HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

ita Institute for Internet Technologies and Applications

TPM-basierter System Health Agent unter Windows 7

Bachelorarbeit

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

Wolfgang Altenhofer, Lukas Studer

17. Juni 2011

Betreuer: Prof. Dr. Andreas Steffen

Gegenleser: Prof. Dr. Peter Heinzmann

Experte: Dr. Ralf Hauser

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 Microsoft Network Access Protection (NAP) 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 Bachelorarbeit hatte das Team den Auftrag, ein SHA-/SHV-Paar zu erstellen, das mit Hilfe des TPMs Manipulationen am Windows SHA/SHV erkennen kann und dazu führt, dass der infizierte Computer nicht mehr in das Netzwerk gelassen wird.

Ergebnis

Mit Trusted NAP konnte die bestehende NAP-Lösung durch Trusted Computing Aspekte erfolgreich erweitert werden. Die Funktionalität des TPMs stellt sicher, dass Systeme mit manipulierten NAP-Dateien nicht mehr in das sichere Unternehmensnetzwerk gelangen können. Auch beliebig andere Dateien auf dem Client können auf ihre Unverändertheit getestet werden. Für Systemadministratoren wird es dadurch besser möglich, Sicherheitspolicies des Unternehmens durchzusetzen.

Trusted NAP lässt sich in bestehende NAP-Umgebungen integrieren. Clientseitig vorausgesetzt ist das Vorhandensein eines TPM sowie des Betriebssystems Windows 7; serverseitig wird ein Windows Server 2008 R2 benötigt.

Abbildung 1: Logo des resultierten Produktes *Trusted NAP*

Ausblick

Das Produkt ist nun bereit in einem produktiven System ausgetestet zu werden. Es müssen Erfahrungen zu Stabilität und Performance des Programms innerhalb von Netzwerken mit mehreren (hundert) Clients gesammelt werden. In der Testumgebung konnten Performancemessungen bisher nur im kleinen Rahmen ausgeführt werden.

Zur weiteren Sicherheitserhöhung könnte der Trusted SHA um die Technologie Trusted Execution Technology (TXT) erweitert werden, um Manipulationen durch Rootkits erkennen zu können.

Inhaltsverzeichnis

Erklärung über die eigenständige Arbeit **i**

Abstract **iii**
Zielsetzung . . . iii
Ergebnis . . . iii
Ausblick . . . iv

Inhaltsverzeichnis **v**

1 Aufgabenstellung **1**

2 Einleitung **5**
2.1 Umfeldanalyse . . . 6
2.2 Ausgangslage . . . 8
2.3 Dokumentation . . . 9

3 Grundlagen **11**
3.1 Trusted Computing . . . 11
3.1.1 Beschreibung . . . 11
3.2 Trusted Platform Module (TPM) . . . 11
3.2.1 Aufbau des TPMs . . . 12
3.2.2 Verwendete Schlüsselkategorien . . . 13
3.2.3 Spezielle Schlüssel . . . 15
3.2.4 Schlüsselhierarchie . . . 16
3.3 Network Access Control (NAC) . . . 17
3.3.1 Beschreibung . . . 17
3.3.2 TNC . . . 17
3.3.3 NAC-Umsetzungen . . . 18
3.3.4 Microsoft Network Access Protection (NAP) . . . 19
3.4 Ablauf NAP Verbindungsprozess (mit IPsec) . . . 19

3.4.1 Konformer Client 19
3.4.2 Nicht konformer Client 20
3.5 TCG Software Stack (TSS) 22
3.5.1 TSS-Grundlagen 22
3.6 Trusted Third Party - PrivacyCA.com 24
3.6.1 Zertifikate 24
3.6.2 Zertifikatlevel 25
3.7 Platform Configuration Register (PCR) 25
3.8 Attestation 26
3.9 Testumgebung 26
3.9.1 Aufbau Netzwerk 26
3.9.2 Logische Netzwerke 26
3.9.3 Computer der Testumgebung 28
3.9.4 Komponenten von NAP (IPsec-Enforcement) 29

4 Anforderungsspezifikation **33**
4.1 Einführung 33
4.1.1 Zweck 33
4.2 Allgemeine Beschreibung 33
4.2.1 Ergebnisse der Ist-Analyse 33
4.2.2 Ziel 34
4.3 Produktperspektive 34
4.3.1 Systemschnittstellen 34
4.3.2 Benutzerschnittstelle 34
4.3.3 Hardwareschnittstellen 34
4.3.4 Softwareschnittstellen 35
4.3.5 Kommunikationsschnittstellen 35
4.4 Funktionale Anforderungen 36
4.4.1 FAnf01: SHA / SHV Paar 36
4.4.2 FAnf02: Clientseitige Manipulationsdetektion 36
4.4.3 FAnf03: Attestation mit TPM-Chip 37
4.4.4 FAnf04: Attestation-Vorgang mit dem quote-Befehl . . 37
4.4.5 FAnf05: Attestation-Vorgang mit dem quote2-Befehl . 37
4.4.6 FAnf06: Attestation mit Attestation Identity Key (AIK) 37
4.4.7 FAnf07: Trusted by PrivacyCA.com / AIK-Zertifikat . 37
4.4.8 FAnf08: Level 1 AIK-Zertifikat (PrivacyCA.com) 38
4.4.9 FAnf09: Paralleles Ausführen des SHV 38
4.4.10 FAnf10: Fehlerlogging in Protokolldatei 38
4.4.11 FAnf11: Fehlerlogging in Event Viewer 38
4.4.12 FAnf12: Platform Configuration Register (PCR) sperren 38
4.4.13 FAnf13: Überprüfung der Dateien im Arbeitsspeicher 39
4.4.14 FAnf14: Dynamisches Laden der SHAs auf dem Client überprüfen 39
4.5 Priorisierung der funktionalen Anforderungen 39

4.6 Nichtfunktionale Anforderungen 40
4.6.1 NfAnf01: Zuverlässigkeit 40
4.6.2 NfAnf02: Sicherheitsanforderungen 40
4.6.3 NfAnf03: Korrektheit 40
4.6.4 NfAnf04: Automatisierter Ablauf 40
4.7 Use Cases 41
4.7.1 Mitarbeiter 41
4.7.2 Systemadministrator 42
4.8 Benutzercharakteristik 42
4.9 Einschränkungen 42
4.9.1 E01: Performance in grossen Netzwerken 42
4.9.2 E02: Hardwarebeschränkungen 43
4.9.3 E03: Serverseitige Sicherheit 43
4.9.4 E04: Windows Vista und Windows Server 2008 43
4.10 Lizenzanforderungen 43
4.10.1 Urheberrecht 43
4.10.2 Verwendung 43

5 Analyse **45**
5.1 Trusted NAP Funktionsweise (FAnf01, 02) 45
5.2 TPM-Integration 47
5.2.1 Kommunikation mit dem TPM (FAnf02) 47
5.2.2 Zusätzliche Übermittlung von Daten (FAnf01, 02) . . . 48
5.2.3 Challenge-Response mit Nonce (FAnf03, 04, 14) 51
5.2.4 Ablauf der Attestation (FAnf03, 04, 06) 51
5.2.5 TPM Initialisierung 53
5.2.6 Zusätzliche Technologieentscheide 54
5.3 Domain Model 54
5.3.1 Trusted System Health Agent (SHA) 55
5.3.2 Trusted System Health Validator (SHV) 58

6 Design **61**
6.1 Architektur / Komponenten 61
6.1.1 Client-PC 61
6.1.2 TPM 61
6.1.3 NAP Health Policy Server 61
6.1.4 Windows-based NAP Enforcement Point 62
6.1.5 Privacy CA 62
6.2 Projektstruktur 63
6.3 Klassen 63
6.3.1 TPM 63
6.3.2 Registry 64
6.3.3 Logging 65
6.4 Datenspeicherung 65

6.5 Schnittstellen und Formate 65
6.5.1 Privacy CA 66
6.5.2 TPM 66
6.5.3 Zwischen Trusted SHA und Trusted SHV 66
6.5.4 Zwischen Enforcement Server und Enforcement Client 66
6.5.5 Zwischen NPS Service und NAP Enforcement Server . 66
6.6 Testing 67

7 Implementation **69**
7.1 Schnittstelle zu Windows NAP (FAnf01) 69
7.1.1 TrustedShaInfo.dll 69
7.1.2 TrustedSha.exe 70
7.1.3 TrustedShv.dll 70
7.2 Trusted SHA / Trusted SHV (FAnf01) 70
7.2.1 Windows Dienst 70
7.2.2 Type Length Value (TLV) 71
7.2.3 VendorDataWriter (Class) 72
7.2.4 VendorDataReader (Class) 73
7.3 Attestation (FAnf02, 03, 04, 06, 14) 73
7.3.1 TPM 73
7.3.2 TpmObject (Class) 74
7.3.3 TpmAttestation (Class) 75
7.4 Privacy CA (FAnf07, 08 78
7.4.1 Ablauf 79
7.4.2 PrivacyCaConnector (Class) 81
7.4.3 WinHttp (Class) 81
7.4.4 Verarbeitung der Antwort auf ein Request 82
7.5 Verifikation der Messwerte / Verifier (Class) (FAnf 02) 82
7.5.1 Überprüfung des Zertifikates 82
7.5.2 Überprüfung der Challenge 83
7.5.3 Überprüfung der Signatur 83
7.5.4 Überprüfen der mitgesendeten PCR-Daten 83
7.5.5 Überprüfung der gesendeten Messungen 83
7.6 Administration 84
7.6.1 Windows Registry 84
7.6.2 BaseRegistryWrapper (Class) 86
7.6.3 RegistryWriter (Class) 87
7.6.4 ConfigWrapper (Class) 87
7.6.5 Rechtevergabe 88
7.6.6 Logging (FAnf 10) 88
7.6.7 BaseLogger (Class) 88
7.6.8 stdOutLogger (Class) 90
7.6.9 FileLogger (Class) 90
7.6.10 PolicyManager (Class) 90

7.6.11 TrustedSHAconfig/TrustedSHVconfig 90
7.6.12 ToolOptions (Class) 91
7.6.13 ArgumentParser (Class) 91
7.6.14 Executer (Class) 91
7.6.15 TpmInitOwner (Class) 92
7.7 Fehler während der Implementation 93
7.7.1 Einbezug eines TSS 94
7.7.2 Internal error, source unknown 94
7.7.3 Extend 94

8 Bedienungsanleitung **97**
8.1 Installation 97
8.1.1 Client - Trusted SHA 97
8.1.2 Server - Trusted SHV 99
8.2 Deinstallation 103
8.2.1 Client - Trusted SHA 103
8.2.2 Server - Trusted SHV 105
8.3 Konfigurieren mit TrustedSHAconfig/TrustedSHVconfig . . . 105
8.3.1 Allgemeine Konfigurationen 105
8.3.2 Konfigurationen Trusted SHA / Trusted SHV 107
8.3.3 Key löschen 109
8.3.4 Initialisierung 109
8.4 TPM-Konfiguration im BIOS 111
8.4.1 Zurücksetzen des TPMs 112
8.4.2 Einschalten des TPMs im Bios 113
8.5 Analysetools von Windows für NAP und TPM 113
8.5.1 Trusted Platform Module (TPM) Management 113
8.5.2 Napstat 115
8.6 Troubleshooting mit NetShell (netsh) 116
8.7 Fehlerquellen 117
8.7.1 Gruppenrichtlinien blockieren TPM-Funktionalitäten . 117
8.7.2 Command Prompt mit Administratorrechten starten . 119

9 Projektstand **121**
9.1 Zielerreichung 121
9.2 Ausblick 122

A Projektmanagement **125**
A.1 Einführung 125
A.2 Involvierte Personen 125
A.2.1 Projektmitglieder 125
A.2.2 Externe Schnittstellen 126
A.3 Management Abläufe 126
A.3.1 Projekt Aufwand 126

A.3.2 Zeitplan . . . 126
A.3.3 Risikomanagement . . . 126
A.4 Meilensteine . . . 127
A.5 Projektplan . . . 129
A.6 Infrastruktur . . . 129
A.6.1 Hardware . . . 129
A.6.2 Software . . . 129
A.7 Qualitätsmassnahmen . . . 130
A.7.1 Dokumentation . . . 130
A.7.2 Protokollierung der Sitzungen . . . 130
A.7.3 Verantwortlichkeiten . . . 130
A.7.4 Zeiterfassung . . . 131
A.7.5 Qualitätssicherung . . . 131
A.7.6 Versionsverwaltungssystem . . . 131

B Projektplanung **133**

C Erfahrungsberichte **139**
C.1 Wolfgang Altenhofer . . . 139
C.2 Lukas Studer . . . 140

D Abkürzungsverzeichnis **141**

E Inhalt der CD **145**

F Sitzungsprotokolle **147**
F.1 Protokoll vom 21.02.2011 - Woche 1 - Kickoff . . . 147
F.2 Protokoll vom 28.02.2011 - Woche 2 . . . 149
F.3 Protokoll vom 07.03.2011 - Woche 3 . . . 150
F.4 Protokoll vom 14.03.2011 - Woche 4 . . . 151
F.5 Protokoll vom 21.03.2011 - Woche 5 . . . 152
F.6 Protokoll vom 28.03.2011 - Woche 6 . . . 153
F.7 Protokoll vom 04.04.2011 - Woche 7 . . . 154
F.8 Protokoll vom 12.04.2011 - Woche 8 . . . 155
F.9 Protokoll vom 18.04.2011 - Woche 9 . . . 156
F.10 Protokoll vom 02.05.2011 - Woche 11 . . . 157
F.11 Protokoll vom 09.05.2011 - Woche 12 . . . 158
F.12 Protokoll vom 23.05.2011 - Woche 14 . . . 159
F.13 Protokoll vom 30.05.2011 - Woche 15 . . . 160
F.14 Protokoll vom 06.06.2011 - Woche 16 . . . 161
F.15 Protokoll vom 10.06.2011 - Woche 16 . . . 162

G Errorcodes **163**

Literaturverzeichnis **169**

Tabellenverzeichnis **171**

Abbildungsverzeichnis **172**

Listings **174**

Kapitel 1

Aufgabenstellung

Auf der nächsten Seite folgt die Aufgabenstellung.

Institute for Internet Technologies and Applications

HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL
FHO Fachhochschule Ostschweiz

Bachelorarbeit 2011

TPM-basierter System Health Agent unter Windows 7

Studenten: Wolfgang Altenhofer, Lukas Studer

Betreuer: Prof. Dr. Andreas Steffen

Ausgabe: Montag, 21. Februar 2011

Abgabe: Freitag, 17. Juni 2011

Einführung

Trusted Network Access Control (TNAC) 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 Betriebsystem-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 einer vorausgegangenen Studienarbeit wurden durch die Studentengruppe die Mechanismen und Protokolle von Network Access Protection (NAP), der Microsoft TNAC Variante mit Windows 7 als Client und Windows Server 2008 R2 als Network Access Authority genauer untersucht. Dabei wurde festgestellt, dass Microsoft noch keinerlei Gebrauch vom Trusted Platform Module (TPM) macht, um das Problem des „Lying Endpoint“ zu lösen, d.h. zu vereiteln, dass der System Health Agent (SHA), vorgetäuschte Messungen an den System Health Validator (SHV) senden kann.

Im Rahmen dieser Bachelorarbeit sollen Messungen eines SHA durch das auf dem Motherboard des Clients vorhandene TPM mit einem vertrauenswürdigen Schlüssel signiert werden. Das TNC@FHH Projekt der FH Hannover hat auf Linux Plattformen schon eine gewisse Vorarbeit im Gebrauch von Attestation Identity Keys (AIKs) geleistet. Weiter soll auch das dynamische Laden der System Health Agents, soweit dies das Windows Betriebssystem und der Prozessor erlaubt, durch das TPM überwacht werden.

Aufgabenstellung

- Einarbeitung in die Attestation von Messungen durch ein TPM.
- Erstellen eines SHA / SHV Paars unter Windows 7 / Windows Server 2008 R2, das einen Attenstation Identity Key und ein dazugehöriges AIK Zertifikat verwendet, um Messungen zu signieren.
- Erstellen eines SHA /SHV Paars, welches das korrekte dynamische Laden der SHAs auf dem Client überwacht.

Links

- Windows Server 2008 R2 Network Access Protection (NAP)
  http://www.microsoft.com/windowsserver2008/en/us/nap-technical-resources.aspx
- TNC@FHH Projekt der Fachhochschule
  http://trust.inform.fh-hannover.de/joomla/index.php/projects/tncfhh
- TCG Architecture Overview Version 1.4
  http://www.trustedcomputinggroup.org/resources/tcg_architecture_overview_versiov_14
- TPM Main Specification
  http://www.trustedcomputinggroup.org/resources/tpm_main_specification
- Rafal Wojtczuk, Joanna Rutkowska, Attacking Intel Trusted Execution Technology
  http://blackhat.com/presentations/bh-dc-09/Wojtczuk_Rutkowska/BlackHat-DC-09-Rutkowska-Attacking-Intel-TXT-slides.pdf

Rapperswil, 21. Februar 2011

Prof. Dr. Andreas Steffen

Kapitel 2

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[15].

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 sollen.

In dieser Arbeit sollen die erarbeiteten Kenntnisse zum Network Access Control (NAC) sowie zum Trusted Platform Module (TPM), welche bereits in der Semesterarbeit betrachtet wurden, vertieft umgesetzt werden. Mit dem Network Access Control wird ermöglicht, dass nur Clients mit gewissen Eckwerten auf das Netzwerk zugreifen können. Die beiden Technologien werden kombiniert, um die Möglichkeit einer Manipulation im NAC zu verkleinern.

Das TPM und die NAC-Lösung von Microsoft, das Network Access Protection (NAP), sollen kombiniert werden. Das TPM überwacht die Messung auf dem Client und attestiert die Berechnung. Fehlt bei der Beglaubigung die Zustimmung des Client-TPMs, so muss der Server davon ausgehen, dass während des Messungsprozesses unerwünschte Dritt-Einflüsse mitspielten. Bestandteil der Arbeit ist die Einarbeitung in die Attestierung von Messungen durch das TPM.

2.1 Umfeldanalyse

Im Bereich NAP und TPM existieren viele Umfeld- und Technologieeinflüsse, die auf das gesamte Projekt und damit das NAP-System einwirken. Diese sollen hier erfasst und beurteilt werden. In Abbildung 2.1 werden die Einflüsse zusammenhängend dargestellt.

Beim Begriff des Zugriffsschutzes Network Access Protection (NAP) steht der Schutz des Netzwerks im Fokus. Diese Technologie wurde mit den Serverbetriebssystemen Windows Server 2008 und Clientbetriebssystemen Windows Vista sowie 7 standardmässig verfügbar gemacht. Für Windows XP wurde die NAP-Funktionalität mit dem Service Pack 3 nachgerüstet. Durch die generelle Verfügbarkeit in den Betriebssystemen wird es für Unternehmen interessant, diese Technologie einzusetzen. Was allerdings nicht unbeachtet gelassen werden darf, ist der damit erhöhte Management- und Konfigurationsaufwand, der mit der Einführung von NAP resultiert.

Die Verbreitung der TPM-Technik ist bereits beachtlich, da bereits 15 namhafte Hersteller bei neuen Computern dieses Modul standardmässig verbauen und mit dem Infineon TPM Professional Package ausliefern[14]. Die Verfügbarkeit dieses Moduls auf den Clientrechnern stellt längerfristig gesehen ein immer kleiner werdendes Hindernis dar. Applikationen, welche das Hardwaremodul TPM unter Windows verwenden, sind zum heutigen Zeitpunkt rar. Eines der wenigen verbreiteten Programmen ist die Festplattenverschlüsselung Bitlocker.

Die Bedienung des Computers wird durch die Installation für den Benutzer von Trusted NAP nicht verändert. Das bereits bei NAP vorhandene Ziel, keinerlei Interaktion mit dem Benutzer während des Verbindungs-Ablaufs zu haben, wird weiterhin verfolgt.

Die Benutzer haben mit einem um ca. drei Sekunden längerdauernden Verbindungsaufbau zu rechnen. Dies liegt daran, dass zusätzliche Überprüfungen und Verifizierungen stattfinden. Doch die eigentliche Geschwindigkeitsbremse stellt das TPM dar, denn es benötigt für die verwendeten kryptographischen Funktionalitäten eine bemerkbare Zeit. Auf der Serverseite nimmt die erweiterte Verifikation ebenfalls mehr Zeit in Anspruch. Diese ist aber so klein, dass lediglich mit einer minimen Verzögerung zu rechnen ist, welche erste bei vielen (geschätzt ab hundert) gleichzeitigen Client-Anfragen ins Gewicht fallen.

Die clientseitige Kommunikation zwischen Betriebssystem und TPM erfolgt auf dem Client via TrouSerS, einer Abstraktionsebene für die Low-Level-Kommunikation mit dem TPM. Da dieses Framework auf den TPM Base Services (TBS) aufsetzt, wird auf Clientseite ein Windows ab Windows Vista vorausgesetzt.

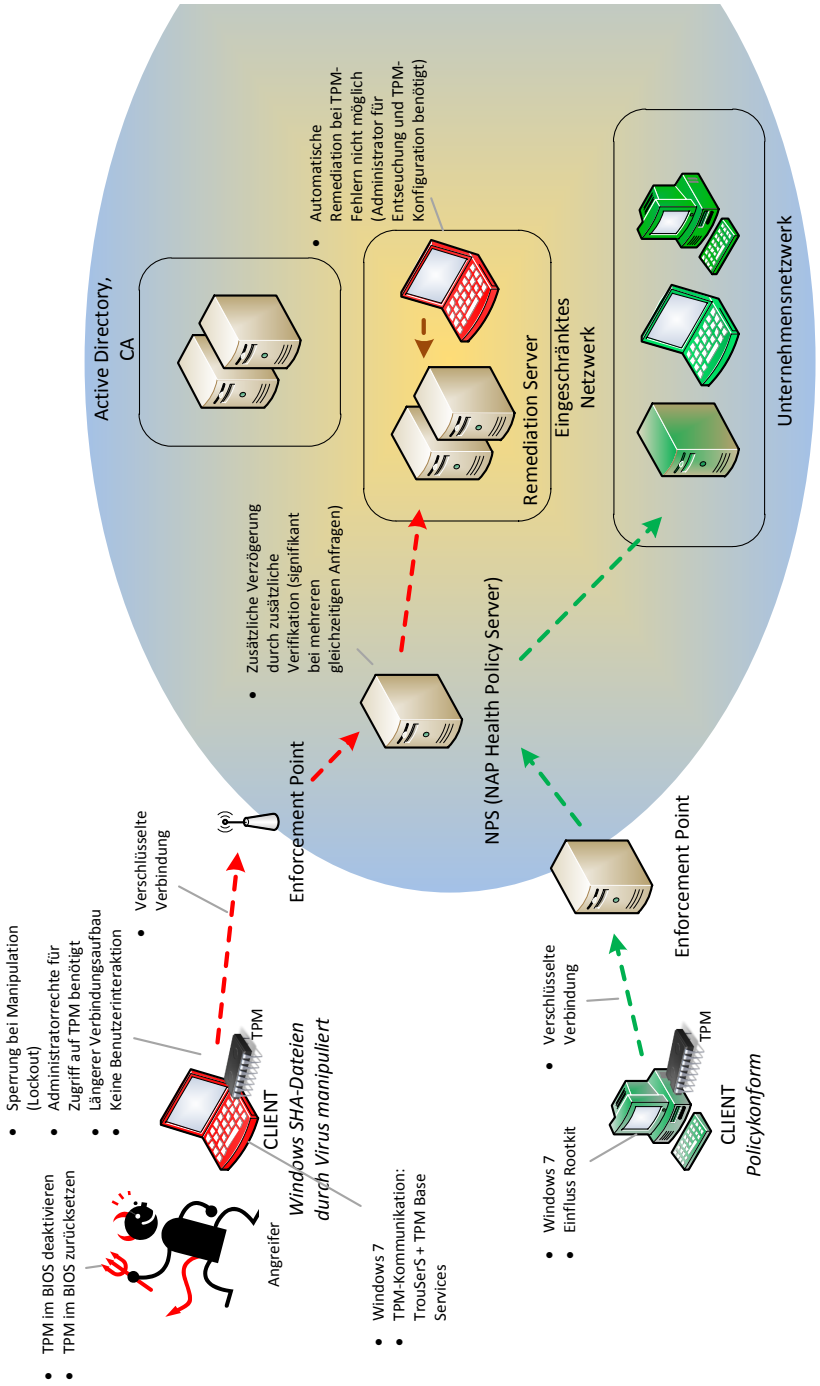


Abbildung 2.1: Übersicht Umfeldanalyse

Trusted NAP kann die Sicherheit des NAPs erhöhen. Wenn Schadsoftware auf dem Clientcomputer versucht, die NAP-Dateien zu manipulieren, dann

wird der Client später nicht mehr in das sichere Unternehmensnetzwerk gelassen, sondern erhält lediglich Zugriff auf ein spezielles Subnetz (Remediation Subnetz). Es darf allerdings nicht unbeachtet bleiben, dass sich die Infektion nicht automatisch korrigieren lässt und deshalb keine automatische Nachbesserung durch die Remediation Server erfolgt. Solange der Computer nicht durch einen Systemadministrator repariert werden kann, kann der Mitarbeiter nicht auf unternehmensinterne Datenablagen zugreifen. Durch diesen Sachverhalt kann dem Unternehmen ein Schaden entstehen, denn der Mitarbeiter ist nicht mehr vollumfänglich fähig, seiner Arbeit nachzugehen.

Die Verbindung zwischen Client und dem Unternehmensnetzwerk wird bereits durch das NAP-System verschlüsselt. Da dieselbe Verbindung verwendet werden kann, muss keine zusätzliche Sicherung der Übertragung erfolgen.

Finden auf dem Clientcomputer wiederholt Zugriffe mit falschem Passwort (beispielsweise durch eine Malware) statt, so verweigert das TPM die weitere Funktionsweise. Somit wird die erfolgreiche Verbindung in das Unternehmensnetzwerk verunmöglicht. Für den Mitarbeiter ist es dann nicht mehr möglich, sich in das Unternehmensnetzwerk zu verbinden, bis der Systemadministrator das TPM entsperrt, die Ursache der TPM-Sperrung ausfindig gemacht und den Computer wieder in einen manipulationsfreien Zustand gebracht hat.

Das TPM kann mit Hilfe des BIOS deaktiviert oder gelöscht werden. Dadurch wird ebenfalls die erfolgreiche Verbindung durch das Trusted NAP in das Unternehmensnetzwerk verunmöglicht. Dieses Risiko kann sehr leicht minimiert werden, indem der Systemadministrator ein Zugangspasswort für das BIOS definiert.

Rootkits, welche sich auf dem Clientcomputer eingenistet haben, können durch das Trusted NAP-System nicht erkannt werden. Gegen diese Art der Attacke ist NAP sowie auch Trusted NAP machtlos, da das Rootkit während der Messung bei einem Zugriff jeweils dem Betriebssystem die originale Datei liefert und während des Ladevorgangs eine manipulierte Datei dem DLL-Loader mitgibt. Solche Manipulationen könnten mit Technologien wie Trusted Execution Technology (TXT) erkannt werden.

