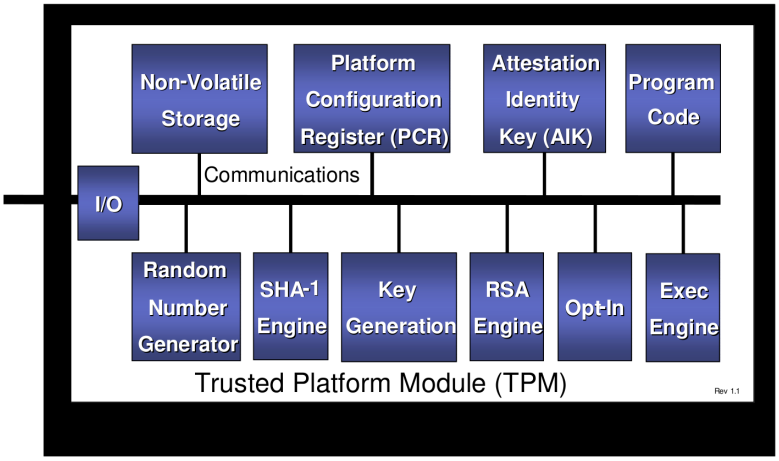

Abbildung 3.1: TPM: Aufbau und Funktionalitäten[8]

Random Number Generator

Der Zufallszahlgenerator wird TPM-intern für die Erzeugung von kryptographischen Schlüsseln verwendet. Diese Funktionalität ist aber ebenfalls von ausserhalb, mit der Methode *TPM_GetRandom*, nutzbar.

SHA-1-Hashgenerator

Die SHA-1-Komponente des TPMs ermöglicht dem TPM die Berechnung von SHA-1 Hashes. Diese Funktionalität wird vor allem für die Berechnung von HMACs verwendet. Diese Funktionalität kann ebenfalls durch Kommandos TPM-extern von Programmen verwendet werden.

RSA-Engine

Diese Komponente implementiert den asymmetrischen Algorithmus RSA. Meist werden für diese Funktionen 2048 Bit lange Schlüssel verwendet (obwohl gemäss Spezifikation auch 512, 768 und 1024 möglich wären).

Die RSA-Engine wird für die Ver- und Entschlüsselung von digitalen Signaturen benötigt. Ebenfalls werden hiermit sichere RSA-Schlüsselpaare generiert

3.2.2 Verwendete Schlüsselkategorien

Das TPM verwaltet die kryptographischen Schlüssel in einer Schlüsselhierarchie. Ausserdem besitzt es die Fähigkeit, eigene Schlüssel intern zu erzeu-

gen. Weil die Erzeugung innerhalb des TPMs erfolgt, kann sichergestellt werden, dass der private Teil des Schlüsselpaars nie durch Dritte ausgelesen werden kann. Bevor ein Schlüsselpaar generiert wird, kann zusätzlich bestimmt werden, ob es lediglich auf dem erzeugenden TPM genutzt werden kann (*non-migratable*) oder ob es sich auch auf andere TPMs transferiert lässt (*migratable*). Im letzteren Fall kann bei der Generierung ein Migrationspasswort mitgegeben werden.

Das TPM verwendet verschiedene Schlüsseltypen, welche je nach Anwendungszweck genutzt werden können. Die einzelnen Schlüsselkategorien werden nachfolgend erläutert:

Storage Keys

Dieser Schlüsseltyp wird für die Verschlüsselung von beliebigen Daten benutzt. Er wird ebenfalls von anderen Schlüsseln verwendet, um die Schlüsselhierarchie aufzubauen und die Schlüssel ausserhalb des TPMs sicher abzulegen.

Signing Keys

Signing Keys werden für das Signieren von Daten benutzt und können plattformgebunden oder migrierbar sein.

Bind Keys

Diese Schlüssel dienen der Verschlüsselung von kleineren Datenmengen, wie beispielsweise symmetrische Schlüssel, welche wieder für die Verschlüsselung von grösseren Mengen an Daten ausserhalb des TPMs verschlüsselt werden.

Legacy Keys

Diese Keys sind eine Kombination aus den Signing Keys und Bind Keys und werden somit für die Verschlüsselung und das Erstellen von digitalen Signaturen verwendet.

3.2.3 Spezielle Schlüssel

Endorsement Key (EK)

Der Endorsement Key (EK) ist ein 2048-bit langes RSA-Schlüsselpaar. Der Chip-Hersteller generiert dieses Paar während des Produktionsprozesses innerhalb des TPM. Dieses Schlüsselpaar kann zu einem späteren Zeitpunkt weder geändert noch gelöscht werden.

Das Schlüsselpaar enthält zudem ein EK-Zertifikat, das garantiert, dass der Generierungsprozess von einem autorisierten Hersteller ausgeführt wurde.

Der Endorsement Key ermöglicht es, das TPM eindeutig zu bestimmen. Dies ist allerdings aus Sicht des Datenschutzes kritisch. Um dieses Problem zu umgehen wurden für die Signierung und Verschlüsselung der Attestation Identity Key (AIK) eingeführt. Der Endorsement Key wird deshalb lediglich für das Entschlüsseln des im TPM hinterlegten Owner-Passworts und zur Entschlüsselung und Signatur während der Erzeugung von AIKs eingesetzt.

Storage Root Key (SRK)

Beim Storage Root Key (SRK) handelt es sich, genau wie beim Endorsement Key, um ein RSA-Schlüsselpaar mit einer Länge von 2048 Bit. Der SRK wird während der Einrichtung des Eigentümers auf dem TPM generiert und abgespeichert.

Neben dem EK ist dies auch bei diesem Schlüssel nicht möglich, den privaten Teil des Schlüssels auszulesen.

Der SRK wird für alle weiteren kryptographischen Funktionalitäten genutzt und bilden den Root of Trust for Storage (RTS). Dieser Schlüssel dient für alle weiteren Operationen als Ausgangslage.

Attestation Identity Key (AIK)

Beim Attestation Identity Key (AIK) handelt es sich um ein RSA-Schlüsselpaar mit einer Länge von 2048 Bit. Er wurde eingeführt, um ein datenschutzspezifisches Problem zu lösen. Würde nämlich die Signierung mit dem Endorsement Key vorgenommen, so könnte man bei einer erneuten Durchführung das System eindeutig identifizieren. Dies ist möglich, weil der Key auf dem TPM weder verändert noch gelöscht werden kann). Um dieses Problem zu umgehen werden Nachrichten nicht mit dem Endorsement Key, sondern mit dem AIK durchgeführt. Somit kann für jede Remote Attestation ein neuer temporärer AIK verwendet werden.

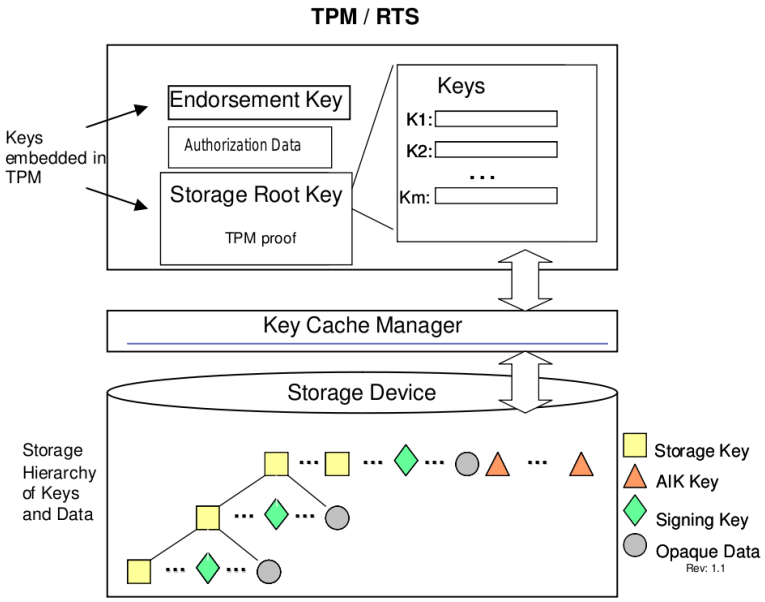

Abbildung 3.2: TPM: Schlüsselhierarchie[8]

3.2.4 Schlüsselhierarchie

Ausser dem Endorsement Key (EK) und dem Storage Root Key (SRK) werden alle durch das TPM verwalteten Keys ausserhalb des TPMs gespeichert. Damit die Schlüssel nicht im Klartext auf externen Medien gespeichert werden, werden sie, wie in Abbildung 3.2 ersichtlich, durch eine Schlüsselhierarchie verschlüsselt. Die Erzeugung des SRK erfolgt während des Take-Ownership-Prozesses. Er wird durch den EK verschlüsselt und mit dem Owner-Passwort geschützt.

In der obersten Ebene der Storage Hierarchy können sämtliche Schlüsseltypen abgelegt werden. Diese werden dann direkt durch den SRK verschlüsselt. Ein Storage Key kann dann wieder sämtliche Schlüsseltypen exkl. der AIK als Kinder besitzen. Das Kind wird dann durch den Elternschlüssel verschlüsselt. Diese vielfältige Möglichkeit kann für die Verwaltung von Schlüsseln mehrerer Benutzer interessant sein.

Um ein in der Hierarchie gespeicherter Schlüssel zu benutzen, muss der private Schlüsselteil, der verschlüsselt auf einem externen Medium abgelegt ist, in das TPM geladen werden. Falls der Schlüssel nicht direkt das Kind des SRK darstellt, müssen noch sämtliche Schlüssel dazwischen geladen werden, damit die vollständige Entschlüsselung erfolgen kann.

3.3 Network Access Control (NAC)

3.3.1 Beschreibung

NAC ist eine Strategie, um die Sicherheit in Computernetzwerken zu erhöhen. Es ermöglicht Unternehmen, Sicherheitsrichtlinien durchzusetzen, um zu verhindern, dass ungewünschte oder unsichere Geräte Zugang zum Netzwerk erhalten. Dazu muss ein Clientgerät, bevor es Zugriff auf das Netzwerk erhält, beweisen, dass es policykonform (z.B. den Virenscanner auf dem aktuellsten Stand) ist. Erst nach einem erfolgreichen Ablauf erhält der Client Zugang in das Unternehmensnetzwerk.

3.3.2 TNC

Die Trusted Computing Group (TCG) hat im Jahr 2006 eine Umsetzung (Trusted Network Connect (TNC)) spezifiziert. Das Ziel ist es, einen offenen Standard für die Clientauthentisierung zu schaffen, das zusätzlich Integritätsmesswerte der Clients übertragen kann. Die TCG definiert dabei die Architektur und die Schnittstellen für die Einführung dieses Konzeptes. In Abbildung 3.3 wird ersichtlich, wie die Kommunikation der einzelnen Knoten abläuft.

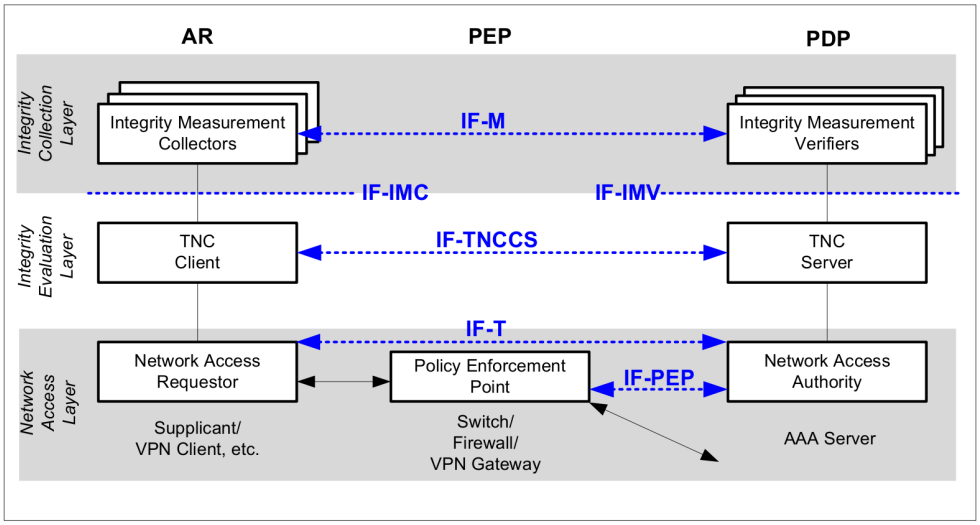

Abbildung 3.3: Architektur-Übersicht: Trusted Network Connect[9]

Das TNC sieht vor, dass es auf bereits verbreiteten Protokollen, wie beispielsweise RADIUS, aufbauen kann. Zusätzlich zu den Authentisierungsinformationen wird ebenfalls der Zustand eines Systems übertragen.

Dabei gibt es vorallem folgende Hauptkomponenten:

Access Requestor (AR)

Der Computer, der Zutritt zum Netzwerk erhalten will, wird um den TNC Client erweitert (siehe Abbildung 3.3). Der TNC Client sammelt Informationen zum momentanen Systemzustand und übermittelt die Resultate an den TNC Server. Um den Zustand zu bestimmen delegiert er diese Aufgabe an ein oder mehrere Integrity Measurement Collector (IMC). Diese können verschiedene, auch unabhängige Prüfungen einleiten. Da sie voneinander unabhängig sind können sie ebenfalls von verschiedenen Herstellern stammen. Eine solche Prüfung könnte z.B. die Version der Virensignaturen, den Zustand des Virenscanners und den Zustand der Firewall umfassen.

Policy Enforcement Point (PEP)

Meist wird als PEP eine aktive Netzwerkkomponente (Router, Switch, WLAN-Accesspoint) eingesetzt. Der PEP besitzt die Aufgabe, die vom Policy Decision Point (PDP) getroffenen Entscheidungen über den Netzwerkzutritt durchzusetzen.

Policy Decision Point (PDP)

Die Bewertung des übermittelten Client-Systemzustandes wird durch den PDP durchgeführt. Der PDP wird zu diesem Zweck durch den TNC Server erweitert, der die Daten des TNC Clients entgegennimmt. Die Auswertung der Werte wird durch die Integrity Measurement Verifier (IMV) durchgeführt, welche das Gegenstück zu den IMC bilden.