Die angesprochenen Einflüsse werden nochmals übersichtlich in einem Mind-Map in Abbildung 2.2 aufgezeigt.

2.2 Ausgangslage

Das Projekt kann direkt an die Resultate und Erkenntnisse der Semesterarbeit aufsetzen. Dies beinhaltet einen Prototypen, der als SHA / SHV in eine

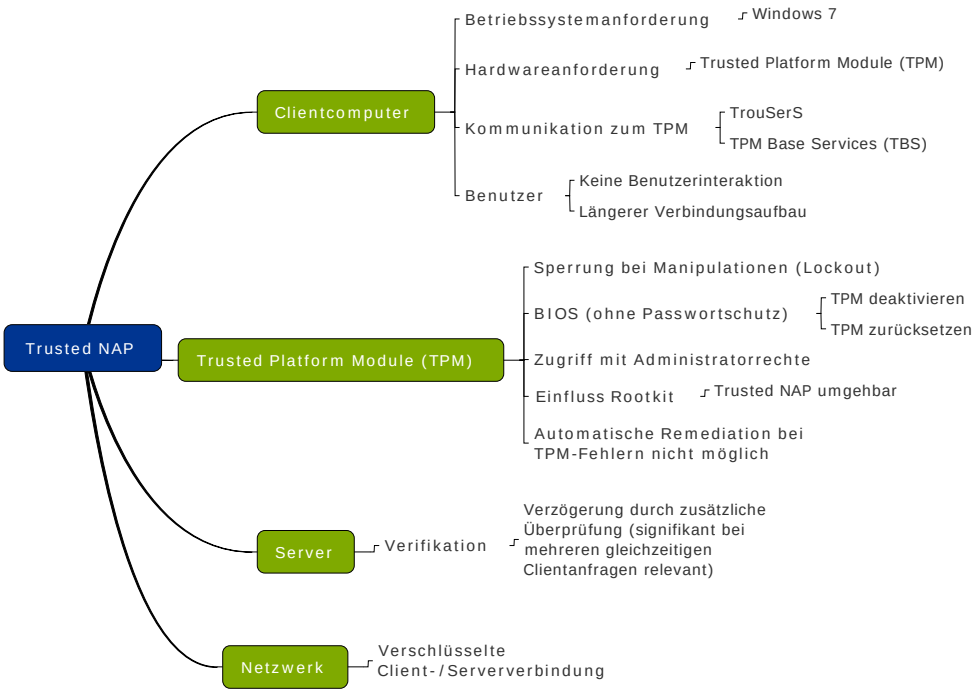


Abbildung 2.2: Mind-Map Umfeldanalyse

NAP-Umgebung integriert werden kann. Zudem ist es möglich, eine Kommunikation mit dem TPM-Sicherheitsmodul aufzubauen und erste Funktionen darauf aufzurufen. Diese Funktionalitäten sind in einem Wrapper abgelegt, der sich um weitere TPM-Funktionen erweitern lässt.

Gleichzeitig konnte während der Semesterarbeit das notwendige Know-How im Umgang mit dem TPM und dem NAP aufgebaut werden. Ausserdem ist bereits eine vollständig funktionsfähige Testumgebung vorhanden, die für Tests innerhalb des NAPs verwendet werden kann.

2.3 Dokumentation

Da diese Arbeit auf der Semesterarbeit aufbaut, wurden für die Erstellung dieser Dokumentation folgende Kapitel aus der Semesterarbeit-Dokumentation übernommen:

- Abstract: Erste drei Paragraphen
- Einleitung (Kapitel 2): Erste fünf Paragraphen

- Grundlagen (Kapitel 3.1 bis und mit 3.4): innerhalb dieser Kapitel wurden kleinere Änderungen vorgenommen worden
- Projektmanagement (Kapitel A): einzelne Aspekte übernommen

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 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 TPMs

Das TPM besteht aus drei Hauptbereichen:

- Functional Units, welche die kryptographischen Funktionalitäten zur Verfügung stellen
- Nichtflüchtiger Speicher, welcher für die permanente Hinterlegung von Informationen (beispielsweise Schlüsseln) dient
- Flüchtiger 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 vor allem 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.

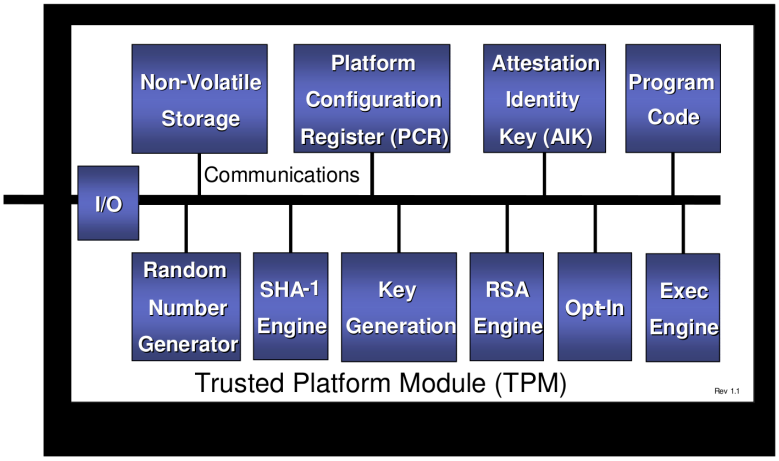


Abbildung 3.1: TPM: Aufbau und Funktionalitäten[7]

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 dies zu verbessern 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 des Einrichtungsprozesses 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, damit die Signierungen nicht mit dem Endorsement Key (EK) erfolgen müssen. Würde nämlich die Signierung mit dem EK 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.

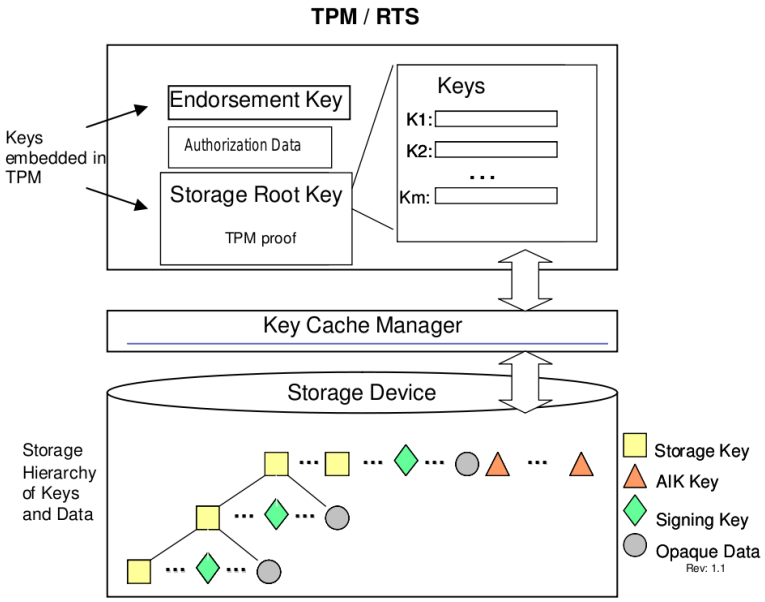


Abbildung 3.2: TPM: Schlüsselhierarchie[7]

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 gelangen, 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 und dadurch z.B. der 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.

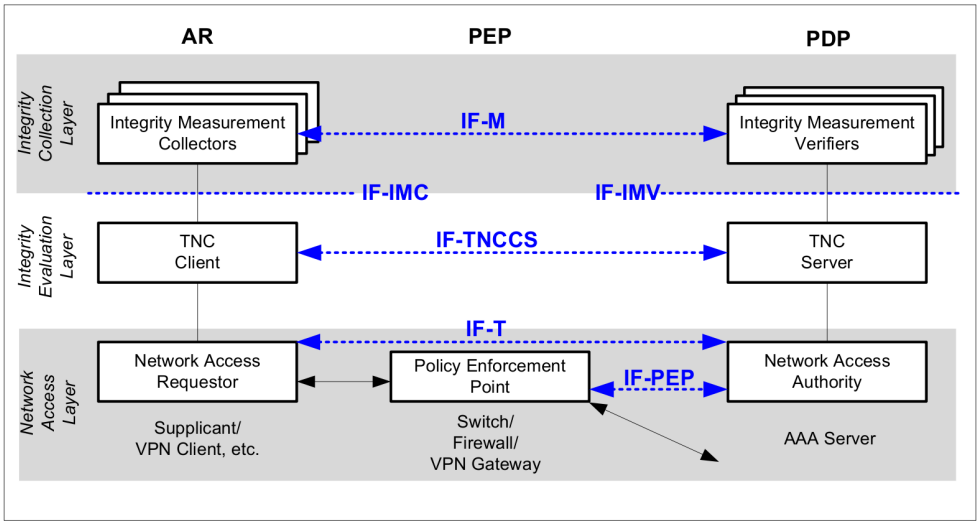


Abbildung 3.3: Architektur-Übersicht: Trusted Network Connect[8]

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 vor allem 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)

Eine Gegenüberstellung über die verschiedenen Bezeichnungen wird in Tabelle 3.1 gezeigt.

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

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 Tabelle 3.1 werden diese unterschiedlichen Bezeichnungen gegenüber gestellt.

3.4 Ablauf NAP Verbindungsprozess (mit IPsec)

3.4.1 Konformer Client

Ein konformer Client besitzt für den Verbindungsaufbau folgenden Ablauf (Abb. 3.4):

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, welche sich meist im selben Server wie der NPS befindet, sendet die SSoH-Daten an den NAP Health Policy Server (NPS).

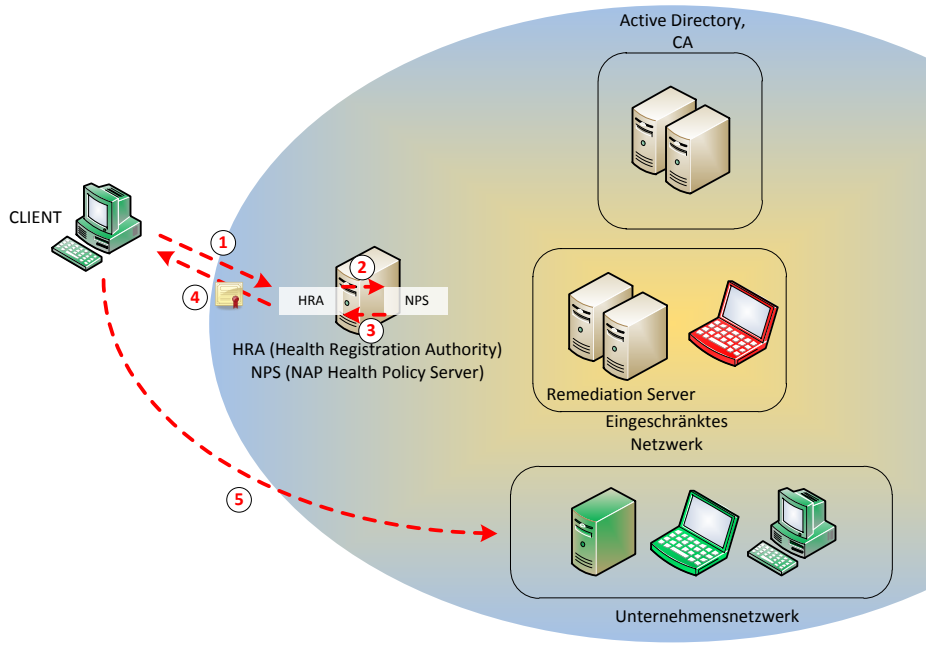


Abbildung 3.4: Verbinden in das Unternehmensnetz mit einem policykonformen Client

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.
4. Die HRA übermittelt dem NAP Client ein gültiges Health Certificate.
5. 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.

3.4.2 Nicht konformer Client

Ein nicht konformer Client besitzt für den Verbindungsaufbau folgenden Ablauf (Abb. 3.5):

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).

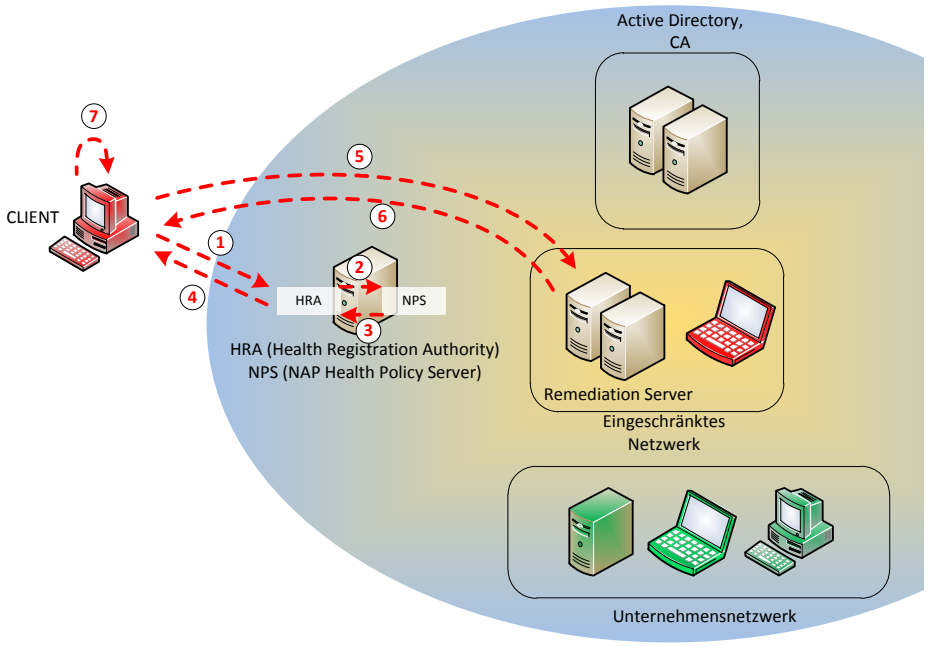


Abbildung 3.5: Verbinden in das Unternehmensnetz mit einem *nicht* policykonformen Client

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.

4. Weil der Gesundheitsstatus des Clients nicht konform ist, sendet die HRA ein SSoHR zum NAP Client. Ein Health Certificate wird nicht ausgeliefert. Es können deshalb keine Verbindungen zum sicheren Netzwerk durchgeführt werden; dem Client fehlt das für die IPsec-Verschlüsselung benötigte Health-Certificate. 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.

   Nachdem der NAP Client seinen Status aktualisiert hat, beginnt das Vorgehen wieder von vorne. Der Client erhält erst Zugang zum Netzwerk, wenn er in einem konformen Zustand ist. In diesem Fall ist mit Kapitel 3.4.1 fortzufahren, ansonsten wiederholt sich der Ablauf des nicht konformen Clienten nochmals. Es kann aber auch sein, dass der nicht policykonforme Zustand nicht durch den Remediation Server

gelöst werden kann. So muss ein manueller Eingriff durch den Systemadministrator erfolgen.

Anmerkung zur Testumgebung mit IPsec: Anders als bei der Enforcement-Methode VPN oder 802.1X ist in IPsec ein externer Policy Enforcement Point (PEP) nicht erforderlich. Denn ein Client kann, solange er kein Healthzertifikat besitzt, keine gesicherte Verbindung mit anderen Computern aufbauen.

3.5 TCG Software Stack (TSS)

3.5.1 TSS-Grundlagen

Beim TCG Software Stack (TSS) handelt es sich um eine Software-Schnittstelle, über die Programme auf die Funktionen des TPMs zugreifen können. Sie besteht aus verschiedenen Komponenten, welche unterschiedlichste Aufgaben auf verschiedensten Ebenen (Abb. 3.6) übernehmen. Zu diesen Komponenten, resp. Modulen, gehören Geräte-Treiber, Kernfunktionalitäten sowie allgemeine Trusted Platform Dienste. TSS verfolgt das Ziel, TPM-unterstützte Applikationen zu schreiben, die unabhängig von der Plattform und vom Betriebssystem sind. Zudem sorgt das TSS dafür, dass der Zugriff gemultiplext wird, da das TPM gleichzeitig nur eine Verbindung besitzen kann. Darüber bewirkt das TSS eine Reduktion der Komplexität im Umgang mit dem TPM.

In Abbildung 3.6 ist veranschaulicht, wie das TSS aufgebaut ist. In der untersten Ebene befinden sich die Module des Kernel-Modes. Der Kernel-Mode ist auf derselben Ebene wie die Kernkomponenten des Betriebssystems. Der Zugriff auf das TPM erfolgt von dieser Ebene aus direkt. Im Kernel-Mode dürfen Programme sämtliche Befehle ausführen und besitzen vollständige Zugriffe auf alle Resourcen (beispielsweise kann auf Memory-Adressen ohne Limitation zugegriffen werden). Im Kernel-Mode dürfen deshalb nur sichere Programme ausgeführt werden, weil sonst die Sicherheit des gesamten Systems eingeschränkt würde. Im User-Mode hingegen besitzen die Programme nur einen eingeschränkten Zugriff auf die Ressourcen.

TrouSerS ist eine Umsetzung des TSS-Standards. Es wurde ursprünglich für Linux entwickelt. Die Portierung für Windows hat zusätzlich eine weitere Schicht als das TSS, weil der Zugriff auf das TPM über die TBS erfolgen (gelber Layer in Abb. 3.6).

Auf den höhergelegenen Ebenen, welche sich im User Mode befinden, werden unter Windows Dienste und Schnittstellen, bzw. unter Linux Daemons, verfügbar gemacht, damit die Anwendungsprogramme die TPM-Funktionalitäten nutzen können.

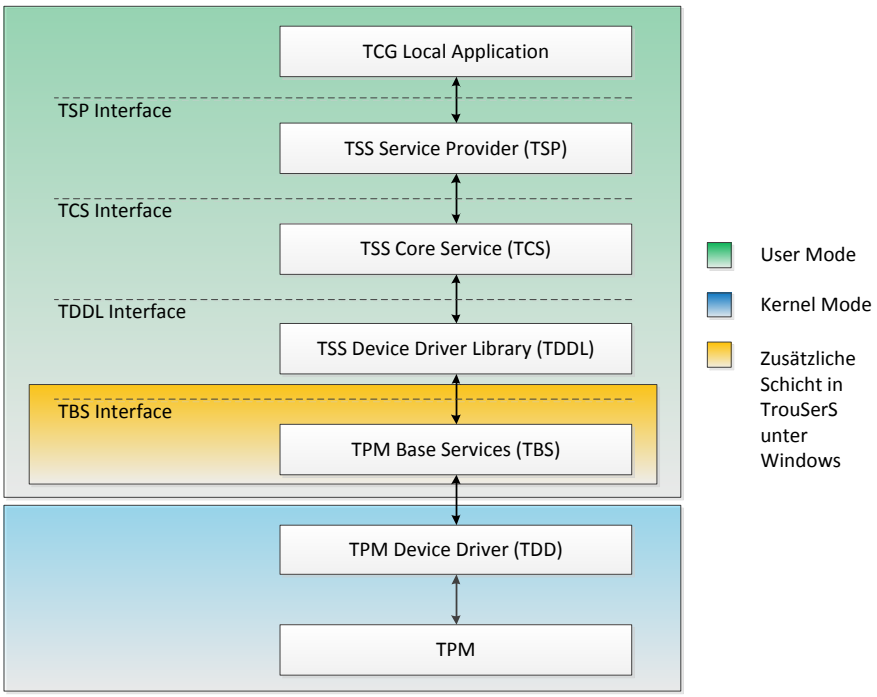


Abbildung 3.6: Der TCG Software Stack

Das TSS besteht hauptsächlich aus drei Komponenten: dem TSS Service Provider (TSP), den TSS Core Services (TCS) und der TSS Device Driver Library (TDDL). Die entsprechenden Kommunikationsinterfaces werden als TSP interface (TSPi), TCS interface (TCSi) und TDDL interface (TDDLi) bezeichnet.

TDD Der TPM Device Driver (TDD) wird von den TPM-Herstellern zur Ansteuerung des TPMs ausgeliefert und ist deshalb nicht Bestandteil des TSS. Er wird über das TSS Device Driver Library (TDDL) für die obliegende Software verfügbar gemacht.

TDDL TSS Device Driver Library (TDDL) stellt eine einheitliche Schnittstelle zu verschiedenen TPM-Treiberimplementationen zur Verfügung; zudem stellt das TDDL den Übergang zwischen Kernel und User Mode dar. In dieser Ebene werden Funktionen wie das Absenden oder Abbrechen von TPM-Kommandos (Transmit() / Cancel()) ausgeführt.

TCS Pro TPM existiert genau eine Instanz der TSS Core Services (TCS). Aus diesem Grund ist es wichtig, dass die TCS-Instanz die Zugriffe der einzelnen Anwenderprogramme multiplexen. Diese Ebene ist ausserdem zuständig für Verwaltungsaufgaben wie die Synchronisation und die Speicherverwaltung, aber auch komplexeren TPM-Funktionalitäten wie das Schlüsselmanagement, wo der Schutz der Schlüssel vollzogen wird.

TSP Dieses Interface bietet dem Programm High-Levelfunktionalitäten sowie zusätzliche Hilfsfunktionalitäten wie Hashing- und Signierfunktionalitäten. Applikationen greifen üblicherweise direkt über das Interface des TSS Service Provider (TSP) auf das TPM zu. Jede Applikation erhält ihr eigenes (virtuelles) TPM-Objekt, welches dann in der TCS-Ebene auf das physikalische TPM multiplext wird. Die Kapselung der verschiedenen Schichten erhöht die Sicherheit, da sich so der Entwickler nicht um den Umgang mit kritischen Methoden kümmern muss und vereinfacht den Umgang mit den TPM-Funktionalitäten, da die Funktionen direkt verwendet werden können und nicht selber umgesetzt werden müssen.

3.6 Trusted Third Party - PrivacyCA.com

Die Trusted Third Party stellt sicher, dass AIKs durch konforme Trusted-Computing-Plattformen erzeugt werden. Die durch die Clientcomputer generierten AIK-Schlüssel werden aus diesem Grund nach der Erzeugung durch die Trusted Third Party (in diesem Fall PrivacyCA.com) bestätigt. Diese Bestätigung erfolgt in Form eines AIK-Zertifikats.

3.6.1 Zertifikate

Durch PrivacyCA.com werden folgende Zertifikattypen unterstützt:

Endorsement Zertifikat Das Endorsement Zertifikat bescheinigt, dass das TPM durch ein autorisiertes Unternehmen hergestellt wurde. Ein solches Zertifikat besagt, dass durch den AIK Key erstellte Signaturen und Quote-Befehle die Zustände des TPM-Systems wiedergeben und der AIK durch ein gültiges TPM administriert wird.

Plattform Zertifikat Dieses Zertifikat bestätigt, dass das TPM durch eine vertrauenswürdige Computer-Umgebung betrieben wird, welche der TCG-Spezifikation entspricht. Das TPM selber muss ebenfalls konform sein. Dieses Zertifikat wird durch den Computerhersteller ausgestellt.

3.6.2 Zertifikatlevel

Die Privacy CA kann drei verschiedene AIK-Zertifikatlevel ausliefern:

Level 0 Diese Zertifikatart ist lediglich für das Testen ausgelegt und beinhaltet keine Validierung der TPM-Client-Umgebung.

Level 1 Für diesen Level der Zertifikate wird das im TPM gespeicherte Endorsement Zertifikat, welches durch den TPM-Hersteller herausgegeben wurde, validiert. Es wird überprüft, ob es von einem vertrauenswürdigen Hersteller ausgestellt wurde.

Level 2 Um dieses Zertifikat zu erhalten, werden das Endorsement Zertifikat sowie das Plattform Zertifikat des TPMs validiert. Diese beiden Zertifikate müssen von vertrauenswürdigen TPM-Herstellern ausgestellt worden sein, damit dieser Level des Zertifikats ausgestellt wird.

Derzeit ist Infineon der einzige TPM-Hersteller, der Endorsement Zertifikate auf den TPMs ausliefert. Aus diesem Grund können zum jetzigen Zeitpunkt lediglich für Infineon-TPMs Level 1-Zertifikate durch die Privacy CA ausgestellt werden.

Level 2-Zertifikate können noch nicht ausgestellt werden, da derzeit kein Hersteller Plattformzertifikate ausstellt.

3.7 Platform Configuration Register (PCR)

Platform Configuration Register (PCR) sind im TPM eingebaute Register, welche SHA-1-Hashwerte zwischenspeichern können. Einige TPM-Funktionalitäten benötigen diese Werte, um nachvollziehen zu können, ob der Zustand des Computers noch immer derselbe ist. Im Zusammenhang mit Attestation werden die Resultate der gemessenen Dateien in den Registern zwischengespeichert. Dadurch, dass die Register im TPM liegen, können sie nachträglich nicht mehr manipuliert werden. Das TPM ist gemäss dem TCG-Standard mit minimal 16 PCRs ausgerüstet.

Um die Register mit Werten füllen zu können, muss sichergestellt sein, dass diese noch keinerlei Informationen enthalten und somit den Ausgangswert besitzen. Um diese Ausgangslage nach Benutzung erreichen zu können, bietet das TPM einen Resetbefehl an. Es ist aber auch möglich, mehrere Messwerte in einem Register zu speichern. Ist bereits ein Hashwert in einem Register gespeichert, so wird der neue Wert an den im PCR Gespeicherten angehängt (konkateniert) und mit der TPM-eigenen SHA-1-Funktion zu einem 20 Byte Hash berechnet und dieser im TPM gespeichert. Dieser so gewonnene Hash ist nun durch mehrere Hashwerte berechnet worden.

Während des Bootvorgangs werden alle Register initialisiert. Dabei wird zwischen zwei verschiedenen Initialisierungswerten unterschieden: Die PCR 1-16 werden mit 0, die höhergelegenen mit -1 (0xFFFF...) initialisiert.

3.8 Attestation

Mit der Attestation kann nachgeprüft werden, ob eine Computerkonfiguration z.B. durch Malware manipuliert wurde. Für das NAP bedeutet dies zu prüfen, ob die Dateien und Bibliotheken, welche für die NAP-Überprüfung der Computer-Policykonfirmität verwendet werden, unverändert sind. Um dies zu erreichen, werden die gemessenen Dateichecksummen mit einer Signatur versandt. Werden nachträglich Veränderungen angebracht, so ist dies später nachvollziehbar, da die Signatur nicht mehr zum veränderten Datenteil passt. Der Server (Verifier) erhält dadurch die Möglichkeit, die übertragenen Messdaten auf die Richtigkeit hin zu prüfen. Die Signatur wird durch das TPM ausgeführt.

3.9 Testumgebung

Die Testumgebung wurde anhand des Microsoft-Dokumentes *Step-by-Step Guide: Demonstrate NAP IPsec Enforcement in a Test Lab[6]* aufgebaut.