3.3.3 NAC-Umsetzungen

Neben der Trusted Computing Group mit dem Trusted Network Connect existieren auch noch Konzepte von anderen Unternehmen. Die folgenden Produkte stellen, gemeinsam mit dem TNC, die bekanntesten Konzepte dar:

- Trusted Network Connect (TNC) von der TCG
- Network Access Protection (NAP) von Microsoft
- Cisco Network Admission Control (Cisco NAC)

Aufgrund der sehr unterschiedlichen Umsetzung wurde für den weiteren Verlauf der Arbeit beschlossen, den Fokus auf ein einziges Produkt, dem

NAP	TNC
NAP Client	TNC Client
Network Policy Server (NPS)	TNC Server
System Health Agent (SHA)	Integrity Measurement Collector (IMC)
System Health Validator (SHV)	Integrity Measurement Verifier (IMV)

Tabelle 3.1: Gegenüberstellung der Begriffe von NAP und TNC

Microsoft NAP, zu legen. Aus diesem Grund wird nachfolgend detaillierter auf die Konzepte und Bezeichnungen von NAP eingegangen.

3.3.4 Microsoft Network Access Protection (NAP)

Network Access Protection (NAP) ist das Netzwerkzugangkontroll-Produkt von Microsoft. Das Produkt wurde mit Windows Server 2008 und Windows Vista eingeführt. Die Architektur des NAP ist sehr ähnlich zu jener des TNC.

NAP ist für vier möglichen Anwendungsfälle geeignet:

- IPsec NAP Enforcement
- 802.1X NAP Enforcement
- VPN NAP Enforcement
- DHCP NAP Enforcement

Die meisten Komponenten sind, verglichen mit dem TNC-Standard, sehr ähnlich. In der folgenden Tabelle sollen diese unterschiedlichen Bezeichnungen gegenüber gestellt werden:

3.4 Testumgebung

Die Testumgebung wurde anhand des Microsoft-Dokumentes *Step-by-Step Guide: Demonstrate NAP IPsec Enforcement in a Test Lab[5]* vorgenommen.

3.4.1 Aufbau Netzwerk

In der Testumgebung, wie in Abbildung 3.4 ersichtlich, wird für die Abbildung eines funktionalen NAP ein Netzwerk mit zwei Windows Servern und zwei Clients gebildet. Für die Zugangskontrolle wird IPsec verwendet.

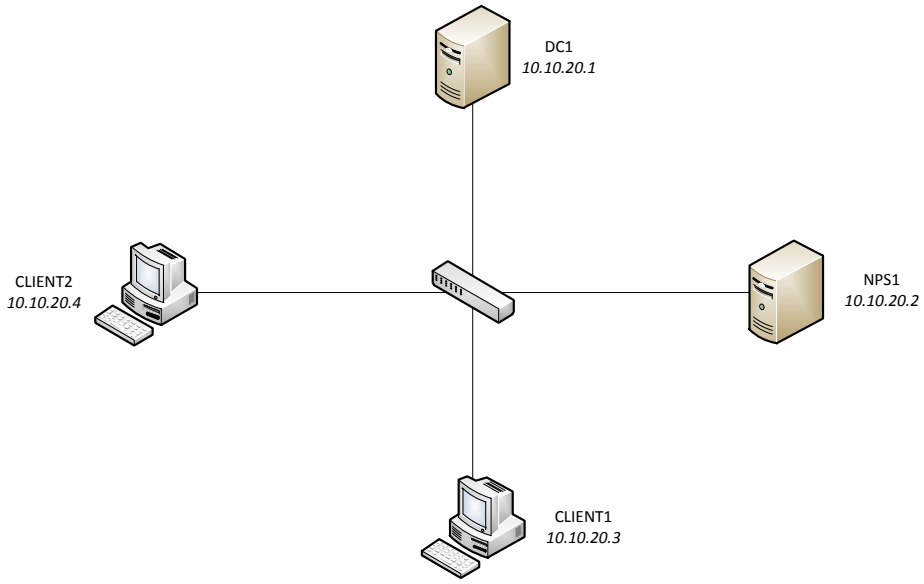

Abbildung 3.4: Netzwerkübersicht Testumgebung

3.4.2 Logische Netzwerke

NAP unterteilt das physikalische Netzwerk in drei logische Netzwerkteile. Diese Teilung wird in der Testumgebung mit Hilfe von IPsec vorgenommen. Ein Clientcomputer wird gemäss seinem Health-Status mit einem dieser Netzwerke verbunden. Mit dieser Teilung wird ermöglicht, dass nicht policykonforme Computer nur eingeschränkte Kommunikationsmöglichkeiten erhalten und somit das Kernnetzwerk nicht erreichen können. Diese Einschränkung wird erst aufgehoben, wenn der Clientcomputer nachweislich seinen Health-Status verbessert hat und schliesslich den Policyrichtlinien des Netzwerkes entspricht.

Clients aus dem eingeschränkten Netzwerk können nicht mit den Rechnern im sicheren Netzwerk kommunizieren (und umgekehrt). Wie in Abbildung 3.5 ersichtlich ist, können Computer nur mit Komponenten aus neben liegenden Schichten kommunizieren.

In den nachfolgenden Unterkapiteln soll auf die drei logischen Netzwerke noch detaillierter eingegangen werden.

Sicheres Netzwerk

Computer, welche die Sicherheitsanforderungen erfüllen, erhalten ein NAP Health Certificate und werden für das sichere Netzwerk zugelassen. Dieses

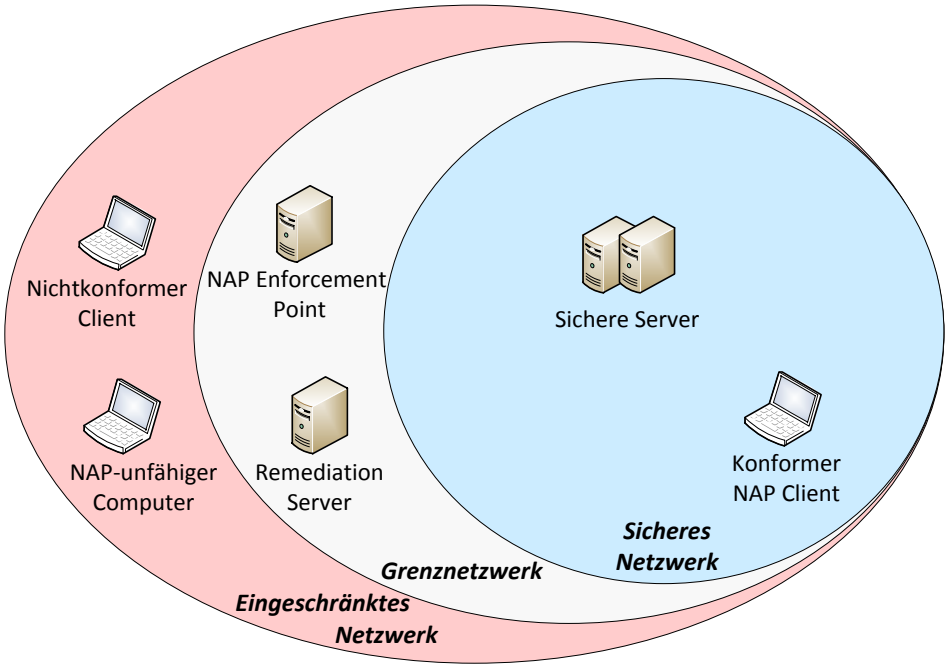

Abbildung 3.5: Übersicht virtuelle Netzwerke des NAP [5]

Netzwerk entspricht dem internen Netzwerk, in welchem Server und Computer, welche ebenfalls über Health Certificates verfügen, eingebunden sind. Die Kommunikation zu den anderen Clients und zum NAP Enforcement Point wird mit Hilfe der Health Certificates über IPsec verschlüsselt.

Computer, die dem sicheren Netzwerk angehören, können eine Verbindung mit Computern aus dem sicheren Netzwerk und dem Grenznetzwerk herstellen.

Grenznetzwerk

Im Grenznetzwerk befinden sich jene Rechner, welche Zugriff auf das sichere sowie das eingeschränkte Netzwerk benötigen. Diese verfügen über ein gültiges Health Certificate.

In diesem Netzwerk werden typischerweise NAP Enforcement Points und Remediation Server positioniert, da sie vom sicheren wie auch vom eingeschränkten Netzwerk Zugriff erhalten sollen. Clients befinden sich nicht in diesem Netzwerk.

Eingeschränktes Netzwerk

Computer, die den Gesundheitsstatus nicht durchführten oder den Netzwerkrichtlinien nicht genügen, werden diesem virtuellen Netzwerk zugeordnet. Gastcomputer (und andere nicht-NAP-kompatible Rechner) wiederfinden sich ebenfalls in diesem virtuellen Netzwerk.

3.4.3 Computer der Testumgebung

Folgend werden die Computer aufgelistet, welche in der Testumgebung verwendet werden:

DC1

Betriebssystem:	Windows Server 2008 R2 Enterprise Edition
Funktionen:	Domain Controller
	DNS Server
	Zertifizierungsstelle

NPS1

Betriebssystem:	Windows Server 2008 R2 Enterprise Edition
Funktionen:	NAP Health Policy Server
	Health Registration Authority
	Subzertifizierungsstelle (Sub-CA)
	RADIUS-Server
	NAP

CLIENT1

Betriebssystem:	Windows 7 Professional
Funktionen:	NAP Client

CLIENT2

Betriebssystem:	Windows 7 Professional
Funktionen:	NAP Client

3.4.4 Komponenten von NAP (IPsec-Enforcement)

Der Zugriffsschutz für das Netzwerk besteht aus verschiedenen Client- und Serverkomponenten. Diese sollen hier genauer erläutert werden.

Clientkomponenten

System Health Agent (SHA) Der System Health Agent (SHA) führt den Gesundheitscheck auf dem Client aus und teilt dessen Resultate per Statement of Health (SoH) dem NAP-Agent mit. Jeder SHA weiss, wo er welche Informationen zur Prüfung der Gesundheit erheben muss. Auf einem System können mehrere SHAs eingebunden sein. Auf Serverseite wird der Status dann durch das entsprechende pendant, dem SHV, analysiert.

NAP Agent Der NAP Agent stellt ein Dienst dar, der die clientseitigen Integritätsinformationen der SHA sammelt, verwaltet und als SSoH-Nachricht an den NAP-Server sendet.

Enforcement Client (EC) Der Enforcement Client ist zuständig für das Anfordern des Zugriffs auf das Netzwerk, das Mitteilen des Integritätsstatus des Clientcomputers an den NAP-Server und das Kommunizieren des Einschränkungsstatus des Clients an weitere clientseitige NAP-Komponenten. Enforcement Clients sind je nach NAP-Anwendungszweck unterschiedlich: so gibt es einen EC für IPsec, VPN, DHCP etc.

Statement of Health (SoH) Das Statement of Health (SoH) ist das Protokoll, welches zwischen den SHAs/SHVs eingesetzt wird, um den Gesundheitsstatus oder das Resultat des Gesundheitschecks eines einzelnen Agents zu übermitteln.

System Statement of Health (SSoH) Das SSoH ist ebenfalls ein Protokoll. In diesem werden SoH-Nachrichten der SHA/SHV zusammengefasst, sodass alle einzelnen Werte der SHAs/SHVs gemeinsam übertragen werden. SSoH dient für die Kommunikation zwischen dem NAP Agent und dem NPS Service.

Serverkomponenten

System Health Validator (SHV) Die System Health Validators stellen das serverseitige Gegenstück zu den SHA dar. Für jeden SHA des Clients existiert ein entsprechender SHV auf dem NAP-Server. Der NPS wertet mithilfe des SHV die gesammelten Client-Informationen des SHA aus und meldet dem NPS Service das erhaltene Resultat (Compliant oder Non-Compliant mit dem entsprechenden Fehlercode und der Fehlermeldung).

NAP Administration Server Ist zuständig für die richtige Zuordnung der einzelnen SoH an die SHV. Er kommuniziert über SSoH mit dem NPS-Service.

NPS Service Der NPS Service empfängt die Zugangsanfragen der Clients und leitet die SSoH an den NAP Administration Server weiter.

NAP Enforcement Server (ES) Der NAP Enforcement Server leitet die Anfragen und den Status des Clients auf die höheren Schichten des Network Health Policy Servers weiter. Zudem ist er für die Durchsetzung der Netzwerklimitierung der Clients verantwortlich. Er teilt den Client je nach Gesundheitsstatus in das richtige Teilnetzwerk zu.

Remediation Server Auf dem Remediation Server werden Updates bereitgestellt, damit ein Computer, der nicht den NAP-Zugangsrichtlinien entspricht, seinen Gesundheitsstatus aktualisieren kann, um schliesslich Zugang zum Netzwerk zu erhalten.

Integrity Rules Diese Regeln werden durch die Konfiguration einzelner SHVs erstellt. Dazu können Richtlinienbedingungen (Policies) konfiguriert werden, welche dann durch den NPS erzwungen werden.

Health Registration Authority (HRA) Falls ein Client die Richtlinien erfüllt, erwirbt die HRA im Auftrag des Clients ein Health Certificate, damit der Client Zugang in das geschützte IPsec-Netzwerk erhält.

Statement of Health Response (SoHR) Dies stellt die Antwort des NPS auf ein SoH dar. Das SoHR enthält die Mitteilung über das Überprüfungsresultat des Client-Gesundheitszustandes.

Falls der Client die Richtlinien erfüllt, so erhält das Statement of Health Response (SoHR) die Zulassung zum Netzwerk. Ansonsten enthält es Informationen über die nichterfüllten Richtlinien und bei welchen Remediation Servern der Client die notwenigen Updateinstruktionen erhält.

System Statement of Health Response (SSoHR) Für den Transport werden alle SoHR-Mitteilungen in einem einzigen SSoHR zusammengefasst.

Kapitel 4

Analyse des NAP-Protokolls

In einer ersten Phase dieser Semesterarbeit wurde der Network Access Protection (NAP)-Kommunikationsprozess genauer unter die Lupe genommen. Es soll aufzeigen, wie der Ablauf funktioniert, welche Protokolle eingesetzt werden und inwiefern der Standard der Trusted Computing Group (TCG) umgesetzt wurde.

4.1 Ablauf NAP Verbindungsprozess (mit IPsec)

1. Damit ein Clientcomputer Zugriff auf das sichere Netzwerk erhält, muss er dem Network Policy Server (NPS) seinen Gesundheitszustand bekanntgeben. Dazu sendet der Enforcement Client seinen Healthstatus (als System Statement of Health (SSoH)) an die Health Registration Authority (HRA).
2. Die HRA sendet die SSoH-Daten an den NAP Health Policy Server (NPS).
3. Der Network Policy Server (NPS) wertet das SSoH-Pakets des NAP-Clients aus und teilt dem HRA mittels System Statement of Health Response (SSoHR) mit, ob der Client die Policybedingungen erfüllt. Erfüllt er sie nicht, so sendet der HRA Informationen für die Remediation mit, damit der Client seinen Gesundheitsstatus verbessern kann.
4. In diesem Schritt müssen zwei Fälle unterschieden werden.
   (a) *Der Gesundheitszustand entspricht den definierten Netzwerk-Richtlinien:* Es kann direkt zu Schritt 9 gesprungen werden.

   (b) *Der Gesundheitszustand genügt den definierten Netzwerk-Richtlinien nicht:*

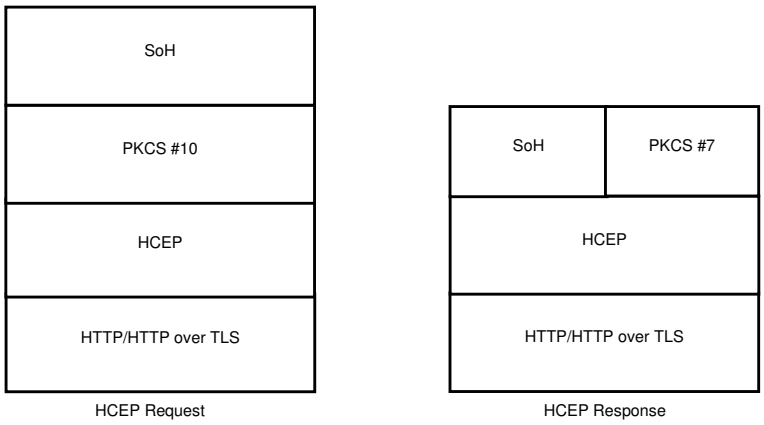

Abbildung 4.1: Protokoll Stack [2]

Die HRA sendet ein SSoHR zum NAP Client. Ein Health Certificate wird nicht ausgeliefert. Es können deshalb keine Verbindungen zum sicheren Netzwerk durchgeführt werden, da dem Client das Health-Certificate für die IPsec-Verschlüsselung fehlt. Er kann hingegen eine Verbindung zum Remediation Server herstellen, von welchem er die fehlenden Updates erhält.

5. Der NAP-Client sendet eine Updateanfrage an den Remediation Server.

6. Der Remediation Server liefert dem NAP-Client die nötigen Updates.

7. Der Clientrechner installiert die erhaltenen Updates.

8. Nachdem der NAP Client seinen Status aktualisiert hat, sendet er ein neues SSoH zum HRA. Nun wird mit Schritt 3 fortgefahren.

9. Die HRA übermittelt dem NAP Client ein gültiges Health Certificate. Mit diesem Health Certificate kann der NAP Client nun IPsec-verschlüsselt mit anderen Computern im sicheren Netzwerk kommunizieren. Der Client befindet sich nun logisch gesehen im sicheren Netzwerk.

4.2 Übertragungsprotokoll

Die Erkenntnisse dieses Kapitels beziehen sich auf den Wireshark-Dump, der auf der CD gefunden werden kann.

4.2.1 Protokoll Stack

Für die Übertragung der NAP Nachrichten wird der Protokoll-Stack verwendet (siehe Abbildung 4.1). Dieser konnte durch Analysieren des Netzwerktraffics nachgewiesen werden. Microsoft setzt diese Nachrichten ein, um den NAP-Status zu überprüfen, bevor eine IPsec Verbindung aufgebaut wird. Es werden aber nicht für alle Nachrichten alle Schichten des Stacks benutzt. Teilweise werden auch lediglich die unteren Layer eingesetzt.

4.2.2 Ablauf im Testszenario

Das Testszenario umfasste eine NAP Verifikation zwischen CLIENT1 und NPS1. Alle dafür verwendeten Nachrichten besassen denselben Grundaufbau. Sie verwenden als Basis immer MS-HCEP über HTTP (optional mit TLSv1). Der weitere Aufbau kann je nach Aufgabe (wie im nächsten Abschnitt gezeigt wird) etwas von Abbildung 4.2 abweichen. Der konkrete Ablauf wird folgend beschrieben:

1. CLIENT1 sendet einen Request an den Server (NPS1). Dieser besteht aus einer PKCS#10 Anfrage, welche die gewünschten Zertifikate anfordert.
2. Der NPS1 antwortet mit einem 401 (Unauthorized), da die Anfrage keine SoH-Informationen enthält. In der Antwort weist der NPS1 den CLIENT1 darauf hin, dass ihm die benötigten Berechtigungshinweise fehlen (*Access is denied due to invalid credentials.*). In diesem Schritt wird lediglich HTTP (über TLSv1) eingesetzt. Zudem setzt Microsoft hier kein MS-HCEP ein.
3. Nun sendet der CLIENT1 eine erneute Anfrage, welche im Paket den vollständigen SoH-Teil enthält, womit der jetzige Request der Abbildung 4.1 entspricht.
4. Falls die SoH-Anfrage gültig und der Client policykonform ist, sendet der NPS1 über PKCS#7 das benötigte Zertifikat an CLIENT1.

Die ersten zwei Schritte dieses Datenaustausches widersprechen sich mit der Definition des MS-HCEP. Gemäss Protokolldefinition von Microsoft muss ein Request zwingend eine SoH-Nachricht enthalten. Diese dürfte nicht fehlen. Auch kann für den darauffolgenden 401 HTTP-Error keinerlei Dokumentationen gefunden werden. Im MS-HCEP ist definiert, dass bei fehlerhaftem oder nicht konformem Request (z.B. kein SoH Teil enthaltend) ein HTTP-Internal Server Error (HTTP-Error 500) zurückgegeben werden muss. Der zweite Request des CLIENT1 erfolgte hingegen gemäss der von Microsoft bereitgestellten Dokumentation. Ebenfalls entspricht auch die Response dem Standard von Microsoft.

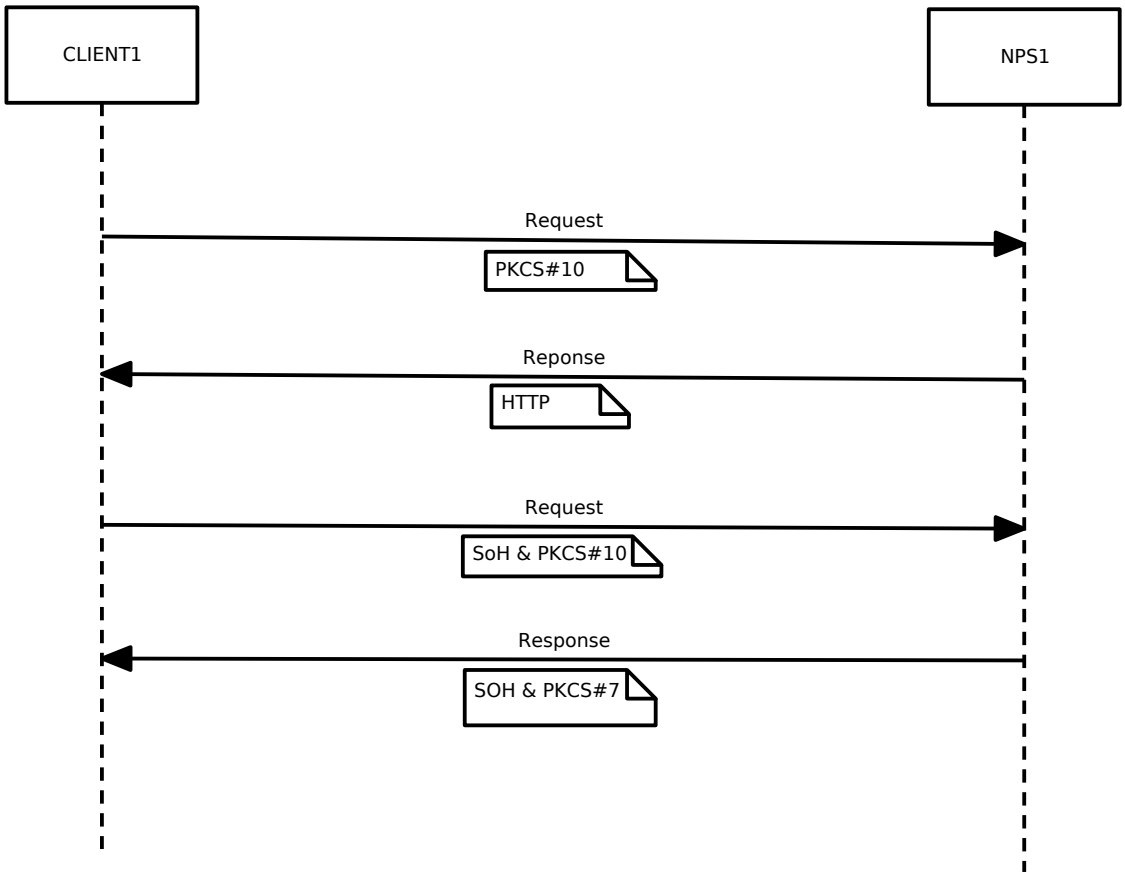

Abbildung 4.2: Ablaufdiagramm NAP

4.2.3 Protokolle

MS-HCEP

Beim Microsoft Health Certificate Enrollment Protocol (MS-HCEP) handelt es sich um das Protokoll, in welchem der SoH oder ein Zertifikat enthalten ist. Es kümmert sich nicht selbst um die Sicherheit der Daten und überlässt dies den anderen Layern. Der Header ist als HTTP Request oder Response aufgebaut und muss einigen Richtlinien Rechnung tragen. Im wesentlichen wird durch dieses Protokoll die Nachricht als Teil von NAP erkannt. Details zum Aufbau und allgemein zum Microsoft Health Certificate Enrollment Protocol können im Kapitel 3 des Dokumentes [MS-HCEP][2] entnommen werden.

MS-SoH

Im Microsoft Statement of Health for Network Access Protection (MS-SoH) sind die Nachrichten der verschiedenen SHAs zusammengefasst. Ausserdem werden die Mitteilungen der SHV im MS-SoH übertragen. Im Protokoll ist genau definiert, wo und wie die Nachrichteninformationen übertragen werden. Details zum Microsoft Statement of Health for Network Access Protection können im Kapitel 3 des Dokumentes [MS-SoH][3] entnommen werden.

Kapitel 5

Umsetzungskonzept

Dieses Kapitel stellt die Entscheide dar, welche im Verlaufe des Projektes aufgrund der erarbeiteten Grundlagen getroffen wurden. Zudem wird hier der Prototyp und seine Funktionalitäten genauer beschrieben.

5.1 Wahl der Programmiersprache

5.1.1 NAP - Agent und Server

Für die Umsetzung wurde zu Beginn abgeklärt, in welchen Programmiersprachen zusätzliche Funktionalitäten für das NAP-System entwickelt werden können. Microsoft bietet in ihrem neuesten Software Development Kit (SDK) (Version 7.1a) eine NAP-Beispielimplementation in C++ an. Zusätzlich zu C++ wurden ebenfalls andere Möglichkeiten wie etwa C# gesucht in Betracht gezogen. Microsoft stellt jedoch lediglich Informationen zu C++ zur Verfügung; andere Programmiersprachen werden nicht beschrieben. Da der Beispielcode ein sinnvoller Grundbaustein darstellt und dadurch die Einarbeitungs- und Programmierzeit zusätzlich gesenkt werden konnte, wurde beschlossen, die Entwicklung für die NAP-Funktionalität in C++ zu beginnen.

5.1.2 Zugriff auf das TPM

Für die Kommunikation mit dem TPM werden von Microsoft zwei Libraries für C++ angeboten. Bei der Recherche wurden ebenfalls Libraries für Java genauer unter die Lupe genommen. Schliesslich wurde entschieden, dass die Entwicklung im TPM-Bereich ebenfalls in C++ stattfindet, da bereits

die NAP-Einbindung in C++ geschrieben wurde und somit die Interaktion zwischen beiden Programmteilen erheblich vereinfacht wird.

5.2 TPM-Library

Für die Programmierung stehen, wie bereits erwähnt, zwei C++-Libraries für Windows zur Verfügung. Namentlich sind dies die Win32_Tpm-Klasse und die TPM Base Services (TBS). Es wurde abgewogen, welche sich für die Zwecke dieser Arbeit besser eignen. Die Win32_TPM-Library überzeugte zu Beginn durch die grosse Funktionsvielfalt. Die Kommunikation mit dem TPM konnte bereits durch einfache Aufrufe realisiert werden. Während die Verwendung der Library Probleme verursachte, wurde ebenfalls festgestellt, dass einige benötigte Funktionalitäten über *Win32_TPM* nicht vorhanden und auch nicht umgesetzt werden können. Aufgrund dieser Erkenntnis wurden dann schliesslich die TPM Base Services gewählt. Die verfügbare Funktionsvielfalt der Library ist minimal und deshalb musste die TPM-Kommunikation gemäss Standard selber nachträglich implementiert werden. Aus diesem Grund wurde dann durch das Team ein TBS-Wrapper implementiert. Auf diesen wird in Kapitel 6.2.4 detaillierter eingegangen.

5.3 Prototyp-Funktionalitäten

Der Prototyp soll die Zusammenarbeit zwischen dem NAP und dem TPM aufzeigen. Für die Umsetzung des Prototyps wird als Basis der NAP-Beispielcode aus dem SDK von Microsoft verwendet.