3.9.1 Aufbau Netzwerk

In der Testumgebung wird für die Abbildung eines funktionalen NAP ein Netzwerk mit zwei Windows Servern und einem Client gebildet. Für die Zugangskontrolle wird IPsec verwendet. Dieser Aufbau ist in Abbildung 3.7 ersichtlich.

3.9.2 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.

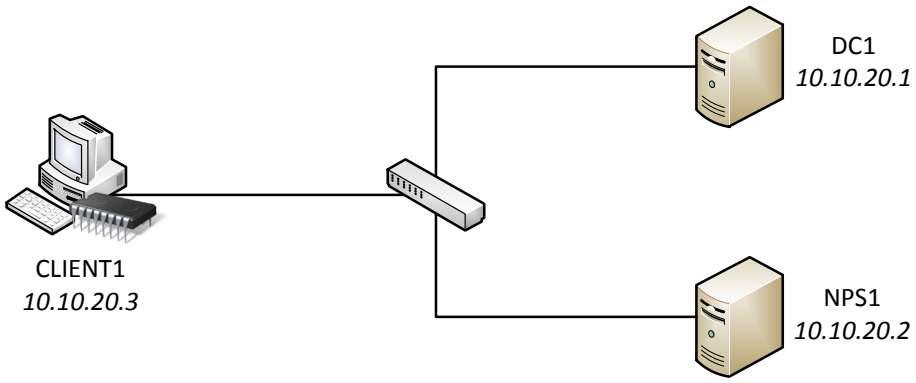


Abbildung 3.7: Netzwerkübersicht Testumgebung

Clients aus dem eingeschränkten Netzwerk können nicht mit den Rechnern im sicheren Netzwerk kommunizieren (und umgekehrt). Wie in Abbildung 3.8 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.

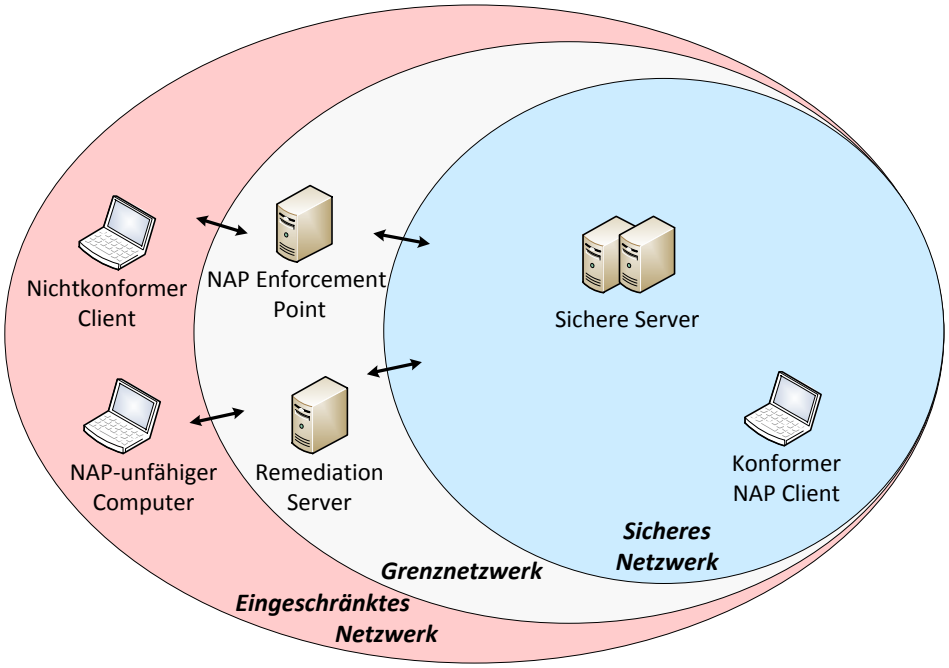


Abbildung 3.8: Übersicht virtuelle Netzwerke des NAP [6]

Sicheres Netzwerk

Computer, welche die Sicherheitsanforderungen erfüllen, erhalten ein NAP Health Certificate und werden für das sichere Netzwerk zugelassen. Dieses 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 Gesundheitscheck nicht durchführen oder den Netzwerkrichtlinien nicht genügen, werden diesem virtuellen Netzwerk zugeordnet. Gastcomputer (und andere nicht-NAP-kompatible Rechner) wiederfinden sich ebenfalls in diesem virtuellen Netzwerk.

3.9.3 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

3.9.4 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 SHAs/SHVs 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 SHAs dar. Für jeden SHA des Clients existiert ein entsprechender SHV auf dem NAP-Server. Der NPS wertet mit Hilfe des SHVs die gesammelten Client-Informationen des SHAs 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 die Angabe, 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

Anforderungsspezifikation

4.1 Einführung

4.1.1 Zweck

Dieses Kapitel dient zur Spezifizierung der Anforderungen in diesem Projekt und umfasst die funktionale Einbindung der Technologie und umsetzungsrelevante Aspekte. Die Anforderungen, welche an dieses Projekt gestellt werden, werden auf diesem Weg schriftlich festgehalten. Diese Anforderungen sind verbindlich und können während des Projektes nur in gegenseitiger Absprache mit Projektteam und Auftraggeber abgeändert werden.

4.2 Allgemeine Beschreibung

4.2.1 Ergebnisse der Ist-Analyse

Aufbauend auf dem Stand der Semesterarbeit wird das Thema Trusted Network Access Control (TNAC) nochmals aufgearbeitet und die Erkenntnisse daraus weiter vertieft. Als Ausgangslage steht dem Team ein Prototyp zur Verfügung, der sich als SHA und SHV in die NAP-Umgebung integriert. Zudem kann er mit dem TPM kommunizieren und auf erste TPM-Funktionalitäten zugreifen. Das erworbene Wissen, wie das API des NAP angesprochen werden muss, kann nun verwendet werden, um neue Funktionalitäten in das NAP-System integrieren zu können.

Neben dem Prototyp steht ebenfalls eine voll funktionsfähige NAP-Testumgebung zur Verfügung.

4.2.2 Ziel

Mit diesem Projekt wird das Ziel verfolgt, das Microsoft Network Access Protection (NAP) zusätzlich mit einem Trusted Platform Module (TPM)-Chip abzusichern. Damit wird ein bisher noch immer ungelöstes Problem angegangen: das Lying Endpoint Problem. Beim NAP muss der Client nämlich selbständig dem Server mitteilen, ob sich Schadprogramme auf seiner Plattform befinden, resp. ob eine sicherheitskritische Konfiguration verändert wurde (Firewall-Einstellungen, automatische Updates, Anti-Virensoftware). Das bedeutet, dass der Server dem Client vertrauen muss. Eine Schadsoftware kann nun die Programmteile, welche für die Messung zuständig sind, austauschen. So können die Mitteilungsinformationen gefälscht werden. Dem Server wird dann vorgegaukelt, dass keine Infizierung vorliegt. Mit Hilfe des TPMs hat der Validierungsserver zusätzliche Möglichkeiten, eine solche Messungsmanipulation zu entlarven.

4.3 Produktperspektive

4.3.1 Systemschnittstellen

Diese Software wird für die neust erhältlichen Windows Softwareversionen entwickelt. Clientseitig wird ein Windows 7, serverseitig ein Windows Server 2008 R2 vorausgesetzt. Zusätzlich muss auf dem Clientcomputersystem ein TPM vorhanden sein, das nicht anderweitig verwendet wird. Darüber hinaus ist während der SHA-Initialisierung ein Zugang zum Internet auf dem Client notwendig, damit die Anbindung der Third Party privacyca.com erfolgen kann. Um die Software zu verwenden, muss eine NAP Infrastruktur existieren. In Abbildung 4.1 wird der Zusammenhang in einem Diagramm dargestellt.

4.3.2 Benutzerschnittstelle

Als Benutzerschnittstelle wird das Statusfenster des bereits in Windows integrierten *napstat* (Abb. 4.2) verwendet. Eine zusätzliche Benutzerschnittstelle, bei welcher eine Benutzerinteraktion möglich wird, wird explizit ausgeklammert, da das NAP-System ohne Nutzerinteraktion funktionieren soll.

4.3.3 Hardwareschnittstellen

Die Schnittstelle zur Computerhardware, wie auch zum TPM, wird durch das Betriebssystem bereitgestellt. Die Anforderungen an die Hardware sind in Tabelle 4.1 aufgeführt.

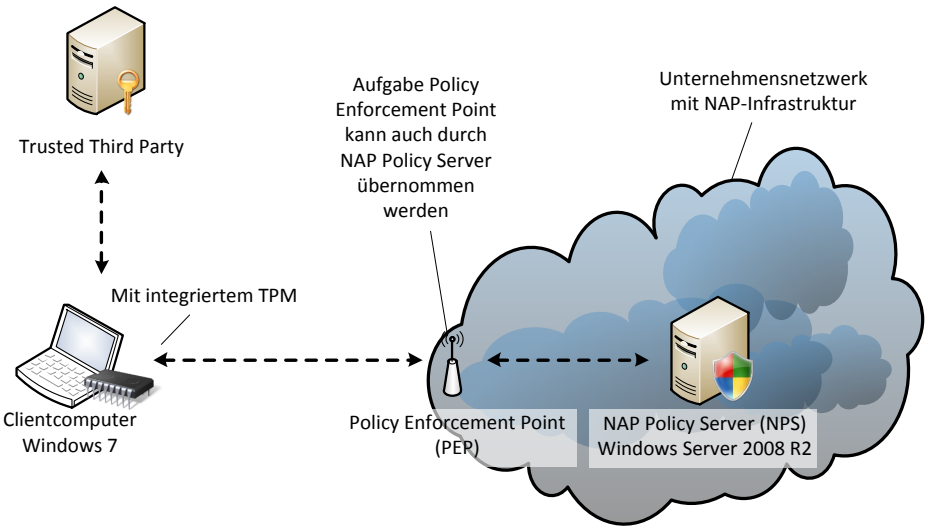


Abbildung 4.1: Systemschnittstellen

System	Anforderungen
Clientcomputer	TPM Version 1.2 gemäss TCG-Spezifikation Internetanbindung, Netzwerkkarte
Servercomputer	Netzwerkkarte

Tabelle 4.1: Anforderungen an die Hardware

4.3.4 Softwareschnittstellen

Das Softwaresystem wird für die Windows-Betriebssysteme Windows 7 und Windows Server 2008 R2 entwickelt. Clientseitig wird das TPM über die Schnittstelle TPM Base Services (TBS) angesprochen.

4.3.5 Kommunikationsschnittstellen

Die Kommunikation zwischen dem Server und dem Client erfolgt über das bereits von NAP zur Verfügung gestellte Statement of Health (SoH)-Protokoll. Zusätzliche Informationen, welche durch den Client zum Server versendet werden, werden im Bereich Vendor Specific Data des SoH-Protokolls im TLV-Format übertragen.

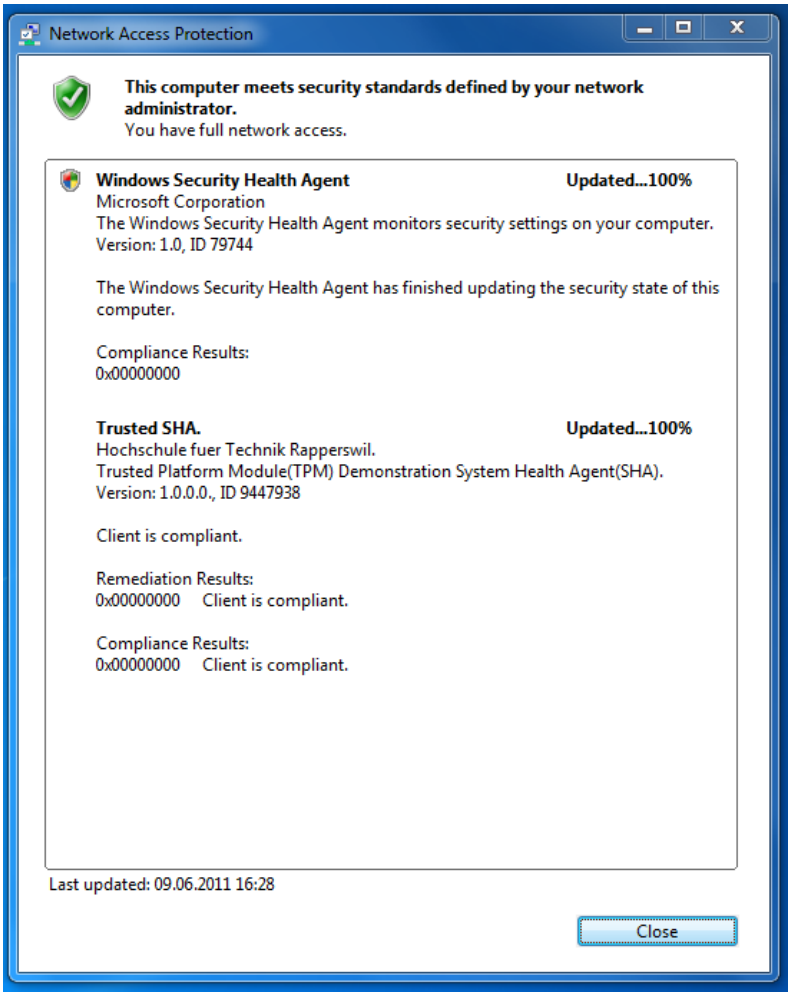


Abbildung 4.2: Programm *napstat* für die Ausgabe von NAP-Stati

4.4 Funktionale Anforderungen

4.4.1 FAnf01: SHA / SHV Paar

Die anzustrebende Lösung muss als System Health Agent (SHA) und System Health Validator (SHV) auf einem bestehenden NAP-System einbindbar sein. Auf Clientseite lässt sich das Programm als System Health Agent (SHA) und auf Serverseite System Health Validator (SHV) einbinden.

4.4.2 FAnf02: Clientseitige Manipulationsdetektion

Es kann nachvollzogen werden, ob beliebige Dateien (wie beispielsweise die für den Windows SHA benötigt werden) auf dem Client nicht manipuliert

wurden. Das bedeutet, dass diese weder verändert noch ersetzt wurden. Somit kann nachgewiesen werden, dass das NAP nicht umgangen wird.

4.4.3 FAnf03: Attestation mit TPM-Chip

Die auf dem Client gemessenen Werte werden zusätzlich vom TPM signiert, damit eine nachträgliche Manipulation der Messwerte auf Serverseite nachgewiesen werden kann. Sobald alle Konfirmitätswerte und Checksummen der ausgeführten Dateien auf Clientseite gelesen wurden, werden die Werte mit dem auf dem TPM gespeicherten Schlüssel signiert. Auf der Verifizierseite werden die erhaltenen Informationen ausgewertet. Fehlt die Signatur oder sind Veränderungen ersichtlich, so darf der Server das erhaltene Paket nicht als gültig werten, auch wenn die NAP-Messwerte korrekt sind.

4.4.4 FAnf04: Attestation-Vorgang mit dem quote-Befehl

Die Dateifingerprints der Messdateien sollen aus den TPM internen Register (PCR) mit quote ausgelesen und mit dem AIK signiert werden.

4.4.5 FAnf05: Attestation-Vorgang mit dem quote2-Befehl

Um die Messwerte aus dem TPM auszulesen soll der quote Befehl mit dem neueren Befehl quote2 Befehl abgelöst werden. Dieser liefert dem Aufrufenden mehr Informationen über die PCR zurück. Dadurch kann ein besserer Überblick über den TPM-Status erlangt werden.

4.4.6 FAnf06: Attestation mit Attestation Identity Key (AIK)

Die Signierung auf Seite des Clients soll durch das TPM mit einem AIK-Schlüssel erfolgen. Der Attestation Identity Key (AIK)-Schlüssel ist dazu vorgesehen, dass mit ihm Remote Attestation Prozeduren durchgeführt werden können.

4.4.7 FAnf07: Trusted by PrivacyCA.com / AIK-Zertifikat

Der verwendete Signierschlüssel (AIK) wird zusätzlich von der Zertifizierungsstelle PrivacyCA.com beglaubigt. Der Client bezieht ein Level 0-Zertifikat von dieser Zertifizierungsstelle und sendet es über das NAP bei jeder Messwertübertragung zum Server. Damit soll sichergestellt werden, dass nur konforme TPMs gültige AIKs erstellen können. Ist das TPM konform,

so wird zum AIK-Schlüssel durch die Trusted Third Party (PrivacyCA) ein AIK-Zertifikat ausgeliefert.

4.4.8 FAnf08: Level 1 AIK-Zertifikat (PrivacyCA.com)

In Verbindung mit der PrivacyCA.com wird grundsätzlich mit Level 0 Zertifikaten gearbeitet. Neuere Infineon TPMs unterstützen Level 1 Zertifikate, welche im Vergleich zu Level 0 Zertifikaten als sicherer gelten, da der TPM-Hersteller ein Endorsement Certificate auf dem TPM bereitstellt. Für Infineon-TPM soll deshalb eine Unterstützung von Level 1 Zertifikaten möglich sein.

4.4.9 FAnf09: Paralleles Ausführen des SHV

Auf dem Server soll der SHV fähig sein, mehrere Clientanfragen gleichzeitig abzufertigen. Das NAP-System sieht bereits vor, dass mehrere Threads für Validierungen gestartet werden, damit Anfragen parallel beantwortet werden können. Aus diesem Grund müssen innerhalb des SHV die Funktionalitäten für eine parallele Ausführung ausgelegt werden.

4.4.10 FAnf10: Fehlerlogging in Protokolldatei

Da der gesamte NAP-Ablauf ohne Interaktion mit dem Benutzer resp. ohne Ausgabe der derzeit ausgeführten Tätigkeit geschieht, sollen die Informationen in einer Datei abgelegt werden. Ebenfalls werden hier Fehler notiert, welche in der Abwicklung mit dem TPM auftraten.

4.4.11 FAnf11: Fehlerlogging in Event Viewer

Die Info- und Fehlerausgaben (wie auch bereits bei der Anforderung 4.4.10 beschrieben) sollen so abgelegt werden, dass sie im EventViewer angezeigt werden können. Dies ergibt eine bessere Integration der Informations- und Fehlerabfrage im Windows Betriebssystem.

4.4.12 FAnf12: Platform Configuration Register (PCR) sperren

Die PCR sollen lediglich durch den SHA veränderbar sein. Dies bedeutet, dass Dritt-Programme die PCR-Werte nicht mit TPM_Extend erweitern dürfen.

4.4.13 FAnf13: Überprüfung der Dateien im Arbeitsspeicher

Ziel ist es, die Überprüfung weiter zu verbessern, denn bis anhin werden die Dateien auf der Festplatte auf Ihre Unveränderheit überprüft. Einen Schritt weiter geht der Ansatz, dass die Dateien direkt im Arbeitsspeicher überprüft werden, nachdem sie dorthin geladen wurden.

4.4.14 FAnf14: Dynamisches Laden der SHAs auf dem Client überprüfen

Das dynamische Laden der DLL soll auf dem Client überprüft werden. Es soll verhindert werden, dass manipulierte DLLs geladen werden. Ebenso ist zu verifizieren, ob der DLL-Loader nicht manipuliert wurde.

4.5 Priorisierung der funktionalen Anforderungen

Die zuvor aufgeführten funktionalen Anforderungen wurden in **Must** und **Optional** unterteilt.

Must

- FAnf01: SHA / SHV Paar
- FAnf02: Clientseitige Manipulationsdetektion
- FAnf03: Attestation mit TPM-Chip
- FAnf04: Attestation-Vorgang mit dem quote-Befehl
- FAnf06: Attestation mit Attestation Identity Key (AIK)
- FAnf07: Trusted by PrivacyCA.com / AIK-Zertifikat
- FAnf10: Fehlerlogging in Protokolldatei
- FAnf14: Dynamisches Laden der SHAs auf dem Client überprüfen

Optional

- FAnf05: Attestation-Vorgang mit dem quote2-Befehl
- FAnf08: Level 1 AIK-Zertifikat (PrivacyCA.com)
- FAnf09: Paralleles Ausführen des SHV
- FAnf11: Fehlerlogging in Event Viewer

- FAnf12: Platform Configuration Register (PCR) sperren
- FAnf13: Überprüfung der Dateien im Arbeitsspeicher

4.6 Nichtfunktionale Anforderungen

4.6.1 NfAnf01: Zuverlässigkeit

Das zu entwickelnde Produkt muss stets lauffähig sein. Ein Nichtfunktionieren dieses SHAs hat zur Folge, dass die Clients nicht mehr mit den internen Servern kommunizieren können, was für die Firma hohe finanzielle Schäden zur Folge hätte.

4.6.2 NfAnf02: Sicherheitsanforderungen

Da es sich um eine Sicherheitssoftware handelt, soll während des Entwicklungsprozesses speziell darauf geachtet werden, dass keine Manipulationen durch Dritte möglich sind. Ebenso ist es wichtig, keine zusätzlichen Sicherheitslöcher zu produzieren. Aus diesem Grund wird der Quellcode durch beide Teammitglieder durchgesehen, bis er als final eingestuft wird. Zudem sind in den funktionalen Anforderungen Ziele angegeben, welche die Erreichung dieses Ziels unterstützen.

4.6.3 NfAnf03: Korrektheit

Der vom Client übermittelte Wert muss korrekt sein. Es ist eine der Hauptaufgaben dieser Software, dies zu erreichen. Die Werte, die der Server vom Client erhält, müssen manipulationssicher sein. Treten nämlich Manipulationen auf, so muss der Server dies erkennen können.

4.6.4 NfAnf04: Automatisierter Ablauf

Es ist wichtig, dass die neuen Funktionalitäten (wie bisher) ohne Benutzer-Interaktion ablaufen. Das Einzige, was der Computerbediener sehen darf, ist die Rückmeldung, wenn ein NAP-Fehler auftritt und der Client somit nicht policykonform ist. Er selber darf aber keine Funktionen auf dem TPM ausführen respektive die Konfiguration der Software nicht verändern.

4.7 Use Cases

Nachfolgend wird eine Übersicht der Use Cases gegeben (Abb. 4.3). Als Akteure ist der Mitarbeiter sowie der Systemadministrator involviert.

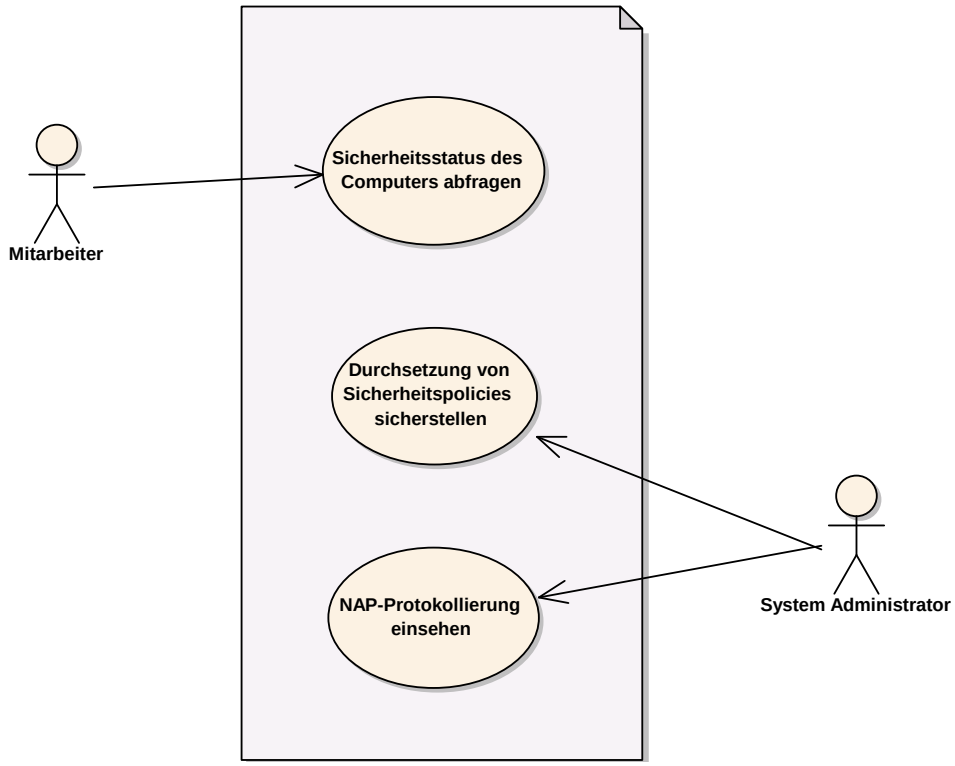


Abbildung 4.3: Use Cases

4.7.1 Mitarbeiter

UC M01: Sicherheitsstatus des Computers abfragen

Der Mitarbeiter erhält die Möglichkeit, sich über den Sicherheitsstatus seines Computers zu informieren.

1. Im Fehlerfall erscheint eine Statusmeldung, welche über die Verseuchung des Computers berichtet.
2. Der Benutzer erhält nun die Möglichkeit, diese Statusmeldung zu öffnen und weitere Informationen zum Fehler anzuzeigen.

4.7.2 Systemadministrator

UC S01: Durchsetzung von Sicherheitspolicies sicherstellen

Der Systemadministrator kann die Sicherheitspolicies, welche in seinem Unternehmen gelten, durchsetzen. Die Systemdateien des Windows System Health Agent (SHA) werden überwacht. Wird eine Manipulation an diesem festgestellt, so wird der Computer als verseucht angesehen.

UC S02: NAP-Protokollierung einsehen

Serverseitig kann die Protokollierung des SHV ausgelesen werden, damit nachvollzogen werden kann, weshalb Fehler auftreten.

1. Der Systemadministrator öffnet die Protokolldatei.
2. In diesem kann er nun die Unregelmässigkeiten und Fehler einsehen.

4.8 Benutzercharakteristik

Die Zielgruppe sind Unternehmen, welche bereits ein NAP-System im Einsatz haben. Da die Applikation ständig im Hintergrund läuft und für den Anwender keinerlei Unterschiede in der Bedienung bemerkbar sind, muss und kann die Applikation nicht für eine bestimmte Nutzergruppe optimiert werden. Ausserdem erfolgt keine Interaktion mit dem Benutzer.

Es wird davon ausgegangen, dass der Netzwerkadministrator, der das System einführt und wartet, über fundiertes technisches Fachwissen (u.a. in der Technologie NAP) verfügt und mit der technischen Dokumentation bzw. dem Handbuch die Applikation in sein Netzwerk einbinden kann.

4.9 Einschränkungen

4.9.1 E01: Performance in grossen Netzwerken

Die zu entwickelnde Software wird mit einer einfachen NAP-Testumgebung getestet. Während der Entwicklungszeit können keine Tests auf mittelgrossen (mehr als 10 Clients) oder grossen Unternehmensnetzwerken (mehr als 100 Clients) gemacht werden. Aus diesem Grund können keine Angaben zur Performance in grösseren Netzwerken gemacht werden.

4.9.2 E02: Hardwarebeschränkungen

Auf die Hardwarebeschränkungen wird in Kapitel 4.3.3 auf Seite 34 eingegangen.