Im NAP-Bereich wurde ein eigener SHA/SHV umgesetzt. Der TPM-Teil wurde in einem eigenen Projekt implementiert und nur testweise mit dem NAP kombiniert. Im TPM-Prototyp ist es möglich, erste grundlegende Funktionalitäten des TPMs auszuführen.

Vertiefte Informationen zur Prototypumsetzung und -funktionalität ist in Kapitel 6.1.1 zu finden.

Kapitel 6

Implementation

6.1 Network Access Protection

Im Rahmen der Semesterarbeit wurde ein bestehender SHA/SHV-Code analysiert und angepasst. Der System Health Validator (SHV) wird auf dem Server als Windows-dll-Datei eingebunden, der *Shv.dll*. Bei Clientanfragen erstellt er für jede Anfrage einen einzelnen Thread, sodass mehrere gleichzeitige Client-Anfragen (von NAP-Agents) beantwortet werden können.

Der System Health Agent (SHA) besteht aus zwei Komponenten. Einerseits einer dll, der *ShaInfo.dll*. Andererseits der ausführbaren *Sha.exe*, worin der eigentliche SHA implementiert ist. Im produktiven Betrieb wird diese vorzugsweise als Windows-Service realisiert.

6.1.1 Konzeptionelles

Der SHA initialisiert sich beim Starten mit der Component Object Model (COM)-Schnittstelle von Windows. Anschliessend muss er sich beim NAP-Agent über die SHA-API (s. Abbildung 6.1) registrieren, damit dieser über den neuen Health Agent in Kenntnis gesetzt wird.

Der SHA implementiert das Interface *INapSystemHealthAgentCallback*. Dieses stellt die Schnittstelle zum NAP-Agent über die SHA-API dar. Ab dem Zeitpunkt der Registrierung läuft der SHA im Hintergrund und kann lediglich von der darunterliegenden Schicht über dieses Interface aufgerufen. Konkret bedeutet dies: wenn der Server den System Statement of Health, also den Systemstatus des Clients, erfragt, ruft der NAP-Agent die Methode über dieses Interface auf, welche die dafür nötigen Prüfungen zur Ermittlung des Systemzustands einleitet. Anschliessend wird das Resultat beziehungsweise

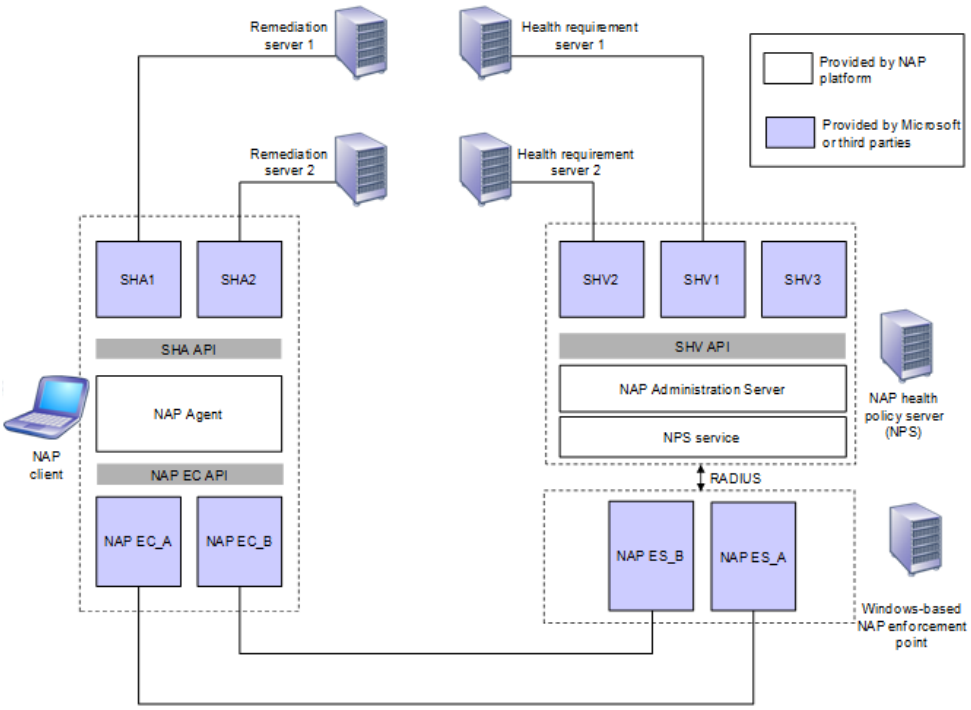

Abbildung 6.1: Architektur der NAP-Plattform auf Client- und Serverseite[4]

die zu übermittelnde Nachricht wieder zurückgesendet, um dann von den weiteren Schichten weitergereicht zu werden.

Grundsätzlich erfolgen jegliche Aufrufe an den SHA/SHV immer aus den darunterliegenden Schichten. Beim NAP-Client werden Prüfungsabfragen durch den NAP-Agent gehandhabt, welcher anschliessend von allen registrierten SHAs den Status anfordert. Es ist auf Client-Seite nicht möglich, lediglich einzelne System Health Agents zu überprüfen, da die Mitteilung an den Server (SSoH) immer alle Zustände beinhalten muss.

Der SHV implementiert seine Funktionalität über das Interface *INAPSystemHealthValidator*. Dieses Interface stellt die Schnittstelle zur darunterliegenden Schicht dar und bietet der untenliegenden Schicht die Möglichkeit, alle Funktionen der obenliegenden Schicht zu nutzen. Dies erfolgt analog wie beim Interface des SHA.

Für die Übertragung zwischen dem SHA und dem SHV können beliebige Nachrichten verwendet werden. Die zu übertragende Nachricht muss als BYTE-Array abgespeichert und ebenfalls als BYTE-Werte übertragen werden (als Struct: *VendorSpecificData*).

6.1.2 Umsetzung

Der SHA/SHV-Teil dieser Arbeit basiert grösstenteils auf dem Beispielcode von Microsoft[6]. Angepasst wurde primär nur der SHA und einige Details, wie etwa der Modulanzeigename sowie die Zuordnung zur Schule (HSR).

Der SHA wurde einerseits bei der Prüfung des SSoH modifiziert, andererseits wurde das Registrieren und Starten überarbeitet. Die Prüfung des Gesundheitsstatus wird mit der Funktion *FillSoHRequest* angestossen. Momentan wird für die Prüfung die Datei *nap.txt* benötigt, welche sich im selben Verzeichnis wie der SHA befindet muss, verwendet. Wenn zu Beginn der Datei der ASCII-Charakter 0 steht, stellt dies ein nicht konformer Gesundheitsstatus dar. Zudem ist dies ebenfalls der Ort, wo die Einbindung des TPMs erfolgt. Die Klasse *SHA* wurde überarbeitet und stellt nun die Funktionen *onStart* und *onStop* zur Verfügung. Diese werden im Zusammenhang mit den Windows-Services benötigt. Für das Registrieren des SHA wird noch die Hilfsklasse *CShaModule* verwendet. Diese stellt die benötigte Funktionalität und Aufrufe für das Registrieren und Entfernen bereit.

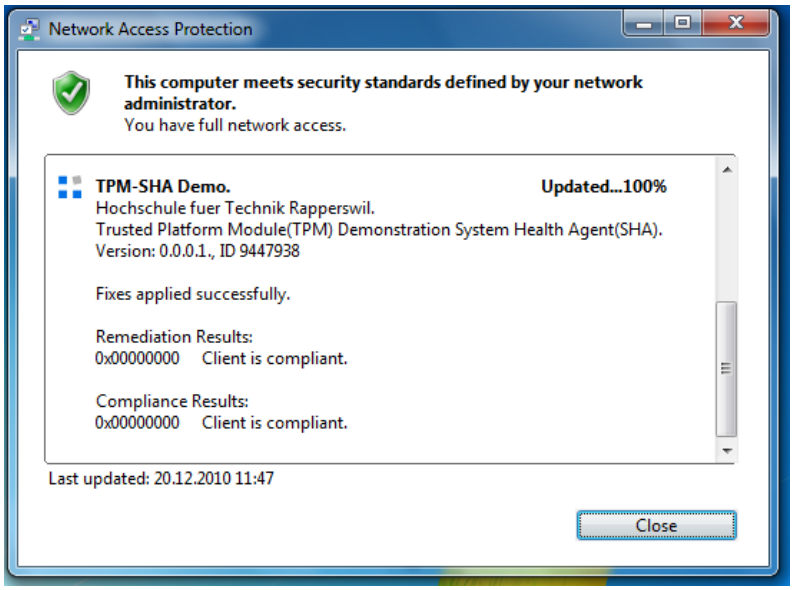

Abbildung 6.2: Ansicht des Prototyps auf Clientseite

6.2 Trusted Platform Module

Die verwendete Library TPM Base Services bietet nur eine sehr geringe Abstraktion zum TPM. Er nimmt primär Arbeit beim An- und Abmelden zum TPM ab. Beim Ausführen der Anweisungen ans TPM bietet er nur eine

geringe Vereinfachung. Aus diesem Grund wurde in der Umsetzung ein zusätzlicher Wrapper erstellt, um das Handling mit dem TPM zu vereinfachen. Auch hinsichtlich der Integration des TPMs in das NAP wird dies die weitere Arbeit vereinfachen. Zum jetzigen Zeitpunkt ist dieser Wrapper mit einer Test-Klasse zur Demonstration versehen und wird noch nicht aus dem NAP direkt aufgerufen. Es ist jedoch geplant, dass die Wrapperklasse in den nächsten Versionen mit dem SHA/SHV verbunden wird.

6.2.1 Konzeptionelles

Anweisungen an das TPM über den TBS werden als *BYTE array (unsigned char)* übertragen. Dabei wird keine Abstraktion zum Standard[10] vorgenommen. Die Befehle beinhalten jeweils minimal 10 Byte lange Arrays, welche die Art des Kommandos, gefolgt von der Länge der gesamten Anweisung sowie einer Kennungsnummer der Anweisung selbst enthält. Anschliessend folgen je nach Anweisung zusätzlich notwendige Informationen zum Ausführen der Anweisung. Dies können Informationen zur Identifikation am TPM, Informationen zum benötigten Schlüssel oder ähnliches sein.

Als Antwort erhält man immer ebenfalls ein BYTE Array in der minimalen Länge von 10 Bytes. In der minimalen Form sind dabei wieder die Art des Kommandos, die gesamte Länge sowie die Kennung des ausgeführten Befehls enthalten. Der Status ist im erfolgreichen Fall *TPM_SUCCESS*, im Fehlerfall entsprechend ungleich *TPM_SUCCESS* (Die Fehlercodes sind in *tpm_error.h* definiert). Auch der TBS gibt einen Fehlercode (*TBS_RESULT*) nach dem Absenden des Befehls zurück. Dieser gibt darüber Auskunft, ob der Befehl (das BYTE Array) erfolgreich an das TPM weitergegeben wurde, oder ob die Übermittlung fehlerhaft war.

6.2.2 Beispiel einer TPM Anweisung

Als konkretes Beispiel wird hier auf die Funktion TPM_GetRandom vertieft eingegangen. In dieser Funktion werden Zufallszahlen auf dem TPM generiert. In der Abbildung 6.3 ist die Funktionsdefinition der Anweisung ersichtlich.

Die ersten drei Input-Parameter (*tag, paramSize, ordinal*) sind in jeder Anweisung an das TPM enthalten. *tag* beschreibt die Art der Anweisung als UINT16 (unsigned 16-Bit Integer) Wert des Structs *TPM_TAG*. Der nächste Parameter, *paramSize* enthält die Länge der gesamten Anweisung als UINT32 (unsigned 32-Bit Integer). Der Letzte der drei Grundwerte einer TPM-Anweisung (*ordinal*) ist ein *TPM_COMMAND_CODE*, der einem UINT32 entspricht und die Befehlkennung als Nummer repräsentiert.

Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_GetRandom.
4	4	2S	4	UINT32	bytesRequested	Number of bytes to return

Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_GetRandom.
4	4	3S	4	UINT32	randomBytesSize	The number of bytes returned
5	<>	4S	<>	BYTE[ ]	randomBytes	The returned bytes

Abbildung 6.3: TPM_GetRandom[7]

Feld	Typ	Wert
tag	TPM_TAG (UINT16)	TPM_TAG_RQU_COMMAND (0x00c1)
parmsize	UINT32	14 (0x000000e)
ordinal	TPM_COMMAND_CODE (UINT32)	TPM_ORD_GetRandom (0x00000046)
bytesRequested	UINT32	4 (0x00000004)

Tabelle 6.1: TPMGetRandom

Die Anweisung TPM_GetRandom besitzt zusätzlich noch einen vierten Parameter: *bytesRequested*, bei welchem beim Funktionsaufruf vier Bytes mitgegeben werden müssen. Damit wird angegeben, wieviele Bytes an Zufallsdaten berechnet werden sollen. In Tabelle 6.1 ist die Übersicht des gesamten Aufrufs von TPM_GetRandom ersichtlich.

Konkret sieht das 14-Byte lange Array nun folgendermassen aus: 0x00, 0xc1, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x00, 0x00, 0x46, 0x00, 0x00, 0x00, 0x04.

Damit das Abfüllen dieser Werte in das BYTE-Array etwas komfortabler vonstatten geht, bietet die implementierte TPM-Wrapperklasse einige Hilfsfunktionen an. Die Funktion convert wandelt den Input (einen 16-Bit Unsigned Integer oder 32-Bit Unsigned Integer) um, so dass der Wert direkt der Startposition im Array zugeordnet werden kann. Als Beispiel wird die *tag*-Wertkonvertierung gezeigt:
$(UINT16\&)array[0] = convert(TPM\_TAG\_ORD\_COMMAND)$ erzeugt im *array* die Einträge 0x00, 0xc1 auf den ersten beiden Positionen (in *array[0]*).

Dieses Array wird nun über die TBS-Schnittstelle an das TPM übermittelt. Dabei kann ebenfalls die Ausführungspriorität (low, normal, high) der Anweisung für das System festgelegt werden. Ebenfalls muss ein BYTE-Array mitgegeben werden, in welchem das TPM die Rückgabewerte speichern kann. Dieses Array muss im Minimum die Grösse der erwarteten Rückgabewerte aufweisen. Im Fall von GetRandom wären das 18 Bytes (14 + 4 Bytes die erwarteten Random-Bytes - siehe Abbildung 6.3: *Outgoing Operands and Sizes*).

In Abbildung 6.3 sind die Rückgabewerte ersichtlich. Diese werden in das mitgegebene Array gespeichert. Bei den ersten drei Werten (*tag, paramSize, returnCode*) handelt es sich um Werte, die jede TPM-Anweisung zurückgibt. Dabei sollte zuerst überprüft werden, ob *paramSize* der erwarteten Grösse entspricht, anschliessend ob der *returnCode* dem TPM_SUCCESS entspricht. Falls einer der beiden Werte nicht den erwarteten Inhalt aufweist, muss eine Fehlerbehandlung durchgeführt werden. Im Fall des hier vorgestellten Wrappers wird der *returnCode* weitergereicht; der Rest wird verworfen.

Im Anschluss an diese Werte folgen nun spezifische Rückgabewerte der Anweisung. Im Fall von TPM_GetRandom handelt es sich um die Länge der angeforderten Random-Bytes, sowie die vier Random-Bytes selbst.

6.2.3 TPM-Übergabe-Parameter

In der Liste der Attribute, welche das TPM für die Anweisungen benötigt, wurden im Verlauf der Arbeit einige weniger geläufige Werte festgestellt. Eine Liste jener, welche bis zum jetzigen Zeitpunkt durch das Team analysiert wurden, sind in der Tabelle 6.2 aufgeführt.

Bei allen Funktionen (beispielsweise in Abbildung 6.3 ersichtlich) werden in der zweiten grossen Spalte (Spaltenname: HMAC) die Parameter für HMAC Aufrufe definiert. Dabei werden die Werte angegeben, welche für die entsprechenden Berechnungen (Tabelle 6.3) benötigt werden, dargestellt.

6.2.4 TPM-Wrapper

Das Ziel des TPM-Wrappers ist es, dem Programmierer den Umgang mit dem TPM zu erleichtern. In der jetzigen Form dient er nur zum Evaluieren der Technologie und stellt einen Ansatz zur Umsetzung des gesamten Wrappers, welcher in der Folgearbeit umgesetzt und vom NAP eingesetzt wird, dar. Momentan beherrscht er lediglich eine kleine Auswahl an Anweisungen für das TPM, diverse Hilfsfunktionen (um Werte für die Verwendung mit dem TPM umzuwandeln) sowie Hilfsfunktionen im Crypto-Bereich.

Parameter	Bedeutung
nonceOdd	Zufalls-Bytes vom Benutzer
nonceEven	Zufalls-Bytes vom TPM
sharedSecret	$HMAC(parentpw, nonceEvenOSAP \| \| nonceOddOSAP)$
inAuth	$HMAC(sharedSecret, inParamDigest \| \| inAuthSetupParams)$
pubAuth	$HMAC(sharedSecret, SHA1(inParamDigest) \| \| inAuthSetupParams)$
dataUsageAuth	$SHA1(sharedSecret \| \| nonceEven)$ *xor password*
dataMigrationAuth	$SHA1(sharedSecret \| \| nonceOdd)$ *xor password*

Tabelle 6.2: TPM-Werte

Parameter	Zusammensetzung
inParamDigest	$SHA1(1S \| \| 2S \| \| 3S..nS)$ des Inputs
outParamDigest	$SHA1(1S \| \| 2S \| \| 3S..nS)$ des Outputs
inAuthSetupParams	$2H1 \| \| 3H1 \| \| 4H1..nH1$ des Inputs
outAuthSetupParams	$2H1 \| \| 3H1 \| \| 4H1..nH1$ des Outputs

Tabelle 6.3: Parameter für den HMAC aufruf

Die verwendeten TPM-Header *tpm.h*, *tpm_ordinal.h*, *tpm_error.h* und *platform.h* wurden aus dem TCG Software Stack (TSS)[12] übernommen. Es handelt sich in diesen Headern primär um die Definition von Konstanten und Structs, welche der Vereinfachung der TPM-Programmierung dienen.

Im Wesentlichen stellt der Wrapper die BYTE Arrays für das TPM zusammen, übermittelt diese und zerlegt die Antwort später in einfacher nutzbare Structs oder kleinere BYTE Arrays. Ebenfalls soll er die Übermittlung von Passwörtern vorbereiten, da das TPM diese gemäss Standard und aus Sicherheitsgründen im Klartext nicht akzeptiert. Dieser Teil ist jedoch momentan unvollständig und somit nicht funktionsfähig.

SHA-1

Das TPM setzt als Hash-Funktion primär auf SHA-1. Die SHA-1-Implementation wurde nicht selbst vorgenommen, sondern stammt von Dominik Reichl[11]. Es ist jedoch geplant, diese Funktion in der Folgearbeit selbst zu implementieren.

HMAC

Das TPM setzt den HMAC Algorithmus in Kombination mit SHA-1 ein. Der Algorithmus kommt für die Identifizierung des Nutzers am TPM zum Einsatz.

Im Projekt wird die HMAC Implementation von Jim Chung[1] eingesetzt. Es ist jedoch hier ebenfalls geplant, den Algorithmus in der Folgearbeit selbst zu implementieren.

RSAES-OAEP (PKCS#1 v2.0)

Die *RSA Encryption Scheme - Optimal Asymmetric Encryption Padding (RSAES-OAEP)* Funktion wurde im Wrapper ansatzweise implementiert. Jedoch ist diese aus Zeitgründen noch nicht vollständig einsatzfähig. Das TPM setzt diese Funktion ein, um Passwörter für neue Schlüssel sicher an das TPM zu übertragen.

6.3 Installation

In diesem Kapitel wird darauf eingegangen, wie der implementierte Prototyp in einer bestehenden NAP-Umgebung eingebunden wird.

6.3.1 NAP

Um den SHA und SHV auszuführen, ist eine NAP Umgebung erforderlich. In dieser Arbeit wurde dazu der Step-by-Step Guide: Demonstrate NAP IPsec Enforcement in a Test Lab[5] verwendet. Es ist nötig, den Guide bis *Verifying NAP functionality (Seite 41)* zu bearbeiten. Falls die gesamte IPsec-Umgebung aufgebaut werden soll, so muss der gesamte Guide befolgt werden.

Nachdem das NAP Demonstrationsprogramm kompiliert wurde, sind noch die folgenden Schritte abzuarbeiten:

Hinweis: Solange nichts anderes angegeben ist, benötigen alle Anweisungen im Command-Window (cmd) Administratoren-Rechte.

1. Auf dem NPS1 muss die dll-Datei Shv.dll in das Verzeichnis *%systemroot%/system32/* kopiert werden und im cmd-Window mit der Anweisung *regsvr32 Shv.dll* im selbigen Verzeichnis registriert werden. Anmerkung, wenn das Programm in 32bit kompiliert wurde: die Datei muss im Ordner *%systemroot%/SysWOW64* abgespeichert werden, da

Windows Server 2008 R2 ein 64bit Betriebssystem ist. Sonst kann die Registrierung der dll-Datei nicht fehlerfrei durchgeführt werden.

2. Im *Network Policy Server* muss der Validator in den gewünschten Policies hinzugefügt werden. Im Fall der Testumgebung wären dies die beiden definierten Policies *NAP IPsec with HRA Compliant* und *NAP IPsec with HRA Noncompliant*.
3. Nach der Kompilierung kann es vorkommen, dass die *message.h*-Datei nicht gefunden wird. Um das Problem zu beheben, muss das Projekt nochmals kompiliert werden.
4. Auf Clientseite muss die Datei *ShaInfo.dll* ebenfalls mit *regsvr32* registriert werden. Dabei kann der Ablageordner der Clientdateien frei gewählt werden (z.B. C:/NAPdemo).
5. Im selben Verzeichnis muss nun eine Text-Datei *nap.txt* angelegt werden. Diese beinhaltet den Gesundheitsstatus des Clients für den Prototyp. Zusätzlich muss in der Datei der Health Status gesetzt werden: Das Zeichen 0 für Unhealthy, 1 für Healthy.
6. Nun kann die Datei Sha.exe aus dem cmd-Fenster aufgerufen und anschliessend mit der Eingabe von *1* gefolgt mit *Enter* gestartet werden.

Der Gesundheitsstatus kann nun im GUI über den Befehl *napstat* (Start / Ausführen) betrachtet werden. Detailliertere Informationen können in der cmd über den Command *netsh nap client show stat* abgefragt werden.

Es ist wichtig, dass der SHA mit der Eingabe von *2* gefolgt mit *Enter* beendet wird, damit er vom System korrekt abgemeldet wird. Dies kann einfach in das Fenster eingegeben werden, wobei die Debugausgaben zu ignorieren sind.

Um die Programme vollständig vom System zu entfernen sind nach dem Beenden des SHA folgende Schritte notwendig:

1. Beim NPS muss der SHV zuerst aus beiden Policies entfernt werden.
2. Anschliessend kann die Dynamic Link Library (DLL) im *system32*-Verzeichnis (oder im *SysWOW64*-Verzeichnis) mit *regsvr32 Shv.dll /u* beim System abgemeldet werden.
3. Auf Clientseite muss dasselbe mit der DLL *ShaInfo.dll* gemacht werden.

Fehlerquellen

Falls beim Starten der sha.exe-Datei ein Fehler auftritt so sollte überprüft werden ob die Datei *nap.txt* existiert.

In einer 64bit Umgebung kann das Kompilieren der Debug-Variante mit der Datei Common.lib vom Linker Fehler ergeben. In diesem Fall sollte das Programm für 32bit kompiliert werden oder auf die Debug-Symbole verzichtet werden (Release kompilieren).

Falls der SHA nach erfolgter SHA/SHV-Einrichtung keine Antwort vom SHV erhält, kann es daran liegen, dass die dll auf dem NPS am falschen Ort platziert wurde. Wie beim Einrichten erwähnt gehört sie primär in den *system32* Ordner. Falls jedoch eine 32bit dll auf einem 64bit System verwendet wird, muss die Datei in den Ordner *%systemroot%/SysWOW64* kopiert und mit der in diesem Verzeichnis vorhandenen *regsvr32.exe* registriert werden.

Allgemein zu Informations- oder Debugzwecken kann auf Client-Seite kann der detaillierte Status in der *net-shell* von Windows abgefragt werden. Dazu muss in einem cmd-Fester folgende Anweisung ausgeführt werden: *netsh nap client show stat*. Im unteren Bereich der Ausgabe werden Details zu den SHAs und dessen Stati angezeigt.

6.3.2 TPM

Um die TPM Funktionalität nutzen zu können muss lediglich das TPM im BIOS aktiviert sein. Die Option findet man im BIOS meist unter dem Abschnitt Security.

Fehlerquellen

Die Bedeutung von auftretenden Fehlercodes, welche im Umgang mit dem TPM erscheinen, können in der Datei *tpm_error.h*[12] nachgeschlagen werden. Da zum jetzigen Zeitpunkt jedoch noch keine Fehlerbehandlung existiert, können solche Fehler nur im Debug-Modus des Visual-Studios nachvollzogen werden.

Kapitel 7

Tests

7.1 NAP

Bei der bisherigen Entwicklung des NAPs wurden keine automatisierten Tests verwendet. Der grösste Teil der Arbeit bestand darin, den Beispielcode für den SHA und den SHV zu analysieren und verstehen. Da es bestehende Software ist, wurde beschlossen, diesen Teil nicht automatisiert zu testen.

Jedoch wurde ein manueller Test definiert und ausgeführt. Dieser beinhaltet die Installation des SHA und SHV in der Testumgebung (siehe Kapitel 6.3.1). Anschliessend wurde durch anpassen der Datei *nap.txt* der Status des SHA verändert. Durch die Ausgaben der Debug-Informationen *(im sha.exe)* sowie der Ausgabe in der netshell (*netsh nap client show stat*) konnte getestet werden, ob das Programm geplant verlief oder Fehler auftraten.

7.2 TPM

Die implementierten TPM-Funktionalitäten wurden getestet. Jedoch wurde dazu kein Testframework eingesetzt. Es wurde mit einer selbstgeschriebenen Testklasse getestet. Da der TPMWrapper noch nicht genügend weit entwickelt wurde um Tests der TPM Funktionen sinnvoll zu automatisieren, wurden nur die Tests für die Hilfsfunktionen in dieser Klasse getestet.

Die Funktionalität des TPM wurde bis zum jetzigen Zeitpunkt manuell getestet. Es ist geplant, dass die TPM-Funktionalitäten jeweils automatisiert getestet werden.

Kapitel 8

Projektstand

Als Ergebnis dieser Arbeit ist ein Prototyp entstanden, der die NAP-Funktionalität beinhaltet und unterstützt. Ebenfalls ist es möglich, erste Funktionalitäten des TPMs anzusprechen.

Die Funktionsweise des NAPs wurde unter die Lupe genommen und das erarbeitete grundlegende Wissen schriftlich in Form einer Dokumentation festgehalten. Zudem wurde das erworbene Wissen anhand eines eigenen Collectors (SHA) sowie Verifiers (SHV) umgesetzt werden.

Die angestrebte Funktionsvielfalt des Prototyps im Bereich des TPMs konnte nicht vollständig umgesetzt werden. Gründe dafür sind TPM-Funktionalitäten, die komplexe algorithmische Abläufe benötigen. Zusätzlich benötigte das Team zusätzliche Zeit um sich in diese neue Problematik einzuarbeiten. Für die nachfolgende Arbeit ist die vertiefte Auseinandersetzung sicherlich ein Vorteil, doch wurde der jetzige Prototyp schlanker.