4.9.3 E03: Serverseitige Sicherheit

In diesem Projekt wird eine Erhöhung der Sicherheit auf der Clientseite angestrebt. Die Server befinden sich im Unternehmensnetzwerk und geniessen durch die Unternehmensfirewall einen erhöhten Schutz. Es darf aber nicht ausser acht gelassen werden, dass ebenfalls der Server angegriffen werden kann. Im Zusammenhang mit NAP kann das bedeuten, dass die Clients unabhängig von Ihrem Gesundheitsstatus in das sichere Netzwerk zugelassen werden.

Die Sicherheitserhöhung der Server (beispielsweise durch eine TPM-Integration) ist im Rahmen dieser Bachelorarbeit aus zeitlichen Gründen nicht vorgesehen.

4.9.4 E04: Windows Vista und Windows Server 2008

Die in diesem Projekt verwendeten Technologien sind ab Windows Vista und ab Windows Server 2008 lauffähig. Die Entwicklung erfolgt aber explizit auf Windows 7- und Windows Server 2008 R2-Systemen. Auf den erst genannten Betriebssystemen werden keine Tests durchgeführt. Deshalb gibt es für jene Systeme keine Funktionsgarantie.

4.10 Lizenzanforderungen

4.10.1 Urheberrecht

Die Urheberrechte stehen den beiden Studenten Wolfgang Altenhofer und Lukas Studer zu.

4.10.2 Verwendung

Die Ergebnisse der Arbeit dürfen durch die beiden Studenten und die HSR nach Abschluss verwendet und weiterentwickelt werden. Dritten ist es in Absprache mit der HSR und den Studenten erlaubt, das Projekt weiterzuentwickeln. Im neuen Projekt muss allerdings eine Erwähnung der beiden Urheber erfolgen.

Kapitel 5

Analyse

In diesem Kapitel werden das Aufgabengebiet sowie die Schwerpunkte detaillierter betrachtet.

5.1 Trusted NAP Funktionsweise (FAnf01, 02)

Für ein besseres Verständnis wird aufgezeigt, weshalb Trusted NAP durch Überprüfung von Dateien eine zusätzliche Sicherheit bieten kann. In diesem Beispiel werden die Dateien des Windows NAP geprüft. Es ist aber mit entsprechender Konfiguration auch möglich, andere Dateien als die von NAP zu überprüfen.

In Abbildung 5.1 wird aufgezeigt, wie die Überprüfung eines infizierten NAP-Systems funktioniert. Dazu sind folgende Schritte notwendig:

1. Der Trusted SHA wird aufgefordert, seine Messungen zu tätigen und das Resultat zurückzumelden.
2. Dieser System Health Agent (SHA) misst nun relevante Dateien, welche für das Windows SHA benötigt werden. Die vollständige Liste der zu überprüften Dateien ist in Kapitel 5.2.2 ersichtlich. Die so gesammelten Dateifingerprints werden in die PCR des TPM geschrieben und durch dieses signiert.
3. Nach Abschluss der Berechnung werden die Resultate inkl. Signatur an den Agenten auf dem Notebook zurückgemeldet.
4. Nun wird der Windows SHA aufgefordert, seine Messungen zu tätigen und die Resultate zurückzumelden.
5. Da der Windows SHA durch eine manipulierte Version ausgewechselt wurde, werden nicht die tatsächlichen Werte des Antivirus, der Fire-

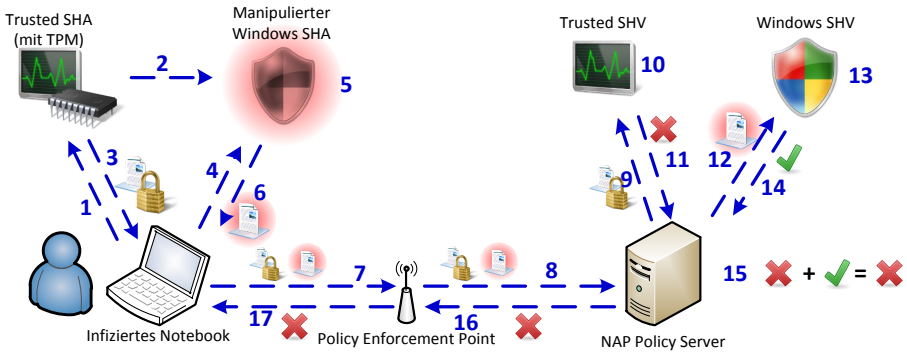


Abbildung 5.1: Ablauf Trusted NAP

wall und der automatischen Updates-Stati berechnet, sondern gefälschte Werte durch den manipulierten Windows SHA zurückgegeben. In diesem Fall: Firewall aktiv, Antivirenschutz aktuell und automatische Updates aktiviert.

6. Die gemessenen Resultate werden dem Agenten zurückgegeben.
7. Der Windows Agent sendet die nun erhaltenen Messresultate (die des Trusted SHA und die des manipulierten Windows SHA) an den Policy Enforcement Point (PEP).
8. Der PEP leitet die NAP Challenge an den Network Policy Server (NPS) weiter.
9. Die soeben erhaltenen Daten werden nun mit den entsprechenden System Health Validator (SHV) geprüft. Als Erstes wird die Angabe des Trusted SHA durch den Trusted SHV geprüft.
10. Die Prüfung ergibt, dass die gemessenen Resultate von den im Validator hinterlegten Prüfsummen abweicht. Er gibt also dem NPS die Mitteilung zurück, dass der Windows SHA auf der Clientseite ausgetauscht ist.
11. Die Meldung über das Nichtbestehen der Prüfung wird an den NPS gesendet.
12. Als Nächstes werden die Werte des Windows SHA durch den Windows SHV überprüft.
13. Die Überprüfung ergibt, dass die Messwerte des Clients korrekt sind und sich der Clientcomputer in einem gesunden Zustand befindet (was aber nicht stimmt).
14. Der Status, dass die Messwerte des Windows SHA gültig sind, wird dem NPS mitgeteilt.

15. Der NPS entscheidet nun gemäss seinen hinterlegten Policies, ob die zurückgegebenen Messwerte einen Zugriff auf das interne sichere Netzwerk zulassen. Der NPS ist so eingerichtet, dass alle SHVs einen positiven Rückgabewert liefern müssen, damit ein Client in das interne Netzwerk gelangen kann. Da dies nicht der Fall ist wird nun dem PEP mitgeteilt, dass der Client lediglich Zugriff auf das Remediation Netzwerk-VLAN erhält.
16. Der NPS meldet dem PEP seinen Entscheid und das Vorgehen, wie sich der PEP gegenüber dem Notebook verhalten muss.
17. Der PEP meldet das Resultat dem Notebook und verbindet ihn in das Remediation-VLAN.
18. Das Notebook hat nun die Möglichkeit, mit diesem eingeschränkten Netzwerk zu kommunizieren.

5.2 TPM-Integration

Folgend soll analysiert werden, wie das TPM in den NAP-Prozess integriert werden kann und mit welchen Schwierigkeiten zu rechnen ist. In Abbildung 5.2 möchte ein Client eine Verbindung in das NAP-Unternehmensnetzwerk herstellen. Dazu muss er seine aktiven SHA dazu auffordern, ihre Messungen zu tätigen. Anschliessend sendet der Client die Messresultate via den PEP an den Server (1). Dieser wertet die Anfrage aus und überprüft, ob die Messwerte gemäss seinen Policies gültig sind. Daraus resultierend sendet der Server die Rückmeldung an den PEP (2), mit Angabe des Subnetzes, mit welchem der Client aufgrund der Messdateien und der Policyrichtlinien eine Verbindung aufbauen darf. Anschliessend erhält der Client durch den PEP Zugriff auf das entsprechende Subnetz (Secure oder Remediation) (3).

5.2.1 Kommunikation mit dem TPM (FAnf02)

Der Zugriff auf das TPM kann über verschiedene Wege erfolgen. Grundsätzlich ist es möglich, TPM-Funktionalitäten zu entwickeln, ohne auf ein TSS zurückzugreifen. Es muss aber beachtet werden, dass die Entwicklung in der Low-Level-Ebene TBS (siehe Abbildung 3.6) aufwendig und fehleranfällig ist. Der Einbezug eines TSS vereinfacht die Entwicklung und ermöglicht es, voll und ganz dem funktionalen Ziel der Software nachzukommen.

Für die Programmierung mit einer TSS-Library wurde abgeklärt, welche Bibliotheken unter Windows für die Entwicklung zur Verfügung stehen. Während der Recherche wurden zwei in Frage kommende Produkte gefunden: Einerseits wird von Infineon das *TPM and TSS Software Develop-*

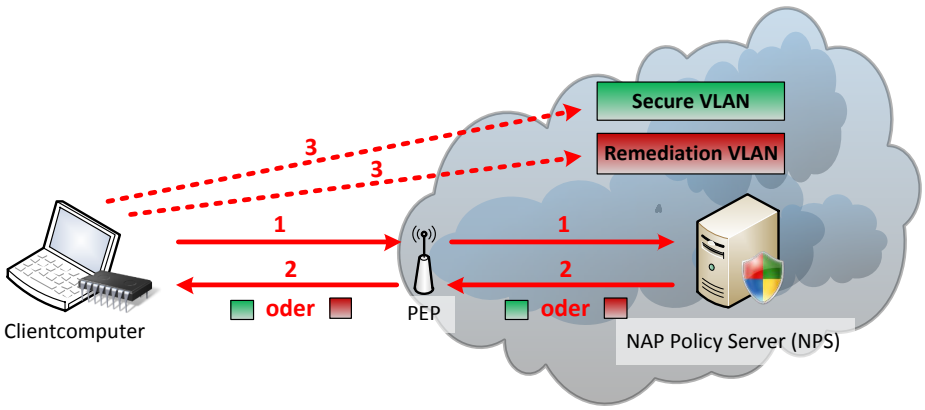


Abbildung 5.2: Übersicht über den Challenge- / Responseablauf

ment Kit (SDK) angeboten, andererseits gibt es eine Portierung des *TrouSerS*-Projektes auf Windows-Rechner (ab Windows Vista).

Das **Infineon SDK** arbeitet mit der Component Object Model (COM)-Schnittstelle des TPMs und ist deshalb für Rechner ab Windows 2000 nutzbar. Es baut grösstenteils auf den Komponenten COM, CryptoAPI und Cryptographic Service Provider auf. Das SDK kann auf schriftliche Anfrage bei Infineon kostenlos bezogen werden.

Das auf Windows portierte **TrouSerS**-Projekt basiert auf den TPM Base Services (TBS) und nicht wie das Infineon auf der COM-Schnittstelle. Die TBS wurden später ins Windows-Betriebssystem aufgenommen. Das TrouSerS-Projekt ist Opensource und noch in Entwicklung. Der Beginn der Portierung von Linux auf die Windows-Plattform begann im Jahre 2010.

Ausschlaggebend für die Auswahl des TSS-Frameworks war die Offenheit des Quellcodes des Produktes. TrouSerS wird unter der Open Source Lizenz vertrieben. Somit ist es möglich nachzuvollziehen, wie die Umsetzung und der Zugriff in tieferen Ebenen funktioniert.

5.2.2 Zusätzliche Übermittlung von Daten (FAnf01, 02)

Der Client muss zusätzliche Daten zum Server übermitteln. NAP bietet die Möglichkeit, zusätzliche Daten dem NAP-Request beizulegen[2]. Die insgesamt verwendbaren 3996 Bytes stellt für die Übermittlung der Daten eine ausreichende Grösse dar.

Die Messung der verschiedenen Dateien werden nacheinander getätigt. Von jeder Datei wird die Checksumme gebildet und dann in einem konfigurier-

baren PCR abgelegt. Als Letztes wird dann noch eine Zufallszahl eingerechnet. Diese Daten werden schliesslich mit quote signiert ausgelesen und dem Server geliefert. So wird es für den Trusted SHV möglich nachzuvollziehen, ob eine Manipulation der SHA-Dateien vorliegt. Die NAP-Dateien von Microsoft werden in den folgenden Verzeichnissen abgelegt:

32-bit Betriebssysteme: C:\Windows\System32\

64-bit Betriebssysteme: C:\Windows\SysWOW64\

Bei den zu prüfenden NAP-Daten handelt es sich um die Folgenden (als Übersicht in Abbildung 5.3 ersichtlich):

mssha.dll Windows SHA, welcher das Windows Security Center überwacht. Dieses umfasst die Windows Firewall, den Antivirenschutz und die automatischen Updates.

msshavmsg.dll Er ist Teil des Windows SHA. Die vollständige Bezeichnung ist Windows Security Health Agent Validator Message. Er ist zuständig für die Darstellung der Informationen im Statusfenster.

qagent.dll Diese Datei repräsentiert den NAP Agent, welcher die obliegenden SHAs aufruft.

napipsec.dll Dies ist der NAP Enforcement Client (in diesem Beispiel für das IPsec Enforcement). Wenn andere Enforcement Methoden gewählt sind, dann werden andere dll-Dateien geladen.

rundll32.exe Der DLL-Loader, der die oben aufgelisteten Dateien ladet und für die Ausführung zuständig ist.

Zusätzlich sind für das Trusted SHA weitere Dateien zu überprüfen. Da TrouSerS eingesetzt wird, müssen ebenfalls die Dateien (Abbildung 5.4), welche für die TPM-Zugriffe verantwortlich sind, überprüft werden:

TrustedSha.exe Die durch dieses Projekt erstellte Erweiterung, die in Zusammenarbeit mit dem TPM überprüft, ob die benötigten Dateien nicht verändert wurden.

TrustedShaInfo.dll Komponente, mit welcher der Trusted SHA Ausgaben im NAP-Informationsfenster tätigen kann.

tcsd.exe Diese Datei repräsentiert den TrouSerS-Service.

tbs.dll Windows Library für den Zugriff auf das TPM.

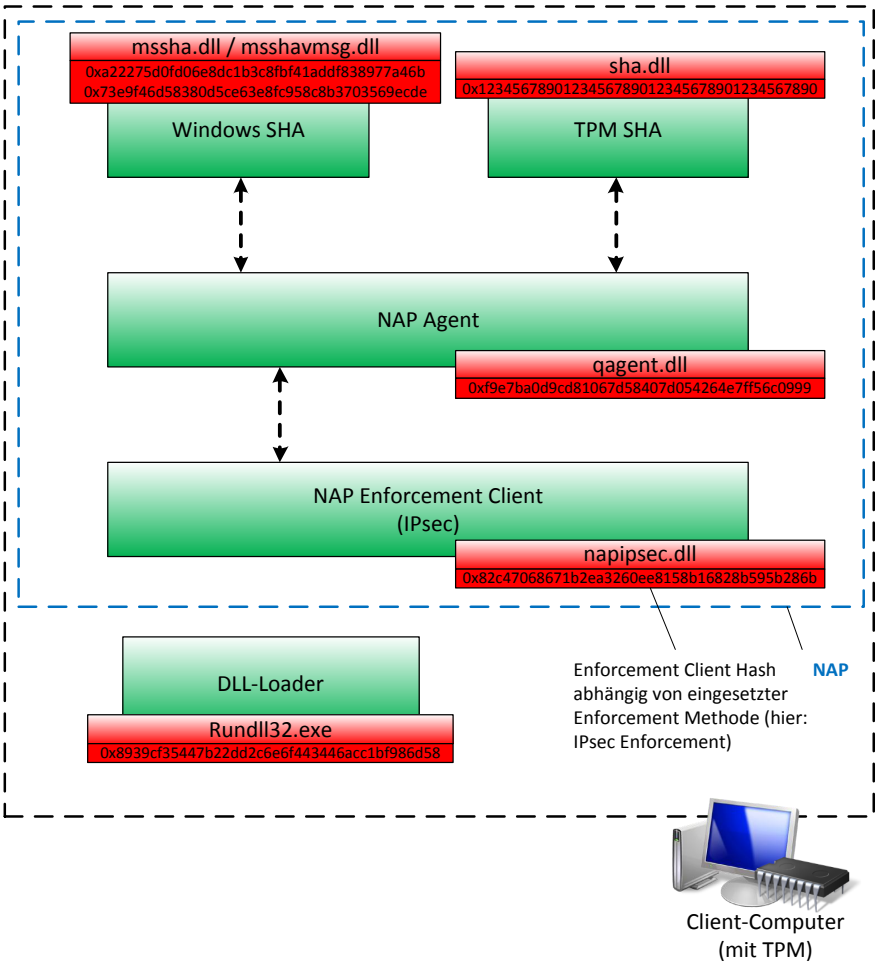


Abbildung 5.3: Messwerte der NAP-Dateien

libcharset1.dll Von TrouSerS benötigte Library.

libiconv2.dll Von TrouSerS benötigte Library.

pthreadVC2.dll Von TrouSerS benötigte Library.

Anmerkung: Die in den Abbildungen 5.3 und 5.4 angegebenen Hashwerte verändern sich mit jeder Versionsveränderung. Aus diesem Grund ist es von nöten die Hashwerte der Original-Dateien bei der Konfiguration neu zu berechnen.

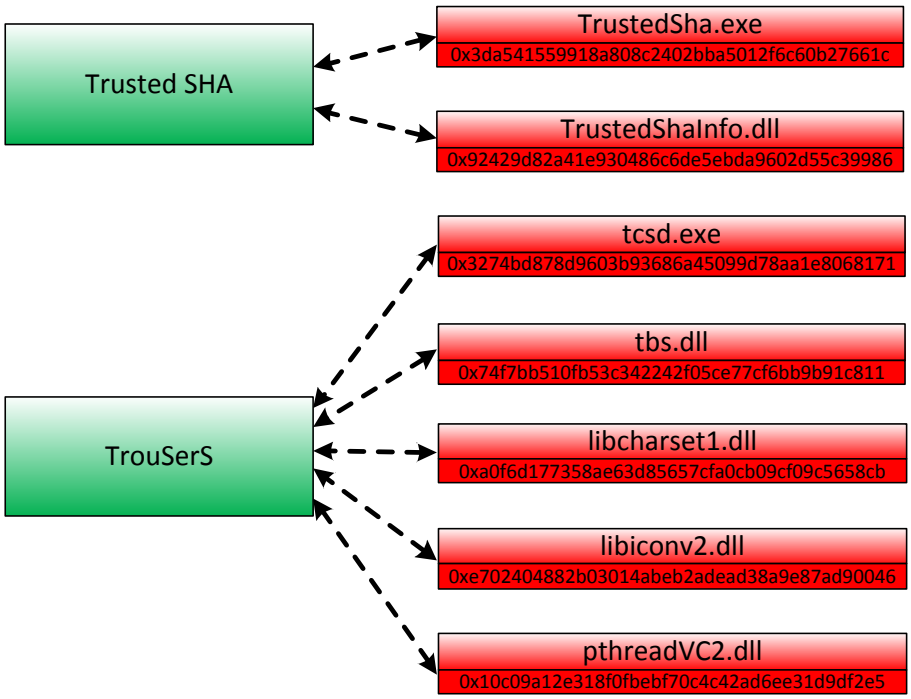


Abbildung 5.4: Messwerte für Trusted SHA und TrouSerS

5.2.3 Challenge-Response mit Nonce (FAnf03, 04, 14)

Eine Zufallszahl (Nonce) wird miteinbezogen, um Replay-Attacken[1] verhindern zu können. Im NAP-Bereich könnten die policykonformen Messwerte ohne Zufallszahl problemlos für spätere Validierungen dem Server gesendet werden. Damit dies nicht möglich ist wird in jeder Kommunikation eine andere Zufallszahl verwendet.

Bei Microsoft NAP ist eine Kommunikation von Server zu Client vor der Übertragung der signierten Messwerten nicht vorgesehen. Darum wurde ähnlich zu Kerberos die Zeit als Challenge gewählt (siehe Abb. 5.5).

5.2.4 Ablauf der Attestation (FAnf03, 04, 06)

Die analysierten Erkenntnisse werden nun in einen vollständigen Ablauf (Abb. 5.6) integriert, wobei dieser im Detail wie folgt aussieht:

[1] In der Replay-Attacke, auch Replay-Angriff genannt, versendet ein Angreifer zuvor aufgezeichnete Daten nochmals zum Server. Somit wird es möglich, eine fremde Identität vorzutäuschen. Um dies zu verhindern, wird in den Datenpaketen eine Zufallszahl mitgesendet.

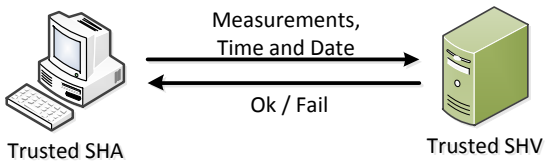


Abbildung 5.5: Challenge-/Response-Vorgang mit Zeit und Datum als Nonce

1. Bei allen benötigten PCR-Registern wird der aktuelle Wert ausgelesen. Dazu wird der Befehl *Tspi_TPM_PcrRead()* verwendet.

2. Der Computer nimmt die Checksummen der Messdateien und legt sie als SHA-1 Hash mit dem Befehl *Tspi_TPM_PcrExtend()* in einem PCR auf dem TPM ab. Dies wird für jede Messdatei und jede Programmbibliothek durchgeführt.

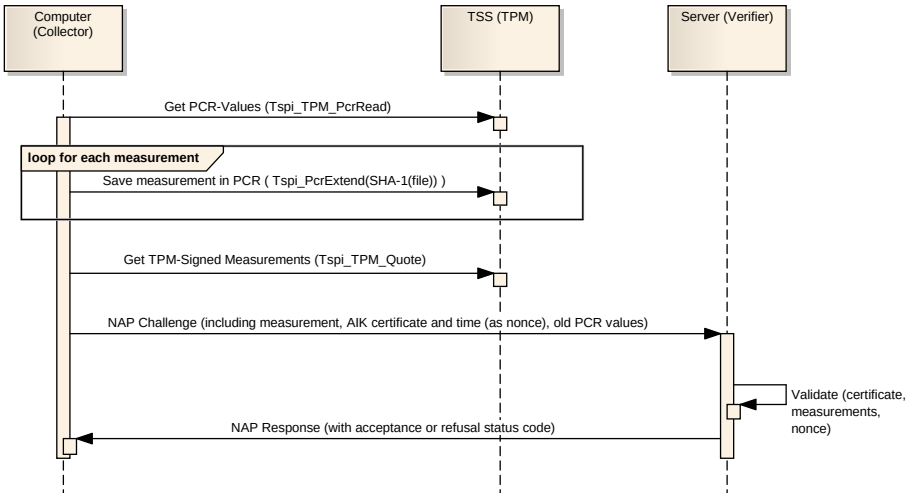


Abbildung 5.6: Ablauf Attestation

3. Der Computer fordert nun die Hashwerte jener PCR an, welche die zuvor ausgeführten Messungen beinhalten. Der *Tspi_TPM_Quote()*-Befehl liefert diese zurück. Zudem werden die zurückgelieferten Werte mit dem AIK-Schlüssel des TPM signiert.

4. Der Computer besitzt nun alle signierten PCR-Hashwerte der Messdateien und sendet nun folgende Werte an den Server:

   - Nonce (Datum und Zeit)

   - Ursprüngliche Hashwerte der PCRs

- Durch das TPM signierte neue PCR-Werte, welche aus den Messungen resultierten
- AIK-Zertifikat ausgestellt durch die PrivacyCA
- Version der Konfiguration

5. Der Server überprüft nun, ob das AIK-Zertifikat gültig ist und ob die übertragenen Daten nicht verändert wurden.
6. Danach wird überprüft, ob die Programmbibliotheken und -dateien sowie die übrigen Daten (nonce, Zertifikat, Windows SHA Werte) gültig sind. Dazu werden die gemessenen Hashwerte der PCR mit Berechnungen des Servers verglichen.
7. Der Server sendet die Rückmeldung an den Client mit der Mitteilung, ob er konform ist oder nicht.

5.2.5 TPM Initialisierung

Die TPM-Initialisierung erfolgt über das Konsolenprogramm. Sie muss während der Clienteinrichtung einmalig durch den Systemadministrator durchgeführt werden. Um das TPM zu aktivieren, muss es vorgängig im BIOS gelöscht und danach eingeschaltet werden (Anleitung in Kapitel 8.4). Nachdem das TPM aktiviert und zurückgesetzt wurde setzt das Initialisierungsprogramm mit der TPM-Einrichtung fort.

Dazu wird dem TPM als allererstes mit dem TPM-Befehl *TPM_TakeOwnership* ein Besitzer zugeordnet (s. Abbildung 5.7). Dieser Befehl verlangt nach einem Passwort für Verwaltungszwecke; dem sogenannten Ownerpassword. Dieses wird für grundlegende TPM-Administrationstätigkeiten benötigt. Zusätzlich wird ein Passwort für die spätere Verwendung des SRKs verlangt. Dieses wird für alle Kommandos vorausgesetzt, welche Zugriff auf den SRK benötigen.

Mit *Tspi_TPM_CollateIdentityRequest()* wird im nächsten Schritt ein AIK-Schlüssel generiert. Dieser wird benötigt, um die Remote Attestation durchzuführen, resp. um die Messwerte aus den PCRs signiert auszulesen. Zusätzlich zur Generierung wird eine Zertifikatsanfragestruktur (certificateRequestStructure) zurückgegeben, welche bei der PrivacyCA.com einzureichen ist. Dieser Request wird via HTTP zur PrivacyCA.com gesendet. Gleichzeitig wird bei der PrivacyCA.com ein verschlüsseltes Zertifikat zurückgegeben, welches später für den Remote Attestation-Vorgang benötigt wird. Der von der PrivacyCA beantwortete Request kann nun als abschliessender Schritt mit dem Befehl *Tspi_TPM_ActivateIdentity()* verarbeitet werden, woraufhin das AIK-Zertifikat verfügbar wird.

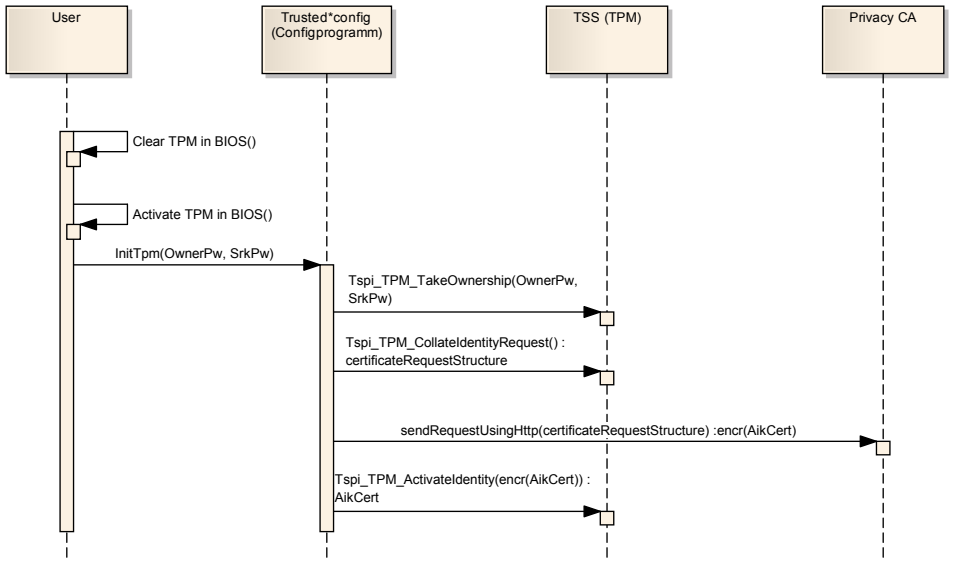


Abbildung 5.7: Initialisierungsablauf des TPMs

5.2.6 Zusätzliche Technologieentscheide

cURL

Im Beispielcode der Privacy CA wird für HTTP-Verbindungen cURL verwendet. Diese Funktionalität sollen durch WinHTTP abgelöst werden, denn mit dieser Abänderung wird erreicht, dass die zusätzliche Drittkomponente cURL nicht benötigt wird. Die WinHttp-Funktionalitäten müssen nämlich nicht zusätzlich eingebunden werden, da sie bereits Bestandteil von Windows sind.