8.1 Zukunftsvision

Damit der Prototyp sinnvoll eingesetzt werden kann, soll er um weitere Funktionalitäten erweitert werden. Im Zusammenhang mit NAP stellen sich ebenfalls sicherheitstechnische Fragen, wie beispielsweise, ob der Client seinen Sicherheitsstatus falsch angeben oder die Mitteilung zwischen Client und Server abgeändert werden kann. Diese Problematik mit dem Client, der ein Falschaussage macht, wird als *Lying Endpoint Problem* bezeichnet. Möglichen Schwachpunkten soll in dieser Thematik nachgegangen und mögliche Lösungsszenarien dargestellt werden.

In diesem Abschnitt wird kurz auf ein mögliches Umsetzungsszenario eingegangen, da in der Zukunftsvision möglicherweise umgesetzt werden könnte.

Wie in Abbildung 8.1 zu sehen ist, wird der Gesundheitszustand auf der Clientseite wie folgt gebildet: Als Erstes wird der Hashwert der SHA-Datei berechnet. Dieser wird dann gemeinsam mit der aktuellen Systemzeit zusammengenommen (konkateniert) und als Hashwert dem TPM übergeben. Der Hashwert wird nun durch einen privaten Signierschlüssel, der auf dem TPM gespeichert ist, signiert. Der Hashwert wird dann zusammen mit der Signatur an den NAP-Server versendet. Der Server kann mit Hilfe des öffentlichen Signierschlüssels nachvollziehen, ob der SHA des Clients manipuliert wurde und deshalb keinen Zutritt ins Netzwerk erhalten darf.

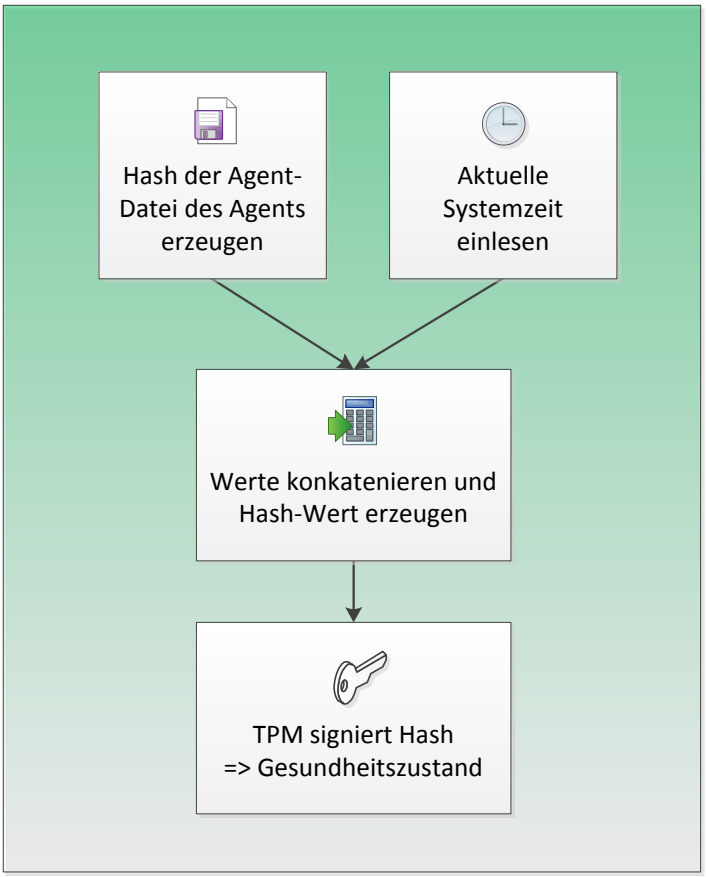

Abbildung 8.1: Signierablauf des Prototyps auf dem Clientcomputer

Kapitel 9

Projektmanagement

9.1 Einführung

Das Projektmanagement soll die Ziele des Projekts, die Zeitplanung, die Realisierung sowie die relevanten Rahmenbedingungen des Projektes dokumentieren. Damit bildet das Projektmanagement ein hilfreiches Werkzeug zur Steuerung des Entwicklungsprozesses.

Die Projektplanung stellte sich zu Beginn vorallem aufgrund des schlecht schätzbaren Einarbeitungsaufwandes als schwierig dar. Deshalb wurde dieser Bereich zu späteren Zeitpunkten dem aktuellen Projektstand angepasst und erweitert.

9.2 Involvierte Personen

9.2.1 Projektmitglieder

Das Team besteht aus zwei Informatikstudenten der HSR, die sich im fünften Semester befinden (Tabelle 9.1).

Wolfgang Altenhofer	wa	wolfgang.altenhofer@hsr.ch
Lukas Studer	ls	lukas.studer@hsr.ch

Tabelle 9.1: Team

Andreas Steffen	as	andreas.steffen@hsr.ch
Walter Sprenger	ws	walter.sprenger@csnc.ch

Tabelle 9.2: Betreuer

9.2.2 Externe Schnittstellen

Das Projekt wird von Prof. Dr. Andreas Steffen betreut und von Herrn Walter Sprenger gegengelesen. (Tabelle 9.2).

9.3 Management Abläufe

9.3.1 Projekt Kostenvoranschlag

Die Zeitplanung sieht pro Student 240 Stunden über 14 Wochen vor. Dies Entspricht einem total von 480 Stunden (17.1 Stunden pro Woche und Student).

9.3.2 Zeitplan

Der Zeitplan wurde zu Beginn geplant und aufgrund der wenig bekannten Projektthematik nachträglich leicht angepasst. Die Endfassung sowie die Auswertung der benötigten Arbeitsstunden sind im folgenden Kapitel ersichtlich.

Die Zeiterfassung erfolgt über das Tool mite.

9.3.3 Risikomanagement

Das Risikomanagement wird in der Tabelle 9.3 aufgeführt. Es wurde Wert darauf gelegt, dass die Risiken realistisch eingeschätzt wurden. Das Risiko R5 wurde als das grösste Wagnis gewertet, da in dieser Arbeit zwei Vertiefungen in unbekanntem Gebiet stattfanden.

9.4 Meilensteine

Die Meilensteine beschreiben die Termine und Zwischenziele, welche für die Entwicklung des Projektes hilfreich sind. Auf diese wird in Tabelle 9.4 näher eingegangen.

Nr.	Datum	Beschreibung	Ziele
MS1	17.10.2010 (Woche 04)	Abschluss Aufgabenstellung Teil 1	Fertigstellen der Dokumentationen über die NAP-Testumgebung, Protokollaufbau NAP und Erkenntnisse, welche in den ersten Projektwochen gesammelt wurden.
MS2	07.11.2010 (Woche 07)	TPM-Einarbeitung	Einarbeitung in Thematik TPM beendet, Programmiersprache und Vorgehensweise, die für Implementation verwendet werden, sind nun bekannt, Ende der Vorbereitungsphase.
MS3	28.11.2010 (Woche 11)	Core-Implementation fertiggestellt	Kernfunktionalität ist ausprogrammiert
MS4	12.12.2010 (Woche 12)	Abschluss der Programmierung	Codefreeze
MS5	20.12.2010 (Woche 13)	Poster- / Abstract-Abgabe	Abgabe Abstract und Poster
MS6	23.12.2010 (Woche 14)	Abgabe	Abgabe Dokumentationen, Kurzzusammenfassung und Bericht

Tabelle 9.4: Meilensteine

9.5 Infrastruktur

9.5.1 Hardware

Grundsätzlich arbeitet jedes Teammitglied auf seinem privaten Notebook. Neben diesen stehen für das gesamte Projekt zwei Computer zur Verfügung. Im Verlauf der ersten Wochen wird mithilfe von VMware eine Testumgebung, bestehend aus zwei Servern (Windows Server 2008 RC2) und zwei Clients (Windows 7 Professional), eingerichtet.

Zur Versionsverwaltung steht ein privater git-Server zur Verfügung.

9.5.2 Software

Für die Entwicklung des Kernprogramms wurde entschieden, Visual C++ zu verwenden. Als Entwicklungsumgebung dient das Visual Studio 2010. Die Dokumentation erfolgt mit LaTeX. Wo dies nicht möglich ist (Bsp. Zeitplanung) wird auf das Openoffice zurückgegriffen.

Für das Projektmanagement wird mite zur Zeiterfassung verwendet. Die Versionsverwaltung wird mit git vorgenommen.

9.6 Qualitätsmassnahmen

9.6.1 Dokumentation

Für uns ist es wichtig, dass wir eine hochwertige Software herstellen. Aus diesem Grund legen wird grossen Wert auf eine verständliche und aktuelle Dokumentation.

9.6.2 Protokollierung der Sitzungen

Ergebnisse, Pendenzen und Diskussionsentscheidungen werden bei den wöchentlichen Sitzungen mit dem Betreuer in kurzen Worten in einem Sitzungsprotokoll festgehalten. Dies bietet die Möglichkeit, besprochene Thematiken und Entscheidungen zu einem späteren Zeitpunkt nachzulesen.

Die Sitzungsprotokolle können im Anhang eingesehen werden.

9.6.3 Zeiterfassung

Die Zeiterfassung wird mit dem Onlinetool mite aufgeschrieben. Diese Software bietet dem Team die Möglichkeit, die geleisteten Arbeitszeiten zu notieren und später entsprechend auszuwerten. Es lassen sich personenbezogene Auswertungen, oder auch solche für das ganze Team erstellen. Somit kann nachträglich die Vorgehensweise nachvollzogen und die benötigte Zeit analysiert werden.

9.6.4 Qualitätssicherung

Um die Qualität der Dokumente hoch zu halten werden diese jeweils von der anderen Person gegengelesen. So können allfällige Unklarheiten gefunden und diskutiert werden.

9.6.5 Versionsverwaltungssystem

Der Quellcode wie auch die Dokumentation werden im git verwaltet, sodass alle Team Mitglieder auf den aktuellen Stand der Dateien Zugriff haben.

	Beschreibung	Auswirkung	Massnahme	max. Schaden	W'keit	Gewichtung
R1	Ausfall von Hardware	Betroffene Person könnte nicht weiterarbeiten	Ersatzhardware auftreiben	10 h	5%	0.5 h
R2	Ausfall des git-Servers	Code und Dokumente können nicht mehr synchronisiert werden	git-Patches per Mail austauschen und zusammenführen, Ergebnisse auf Ersatz-Server laden	2 h	10%	0.2 h
R3	Ausfall eines Team-Mitgliedes durch Krankheit/Unfall	Verlust einer Arbeitskraft	Features des Projektes anpassen. Mehraufwand für das Team.	9 h	10%	0.9 h
R4	Datenverlust	Unwiderruflicher Verlust von erarbeiteten Daten	git regelmässig updaten und auschecken, VMs mindestens wöchentlich sichern	9 h	5%	0.45 h
R5	Aufwandsplanung der Technologie war falsch /	Projektumsetzung nicht, bzw. mit (grosser) Verzögerung möglich	Ziele mit Betreuer neu definieren	100h	20%	20h
R6	Hürden durch nicht vorhandene oder schlechte Hersteller-Dokumentationen	Einarbeitung verzögert sich	Im Projektplan Puffer einplanen	10 h	5%	0.5 h
R7	Hürden durch noch unbekannte Programmier-sprache	Mehr Zeit für Einarbeitung / Implementation notwendig	Grosszügig planen	20 h	10%	2h

Tabelle 9.3: Risikofaktoren

Kapitel 10

Projektplanung

10.1 Zeitabrechnung

In Abbildung 10.1 ist der Verlauf des Aufwands pro Student ersichtlich. Es fällt auf, dass die Arbeitszeit zu Beginn der Arbeit nahe an der Soll-Zeit liegt. Gegen Ende (ab Woche 11) stieg die Arbeitszeit pro Woche stark an. Dies ist auf den unterschätzten Aufwand im Umgang mit dem TPM zurückzuführen. In der Tabelle 10.1 wird der durchschnittliche wöchentliche Aufwand pro Student aufgezeigt.

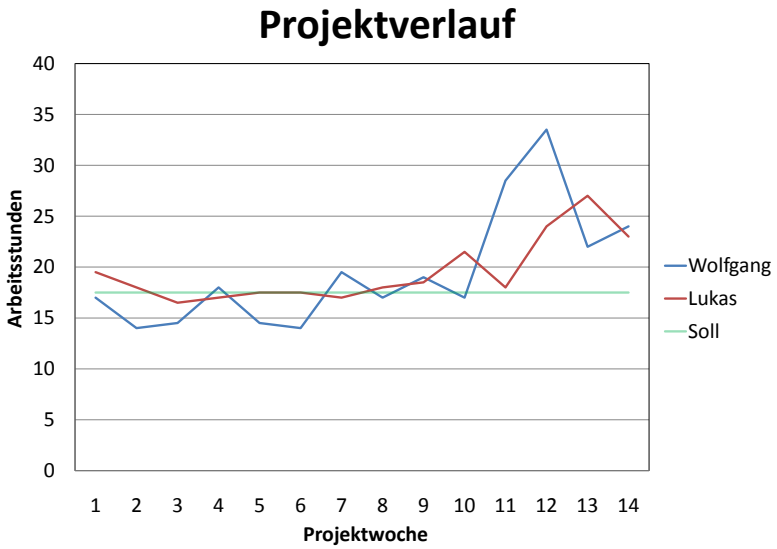

Abbildung 10.1: Aufgewendete Stunden pro Woche

Wer	Ø	∑
Altenhofer Wolfgang	19.3h	271h
Studer Lukas	19.5h	273h
Soll	17.15h	240h

Tabelle 10.1: Zeitauswertung: Wochenschnitt und Total

10.1.1 Aufteilung nach Arbeitspaketen

Die Grafik 10.2 zeigt, wie hoch der Aufwand prozentual zur gesamten geleisteten Arbeitszeit in den einzelnen Themenbereichen war. Es fällt auf, dass ein grosser Teil der Arbeitszeit für den Punkt Technologiestudium verwendet wurde.

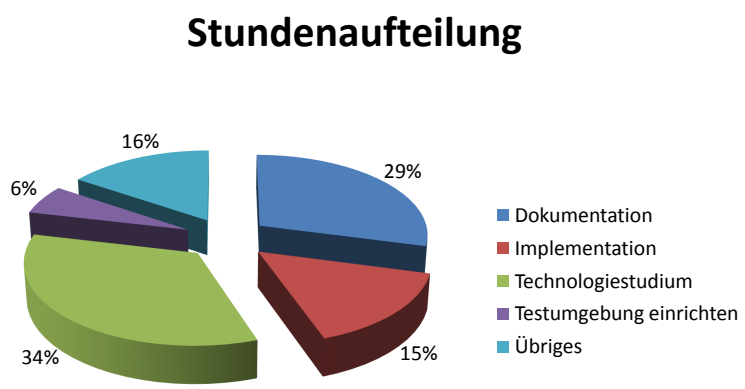

Abbildung 10.2: Prozentuale Aufteilung der Arbeitsstunden über den gesamten Projektverlauf

10.1.2 Vergleich NAP/TPM

Die beiden Diagramme in Abbildung 10.3 und 10.4 stellen genauer dar, wofür die Zeit in der Implementation und im Technologiestudium investiert wurde. Es wird in beiden Diagrammen gezeigt, dass die Handhabung des TPMs um einiges aufwendiger war als das NAP. Dies ist auf den erschienenen Mehraufwand zurückzuführen, für die Kommunikation mit dem TPM anfiel.

Das Verhältnis zwischen NAP und TPM ist ebenfalls deshalb so unausgeglichen, da zuerst mit einer ungeeigneten Library für das TPM gearbeitet wurde und deshalb zusätzlich eine Woche verschwendet wurde.

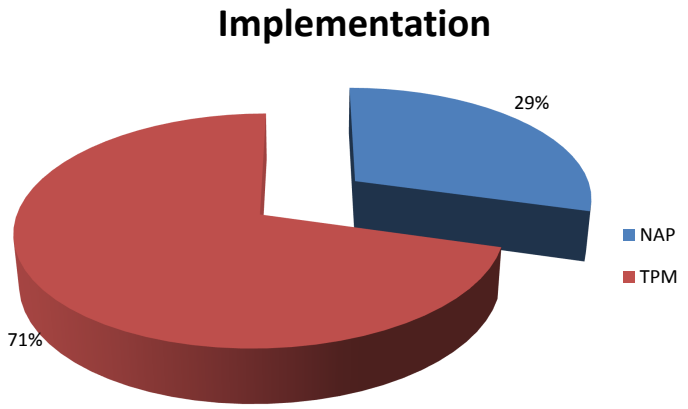

Abbildung 10.3: Aufteilung des Implementationsaufwandes

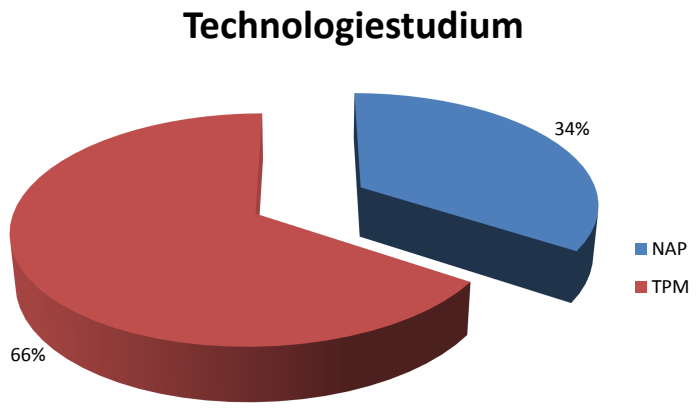

Abbildung 10.4: Aufteilung der benötigten Zeit für das Technologiestudium

10.1.3 Projektzeitplan

Der Projektzeitplan widerspiegelt den Verlauf der Arbeit. Zu Beginn musste viel Zeit für den Aufbau der Testumgebung investiert werden. Zudem wurde in den ersten Wochen viel Zeit für die Einarbeitung in die Technologie NAP aufgewendet.

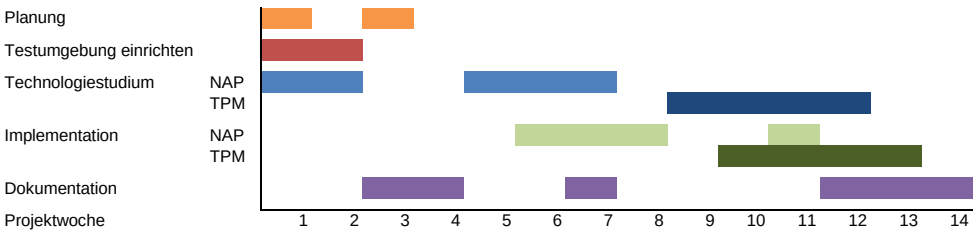

Tabelle 10.2: Projektplan

In der Woche 3 und 4 wurde die NAP-Protokollanalyse dokumentiert. Danach ging es weiter mit dem Technologiestudium, sodass die erste NAP-Implementation in Woche 6 beginnen konnte.

Mehr und mehr wurde dann auch das TPM zum Technologiestudiumsthema. Ab Woche 9 wurde die Thematik TPM aufgegriffen. Der Aufwand für das TPM war aber höher und es musste sehr intensiv gearbeitet werden. In Woche 11 wurden die beiden Prototypteile (NAP und TPM) in Kombination miteinander erfolgreich getestet.

Anhang A

Erfahrungsberichte

A.1 Wolfgang Altenhofer

Im Verlauf der Semesterarbeit habe ich viele neue Konzepte und Ideen rund um die Security mit NAP und speziell dem TPM kennengelernt. Während dem Aufsetzen der Windows-Testumgebung hatte ich nebenher eine gute Gelegenheit, meine eingerosteten Windowskenntnisse als Linux-Nutzer wieder etwas aufzufrischen. Nach einigen Start-Schwierigkeiten gelang anschliessend auch das Analysieren der NAP-Technologien und wir kamen anfangs gut voran.

Nach dem Fehlschlag mit *Win32_tpm*, wodurch viel Zeit verloren ging, gestaltete sich die Entwicklung mit dem TPM als sehr interessant. Ich realisierte während des Implementierens, dass ich in den Sicherheitstechnologien noch nicht so sattelfest bin. Häufig stand ich an vom TPM verwendeten Algorithmen an. Das Studium dieser Algorithmen gestaltete sich zwar als sehr aufwendig, jedoch auch spannend. So konnte ich die Thematik Detail studieren und sah, wie nun die Ver- und Entschlüsselung oder die Sicherung durch Hash-Funktionen im Hintergrund funktionieren. Diesen Punkt der Arbeit habe ich wohl auch am Stärksten unterschätzt. Ich wollte ursprünglich die TPM Funktionalität mit der Semesterarbeit schon weiter entwickelt haben. Jedoch habe ich in meiner Planung anfangs Semester den erwähnten Aufwand durch die Algorithmen unterschätzt.

Rückblickend betrachtet, haben wir wohl etwas zu viel Zeit für NAP eingesetzt und sind zu spät aufs TPM gewechselt.

Die Zusammenarbeit mit Andreas Steffen verlief sehr positiv. Wir erhielten von ihm viele positive Inputs und Hinweise zu den aufgetretenen Problemen. Auch habe ich die Freiheiten, die er uns im Verlauf der Arbeit gewährte, sehr geschätzt.

Ich freue mich darauf, diese Arbeit mit Lukas weiterführen zu können. Wir haben uns als Team gut ergänzt und es kam häufig zu interessanten Diskussionen. Speziell freue ich mich auch darauf, weiterhin mit dem TPM arbeiten zu dürfen. Ich finde die Funktionalität dahinter sehr interessant und hoffe, dass wir einen guten Wrapper für den Chip schreiben können, um die TPM-Funktionalität mit dem NAP einfach zu verbinden.

A.2 Lukas Studer

Sicherheit ist ein interessanter und herausfordernder Bereich, in welchem wir während des letzten halben Jahres vertieft tätig waren. Denn wir hatten in dieser Semesterarbeit die Chance, uns in neue Technologien einzuarbeiten. Zu Beginn stand viel Selbststudium an, doch die Thematik interessierte mich und deshalb war es auch kein Problem, in die Themen NAP und TPM einzutauchen.

Der Weg zu unserem jetzigen Resultat beinhaltete viele Knacknüsse, die es zu lösen galt. Ich sah dies immer als Herausforderung und stellte mich ihnen gerne. Im Team haben wir dann unsere persönlichen Lösungsansätze diskutiert, sodass wir einen möglichst optimalen Weg finden konnten.

In dieser Arbeit arbeiteten wir sehr häufig mit RFCs oder Protokollstandards. Zu Beginn war der Umgang mit diesen Standards etwas gewöhnungsbedürftig, doch mehr und mehr entwickelten sich diese Dokumente als wichtige Begleiter.

Herr Steffen hat uns souverän durch die Semesterarbeit begleitet. Er stand uns bei Fragen und algorithmischen Problemen zur Seite und lieferte immer wieder gute Gedanken und Ideen. Im Gegenzug wurde von uns viel Eigeninitiative gefordert, doch genau diesen Aspekt brachten wir entgegen. Denn durch die vielen Freiheiten, die uns Herr Steffen gab, waren unserer Umsetzungskreativität keine Grenzen gesetzt. Wir konnten auch die Richtung unserer Arbeit beeinflussen - und genau diese flexible Art empfand ich als sehr angenehm.

Es ist schade, dass wir etwas Zeit verloren, weil wir für die Kommunikation mit dem TPM auf der falschen Library aufbauten und schliesslich einsehen mussten, dass sie für unseren Anwendungszweck nicht geeignet war. Dies hat uns aber auch gezeigt, wie schnell der Zeitplan durcheinandergeraten kann und deshalb eine ausgiebige Technologie-Recherche wichtig ist, um mögliche Risiken zu minimieren. Wir verloren ebenfalls Zeit, weil die Kommunikation mit dem TPM das Wissen von einigen Algorithmen voraussetzte. Die Einarbeitung und das Verständnis dieser benötigte viel Zeit.

Ich freue mich, dass wir die Arbeit im Frühling 2011 fortsetzen können. Denn nun sind wir im Besitz von fundiertem Wissen. Das ermöglicht es uns, dass wir direkt mit nur minimalem zusätzlichen Technologiestudium die Ziele erreichen können und deshalb im Vergleich zu diesem Projekt umfangreichere Resultate erreichen können. Im übrigen arbeite ich gerne mit Wolfgang zusammen, da wir zusammen effizient arbeiten können. Ich empfinde den Umgang als angenehm, da wir uns gegenseitig ergänzen.

Anhang B

Abkürzungsverzeichnis

AIK Attestation Identity Key

ASCII American Standard Code for Information Interchange

AR Access Requestor

Cisco NAC Cisco Network Admission Control

COM Component Object Model

DHCP Dynamic Host Configuration Protocol

DLL Dynamic Link Library

EC Enforcement Client

EK Endorsement Key

ES Enforcement Server

HMAC Keyed-Hash Message Authentication Code

HRA Health Registration Authority

HSR Hochschule für Technik Rapperswil

IMC Integrity Measurement Collector

IMV Integrity Measurement Verifier

IPsec Internet Protocol Security

MAC Message Authentication Code

NAC Network Access Control

NAP Network Access Protection

NPS Network Policy Server

MS-HCEP Microsoft Health Certificate Enrollment Protocol

MS-SoH Microsoft Statement of Health for Network Access Protection

PDP Policy Decision Point

PEP Policy Enforcement Point

RADIUS Remote Authentication Dial In User Service

RSA Rivest Shamir Adleman

RTS Root of Trust for Storage

SDK Software Development Kit

SHA System Health Agent

SHV System Health Validator

SoH Statement of Health

SoHR Statement of Health Response

SRK Storage Root Key

SSoH System Statement of Health

SSoHR System Statement of Health Response

TBS TPM Base Services

TCG Trusted Computing Group

TNC Trusted Network Connect

TPM Trusted Platform Module

TSS TCG Software Stack

TXT Trusted Execution Technology

VPN Virtual Private Network

Anhang C

Sitzungsprotokolle

C.1 Protokoll vom 21.09.2010 - Woche 1 - Kickoff

Sitzungsinhalt

- Einführung in Thematik
- Definieren der Projektziele
- Projektziele noch nicht ganz fixiert ... Potenzial für eigene Umsetzungsideen vorhanden.

Pendenzen

- Aufgaben gemäss Aufgabenstellung

Termine

- Nächste Sitzung: Di, 28.09.2010 - 13:00 Uhr

Beschlüsse

- Aufgabenstellung überprüfen und gegebenfalls anpassen.
- Abklären, ob zusätzliche virtuelle Computerinfrastruktur benötigt wird.
- Vertiefen der Protokollfunktionalität von Microsoft NAP.

Aufgetretene Probleme

- -

C.2 Protokoll vom 28.09.2010 - Woche 2

Sitzungsinhalt

- Vorstellen der jetzigen Testumgebung (Microsoft NAP). Es wurde ein Netz mit vier virtuellen Rechnern erstellt. Weitere virtuelle Rechner sind momentan nicht notwendig.
- Analysieren der Protokollpakete aufwendig, aufgrund von verschlüsselter Übertragung.

Pendenzen

- Aufgabenstellung überprüfen und gegebenfalls anpassen.
- Vertiefen der Protokollfunktionalität von Microsoft NAP.

Termine

- Nächste Sitzung: Mi, 06.10.2010 - 14:00 Uhr

Beschlüsse

- Nächste Schritte: Übertragung für NAC analysieren: wie sieht die Kommunikation zwischen Client1 und NPS aus?
- Projektplan soll bis nächstem Termin als Entwurf vorliegen.

Aufgetretene Probleme

- Protokollübertragung erfolgt verschlüsselt. Analyse des entschlüsselten Datenverkehrs soll ermöglicht werden.