OpenSSL

TrouSerS nutzt für kryptographische Operationen die Library OpenSSL. Für solche Funktionalitäten kann die Alternative Microsoft CryptoAPI noch in Betracht gezogen werden. Doch angesichts der Tatsache, dass OpenSSL im Projekt bereits verwendet wird, kann durch die Wahl von OpenSSL eine zusätzliche Abhängigkeit vermieden werden.

5.3 Domain Model

Die komplette Übersicht des Trusted SHA sowie des Trusted SHV in Domain-Model-Form ist in den Abbildungen 5.9 sowie 5.10 ersichtlich. Auf den detaillierten Aufbau wird in Kapitel 7 eingegangen. Um nachvollziehen zu

können, in welchem Bereich die Erweiterung in die NAP-Umgebung eingebunden wird, kann die Abbildung 5.8 verwendet werden. Dort ist ersichtlich, welche Teile bereits durch NAP vorhanden ist. Der Trusted SHA wird dabei durch den NAP Agent und der Trusted SHV durch den NAP Administration Server verwaltet.

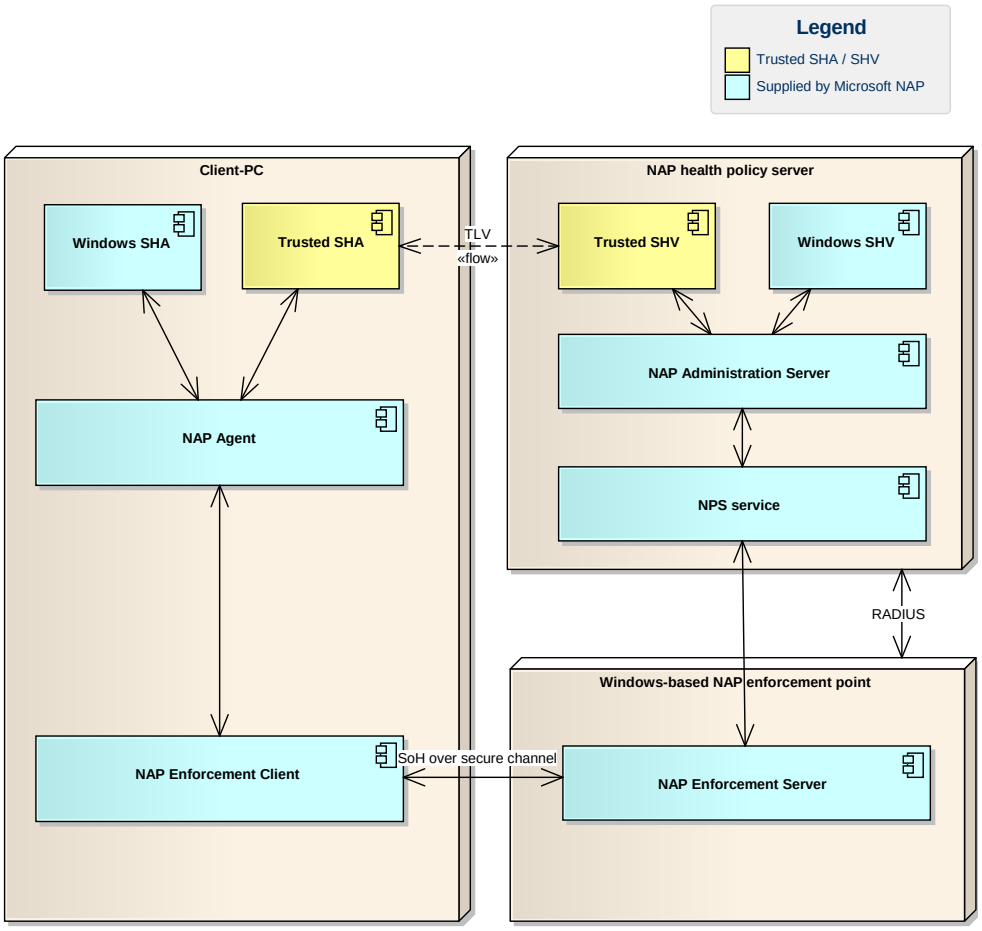


Abbildung 5.8: Übersicht über die Komponenten

5.3.1 Trusted System Health Agent (SHA)

Das Domain Model ist in Abbildung 5.9 ersichtlich. Folgend wird auf die Bestandteile des Trusted SHA eingegangen.

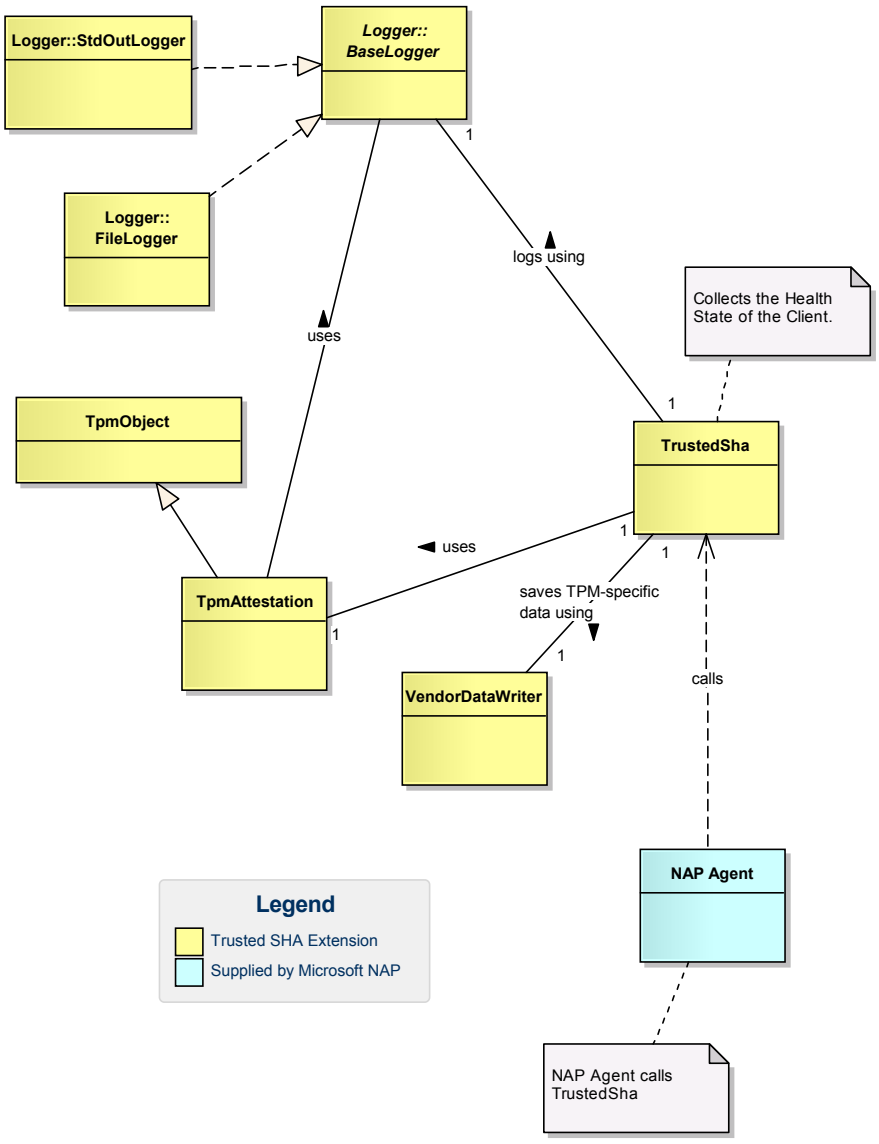


Abbildung 5.9: Trusted SHA Domain Model

NAP Agent

Der NAP Agent veranlasst den TrustedSha (und alle anderen registrierten SHAs von Drittherstellern), Messungen auf dem Clientcomputer durchzuführen. Er ist blau dargestellt, da er durch die NAP-Umgebung zur Verfügung gestellt wird und nicht veränderbar ist.

TrustedSha

Der TrustedSha, welcher das Synonym für den entwickelten Trusted System Health Agent darstellt, koordiniert den Ablauf der Messungen. Die resultierenden Ergebnisse werden zusammen mit weiteren Daten (Zertifikat als PEM(x509), Zeit als Nonce, ursprünglicher PCR-Inhalt, resultierende Quote-Daten) und die Version der Konfiguration als Daten-Blob dem NAP Agent weitergereicht.

TpmObject

Das TpmObject, welches als Synonym für Trusted Platform Module steht, ist für die Kommunikation mit dem TPM-Hardwaremodul verantwortlich und bietet Funktionalitäten für den Verbindungsauf- und den Verbindungsabbau. Die davon abstammenden Objekte *TpmAttestation* und *TpmManagement* bieten darüber hinaus spezifischere Funktionalitäten.

TpmAttestation

Ist für den Attestation TPM-Einsatz verantwortlich und besitzt direkten Zugriff auf das TPM. Funktionalitäten wie das Beschreiben der PCR resp. das Auslesen und Signieren der Registerwerte werden durch dieses Objekt durchgeführt.

VendorDataWriter

Dieses Objekt ist zuständig für das korrekte Schreiben der Messungsdaten, damit sie dann übertragen werden können.

BaseLogger

Der BaseLogger besitzt Ausgabefunktionalitäten, welche vom *FileLogger* bzw. vom *StdOutLogger* verwendet werden.

FileLogger

Legt Informationen und Fehler zu Protokollzwecken in einer Datei ab.

StdOutLogger

Schreibt Informationen und Fehler zu Protokollzwecken auf die Standardausgabe Konsole (std::cout).

5.3.2 Trusted System Health Validator (SHV)

Das Domain Model ist in Abbildung 5.10 ersichtlich. Folgend wird auf die Bestandteile des Trusted SHV eingegangen.

NAP Administration Server

Der NAP Administration Server ist ein Teil des NAP-Systems. Er ruft die einzelnen SHVs des Systems auf, um die Verifikation der vom Client erhaltenen Daten vorzunehmen. Er ist blau dargestellt, weil er durch das System zur Verfügung gestellt wird und deshalb nicht veränderbar ist.

TrustedShv

Der TrustedShv (Synonym für Trusted System Health Validator wird durch den NAP Administration Server aufgerufen. Bei diesem Aufruf erhält er Messungsdaten der einzelnen SHAs eines Clients. Diese übermittelten Daten werden auf Ihre Gültigkeit überprüft und die Messwerte werden schliesslich gemäss gesetzten Policies verifiziert. Anschliessend liefert er das Resultat dem NAP Administration Server zurück.

Verifier

Objekt, welches die Verifizierfunktionalität für die neu hinzugekommenen Daten (TPM Signatur etc.) bereithält.

PolicyManager

Liest die definierten Policies aus und übergibt diese dem *Verifier*, damit dieser weiss, wie er vorzugehen hat.

VendorDataReader

Liest die Messungsdaten des Clients aus und übergibt es dem *Verifier*.

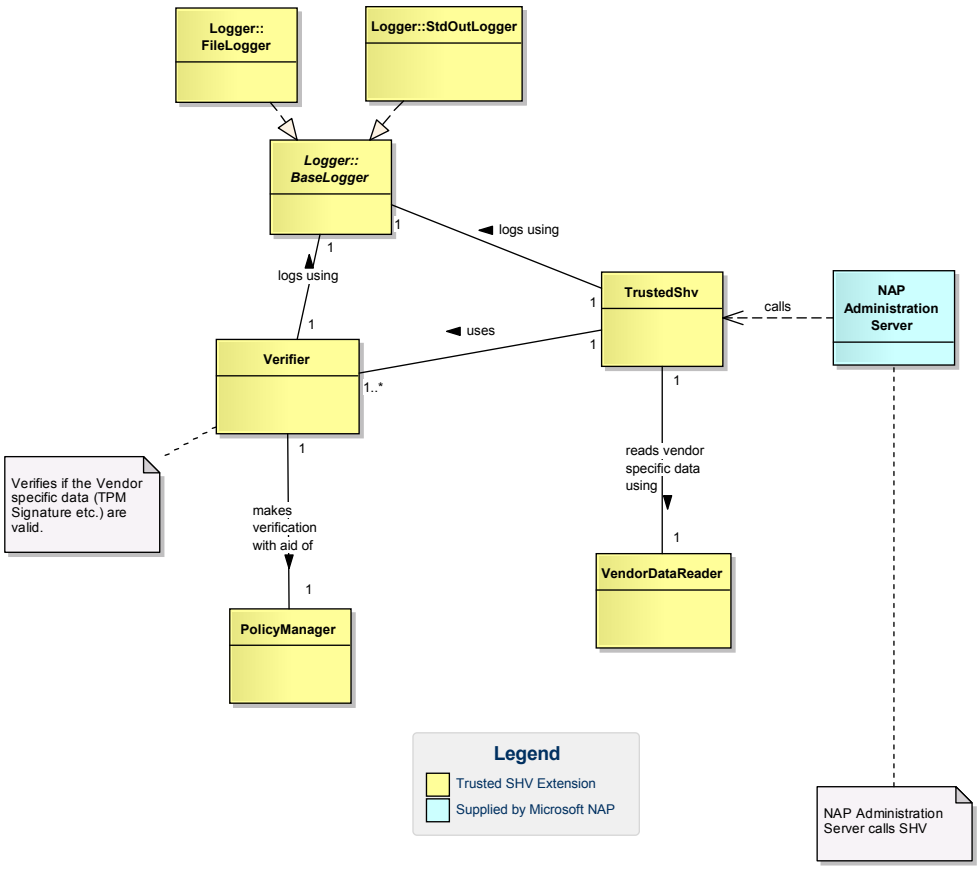


Abbildung 5.10: Trusted SHV Domain Model

BaseLogger

Bei diesem Objekt handelt es sich um dasselbe Objekt wie bereits in Kapitel 7.9 beschrieben wurde.

FileLogger

Bei diesem Objekt handelt es sich um dasselbe Objekt wie bereits in Kapitel 5.3.1 beschrieben wurde.

StdOutLogger

Bei diesem Objekt handelt es sich um dasselbe Objekt wie bereits in Kapitel 5.3.1 beschrieben wurde.

Kapitel 6

Design

6.1 Architektur / Komponenten

In der Abbildung 6.1 wird aufgezeigt, wie die Interaktion zwischen den eingesetzten Komponenten funktioniert.

Die in dieser Abbildung gelb hervorgehobenen Module wurden in das bestehende System des NAPs entwickelt und stellen die Erweiterung der Trusted SHA und Trusted SHV dar. Alle anderen Module und Komponenten bestanden bereits. Auf die einzelnen Komponenten des Microsoft-NAP wird hier nicht detaillierter eingegangen. Diese werden in Kapitel 3.9.4 erklärt.

6.1.1 Client-PC

Dies ist der Clientcomputer. Auf diesem ist das TPM aktiv. Auf diesem Gerät erfolgt die Messung des Sicherheitszustandes des Computers.

6.1.2 TPM

Hardwaremodul, das die Attestation (Bescheinigung) über die korrekte Messung und über die korrekt vorhanden liegenden Dateien geben kann. Auf das TPM wird über TrouSerS zugegriffen. TrouSerS ist ein TSS-Framework, welches den Zugriff auf das TPM abstrahiert und somit vereinfacht.

6.1.3 NAP Health Policy Server

Das ist der Server, welcher die Überprüfung der NAP-Nachricht übernimmt. Im Zusammenhang mit NAP wird diese Komponente Network Policy Server (NPS) genannt.

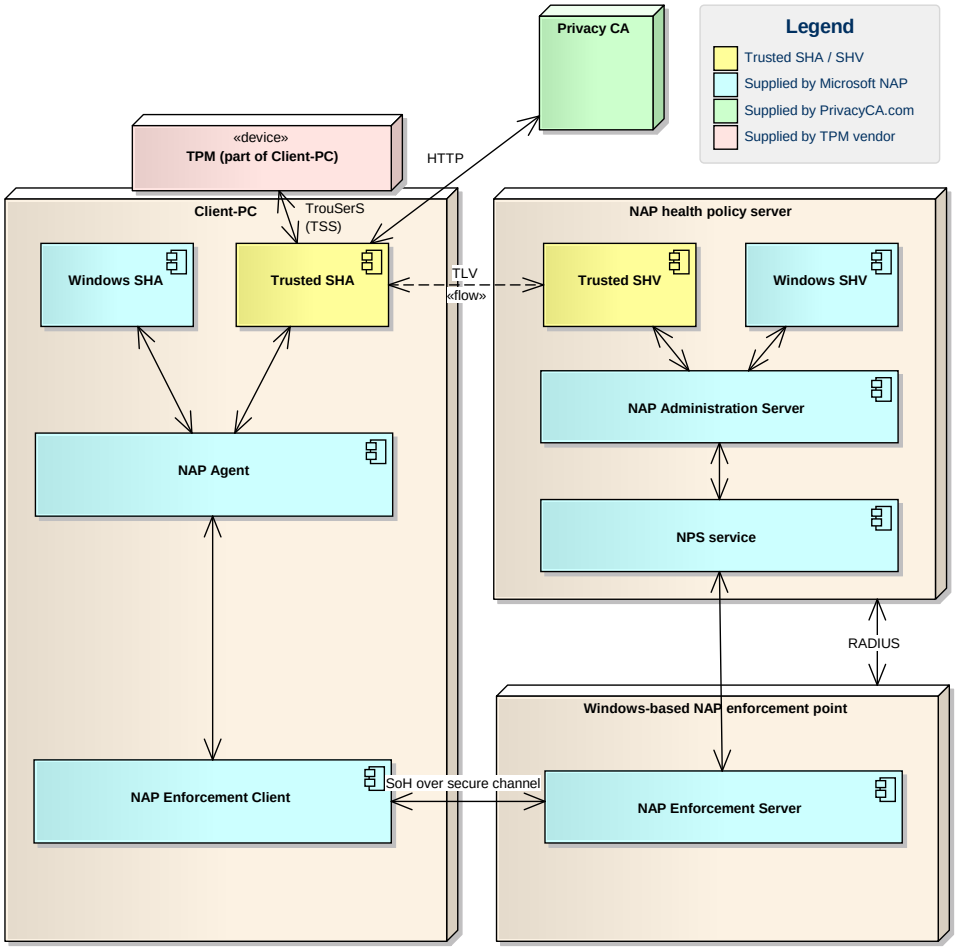


Abbildung 6.1: Architekturübersicht

6.1.4 Windows-based NAP Enforcement Point

Enforcement Point ist der Ort, wo die Richtlinie durchgesetzt wird. Das kann ein separates System sein. In der von diesem Projekt eingesetzten Testumgebung sind NAP Enforcement Point und Health Policy Server auf demselben Rechner eingerichtet.

6.1.5 Privacy CA

Die Privacy CA ist eine extern arbeitende Zertifizierungsstelle und bietet dem Client die Möglichkeit, zusätzlich zum AIK ein AIK-Zertifikat auszustellen, um nachzuweisen, dass das TPM standardkonform ist. Die Privacy CA wird auch als Trusted Third Party bezeichnet.

6.2 Projektstruktur

Die Visual Studio Solution TrustedNAP (Abb. 6.2) beinhaltet verschiedene Unterprojekte, welche die einzelnen Komponenten repräsentieren.

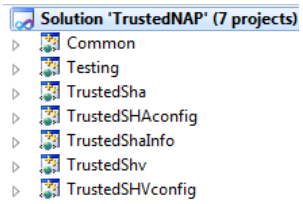


Abbildung 6.2: Projekt in der TrustedNAP Solution

Common beinhaltet alle gemeinsamen Komponenten welche der Trusted SHA und der Trusted SHV für die Integration in die NAP-Umgebung benötigen.

Testing enthält die Unit-Tests.

TrustedSha erstellt den Trusted SHA, welcher auf dem Client als Service installiert und ausgeführt wird.

TrustedSHAconfig ist das Programm, mit welchem sämtliche Konfigurationen auf dem Client vorgenommen werden können.

TrustedShaInfo wird clientseitig für die Daten-Darstellung im Informationsprogramm *napstat* (Kapitel 8.5.2) verwendet.

TrustedShv generiert den serverseitigen Verifizierdienst (Trusted SHA), der die Messdaten des Trusted SHA überprüft.

TrustedSHVconfig ist das Programm, mit welchem sämtliche Konfigurationen auf dem Server vorgenommen werden können.

6.3 Klassen

In den folgenden Diagrammen wird das Zusammenspiel der TPM-, der Registry- und der Logging-Klassen dargestellt.

6.3.1 TPM

Das *TpmObject* (Abbildung 6.3) bietet grundlegende Zugriffsmöglichkeiten, welche für die Verwendung des TPMs benötigt werden. Unter anderem sind

dies der Verbindungsauf- und abbau. Die drei Klassen, welche die Funktionalitäten von *TpmObject* nutzen, sind *PrivacyCAConnector*, *TpmAttestation* und *TpmInitOwner*. Auf den genauen Funktionalitätsumfang wird in Kapitel 7 genauer eingegangen. *PrivacyCAConnector* instanziiert ein Objekt der Klasse *Http*, um über HTTP mit der Privacy CA kommunizieren zu können.

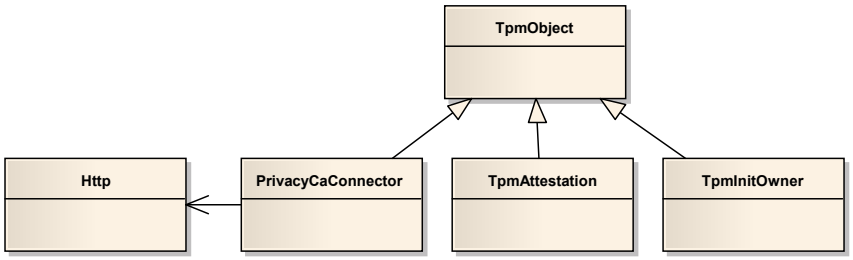


Abbildung 6.3: Übersicht TPM-Klassen

6.3.2 Registry

Der *BaseRegistryWrapper* (Abbildung 6.4) bietet Möglichkeiten, um auf die Schlüssel in der Registry lesend zugreifen zu können. Die davon erbende Klasse *RegistryWriter* ermöglicht zusätzlich die Möglichkeit zum Schreiben und Löschen von Registryeinträgen. Die Klassen *KnownAikRegistryWrapper* und *ConfigRegistryWrapper* verwenden die Funktionalitäten des *BaseRegistryWrapper*.

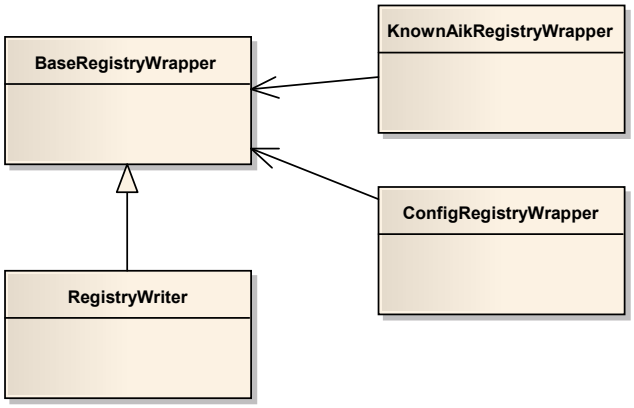


Abbildung 6.4: Übersicht Registry-Klassen

6.3.3 Logging

Der *BaseLogger* (Abbildung 6.5) stellt Funktionalitäten für die formatierte Ausgabe zur Verfügung. Die Klassen *FileLogger* und *StdOutLogger* realisieren das Schreiben der Lognachrichten in einer Datei bzw. ermöglicht die Ausgabe in der Konsole.

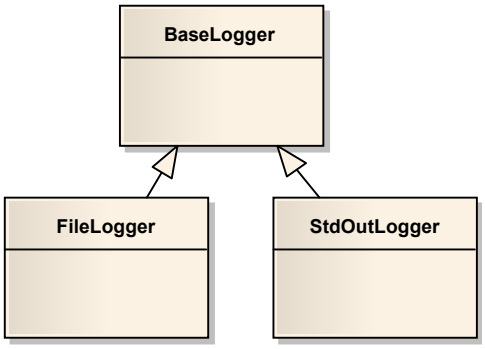


Abbildung 6.5: Übersicht Logging-Klassen

6.4 Datenspeicherung

Die Konfigurationseinstellungen, welche auf dem Client- sowie auf dem Serverrechner anfallen, werden in der Registry gespeichert. Jegliche anderen Dateien (Logging, AIK-Zertifikat, verschlüsselter AIK) werden auf die Festplatte gespeichert. Der AIK sowie das dazugehörige Zertifikat werden im selben Verzeichnis gespeichert, in welchem sich auch der Trusted SHA-Dienst befindet. Die Datei, in welche die Loggingausgaben geschrieben werden sollen, kann beliebig durch eine entsprechende Konfigurationsänderung gewählt werden.

6.5 Schnittstellen und Formate

Folgend werden die Schnittstellen erläutert, welche zwischen den verschiedenen Komponenten verwendet werden (siehe dazu Abbildung 6.1).

6.5.1 Privacy CA

Der Trusted SHA kommuniziert zur Privacy CA über das HTTP-Protokoll. Die API von PrivacyCA.com beschreibt ein REST-API[12]. Die beiden genutzten Funktionalitäten in diesem Programm erfolgen über dieselbe URL:

http://www.privacyca.com/api/pca/levelX

Wobei X mit dem Zertifikatlevel ersetzt werden muss (0 oder 1). Die Unterscheidung der angefragten Funktion wird durch den Request-Typ unterschieden:

- GET-Request: fordert das Zertifikat der Privacy CA an.
- POST-Request: fordert ein neues AIK-Zertifikat an.

Dem POST-Request muss zusätzlich ein durch das TPM erzeugter AIK-Request mitgesendet werden, der die Anforderung schliesslich auslöst.

6.5.2 TPM

Zwischen dem Trusted SHA und dem TPM wird das TrouSerS-Framework eingesetzt. Dieses ermöglicht die Kommunikation zwischen Computer und TPM.

6.5.3 Zwischen Trusted SHA und Trusted SHV

Die Kommunikation zwischen Trusted SHA und Trusted SHV läuft über ein eigen definiertes TLV-Protokoll ab. Dieses überträgt die Version der Konfiguration, die Signatur des Quotes, das AIK-Zertifikat des Clients, die PCR-Zustände vor und die Werte nach hinzufügen der Messdatei-Checksummen, die Konfigurationsversion sowie die Challenge (auch Nonce genannt). Ein vertiefter Einblick in die Übertragung wird in Kapitel 7.2.2 gewährt.

6.5.4 Zwischen Enforcement Server und Enforcement Client

Diese Schnittstelle wurde durch Microsoft definiert. Hier wird für die Übertragung das SoH-Protokoll[3] mit einer Verschlüsselung verwendet.

6.5.5 Zwischen NPS Service und NAP Enforcement Server

Zur Übertragung zwischen dem NPS Service und dem NAP Enforcement Server wird das RADIUS-Protokoll eingesetzt.

6.6 Testing

Das Produkt und die Zuverlässigkeit des Produkts wurden in der Testumgebung mit verschiedensten Einstellungsmöglichkeiten ausführlich getestet. So konnten die funktionalen Anforderungen getestet werden.

Wo es angebracht war, wurden automatisierte Unit-Tests definiert. Dies geschah allerdings nicht mit einem Testframework, sondern mit dem eigens dafür geschriebenen Projekt *Testing*.

Kapitel 7

Implementation

7.1 Schnittstelle zu Windows NAP (FAnf01)

Das TrustedNAP basiert auf dem Beispielcode von Microsoft, welcher auf der CD zu finden ist. Hier soll aufgezeigt werden, welche Teile übernommen und wo Erweiterungen angebracht wurden.

Auf Seite des Trusted SHA (Clientcomputer) wurde die Klasse *TrustedSha* umgeschrieben, um sie als Windows Service verwenden zu können. Die Attestierung mit dem Trusted NAP wird in der Klasse *Callback*, in der Funktion *FillSoHRequest* aufgerufen. Diese Funktion und somit auch alle darin aufgerufenen Klassen wurden neu implementiert. In den dort aufgerufenen Klassen ist der Grossteil der TPM Funktionalität untergebracht. Bei den anderen Funktionen wurden nur kleinere Änderungen vorgenommen, um eine reibungslose Kommunikation zu ermöglichen.

Beim Trusted SHV befindet sich der Einhänge-Punkt für die Verifikation in der Klasse *TrustedShv*, genauer in der Funktion *CheckRequestSoHHealth*. In ihr wird die erhaltene Nachricht des TrustedSha ausgewertet und zur Verifikation an die Klasse *Verifier* weitergeleitet. Die darin enthaltene Logik enthält die Verifizierungsfunktionalität des TrustedSHV. Über diese kann der NAP Administration Server dem Trusted SHV die Dateien des TrustedSha zukommen lassen, um danach die Verifikation mit der Klasse *Verifier* durchzuführen.

7.1.1 TrustedShaInfo.dll

Die *TrustedShaInfo.dll* enthält die Informationen des Trusted SHA, welche für die Ausgabe von Meldungen nötig sind. Diese werden vom *TrustedSha* an den Benutzer via GUI oder netshell weitergegeben.

7.1.2 TrustedSha.exe

Die *TrustedSha.exe* wurde als Windows-Service implementiert. Sie enthält den Trusted SHA und somit die Logik für die Messungsdurchführung. Zudem implementiert die Datei das Interface zum NAP-Agent, um die Anfragen des Systems zu beantworten. Ebenfalls wertet er die Antworten vom Server aus und gibt entsprechende Meldungen über die Systembibliothek *TrustedShaInfo.dll* an den Benutzer weiter.

7.1.3 TrustedShv.dll

Die *TrustedShv.dll* enthält die gesamte Funktionalität des Validators. Sie muss auf dem Server eingebunden werden.

7.2 Trusted SHA / Trusted SHV (FAnf01)

Die nun folgenden Bereiche wurden selber entwickelt.

7.2.1 Windows Dienst

Der TrustedSha wurde als Windows Service implementiert. Er wird mit der Bezeichnung TrustedSHA im System registriert und startet nach der Installation bei jedem Systemstart automatisch.

Das Code-Listing 7.1 zeigt den vereinfachten Ablauf, welche für die Erstellung des Services benötigt wird. Nach dem Öffnen des Service-Managers wird mit *CreateService* ein neuer Service mit der Bezeichnung TrustedSHA und der Beschreibung "HSR Trusted System Health Agent" erstellt. Die weiteren Parameter beschränken den Zugriff auf den eigenen Prozess und setzen die Rechte des Services. Zudem werden Fehler aus dem TrustedSHA für den Rest des Systems als nicht kritisch eingestuft.

Zuletzt wird mit *ChangeServiceConfig2* der Auto-Start des Services verzögert, sodass der NAP-Agent von Windows sowie der Deamon von TrouSerS bereits gestartet sind, bevor der TrustedSha startet.

Die restlichen Funktionen der Quellcode-Datei dienen zum Verwalten des Services. Sie verarbeiten das Start- und Stopp- Signal von Windows und lösen die damit verbundenen Aktionen für den TrustedSHA aus.

```
SC_HANDLE scManager, service;
scManager = OpenSCManager(NULL,NULL,SC_MANAGER_ALL_ACCESS);
service = CreateService(
                scManager,
                "TrustedSHA"
                "HSR Trusted System Health Agent",
                SERVICE_ALL_ACCESS,
                SERVICE_WIN32_OWN_PROCESS,
                SERVICE_AUTO_START,
                SERVICE_ERROR_NORMAL,
                pathToService,
                NULL,
                NULL,
                NULL,
                NULL,
                NULL);
SERVICE_DELAYED_AUTO_START_INFO delay = { true };
ChangeServiceConfig2(service, SERVICE_CONFIG_DELAYED_AUTO_START_INFO, &delay);
CloseServiceHandle(service);
```

Listing 7.1: Registrierung des Windows Services

7.2.2 Type Length Value (TLV)

Die Windows API des NAPs stellt für die Übertragung der Messwerte zwischen entwickelten SHAs und SHVs ein Feld namens *vendorSpecificData*[2] zur Verfügung. Insgesamt können 3996 Bytes für beliebig zu übertragende Daten genutzt werden. 4 Bytes werden als Längenfeld verwendet.

Um eine möglichst einfache und erweiterbare Datenübertragung zu garantieren, wird zwischen dem Trusted SHA und Trusted SHV mit einem TLV-Protokoll gearbeitet.

Aufbau

Abbildung 7.1: Aufbau eines Feldes einer TLV-Nachricht

Abbildung 7.1 zeigt den Aufbau einer Nachricht für die Übertragung. Sie besteht immer aus zwei Bytes für den Typ, gefolgt von zwei Bytes für die Länge des anschliessend folgenden Teils, des Wertes. Die verfügbaren Nachrichten-Typen (Tabelle: 7.1) sind im Projekt *Common* in der Datei *Common.h* definiert. Sie stehen somit dem Trusted SHA und dem Trusted SHV zur Verfügung. Durch die zentrale Definition kann die Liste der Typen auf einfache Art und Weise erweitert werden.

Da für jede übermittelte Nachricht ein Typ definiert ist, ist die Reihenfolge, in welcher die Daten durch den Trusted SHA in die Nachricht abgefüllt wer-

Variable	Wert	Verwendung
MSG_TYPE_VERSION	0x0001	Version der Konfiguration
MSG_TYPE_SIGNATURE	0x0002	Signatur des Quotes
MSG_TYPE_CERT_PEM	0x0003	Zertifikat des Clients im x509 Format
MSG_TYPE_PCR_DATA	0x0004	PCR-Inhalt nach extend
MSG_TYPE_PCR_INITVALUES	0x0005	PCR-Werte vor den Messungen
MSG_TYPE_CHALLENGE	0x0006	Für die Challenge verwendeter Wert (Zeit)

Tabelle 7.1: Verfügbare Datentypen für das TLV

den, nicht relevant. Der Trusted SHV wertet die einzelnen Nachrichtenteile aus und macht sie für die folgende Verifikation verfügbar.

Um das Verarbeiten der Nachrichten zu erleichtern, wurden zwei Klassen geschrieben, welche Funktionalitäten für das Schreiben und Lesen von Nachrichten zur Verfügung stellen. Namentlich sind dies die Klassen *VendorDataWriter* und *VendorDataReader*.

7.2.3 VendorDataWriter (Class)

Diese Klasse befindet sich im Projekt *TrustedSha* und ist für das Zusammenstellen einer Nachricht zwischen Trusted SHA und Trusted SHV zuständig.

addToMessage() Mit der Funktion addToMessage() kann der Nachricht ein weiteres TLV-Feld angehängt werden. Falls die Nachricht keinen Platz für die gewünschte Aktion hat, wird false zurückgegeben. Beim Programmieren muss darauf geachtet werden, dass die Maximalgrösse nicht überschritten wird, da diese von der Application Programming Interface (API) festgelegt ist.

getMessage() Gibt die gesamte Nachricht als BYTE-Pointer zurück.

getMessageLength() Gibt die aktuelle Länge der gesamten Nachricht zurück.

resetMessage() Löscht die Nachricht und setzt alle Zustände zurück.

7.2.4 VendorDataReader (Class)

Diese Klasse befindet sich im Projekt *TrustedShv*. Die Klasse gleicht einem Iterator, da sie durch eine TLV-Nachricht vorwärts navigieren und dabei die gelesenen Daten zurückgeben kann.

Durch diesen Aufbau und die im folgenden Text aufgeführten Funktionen bietet es sich an, die Auswertung einer gesamten Nachricht mit einer Schleife kombiniert mit einem Switch Case Statement durchzuführen. Dies ist als Pseudocode im Listing 7.2 dargestellt.

```
VendorDataReader reader;
do {
  type = reader.getNextType();
  switch(type) {
    case MSG_A:
      ...
      break;
    case MSG_B:
      ...
      break;
  }
} while(reader.goNext());
```

Listing 7.2: Pseudocode für das Auswerten einer Nachricht

getNextType()/getNextLength()/getNextValue() Gibt die entsprechenden Werte des aktuellen Teils der Nachricht zurück.

goNext() Springt zum nächsten Nachrichtenteil. Vergleichbar mit dem ++ Operator des Iterators.

setToBegin() Setzt den Zeiger innerhalb des Objektes auf den Anfang der Nachricht und somit auf den ersten Teil der Nachricht.

7.3 Attestation (FAnf02, 03, 04, 06, 14)

Die für den Ablauf der Attestation benötigte Funktionalität befindet sich in der Klasse *TpmAttestation*. Zur Verifizierung dient die Klasse *Verifier*.

7.3.1 TPM

Die gesamte Kommunikation mit dem TPM findet über TrouSerS statt. Innerhalb des Projektes besitzen alle Klassen, die mit dem TPM interagieren, das Prefix Tpm.

Zusätzlich zum hier gezeigten TPM-Verbindungsaufbau wird in Kapitel 7.6.15 auf das *TPM_TakeOwnership* vertieft eingegangen.

7.3.2 TpmObject (Class)

Die Klasse *TpmObject* vereinfacht die Verwendung des TPMs. Sie besitzt die Möglichkeit, eine Verbindung zum TPM herzustellen. Somit können Objekte, welche Zugriff auf das TPM benötigen, von dieser Klasse erben und dadurch die Funktionalitäten nutzen.

openNewConnection() Diese Funkton öffnet eine neue Verbindung zum TPM. In Abbildung 7.2 ist der Ablauf der Funktion und deren Kommunikation mit TrouSerS ersichtlich. Es wird eine Verbindung zum TPM aufgebaut und danach das TPM-Objekt angefordert. Detailliert läuft der Ablauf folgendermassen ab:

1. Tspi_Context_Create()
2. Tspi_Context_Connect()
3. Tspi_Context_GetTpmObject()

Tspi_Context_Create(): Diese Methode erstellt und initialisiert ein leeres Kontext-Objekt und gibt einen Handle (*hContext*) zurück, welcher dieses Objekt repräsentiert. Dieser Handle wird dann für den weiteren Ablauf benötigt.

Tspi_Context_Connect(): Hiermit wird eine Verbindung zum TPM hergestellt. Diese wird dann in *hContext* für spätere Zugriffe hinterlegt. In *locationOfTPM* kann angegeben werden, zu welchem Remote liegenden TPM eine Verbindung aufgebaut werden soll. In diesem Fall ist die Verbindung lokal und deshalb wird NULL als Parameter eingesetzt.

Tspi_Context_GetTpmObject(): Mittels *GetTpmObject()* wird das TPM-Objekt eines Kontextes zurückgegeben. Pro Kontext kann jeweils nur eine TPM-Instanz existieren. Alle für diese Funktion verwendeten Handler und Objekte sind im Objekt verfügbar. Sie werden solange gespeichert, bis die Verbindung geschlossen oder das Objekt zurückgesetzt wird.

closeConnection() Mit dieser Funktion wird die Verbindung zum TPM abgebaut und die TPM-Objekte werden wieder freigegeben. Diese Funktion wird aufgerufen, um die Verbindung zum TPM zu schliessen.

resetObject() Setzt das TpmObjekt vollumfänglich zurück. Bestehende Verbindungen werden geschlossen und der Speicher wieder freigegeben. Das *TpmObjekt* kann anschliessend wieder für weitere TPM-Zugriffe verwendet werden.

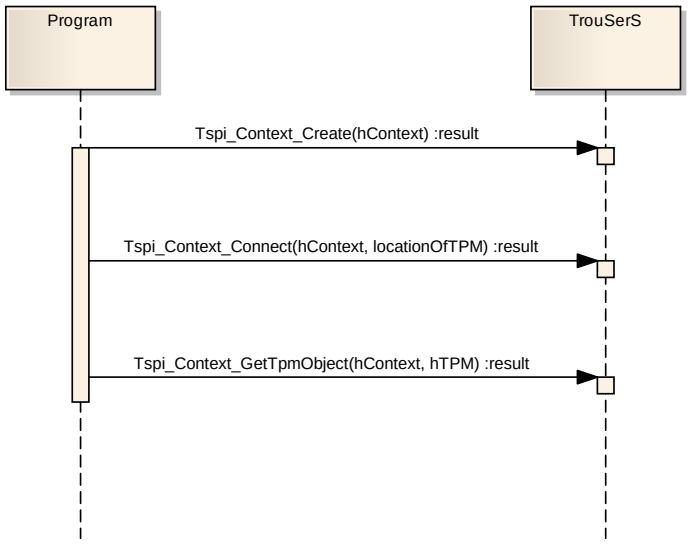


Abbildung 7.2: Ablauf openNewConnection()

checkAndHandleError() Durch diese Funktion kann, um die Fehlerbehandlung zu vereinfachen, ein Fehler des TPMs respektive von TrouSerS erkannt und entsprechend protokolliert werden. Die Funktion überprüft, ob der mitgegebene Error-Code einen Fehler enthält und entscheidet danach über das weitere Vorgehen. Im Fehlerfall wird die Verbindung zum TPM abgebaut. Die mitgegebene Nachricht wird im Fehlerfall als Error ins Log geschrieben. Im Erfolgsfall wird ab dem Log-Detaillierungsgrad INFO ein Logeintrag erstellt, welcher darüber informiert, dass der TrouSerS-Aufruf erfolgreich terminiert hat. Die Errorcodes sind im Anhang in Kapitel G ersichtlich.

7.3.3 TpmAttestation (Class)

Dieses Objekt enthält die gesamte für den Trusted SHA benötigte TPM Funktionalität. Dies umfasst das Lesen und Schreiben der PCR, den Quote Aufruf sowie einige Hilfsfunktionen.

PCR-Mask Um die selektierten PCR einfach zu verwalten wird dieselbe Auswahlform, wie sie ebenfalls in *quote* verwendet wird, gewählt. Es handelt sich dabei bei TPMs mit 24 PCRs um drei Bytes, in welchen jedes Bit ein Register repräsentiert (bei TPMs mit mehr als 24 Registern werden automatisch mehr Bytes verwendet). Die Maske setzt bei den gewählten Registern die Bits auf 1. In Abbildung 7.3 ist die Zuteilung von 24 Registern auf drei Bytes ersichtlich. So sähe die Maske für die Auswahl von Register 4 und 6 bei gesamthaft 8 Registern wie folgt aus: 01010000. Der verwendete

Code, welcher die Auswahl in die Maske schreibt, ist im Code-Listing 7.3 abgebildet.

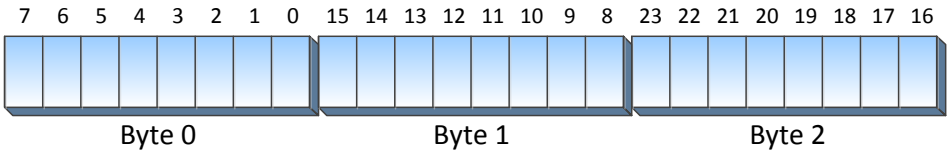


Abbildung 7.3: PCR Maske für 24 Register über 3 Byte

```
addPcrToSelection(int pcr)
{
  selectedPcr[pcr/8] |= 1 << (pcr%8);
}
```

Listing 7.3: Generierung der PCR-Maske (Hinzufügen eines PCR)

Read PCR Values / getPcrValuesOfSelection() PCR Werte werden mit dem TrouSerS Aufruf *Tspi_TPM_PcrRead* gelesen. Es ist dafür keine Authentifikation erforderlich. Die Methode kann nach dem Öffnen einer TPM-Verbindung verwendet werden. Innerhalb des Trusted SHA wird lediglich das Lesen aller ausgewählter Register benötigt. Diese Funktionalität ist in der Funktion *getPcrValuesOfSelection()* implementiert.

In dieser Funktion wird ein BYTE Array erstellt, welches zuerst die PCR-Maske und anschliessend die Werte der ausgewählten Register enthält. Diese Werte sind aufsteigend ohne Trennzeichen sortiert, da jeder Wert exakt zwanzig Bytes lang ist. Somit ist auch die Länge des BYTE Arrays im voraus berechenbar, wobei die PCR-Bytes die Anzahl der Bytes zum Repräsentieren aller PCR darstellt:

$$pcrBytes + numberOfSelectetPcr * 20$$

extendPcr() Schreibt mit Hilfe von TrouSerS einen 20 Byte Wert in ein beliebiges PCR. Dieser Wert wird zuvor mit Hilfe der OpenSSL-Funktion *SHA1* erstellt. Für das Abfüllen der Werte wird der TrouSerS Funktionsaufruf *Tspi_TPM_PcrExtend()* verwendet (vereinfacht im Code-Listing 7.4). Er übergibt den neuen PCR Wert und die Nummer des Registers und erhält den neuen Wert, welcher sich PCR befindet, zurück. Da dieser im Trusted SHA nicht weiter verwendet wird, gibt die Funktion *Tspi_Context_FreeMemory()* den Speicher wieder frei.

Innerhalb des TPMs fliesst bei *extendPcr()* der alte Wert des Registers in die Berechnung ein. Der alte Wert wird zur Berechnung des neuen Wertes

```
TpmAttestation::extendPcr(BYTE* value, UINT32 pcrNr)
{
  BYTE* newPcrValue;
  UINT32  newPcrValueLen;
  Tspi_TPM_PcrExtend(hTPM, pcrNr, 20, value, NULL, &newPcrValueLen, &newPcrValue);
  Tspi_Context_FreeMemory(hContext, newPcrValue);
}
```

Listing 7.4: Speichern eines Wertes in einem PCR

verwendet, indem er mit dem neuen Wert konkatiniert (aneinandergehängt) und anschliessend mit SHA1 der Hash berechnet wird:

$$PCR_i = SHA1(PCR_i||newValue)$$

Da häufig ein Hash von Daten als neuer Wert fungiert muss man dies zusammennehmen, wodurch folgende Operation entsteht:

$$PCR_i = SHA1(PCR_i||SHA1(data))$$

extendMeasurementsToPcrs() Diese Funktion liest die Messungen der Daten aus und schreibt diese mit Hilfe von *extendPcr()* in die Register. Dazu wird jede Datei gemessen, der Messwert in der richtigen Reihenfolge in das definierte PCR geschrieben und schliesslich noch die Challenge in das PCR geschrieben. Der im Code-Listing 7.5 ersichtliche Pseudocode ist folgend noch kurz beschrieben:

1. Lesen der Konfiguration aus der Registry
2. Für jedes konfigurierte PCR:
   - Lese Pfad zur zu überprüfenden Datei
   - Schreibe mit *extendPcr()* den SHA1 Hash der Datei ins Register
   - Wiederhole die letzten beiden Schritte, bis alle zu überprüfenden Dateien für das gewählte PCR abgearbeitet sind
3. Schreibe mit *extendPcr* den SHA1-Hash der Challenge in jedes verwendete Register

```
get PcrConfig
for each pcr in Config
  for each file in pcr
    extendPcr(SHA1(file), pcrNr)
  done
  extendPcr(SHA1(challenge), pcrNr)
done
```

Listing 7.5: Pseudocode für die Funktion *extendMeasurementsToPcrs()*

quote() Der *quote()*-Aufruf beinhaltet den gesamten Ablauf, welcher für die Benutzung des *Tspi_TPM_Quote()* benötigt wird.

Nach dem Aufbau einer Verbindung zum TPM und der Authentifizierung mit dem SRK-Passwort wird der AIK aus dem Datenblob geladen. Anschliessend werden die Vorbereitungen bezüglich der Auswahl der Register getroffen. Dazu wird ein *TSS_PCRS_STRUCT_INFO_SHORT* erzeugt und dem daraus resultierenden Handler die Register zugewiesen, über welche sich das Quote erstrecken soll. Da nun alle benötigten Teile für die Quote-Anweisung bekannt sind (AIK, PCR-Selection), kann die eigentliche Quote-Operation durchgeführt werden. Als Rückgabewert des *Tspi_TPM_Quote* Aufrufes erhält man die Validierungs-Daten in Form eines *TSS_VALIDATION* Structs. Dieses enthält den Hash über den Input, sowie die Signatur und deren Länge.

Um zu überprüfen, ob die erwarteten Daten mit dem Quote-Befehl signiert wurden, wird der Hash durch rekonstruieren des Inputs manuell überprüft. Hierfür ist das *TPM_QUOTE_INFO* struct erforderlich, sowie die Inhalte der Register.

TPM_QUOTE_INFO Struct Die Definition des *TPM_QUOTE_INFO*-Struct ist, wie im Listing 7.6 ersichtlich ist, definiert. Die Version ist dabei immer auf 1.1.0.0 zu setzen. Der Wert *fixed* wird bei der Verwendung von *quote* auf "QUOT" gesetzt. *digestValue* entspricht dem SHA1-Hash über die PCR-Maske, die Inhalte der entsprechenden PCR, sowie deren Länge. Im letzten Parameter kann noch eine beliebige, 20 Byte lange Nonce angegeben werden.

```
typedef struct tdTPM_QUOTE_INFO{
  TPM_STRUCT_VER version;
  BYTE fixed[4];
  TPM_COMPOSITE_HASH digestValue;
  TPM_NONCE externalData;
} TPM_QUOTE_INFO;
```

Listing 7.6: Definition des *TPM_QUOTE_INFO* nach der TCG[9]

7.4 Privacy CA (FAnf07, 08

Als Grundlage für die PrivacyCA.com-Anbindung wurde die von dieser Certification Authority (CA) zur Verfügung gestellte Implementierung[13] verwendet. Im bereitgestellten C-Programmcode wurden kleinere Veränderungen vorgenommen, damit der Code für dieses Projekt nutzbar wurde. Unter anderem wurde die Bibliothek cURL durch WinHttp ersetzt.

7.4.1 Ablauf

Im Programmcode (*PrivacyCaConnector.cpp*) erfolgt die AIK-Zertifikatanforderung und somit die Privacy CA-Anbindung (Abbildung 7.4) wie folgt:

1. Im in der Registry hinterlegten Pfad für die Trusted SHA Dateien werden zwei Dateien für den späteren Ablauf angelegt:
   - In *tpmAIK.blob* wird der generierte AIK-Schlüssel abgelegt, welcher in verschlüsselter Form abgelegt ist.
   - Des weiteren wird in *tpm.pem* das angeforderte AIK-Zertifikat abgelegt.
2. Nun wird das öffentliche Privacy CA Zertifikat angefordert. Die URL, welche für die Anforderung benutzt wird, muss dem entsprechenden Zertifikat-Level entsprechen (siehe API-Beschreibung [12]). TPMs, die ein Endorsement Zertifikat beinhalten (wie derzeit Infineon TPMs), können Level 1 Zertifikate anfordern. Alle anderen fordern ein Level 0 Zertifikat an. Die HTTP-GET-Anforderung erfragt ein Zertifikat im PEM-Format über die Uniform Resource Locator (URL) *www.privacyca.com/api/pca/level%d?ResponseFormat=PEM*, wobei *%d* durch die Level-Nummer ersetzt wird.
3. Aus diesem erhaltenen Zertifikat wird nun der Public Key extrahiert.
4. Mit Hilfe von TrouSerS wird nun das User Password des SRKs geladen. Zusätzlich wird das Owner Password geladen, damit die AIK-Generierung erfolgen kann.
5. Falls es sich nun um eine Level 0 AIK-Zertifikatgenerierung handelt, so muss noch ein Fake-EK-Zertifikat erstellt werden, welches mit Hilfe des EK in einem von Privacy CA vorgegebenen Raster generiert wird. Dies erfolgt lediglich, weil die TPMs im Level 0 keine EK-Zertifikate zur Verfügung stellen. In diesem Schritt wird auch klar, weshalb Level 0 AIK-Zertifikate keine zusätzliche Sicherheit bieten, da die Konfirmität des TPMs nicht überprüft wird.
6. Die Generierung des AIKs wird schliesslich mit dem Befehl *Tspi_TPM_CollateIdentityRequest()* im TPM gestartet. Damit wird ein AIK-Schlüsselpaar mit der Bezeichnung *HSR* erzeugt. Zurückgegeben wird ein binärer AIK-Zertifikatsrequest, welcher im nächsten Schritt zur PrivacyCA.com übermittelt wird. Dieser Request-Datenblob liegt in verschlüsselter Form vor. Er wurde mit einem zufällig generierten symmetrischen Schlüssel verschlüsselt. Dieser symmetrische Schlüssel wurde schliesslich mit dem public Key der Privacy CA verschlüsselt. Durch diese hybride Verschlüsselungsmethode wird gewährleistet, dass der

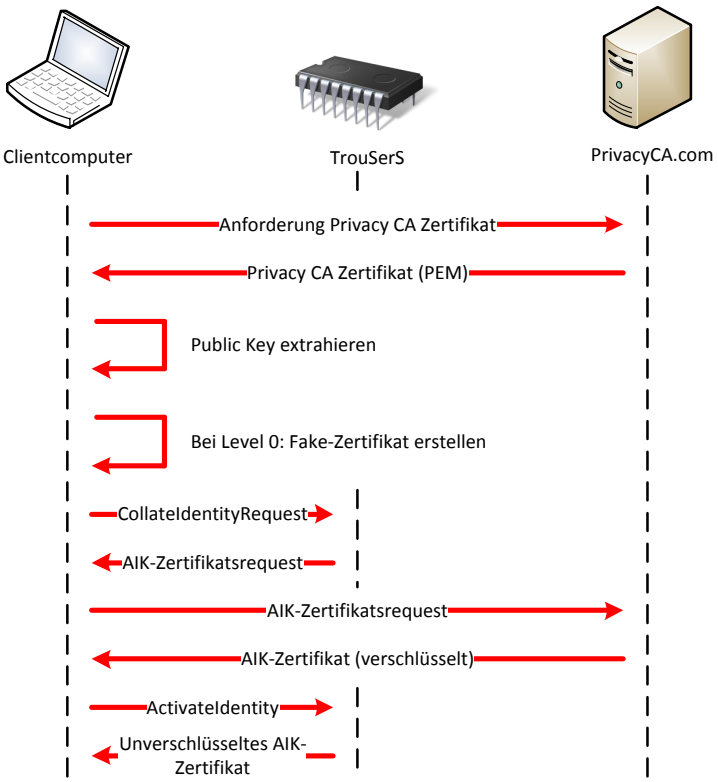


Abbildung 7.4: Ablauf Zertifikatsanforderung

Request lediglich durch die Privacy CA gelesen werden kann. Die gesamten verschlüsselten Daten werden mit einem HTTP-POST-Befehl an die URL *www.privacyca.com/api/pca/level%d?ResponseFormat=Binary* gesendet. Das *%d* wird dabei durch den Zertifikatlevel ersetzt.