C.3 Protokoll vom 06.10.2010 - Woche 3

Sitzungsinhalt

- Analyse des momentanen Projektstandes:
  - Erlangte Erkenntnisse
  - Vorhandene Infrastruktur
  - Aufgetretene Probleme
  - Momentane Knackpunkte
- Neuausrichtung der Thematik:

Pendenzen

- Erkenntnisse über NAP dokumentieren
- Projektplan fertig ausarbeiten

Termine

- Nächste Sitzung: Di, 12.10.2010 - 13:00 Uhr

Beschlüsse

- Aufgabenstellung wird angepasst, da die ursprüngliche Aufgabenstellung eine zu umfangreiche Einarbeitungszeit erfordert, und für ein halbjähriges Projekt unrealistisch erscheint. Deshalb wird nun das Ziel verfolgt, sich in die Thematik des TPMs einzulesen und einen Collector und einen Verifier mit Verwendung von TPM für Windows 7 zu implementieren. Dazu soll das Microsoft-API verwendet werden. Mittels TPM soll erreicht werden, dass der Client den Health-Status nicht gefälscht werden kann, bzw. dass die Manipulation erkannt wird.
- Erkenntnisse (Kommunikationsausschnitt von NAP) dokumentieren.
- Analysieren, ob IP-Sec Enforcement auch mit PEAP anstatt HCEP realisiert werden kann.

Aufgetretene Probleme

- Codierter Protokollteil des SOH-Protokolls interpretieren

C.4 Protokoll vom 12.10.2010 - Woche 4

Sitzungsinhalt

- Erarbeitete Dokumente Projektplan und Zeitplanung für das Projekt wurden dem Betreuer gezeigt.
- Das Ersetzen von HCEP durch PEAP konnte bisher nicht erreicht werden.
- Risikoanalyse beinhaltet viele Risikofaktoren, welche unternehmensweit gelten und deshalb nicht in einer projektspezifischen Risikoanalyse vorkommen sollte. Forschungsrisiken werden wenig erwähnt.
- Als Hilfe für TPM-Entwicklungen existiert ein TPM-Emulator: www.tpm-emulator.berlios.de
- Im Standard von TCG / NAC sind die Bindings von IMV/IMC auf Windows und Linux genau beschrieben. Deshalb sollte für die Implementation dies beachtet werden.

Pendenzen

- Risikoanalyse überarbeiten (Technologische Risiken einbinden)
- Aufgabenstellung anpassen.

Termine

- Nächste Sitzung: Di, 19.10.2010 - 13:00 Uhr

Beschlüsse

- -

Aufgetretene Probleme

- -

C.5 Protokoll vom 19.10.2010 - Woche 5

Sitzungsinhalt

- Vorstellen der Projektdokumentation (Meilenstein 1)
- Protokoll HCEP: Besprechung des Ablaufdiagramms
- TPM: Einführung in TPM-Thematik in den kommenden Tagen.

Pendenzen

- Anmerkung bei Quellennutzung direkt bei der Abbildung oder in der Fusszeile.

Termine

- Nächste Sitzung: Di, 26.10.2010 - 13:00 Uhr

Beschlüsse

- Dump der Netzwerkmitschnitte auf die Abgabe-CD, Health-Certificate auf CD.

Aufgetretene Probleme

- -

C.6 Protokoll vom 26.10.2010 - Woche 6

Sitzungsinhalt

- Die Ruhr University stellt PDF-Präsentationen einer Vorlesung zu Trusted Computing zur Verfügung.
- Besprechung des TPM-Trusted-Bootvorgangs / TPM-Verschlüsslung
- Inwiefern wird die SoH- / TPM-Funktionalität in C# / Java / C++ unterstützt. Es wird noch weiter getestet und abgeklärt.

Pendenzen

- Programmiersprache auswählen / Funktionalität überprüfen.

Termine

- Nächste Sitzung: Di, 02.11.2010 - 13:00 Uhr

Beschlüsse

- -

Aufgetretene Probleme

- -

C.7 Protokoll vom 02.11.2010 - Woche 7

Sitzungsinhalt

- Es muss davon ausgegangen werden, dass der Kommunikationskanal unsicher ist. Dementsprechend müssen Manipulationen des Datenpakets erkannt werden können.
- TPM soll für Signierung der Übertragung verwendet werden.
- Hash des Dienstes (oder der ausführenden Datei) soll mit einem im TPM hinterlegten Hash verglichen werden können.
- Challenge soll garantieren, dass übermittelte Nachricht nur einmalig gültig ist.

Pendenzen

- -

Termine

- Nächste Sitzung: Di, 09.11.2010 - 13:00 Uhr

Beschlüsse

- -

Aufgetretene Probleme

- -

C.8 Protokoll vom 09.11.2010 - Woche 8

Sitzungsinhalt

- Lizenzierung der NAP-Samples von Microsoft ... unter welchen Bedingungen dürfen Codebeispiele weiterverwendet werden?
- Wie soll Request-Response realisiert werden? Problematik, da nur ein Request und eine Response vorhanden sind. Idee mit Challenge: Zeit des Clientcomputers soll als Challenge verwendet werden.
- Ist es möglich, mittels TPM die Dateiintegrität sicherzustellen, ohne dass ein infizierter Computer lügen kann? (Lying-Endpoint-Problem)
- Collector könnte ausgetauscht werden. Mögliche Lösungsvorschläge: Collector verschlüsselt ablegen und ihn temporär für die Überprüfung im Memory unverschlüsselt zwischenspeichern.
- Abschwächung der Gefahr:
  - Zugriffsbegrenzung über Rechtesystem des Betriebssystems.
  - Collector möglichst früh während des Boot-Prozesses laden, da so die Chance von Veränderungen durch Malware gemindert wird.

Pendenzen

- Lizenzierung der NAP-Samples abklären.
- Abklären, wie in anderen Projekten mit dem Lying-Endpoint-Problem umgegangen wird.

Termine

- Nächste Sitzung: Di, 16.11.2010 - 13:00 Uhr

Beschlüsse

- -

Aufgetretene Probleme

- Siehe Sitzungsinhalt

C.9 Protokoll vom 16.11.2010 - Woche 9

Sitzungsinhalt

- Lizenz des von Microsoft bereitgestellten Beispiels (NAP) darf weitergenutzt werden. Es ist lediglich Vermerk in Headerdatei nötig.
- Probleme beim NAP-Collector/-Verifier: Unregister-Funktionalität funktioniert nicht richtig.
- Eigene Test-Collector/-Verifier für NAP programmiert.
- Problematik Betriebssystem ist unsicher: sicherer Microkernel verwenden. Intel/AMD bieten entsprechende Hardwareunterstützung (Trusted Executing Technology / Intel TBoot).

Pendenzen

- Abklären, wie in anderen Projekten mit dem Lying-Endpoint-Problem umgegangen wird.

Termine

- Nächste Sitzung: Di, 23.11.2010 - 13:00 Uhr

Beschlüsse

- Für das TPM wird die API von Microsoft für C++ verwendet.
- Für Abgabe: möglichst eine Dokumentation abgeben.

Aufgetretene Probleme

- Unregister in NAP funktioniert nicht.

C.10 Protokoll vom 23.11.2010 - Woche 10

Sitzungsinhalt

- In der letzten Woche wurde versucht, das TPM über die Schnittstelle Win32_Tpm Windows Management Instrumentation (WMI) anzusprechen. Leider hat sich dies als ein nicht erfolgreicher Weg herausgestellt, sodass nun die TPM Base Services (TBS) von Microsoft verwendet werden. Ein erster erfolgreicher Test konnte am Montag durchgeführt werden.
- Befehlcodes für TPM sind auf der trustedcomputinggroup.org-Seite ersichtlich (ebenfalls als Headerdatei downloadbar).
- Lösungsmöglichkeiten für unsicheres OS weiter abklären (Ansätze: sicheres Laden des OS erzwingen, sicherer Microkernel laden).

Pendenzen

- gemäss Sitzungsinhalt

Termine

- Nächste Sitzung: Di, 30.11.2010 - 13:00 Uhr

Beschlüsse

- -

Aufgetretene Probleme

- Probleme mit der Ansteuerung des TPMs via WMI sind aufgetreten (s. Sitzungsinhalt).

C.11 Protokoll vom 30.11.2010 - Woche 11

Sitzungsinhalt

- Besprechung der Dokumentationsstruktur anhand des jetzigen Dokumentationsstands.
- Abkürzungsverzeichnis ist sinnvoll, Wort das erste Mal ausschreiben, dann als Abkürzung erwähnen. Glossar soll nicht erstellt werden.
- Abstract: Soll für nicht Eingelesene verständlich sein und einen Einblick in die Thematik geben.
- Erfahrungsberichte: selbstkritisch sein, Schlüsse ziehen
- Analyse des SHA-1-Algorithmuses aufgrund der aufgetretenen Probleme.

Pendenzen

-

Termine

- Nächste Sitzung: Di, 07.12.2010 - 13:00 Uhr

Beschlüsse

- -

Aufgetretene Probleme

- Probleme beim Ansprechen der SHA-1-Funktion des TPMs. Welchen Input benötigt das TPM?

C.12 Protokoll vom 07.12.2010 - Woche 12

Sitzungsinhalt

- Aufgabenstellung wird noch angepasst (TPM-spezifisch)
- Intel Trusted Execution Technology (TXT): Möglicherweise ein Teilgebiet für die Bachelorarbeit. Es stellt aber aufgrund noch nicht bestehenden Wissens ein grosses Risiko dar, könnte aber interessant sein.
- Aufgetretene Probleme: s. unten.
- In der letzten Projektwoche wird gegenüber den Experten eine Kurzpräsentation gehalten.

Pendenzen

- -

Termine

- Nächste Sitzung: Di, 14.12.2010 - 13:00 Uhr

Beschlüsse

- -

Aufgetretene Probleme

- Problem mit Authentisierung an TPM. TPM verweigert Login. Möglicherweise ein Problem mit der Nonce / HMAC-Bildung

C.13 Protokoll vom 14.12.2010 - Woche 13

Sitzungsinhalt

- Dokumentationsaufbau wurde besprochen
- Implementation soll ein Schwerpunkt darstellen
- TPM: Beschreiben, welche Funktionalitäten implementiert wurden, und welche aus zeitlichen Gründen ersten begonnen wurden.
- Referenz für RSA Algorithmen (zusätzlich zu RFCs): www.rsalabs.com / Standards Initiatives / Public-Key Cryptography Standards (PKCS)

Pendenzen

- Dokumentation fertigstellen
- Präsentation vorbereiten

Termine

- Nächste Sitzung (prov.): Di, 21.12.2010 - 13:00 Uhr
- Kurzpräsentation: Do, 23.12.2010 - 09:30 Uhr mit Betreuer und Gegenleser

Beschlüsse

- Arbeit am Donnerstag gedruckt und auf CD an Betreuer abgeben / pdf per Email an Betreuer und Gegenleser senden

Aufgetretene Probleme

- -

C.14 Protokoll vom 21.12.2010 - Woche 14

Sitzungsinhalt

- Poster und Abstract wurden besprochen. Kleine Anpassungen wurden getätigt.
- Präsentationstermin wurde bestätigt.

Pendenzen

- Dokumentation fertigstellen

Termine

- Kurzpräsentation: Do, 23.12.2010 - 09:30 Uhr mit Betreuer und Gegenleser

Beschlüsse

- -

Aufgetretene Probleme

- -

Literaturverzeichnis

[1] Jim Chung. C++ HMAC Implementation. http://www.codeproject.com/KB/recipes/HMACSHA1class.aspx. C++ Quellcode, ohne Lizenz, Letzte Konsultation 13. Dezember 2010.

[2] Microsoft Corporation. [MS-HCEP]: Health Certificate Enrollment Protocol Specification. Technical report, Microsoft, August 2010.

[3] Microsoft Corporation. [MS-SOH]: Statement of Health for Network Access Protection (NAP) Protocol Specification. Technical report, Microsoft, August 2010.

[4] Microsoft Corporation. NAP Server-side Architecture. http://msdn.microsoft.com/en-us/library/cc895519%28v=VS.85%29.aspx, 2010. Letzte Konsultation: 12. Dezember 2010.

[5] Microsoft Corporation. Step-by-Step Guide: Demonstrate NAP IPsec Enforcement in a Test Lab. http://www.microsoft.com/downloads/en/details.aspx?FamilyID=298ff956-1e6c-4d97-a3ed-7e7ffc4bed32&displaylang=en, 2010. Letzte Konsultation: 18. Dezember 2010.

[6] Microsoft Corporation. Windows SDK for Windows 7 and .NET Framework 4. http://www.microsoft.com/downloads/en/confirmation.aspx?FamilyID=6b6c21d2-2006-4afa-9702-529fa782d63b&displaylang=en, 2010. Letzte Konsultation: 13. Dezember 2010.

[7] Thrusted Computing Group. TPM Main Specification – Part 3 Commands. http://www.trustedcomputinggroup.org/files/resource_files/E14A09AD-1A4B-B294-D049ACC1A1A138ED/

mainP3Commandsrev103.pdf. Version 1.2, Revison 103, Letzte Konsultation 17. Dezember 2010.

[8] Trusted Computing Group. TCG Architecture Overview. http://www.trustedcomputinggroup.org/resources/tcg_architecture_overview_version_14. Version 1.4, Letzte Konsultation: 18. Dezember 2010.

[9] Trusted Computing Group. TNC IF-TNCCS Specification. http://www.trustedcomputinggroup.org/resources/tnc_iftnccs_specification. TLV Binding Version 2.0, Revision 16, Letzte Konsultation 18. Dezember 2010.

[10] Trusted Computing Group. TPM Main Specification. http://www.trustedcomputinggroup.org/resources/tpm_main_specification. Version 1.2, Revison 103, Letzte Konsultation 17. Dezember 2010.

[11] Dominik Reichel. C++ SHA1 Implementation. http://www.dominik-reichl.de/software.html#csha1. C++ Quellcode, ohne Lizenz, Letzte Konsultation 17. Dezember 2010.

[12] TCG Software Stack (TSS) Specification. http://www.trustedcomputinggroup.org/resources/tcg_software_stack_tss_specification. Version 1.2, Revision 85, Letzte Konsultation 17. Dezember 2010.

[13] Koordinationsstelle zur Bekämpfung der Internet-Kriminalität KOBIK. Jahresbericht 2009. http://www.kobik.ch/report/Rechenschaftsbericht_2009_DE.pdf. Letzte Konsultation 21. Dezember 2010.

Tabellenverzeichnis

Abbildungsverzeichnis