7. Die Privacy CA erzeugt ein AIK-Zertifikat, welches zuerst symmetrisch mit einem zufälligen Schlüssel und dann asymmetrisch mit dem öffentlichen Schlüssel des EKs des Client-TPMs verschlüsselt wird.

8. Nach Erhalt der Rückmeldung können die verschlüsselten binären Daten (das durch die Privacy CA signierte AIK-Zertifikat) mit dem Befehl *Tspi_TPM_ActivateIdentity()* entschlüsselt und verfügbar gemacht werden.

9. Als Letztes wird nun das durch diesen Vorgang generierte AIK-Schlüsselpaar in der Datei *tpmAIK.blob* in verschlüsselter Form (durch den SRK Public Key verschlüsselt) abgelegt. Ausserdem wird das AIK-Zertifikat im PEM-Format in der Datei *tpm.pem* gespeichert.

7.4.2 PrivacyCaConnector (Class)

Die Klasse *PrivacyCaConnector* stellt die Funktionalitäten der PrivacyCA.com für andere Klassen zur Verfügung. Die beiden folgenden Methoden bauen auf dem Beispielcode von PrivacyCA auf und wurden lediglich für die Verwendung im TrustedSHAconfig angepasst.

requestCertificateFromPrivacyCa() Diese Methode ist für die Generierung eines neuen AIKs sowie der Anforderung eines AIK-Zertifikates bei der Privacy CA verantwortlich. Dazu werden für den Zugriff auf das TPM das User Passwort, das Owner Passwort (beides als SHA-1-Hash) sowie der Level des auszustellenden Zertifikats mitgegeben. Der Zugriff auf die Privacy CA erfolgt über die Klasse *Http*.

makeEKCert() Die Methode erstellt ein Fake-EK-Zertifikat für Level 0 TPMs, damit diese ebenfalls fähig sind, sich mit einem AIK-Zertifikat verifizieren zu lassen. Dies geschieht mit Hilfe des EK aus dem TPMs, der in ein vorgegebenes Zertifikatraster eingefügt wird. TPMs mit bereits integrierten EK-Zertifikaten benötigen diese Methode nicht. Das Zertifikat wird mit Hilfe des EK in einem von Privacy CA vorgegebenen Raster generiert. Es ist somit ein nachgebautes AIK-Zertifikat.

7.4.3 WinHttp (Class)

Um bei der Portierung des Codes von PrivaycCa.com auf cURL verzichten zu können wurde Curl durch WinHttp ersetzt. WinHttp ermöglicht es, POST- und GET- Requests an die PrivacyCA.com zu senden. Diese Bibliothek bietet den Vorteil, dass sie bereits in Windows integriert ist und deshalb keine zusätzliche Installation notwendig ist. Im Umgang mit HTTP bietet WinHttp einen ähnlichen Funktionsumfang wie cURL.

Requests (GET/POST)

Um eine HTTP Verbindung mit WinHttp aufzubauen muss zuerst eine WinHttp Session aufgebaut werden. Dieser Session werden anschliessend der Host sowie weitere Verbindungseigenschaften übergeben.

Das Code Listing 7.7 zeigt dieses Vorgehen anhand einer Verbindungsvorbereitung. Zu Beginn wird ein Session-Handler mit Standardeinstellungen erstellt. Danach wird der Session die Verbindung zugeordnet (Host und Port) und abschliessend wird der Request definiert. In diesem Listing handelt es sich um einen GET-Request, der zur URL *wUrl* eine Verbindung herstellt.

```
HINTERNET hSession, hConnection, hRequest;
wURL = L"/api/pca/level0";
hSession =     WinHttpOpen(L"HTTP/1.1", WINHTTP_ACCESS_TYPE_DEFAULT_PROXY,
               WINHTTP_NO_PROXY_NAME,WINHTTP_NO_PROXY_BYPASS,0);
hConnection = WinHttpConnect(hSession, L"www.privacyca.com",
               INTERNET_DEFAULT_HTTP_PORT, 0);
hRequest =     WinHttpOpenRequest(hConnect, L"GET", wUrl, NULL,
               WINHTTP_NO_REFERER, WINHTTP_DEFAULT_ACCEPT_TYPES, NULL);
```

Listing 7.7: Vorbereiten einer Verbindung mit WinHTTP

Nach dem Vorbereiten der Handler kann der Request mit *WinHttpSendRequest()* abgesetzt werden. Diese Funktion erlaubt es zudem, weitere Optionen im Header mitzugeben. Nach Absenden des Requests kann mit dem Empfang und der Verarbeitung der Antwort weitergefahren werden.

7.4.4 Verarbeitung der Antwort auf ein Request

Die private Funktion *Http::readAndSaveResponse()* vereinfacht das Lesen und Speichern einer Antwort eines HTTP-Requests. Sie empfängt die Daten mit Hilfe der WinHttp-Funktion *WinHttpReceiveResponse()* und liest anschliessend diesen Datenstrom mit Hilfe von *WinHttpReadData()* solange, wie weitere Daten zu dieser Verbindung zur Verfügung stehen. Anschliessend werden die Daten im mitgegebenen Datei-Pointer gespeichert.

7.5 Verifikation der Messwerte / Verifier (Class) (FAnf 02)

Die Verifikation der Messwerte ist in der Klasse *Verifier* des Trusted SHV untergebracht. Die Klasse ist lediglich im Besitz einer öffentlichen Funktion. Diese nimmt das X509-Zertifikat, die PCR-Werte vor der Messung, die Quote Daten (PCR Werte nach der Messung) sowie die Challenge entgegen. Alle diese Informationen können mit Hilfe der Klasse *VendorDataReaders* (siehe 7.2.4) aus den übertragenen Daten extrahiert werden. Wenn eine der Prüfungen fehlschlägt, wird die restliche Prüfung abgebrochen und ein Fehler zurückgegeben. Dies ist dann der Fall, wenn eine Datenmanipulation vorliegt.

7.5.1 Überprüfung des Zertifikates

Mit OpenSSL wird aus dem erhaltenen Zertifikat der Fingerprint generiert. Dieser Vorgang ist im Code-Listing 7.8 ersichtlich. Anschliessend wird der erhaltene Fingerprint mit auf dem Server hinterlegten vertrauenswürdigen Fingerprints verglichen, welche in der Registry abgelegt sind. Findet eine

Übereinstimmung statt, so wird mit der Validierung der weiteren Daten fortgesetzt; anderenfalls findet ein Abbruch statt.

```
UINT32 x509DigestLen; BYTE md[EVP_MAX_MD_SIZE];
const EVP_MD *sha1Digest = EVP_sha1();
X509_digest(x509, sha1Digest, md, &x509DigestLen);
```

Listing 7.8: Generierung des Fingerprints mit Hilfe von OpenSSL

7.5.2 Überprüfung der Challenge

Als Nächstes wird überprüft, ob die vom Client mitgesendete Challenge (Zeit seit dem 1. Januar 1970 in Sekunden) zur Serverzeit eine maximale Abweichung von 5 Minuten besitzt (+/-5min). Dieser Abweichungswert entspricht der Abweichung, welche ebenfalls im Kerberos-Netzwerkprotokoll verwendet wird[1].

7.5.3 Überprüfung der Signatur

Nach dem Rekonstruieren des Quote-Info-Struct (siehe Kapitel 7.3.3) wird die RSA-Signatur mit Hilfe von OpenSSL überprüft. Dies erfolgt mit der OpenSSL-Funktion *RSA_verify()*.

7.5.4 Überprüfen der mitgesendeten PCR-Daten

Im nächsten Schritt wird überprüft, ob die gesendeten PCR auch tatsächlich zu verifizieren sind. Es wird festgestellt, ob ein Register fehlt bzw. unnötige PCR gesendet wurden. In beiden Fällen wird die Überprüfung beendet.

Die involvierten Register aus der PCR-Maske gelesen (siehe Kapitel 7.3.3).

7.5.5 Überprüfung der gesendeten Messungen

Als Letztes wird überprüft, ob die gesendeten Messwerte auch mit den erwarteten Werten übereinstimmen. Dies wird in der Funktion *checkPcrValue()* für jeden Messwert einzeln durchgeführt. Dabei wird die im Kapitel 7.3.3 beschriebene Extend-Funktionalität des TPMs nachgebildet.

1. Die Referenz für das gewünschte Register wird in der Konfiguration gesucht.
2. Der erwartete Messwert wird berechnet. Dieser Teil wird durchlaufen, bis die Konfiguration für das zu überprüfende PCR abgeschlossen ist (einmal pro eingetragenen Datenhash, welcher in der Registry abgelegt ist).

a) Ausgangswert in pcrValue schreiben

b) pcrValue in toHash schreiben

c) Den Hash der ersten Datei aus der Registry lesen und ihn an toHash anhängen. toHash ist nun 40Bytes lang.

d) Den SHA1 aus toHash berechnen und diesen in pcrValue speichern

e) Solange noch weitere Dateien für dieses Register mitgehasht werden müssen, wir mit b) fortgefahren

f) pcrValue mit den SHA1 der Challenge konkatinieren und anschliessend mit SHA1 Hashen. Wert in pcrValue speichern

g) Den Messwert Trusted SHA mit dem pcrValue vergleichen. Falls diese übereinstimmen wird beim nächsten Register fortgefahren, ansonsten wird abgebrochen und die Überprüfung schlägt fehl.

3. Wenn alle Messwerte überprüft wurden und es sich um gültige Messwerte handelt, so wird die Funktion erfolgreich beendet und es wird 0 zurückgegeben.

7.6 Administration

7.6.1 Windows Registry

Für das Speichern jeglicher Einstellungen wurde die Windows Registry gewählt. Die einzigen Daten, welche auf der Festplatte abgelegt werden, sind der AIK-Key-Blob, das zugehörige Zertifikat sowie das Logfile.

Struktur für Trusted SHA / Trusted SHV

Alle Schlüssel werden unter *HKEY_LOCAL_MACHINE* gespeichert. Genauer unter dem Pfad $SOFTWARE\backslash HSR\backslash TrustedNAP$ innerhalb des dazugehörigen Bereiches. Die abschliessende Struktur ist in Abbildung 7.5 ersichtlich. Dabei ist zu beachten, dass auf dem TrustedSHA der Schlüssel *shv* entfällt, auf Seiten des TrustedSHV der Schlüssel *sha*.

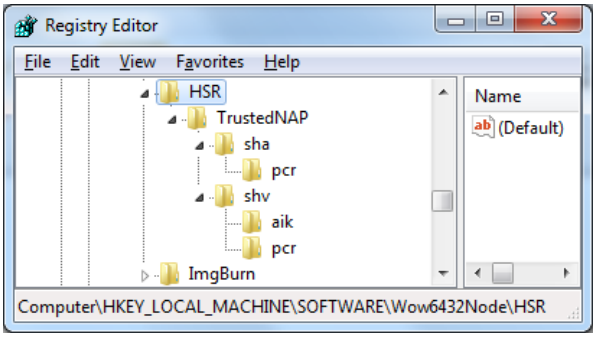


Abbildung 7.5: Registry Struktur des Trusted SHVs und Trusted SHAs

PCR Die PCR werden in einer vorgegebenen Form in der Registry abgelegt. Der Name der Werte entspricht dabei immer der Form [PCR].[Reihenfolge]. Beispielsweise ist das für einen Wert, der für das PCR 15 als ersten Wert zu hashen ist, 15.1.

Für den Trusted SHA entsprechen die gespeicherten Werte dem Pfad zur Messungsdatei. Im Falle des Trusted SHV ist der Wert ein 20 Byte Binary-Value, welcher dem SHA1 Hash der korrespondierenden Datei (auf dem Trusted SHA) entspricht.

AIK Während die Werte für AIKs auf Seite des Trusted SHV den SHA1-Fingerprints entspricht (20 Byte Binary-Values), ist der dazugehörige Name frei wählbar. Es ist empfehlenswert, einen aussagekräftigen Namen, wie z.B. den Computernamen, zu wählen.

Allgemeine Konfiguration Allgemeine Einstellungen, wie beispielsweise der Pfad zur Log-Datei, sind als String-Werte mit vordefiniertem Namen gespeichert (Tabelle 7.2). Während die Einstellungen zur Log-Datei und der Konfigurationsversion unter dem Schlüssel $SOFTWARE\backslash HSR\backslash TrustedNAP$ abgelegt sind, sind die Übrigen unter dem Schlüssel $SOFTWARE\backslash HSR\backslash TrustedNAP\backslash sha$ gespeichert.

Name	Verwendung
logFile	Absoluter Pfad zur Log-Datei
logLevel	Level des Logs (0-4)
shaPath	Pfad zum Verzeichnis des Trusted SHA
tpmUserPw	TPM-User Passwort als SHA1 (Binary-Value)
configVersion	Version der Konfiguration

Tabelle 7.2: Konfigurationswerte in der Registry

7.6.2 BaseRegistryWrapper (Class)

Die Klasse *BaseRegistryWrapper* stellt mit Öffnungs-, Schliess- und einigen Lesefunktionen die grundlegende Funktionalität für den Zugriff auf die Registry bereit. Die Klasse kann einerseits direkt für Lesezugriffe verwendet werden, andererseits kann auch davon abgeleitet werden, um die Funktionalität zu erweitern (z.B. Schreiben).

open() / close() Öffnet bzw. schliesst einen angegebenen Schlüssel in der Registry unter *HKEY_LOCAL_MACHINE*. Es kann pro Objekt jeweils nur ein Schlüssel geöffnet sein. Jedoch ist es möglich, mit mehreren Instanzen des Objektes, mehrere Schlüssel gleichzeitig zu öffnen.

readValue() / readTwentyBytes() Liest den angegebenen Wert aus der Registry und gibt ihn zurück. Im Falle von *readTwentyBytes()* wird von einem Binary-Value die ersten 20 Bytes gelesen. *readValue()* gibt einen String-Value der Maximallänge von 4096 Bytes zurück.

readAllKeys() / readAllTwentyByteKeys() Diese Funktionen verhalten sich wie *readValue() / readTwentyBytes()*. Jedoch werden anstelle des angegebenen Wertes alle Werte innerhalb des geladenen Registry-Schlüssels gelesen und in einer Map mit dem Namen als Key und dem Wert als Value zurückgegeben.

get*Key() Die get*Key()-Funktionen geben die Pfade der Schlüssel, die für das TrustedNap Projekt verwendet werden zurück. In Tabelle 7.3 sind die zurückgegebenen Werte ersichtlich. Alle Werte sind im *HKEY_LOCAL_MACHINE* abgelegt.

Funktion	Rückgabewert
getRootKey()	*SOFTWARE\HSR\TrustedNAP*
getShaConfigKey()	*SOFTWARE\HSR\TrustedNAP\sha*
getShaPcrKey()	*SOFTWARE\HSR\TrustedNAP\sha\shv*
getShvPcrKey()	*SOFTWARE\HSR\TrustedNAP\shv\pcr*
getShvAikKey()	*SOFTWARE\HSR\TrustedNAP\shv\aik*

Tabelle 7.3: Rückgabewerte der get*Key Funktionen

7.6.3 RegistryWriter (Class)

Diese Klasse erbt von *BaseRegistryWrapper* und erweitert die Funktionalität um das Schreiben sowie Löschen von Werten.

writeStringValue() / writeTwentyByteValue() Diese beiden Funktionen schreiben entsprechend ihren Namen Strings respektive 20 Bytes in die Registry. Die Werte werden mit dem angegebenen Namen in der Registry unter den mit *open()* geöffneten Schlüssel geschrieben.

removeKey() Versucht den angegebenen Wert unter den geöffneten Schlüsseln zu löschen.

clearAll() Löscht alle vom Programm für die Konfiguration verwendeten Schlüssel aus der Registry (rekursiv ab und mit $SOFTWARE\backslash HSR$). Nicht gelöscht werden Schlüssel und Werte zum Windows-Service und den Einträgen zu den Programmbibliotheken.

7.6.4 ConfigWrapper (Class)

Die Klasse *ConfigRegistryWrapper* liest die Daten aus der Registry und speichert sie in einem für spätere Prozesse passenden Format ab.

getLogFilePath() Gibt den Pfad der Log-Datei als String zurück.

getLoggingLevel() Gibt das Log-Level als *BaseLogger::loggingLevel* zurück.

getPcrConfigShv() Liest die PCR Konfiguration aus der Registry für den TrustedShv aus. Um die Daten einfacher zur Verfügung zu stellen, wurde eine Schachtelung von zwei Maps gewählt. Der genaue Datentyp der geschachtelten Map lautet:

$$std :: map<UINT32, std :: map<UINT32, std :: string>>$$

Als Beispiel sollen Einträge in der Konfiguration mit den Bezeichnungen 15.1 und 15.3 dienen. Die Nummer des Registers wird zusammengefasst als Wert in die äussere Map geschrieben. Die anschliessend folgenden Zahlen repräsentieren die Reihenfolge für den Hashvorgang (1 und 3). Die unter

diesen Namen gespeicherten 20 Byte Hex-Werte werden schliesslich als Value zu den Keys in der inneren Map gespeichert.

Wenn der Name eines Wertes für ein PCR in der Registry nicht der Struktur [PRC].[Reihenfolge] entspricht, wird dieser ignoriert und der Rest der Konfiguration wird gelesen.

getPcrConfigSha() Gleiches Vorgehen wie bei *getPcrConfigShv()*. Anstelle des 20 Byte Hex Wertes wird der Pfad zur Datei im string gespeichert.

7.6.5 Rechtevergabe

Da der Trusted SHV unter dem Benutzer *Network Service* läuft, ist ein Anpassen der Rechte auf die Log-Datei sowie die Registry Schlüssel nötig. Während in der Registry Leserechte genügen, muss auf die Log-Datei das Recht zum Bearbeiten vorhanden sein.

Auf der Seite des TrustedSha sind keine Anpassungen der Rechte nötig, da dieser als eigenständiger Windows-Service mit genügend Rechten läuft.

7.6.6 Logging (FAnf 10)

Für das Logging wurden in diesem Projekt eigene Klassen geschrieben. Der Aufbau der Klassen erlaubt es, den Logger während des Betriebes auszuwechseln. Ebenfalls ist es einfach möglich, das Logging-Konzept um weitere, individuelle Ausgaben der Nachrichten zu erweitern.

7.6.7 BaseLogger (Class)

Die Klasse *BaseLogger* stellt die grundlegende Funktionalität zum Loggen von Nachrichten zur Verfügung. Neben den Funktionen zum Loggen sind in dieser Klasse virtuelle Funktionen zum Schreiben in das vom spezifischen Logger gegebene Ziel definiert. Diese Funktionen müssen demzufolge zwingend in den davon erbenden Klassen überschrieben werden. Dieser Aufbau ist im Code-Listing 7.9 ersichtlich und wird im Folgenden genauer erläutert.

Der *BaseLogger* wertet anhand des *loggingLevel* aus, in welchen Fällen eine Nachricht tatsächlich in das Log geschrieben oder verworfen werden soll. Hierzu stehen folgende Optionen zur Verfügung:

LOG_DISABLED Es wird nichts geloggt.

```
class BaseLogger
{
public:
  void logFunction(UINT32 errorID, string function);
  void logErrorMsg(UINT32 errorID, string function, string msg);
  void logInfoMsg(UINT32 errorID, string function);
  void logDebugMsg(string msg);
  void logDebugAsHex(string msg, BYTE* data, UINT32 len);
protected:
  virtual void writeLog(string msg) = 0;
  virtual void writeLog(char* msg, UINT32 len) = 0;
};
```

Listing 7.9: Aufbau der wichtigsten Funktionen des BaseLogger

LOG_ERROR Nur Fehler werden geloggt (kritische wie auch unkritische Fehler).

LOG_INFO Informations-Nachrichten werden zusätzlich geloggt.

LOG_DEBUG Es werden zusätzlich Hex-Dumps und bedeutende Variablen geloggt.

Die verschiedenen Level sind unter *BaseLogger::loggingLevel::LOG_** im *BaseLogger* als enum definiert. Das Level kann mit Hilfe der Funktion *setLoggingLevel()* jederzeit angepasst werden.

Die allgemeine Formatierung der Ausgaben eines Log-Files ist in Listing 7.10 ersichtlich. Sie entspricht der Form:

hh:mm:ss Level: Meldung

```
15:32:42 INFO:  Function 'Tspi_TPM_PcrExtend' terminated
```

Listing 7.10: Beispielnachricht in der Log-Datei

writeLog() Diese Funktionen müssen von erbenden Klassen zwingend überschrieben werden. Sie definieren, wo und wie der bereits formatierte Text geschrieben wird.

logFunction() Wird verwendet, um das erfolgreiche Beenden einer Funktion als *INFO* zu loggen, oder im Fehlerfall als *ERROR*.

logErrorMsg() / logInfoMsg() / logDebugMsg() Wird verwendet um eine Nachricht des entsprechenden Levels ins Log zu schreiben. Im Falle von *logErrorMsg* wird ebenfalls der Fehler-Code geschrieben.

logDebugAsHex() Neben der Nachricht wird das BYTE-Array als Hex ins Log geschrieben. Diese Funktionalität ist für das Ausgeben von Hex-Dumps vorgesehen. In ihr werden keine weiteren Formatierungen vorgenommen.

7.6.8 stdOutLogger (Class)

Erbt vom *BaseLogger* und schreibt die Nachrichten auf die Standardausgabe std::cout (meist Command Prompt).

7.6.9 FileLogger (Class)

Erbt vom *BaseLogger* und schreibt die Nachrichten in eine Datei. Wenn keine Datei angegeben ist, sucht er in der Registry nach dem *logFilePath* und *logLevel* und verwendet diese entsprechend.

Für den Zugriff auf die Registry wird der *BaseRegistryWrapper* verwendet.

7.6.10 PolicyManager (Class)

Die Klasse *PolicyManager* verwaltet innerhalb des Trusted SHV die gültigen AIK Fingerprints. Ebenfalls werden die Hashs und deren Reihenfolge für die einzelnen PCR gespeichert und daraus die PCR-Maske (Siehe Kapitel 7.3.3).

Diese Klasse ermöglicht es, dass die erwähnten Daten nur einmal geladen werden und nicht bei jeder Prüfung erneut aus der Registry gelesen werden müssen. Für das Beziehen der AIK- resp. PCR-Policies stehen die Methoden *getAikPolicy()* und *getPcrPoliy()* zur Verfügung.

7.6.11 TrustedSHAconfig/TrustedSHVconfig

TrustedSHAconfig und TrustedSHVconfig sind Tools, mit dem die Konfigurationen auf dem Client- sowie dem Servercomputer vorgenommen werden. Die Folgenden Klassen existieren in beiden Projekten. Einzige Ausnahmen sind der *ArgumentParser*, der zwar gleich heisst, jedoch andere Entscheide zu den Optionen fällt, sowie der *Executer*. Im Falle des *Executer* fehlt beim TrustedSHVconfig-Tool die TPM-Funktionalität.

7.6.12 ToolOptions (Class)

Die Datenklasse *ToolOptions* speichert die den Tool mitgegeben Daten, sowie die vom *ArgumentParser* geparsten Argumente. So ist es möglich, dass ein Teil der Optionen in der Reihenfolge variieren kann.

7.6.13 ArgumentParser (Class)

Der *ArgumentParser* liest die beim Aufruf des Tools mitgegebenen Optionen und entscheidet anhand dessen, welche Operationen ausgeführt werden müssen. Er versucht die verschiedenen Argumente zu interpretieren und beim Erkennen eines Schlüsselwortes (add, remove, init, list, clearreg, getfp, getsha1, loglevel, logfile, setshapath, conversion) anhand der vorherigen Optionen die daraus resultierende Operation zu bestimmen.

Falls eine gültige Kombination erkannt wurde, wird der entsprechende Befehl vom *Executer* aufgerufen und die Aktion ausgeführt. Eine Anleitung zur Benutzung der TrustedSHAconfig-/TrustedSHVconfig-Tools mit den gültigen Parametern findet sich im Kapitel 8.3.

7.6.14 Executer (Class)

Der *Executer* stellt die Funktionalität für die ausgewerteten Parameter indirekt über andere Klassen zur Verfügung. Einerseits wird der *RegistryWriter* verwendet um die Konfigurationen in die Windows Registry zu schreiben beziehungsweise davon zu lesen. Andererseits werden die TPM Klassen *TpmInitOwner* und *PrivacyCaConnector* verwendet.

Im Folgenden werden die interessanteren Funktionen dieser Klasse kurz erwähnt. Daneben existieren Funktionen zum Lesen und Speichern der Einstellungen in der Registry.

initTpm() Diese Funktion initialisiert das TPM. Die im Verlauf abgefragten User und Owner Passwörter werden in ASCII-Codierung mit Null-Termination als SHA1-Hash der Funktion *takeOwnershipSha1* der Klasse *tpmInitOwner* übergeben.

requestNewAik() Generiert mit Hilfe der Klasse *PrivacyCaConnector* einen neuen AIK inklusive eines dazugehörigen Zertifikates.

7.6.15 TpmInitOwner (Class)

Die Klasse *tpmInitOwner* bietet einzig die Funktionalität, den Besitzer des TPMs zu setzen. Die Basis-Funktionalitäten erbt die Klasse von der Klasse *tpmObject*.

takeOwnershipSha1() Diese Funktion ermöglicht es, das Besitzer- und SRK-Passwort zu setzen. Beide Passwörter werden bereits als SHA1-Hash entgegengenommen.

Im Folgenden wird der Ablauf mit TrouSerS genauer erklärt (siehe Abbildung 7.6 und Code-Listing 7.11).

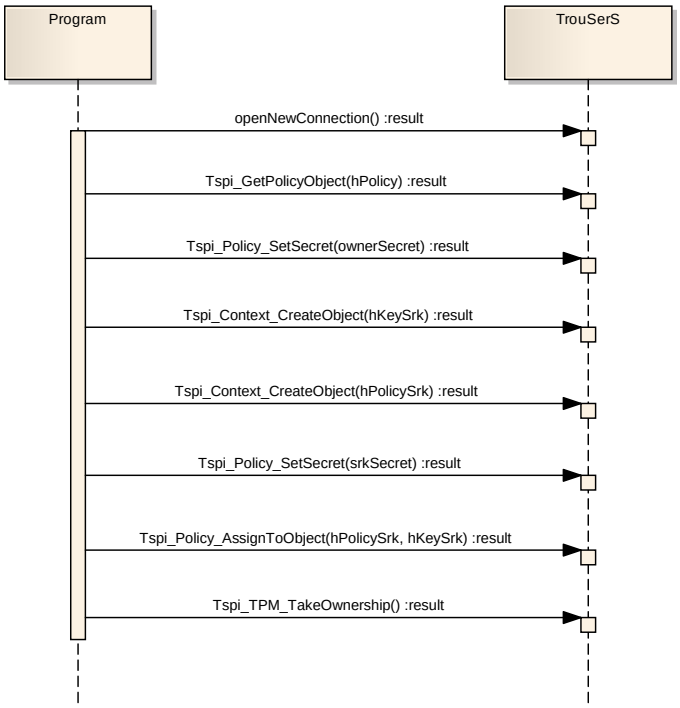


Abbildung 7.6: Ablauf TakeOwnership

1. *openNewConnection()*: Zuerst wird eine Verbindung zum TPM aufgebaut. Auf diese Methode wird genauer unter 7.3.2 eingegangen.
2. *Tspi_GetPolicyObject()*: Das Policyobjekt, welches dem *hTPM* (TPM-Handle) zugeordnet ist, wird zurückgegeben.
3. *Tspi_Policy_SetSecret()*: Das Besitzer-Passwort (OwnerSecret) wird gesetzt. Es wird dem Policy-Handle *hPolicy* zugewiesen.

4. *Tspi_Context_CreateObject()*: Ein leeres Key-Objekt (*hKeySRK*) vom Typ *TSS_OBJECT_TYPE_RSAKEY* wird erstellt. Dieses wird an den bereits bestehenden *hContext* gebunden. Mit den Keyflags werden noch zusätzliche Informationen mitgegeben. Dies sind die Folgenden:

   a) TSS_KEY_TSP_SRK: Bezeichnet, dass es sich bei diesem Schlüssel um einen SRK handelt.

   b) TSS_KEY_AUTHORIZATION: Die Verwendung des Schlüssels benötigt die Eingabe des SRK-Passworts. Dies ist notwendig, da sonst Fehler bei der nächsten Keyverwendung auftreten.

5. *Tspi_Context_CreateObject()*: Ein leeres Policy-Objekt (*hSrkPolicy*) vom Typ *TSS_OBJECT_TYPE_POLICY* wird erstellt. Dieses wird an den bereits bestehenden *hContext* gebunden.

6. *Tspi_Policy_SetSecret()*: Das Passwort für den SRK wird gesetzt. Dieses wird der *hSrkPolicy* zugewiesen.

7. *Tspi_Policy_AssignToObject()*: Der SRK-Handle (*hSRK*) wird der SRK-Policy zugewiesen.

8. *Tspi_TPM_TakeOwnership()*: Diese Methode führt schliesslich die Besitzübernahme durch. Alle vorher getätigten Schritte haben Einfluss auf diese Funktion (Keyflags, Owner- und SRK-Passwort).

```
Tpm\-Init\-Ow\-ner::takeOwnershipSha1(BYTE *ownerSecret, BYTE *srkSecret)
{
  TSS_HPOLICY hPolicy, hSrkPolicy;
  TSS_HKEY hKeySRK;
  TSS_FLAG keyFlagsSrk = TSS_KEY_TSP_SRK | TSS_KEY_AUTHORIZATION;

  openNewConnection();

  Tspi_GetPolicyObject(hTPM, TSS_POLICY_USAGE, &hPolicy);
  Tspi_Policy_SetSecret(hPolicy, TSS_SECRET_MODE_SHA1, 20, ownerSecret);
  Tspi_Context_CreateObject(hContext, TSS_OBJECT_TYPE_RSAKEY,
                            keyFlagsSrk, &hKeySRK);
  Tspi_Context_CreateObject(hContext, TSS_OBJECT_TYPE_POLICY,
                            TSS_POLICY_USAGE, &hSrkPolicy);
  Tspi_Policy_SetSecret(hSrkPolicy, TSS_SECRET_MODE_SHA1, 20, srkSecret);
  Tspi_Policy_AssignToObject(hSrkPolicy, hKeySRK);
  Tspi_TPM_TakeOwnership(hTPM, hKeySRK, (TSS_HKEY)NULL);

  closeConnection();
}
```

Listing 7.11: Code zu takeOwnershipSha1

7.7 Fehler während der Implementation

Hier wird kurz auf die Aspekte eingegangen, welche während der Implementation ungewöhnlich viel Zeit in Anspruch nahmen.

7.7.1 Einbezug eines TSS

Zu Beginn des Projektes wurde versucht, die notwendigen Funktionen des TSS selber zu implementieren. Nachdem bemerkt wurde, dass die Implementation unverhältnismässig viel Zeit benötigt, wurde dies aufgegeben und nach einer Alternative gesucht. Diese Alternative war schliesslich TrouSerS. Durch diesen Sachverhalt gingen rund zwei Wochen Arbeitszeit verloren.

7.7.2 Internal error, source unknown

Bei der Ausführung des *Tspi_Context_LoadKeyByBlob()*-Befehls von TrouSerS kam es zu Beginn zum Fehler *0x00002002*, was *An internal error has been detected, but the source is unknown* bedeutet. Anhand dieses nichtsaussagenden Fehlers ist es sehr schwierig nachzuvollziehen, wodurch der Fehler hervorgerufen wurde. Die Fehlerbehebung dauerte lange, denn der Fehler wurde an einem anderen Ort verursacht, und zwar in der Methode *TpmInitOwner::takeOwnershipSha1()*. Wenn man diese genauer unter die Lupe nimmt kann festgestellt werden, dass in dieser Methode keyFlags für den Umgang für dem SRK gesetzt werden. Der dort verwendete Befehl

$$keyFlags = TSS\_KEY\_TSP\_SRK;$$

war unvollständig.

Mit dem zusätzlichen Flag *TSS_KEY_AUTHORIZATION* wird angegeben, dass für alle Operationen des SRKs ein Passwort benötigt wird. Ohne dieses kommt es später (bei gesetztem Passwort) zu Fehlern. Der vollständige Befehl ist demnach:

$$keyFlags = TSS\_KEY\_TSP\_SRK|TSS\_KEY\_AUTHORIZATION;$$

7.7.3 Extend

Die Programmierung des Befehls zur Überprüfung der Messwerte *Verifier::doVerification()* muss dieselben Operationen wie das TPM durchführen. Die Formel für die Extend-Operation ist bekannt, doch wurden auf Verifier immer andere Hashwerte berechnet. Der Fehler lag darin, dass die Daten, welche an den bisherigen PCR-Wert angehängt werden, zuerst gehasht werden müssen, sodass zwei 20 Byte grosse Hashes konkateniert werden können und schliesslich wieder ein Hash erzeugt werden kann.

$$PCR_i = SHA1(PCR_i||newValue)$$

Somit ist der korrekte Befehl für die Verifikation:

$$PCR_i = SHA1(PCR_i||SHA1(data))$$

Kapitel 8

Bedienungsanleitung

8.1 Installation

Für die Installation des Trusted SHA und Trusted SHV wird von einer funktionierenden NAP-Umgebung ausgegangen. Eine solche könnte beispielsweise durch den *Step-by-Step Guide: Demonstrate NAP IPsec Enforcement in a Test Lab*[6] aufgebaut werden. Anschliessend kann der Server mit der neuen Funktionalität erweitert werden.

8.1.1 Client - Trusted SHA

Auf dem Client müssen folgende Schritte für das Einrichten des Trusted SHAs vorgenommen werden:

Zurücksetzen und Aktivieren des TPMs Zu Beginn muss das TPM im BIOS zurückgesetzt und aktiviert werden. Eine Anleitung für dieses Vorgehen wird in den Kapiteln 8.4.2 und 8.4.1 gegeben.

TrouSerS installieren Danach wird der TrouSerS Installer, welcher auf der Projekt-CD zu finden ist, gestartet und ist Konfigurationsänderung zu durchlaufen. Danach ist ein Neustart des Clients notwendig.

Initialisierung durchführen Mit dem Befehl

TrustedSHAconfig init

werden folgende Konfigurationen durchgeführt:

- Registrystruktur erstellen
- AIK generieren und angefordertes AIK-Zertifikat speichern

- TPM mit *TPM_TakeOwnership* initialisieren

Für die TPM-Initialisierung werden ein frei wählbares Owner- und User-Password (SRK) abgefragt. Während das User Password in der Registry gespeichert wird, wird das Owner Password nirgends hinterlegt.

Die generierten Dateien *tpm.pem* und *tpmAIK.blob* müssen sich im selben Verzeichnis wie die Datei *TrustedSha.exe* befinden. Dies ist automatisch der Fall, falls das TrustedSHAconfig-Tool aus dem SHA-Verzeichnis ausgeführt wurde.

Logfile-Pfad setzen Mit

TrustedNAPconfig logfile "C:\logfile.log"

wird der Pfad zum Logfile gesetzt werden.

Logging-Level setzen Mit

TrustedNAPconfig loglevel 2

werden alle Fehlermeldungen und Informationen in das Logfile geschrieben.

Pfad zu TrustedSHA setzen Mit

TrustedSHAconfig setshapath "C:\TrustedNAP"

wird das Verzeichnis zum SHA in der Registry hinterlegt. Dabei wird überprüft, ob sich eine Datei namens TrustedSHA.exe in jenem Verzeichnis befindet.

Trusted SHA Info installieren Auf dem Client muss die Datei *TrustedShaInfo.dll* registriert werden. Dafür muss mit einem Command Prompt mit Administratorenrechten gestartet werden, in das Verzeichnis des Trusted SHA gewechselt werden und dort mit

regsvr32 TrustedShaInfo.dll

registriert werden. Der erfolgreiche Vorgang wird mit einer erfolgreichen Bestätigung quittiert.

Trusted SHA Service installieren Als Nächstes wird mit einer mit Administratorrechten ausgestatteten Command Prompt der Trusted SHA als Service installiert. Dazu muss in den Ordner gewechselt werden. Dort angekommen, wird der Befehl

TrustedSHA.exe -i

ausgeführt. Die Ausführung dieses Befehls wird mit einer Bestätigung quittiert. Der Service muss nach der Installation einmalig manuell gestartet werden. Der Service ist unter *services.msc* und kann unter dem Namen *HSR Trusted System Health Agent* gefunden werden. Alternativ kann auch der Computer neu gestartet werden, um den Service zu starten.

Zu messende Dateien hinzufügen Nun werden die zu überprüfenden Dateien hinterlegt. Für die Überprüfung der NAP-Dateien sind dies die Dateien, welche in Kapitel 5.2.2 genannt wurden. Es können aber auch andere Dateien für die Überprüfung ausgewählt werden. Es ist wichtig, dass dies die selben Dateien und in der selben Konfiguration wie auf dem Server getätigt wird. Sonst schlägt die Verifikation fehl. Um die Dateien in der Konfiguration hinzuzufügen, wird der Befehl

TrustedSHAconfig add 15.1 "C:\TrustedNAP\TrustedSHA.exe"

benutzt. Dieser Befehl wird nun mit jeder zu messenden Datei durchgeführt.

Version der Konfiguration definieren Für die angegebene Konfiguration muss noch die Version der Konfiguration definiert werden. Sie soll mit dem Server immer gleich sein. Sind sie nicht gleich, so schlägt die Verifikation fehl. Darum muss diese bei jeder Konfigurationsänderung geändert werden. So ist schnell ersichtlich, falls Server und Client verschiedene Konfigurationen aufweisen.

TrustedSHAconfig confversion 1.01

8.1.2 Server - Trusted SHV

Auf dem Network Policy Server (NPS) (in der Testumgebung NPS1 genannt) müssen folgende Schritte durchgeführt werden:

Registry Struktur erstellen Als Nächstes wird nun der Trusted SHV konfiguriert. Dazu wird eine Command Prompt mit Administratorrechten benötigt. Mit dem Befehl

TrustedSHVconfig init

wird die Initialisierung der Registry vollzogen.

Logfile-Pfad setzen Mit

TrustedSHVconfig logfile "C:\logfile.log"

wird der Pfad zum gewünschten Logfile gesetzt.

Logging-Level setzen Mit

TrustedSHVconfig loglevel 2

werden alle Fehlermeldungen und Informationen in das Logfile geschrieben.

Messwerte hinterlegen Nun werden die korrekten Messwerte hinterlegt. Für die Überprüfung der NAP-Dateien sind dies die Dateien, welche in Kapitel 5.2.2 genannt wurden. Es können aber auch andere Dateien für die Überprüfung ausgewählt werden. Auf einem Client, der als infektionslos gilt, müssen diese SHA1-Hashwerte (falls sie noch nicht vorliegen) berechnet werden. Dafür wird der Befehl

TrustedSHAconfig getsha1 "C:\Windows\SysWOW64\mssha.dll"

benutzt. Diese Werte werden nun verwendet, um sie für die Verifikation serverseitig zu hinterlegen. (Da sich diese Dateien beispielsweise durch Updates verändern können ist es wichtig, dass die aktuellen verwendeten Dateien gehasht werden.)

Die Hashes werden mit dem Aufruf

TrustedSHVconfig
add 15.1 [File-Checksum]

hinzugefügt, wobei 15 das PCR und das 1 die Reihenfolge der extend-Vorgänge beschreibt. Die [File-Checksum] wird in der Form 61d028ece6c74f44bb71c86bbe2cd624cadecf0b angegeben. Es ist wichtig, dass dies auf Client und Serverseite gleich ist; d.h. das PCR 15.1 auf Clientseite muss dem nun eingetragenen Hash entsprechen. Alle weiteren Dateien werden mit 15.2, 15.3 etc. hinzugefügt.

Vertrauenswürdige AIK-Zertifikate hinterlegen Anschliessend müssen noch die gültigen AIK-Zertifikat-Fingerprints hinterlegt werden. Auf dem Client, der bereits ein AIK-Zertifikat erhalten haben muss, wird der AIK-Fingerprint mittels

TrustedSHAconfig getfp "C:\TrustedNAP\tpm.pem"

ausgelesen. Dieser wird nun mit

TrustedSHVconfig aik add client1 [AIK-Fingerprint]

hinzugefügt, wobei client1 die Bezeichnung für den Client ist. Diese wird bei der Verifikation nicht überprüft und dient lediglich dem Systemadministrator um nachvollziehen zu können, zu welchem Computer das AIK-Zertifikat gehört. Der [AIK-Fingerprint] muss in der Form 61d028ece6c74f44bb71c86bbe2cd624cadecf0b eingegeben werden.

Version der Konfiguration definieren Für die angegebene Konfiguration muss noch die Version der Konfiguration definiert werden (so wie auf dem

Client). Sie muss gleich sein wie auf dem Client, sonst schlägt die Verifikation fehl.

TrustedSHVconfig confversion 1.01

Leserechte für Network Service setzen Die TrustedShv.exe benötigt lesenden Zugriff auf die Registry. Die Library läuft als *Network Service*. Um diesen Zugriff zu ermöglichen muss der Registry Editor gestartet werden. Dies geschieht mit *Windows-Taste + R* und der Eingabe von *regedit*. Dort wird in den Registryschlüssel *HKEY_LOCAL_MACHINE/SOFTWARE/Wow6432Node/HSR*, bzw. *HKEY_LOCAL_MACHINE/SOFTWARE/-HSR* für 32 bit Systeme, gewechselt. Auf den Schlüssel *HSR* wird ein Rechtsklick getätigt und auf *Permissions...* geklickt (siehe Abbildung 8.1).

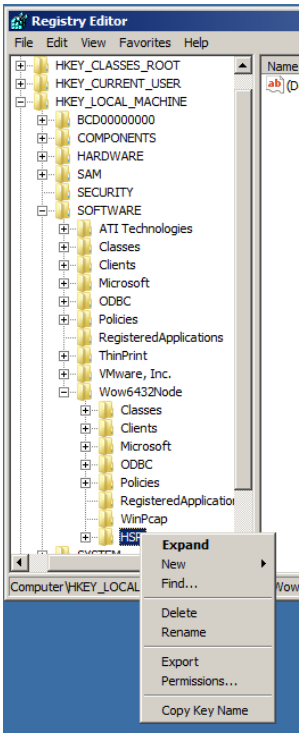


Abbildung 8.1: HSR-Schlüssel in der Registry

Dort wird mit *Add...* ein neues Dialogfenster (Select Users, ...) geöffnet. In diesem wird nun unter *Enter the object names to select* die Bezeichnung *Network Service* eingegeben und auf *Check Names* geklickt. Nun erscheint der eingegebene Text in Grossbuchstaben und unterstrichen. Mit *OK* wird die Auswahl bestätigt. Für den Eintrag *NETWORK SERVICE* muss nun unter *Read* das Häkchen für Allow gesetzt werden (siehe Abbildung 8.2).

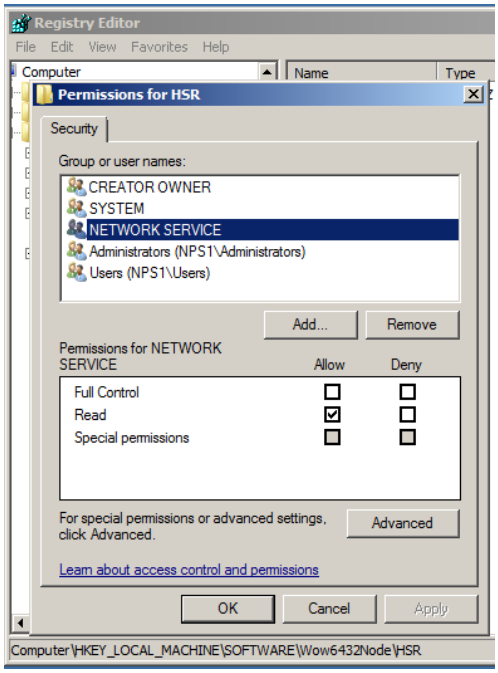


Abbildung 8.2: Vergabe der Leserechte für NETWORK SERVICE am Registry-Key HSR

Schreibrechte auf Logging-File für Network Service setzen Damit der TrustedShv Service die Rechte besitzt, die Logdatei anzupassen, müssen die Dateirechte entsprechend gesetzt werden. Dazu muss mit dem Windows Explorer in das Verzeichnis, in welchem sich die Logdatei befindet, gewechselt werden. Falls die Logdatei noch nicht existiert, so muss eine leere Datei angelegt werden. Auf diese wird nun ein Rechtsklick getätigt, und auf *Properties* geklickt. Im daraufhin öffnenden Dialogfenster mit einem Klick auf *Security*, einem Klick auf *Edit* und einem Klick auf *Add...* wird ein neues Dialogfenster (Select Users, ...) geöffnet. In diesem wird nun unter *Enter the object names to select* die Bezeichnung *Network Service* eingegeben und auf *Check Names* geklickt. Nun erscheint der eingegebene Text in Grossbuchstaben und unterstrichen. Mit *OK* wird die Auswahl bestätigt. Für den Eintrag *NETWORK SERVICE* muss nun unter *Modify* das Häkchen für Allow gesetzt werden.

Trusted SHV installieren Auf dem Server muss die Datei *TrustedShv.dll* im folgenden Verzeichnis abgelegt werden: *C:\Windows\SysWOW64*, bzw. *C:\Windows\System32* für 32 bit Betriebssysteme. Als Nächstes wird eine Command Prompt mit Administratorenrechten gestartet. In jener wird in das Verzeichnis *C:\Windows\SysWOW64* (bzw. *C:\Windows\System32* für 32 bit Betriebssysteme) navigiert. Dort wird nun mit *regsvr32.exe TrustedShv.dll* der Dienst aktiviert. Der erfolgreiche Vorgang wird mit

einer Bestätigung (Abb. 8.3) quittiert.

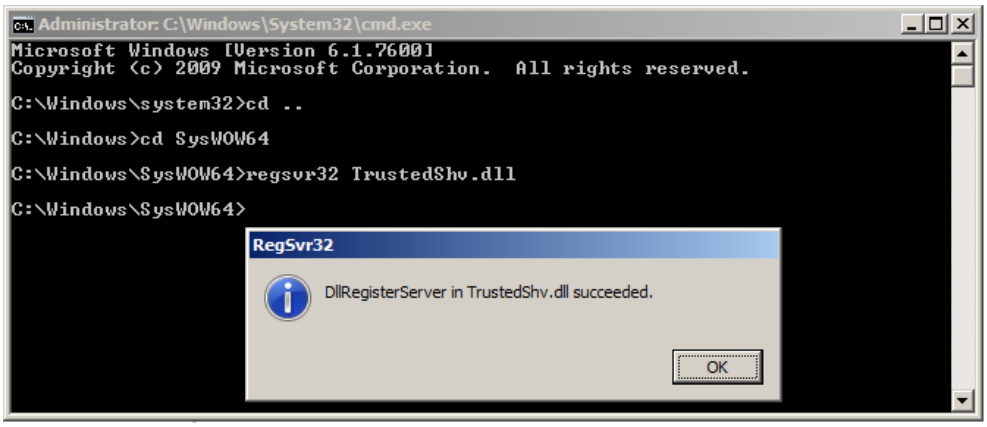


Abbildung 8.3: Erfolgreiches Registrieren der TrustedShv.dll mit Administratorrechten

Trusted SHV Policy aktivieren Im nächsten Schritt wird aus den *Administrative Tools* der *Network Policy Server* gestartet. Unter Policies / Health Policies müssen nun die geltenden Policies um den neuen Trusted SHV erweitert werden. In der Testumgebung sind dies die Compliant- und die Noncompliant Policy. Um diese zu verändern wird die erste Policy mit einem Doppelklick geöffnet. Dort muss nun unter *SHVs used in this health policy* der *Trusted SHV* mit anhaken aktiviert werden (Abb. 8.4). Dieser Schritt muss nun noch bei den übrigen Policies durchgeführt werden.

Anmerkung: Nachdem eine Konfigurationsänderung erfolgte, muss die TrustedShv.dll neu geladen werden. Dies erfolgt mit den folgenden Befehlen (mit einer Command Prompt im Verzeichnis C:\Windows\SysWOW64\):

regsvr32.exe -u TrustedShv.dll

regsvr32.exe TrustedShv.dll

8.2 Deinstallation

8.2.1 Client - Trusted SHA

TrustedSha-Service deinstallieren Der TrustedSha-Service wird über eine mit Administratorrechten ausgestattete Command Prompt (im Verzeichnis der SHA-Dateien) mit

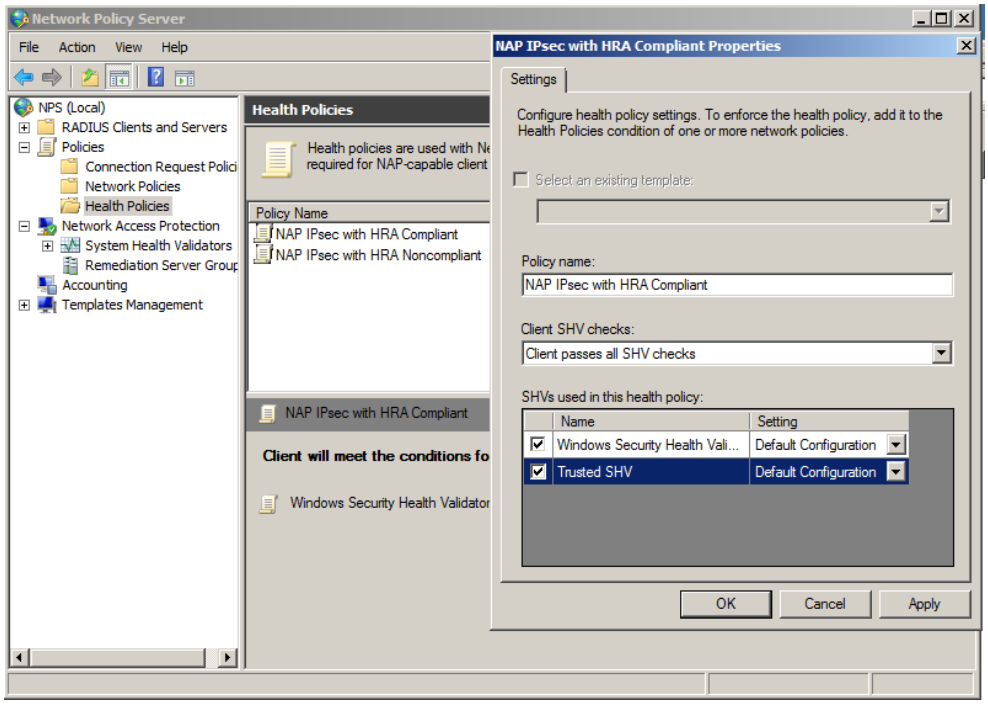


Abbildung 8.4: Aktivieren des Trusted SHV innerhalb der Network Policy Server Health Policies

TrustedSha.exe -u

deinstalliert.

TrustedShaInfo.dll deregistrieren Die Deregistrierung der *TrustedShaInfo.dll* erfolgt mit dem folgenden Befehl:

regsvr32.exe -u TrustedShaInfo.dll

Registry aufräumen Ein Command Prompt mit Administratorrechten öffnen und in das Verzeichnis navigieren, wo sich TrustedSHAconfig befindet. Dort angekommen ist folgender Befehl einzugeben, um die Konfigurationen aus der Registry zu löschen:

TrustedSHAconfig.exe clearreg

TrouSerS deinstallieren Die Deinstallation erfolgt über das Windows Deinstallationsprogramm.

TPM löschen Um das TPM zurückzusetzen, muss in das BIOS gewechselt werden. Dies ist in Kapitel 8.4.1 beschrieben.

8.2.2 Server - Trusted SHV

Policies deaktivieren Die gesetzten Policies im *Network Policy Server* sind zu deaktivieren.

Trusted SHV deregistrieren In einer Konsole (mit Administratorrechten) auf dem NPS in das Verzeichnis wechseln, in welchem sich TrustedShv.dll befindet.

regsvr32 -u TrustedShv.dll

Registry aufräumen Ein Command Prompt mit Administratorrechten öffnen und in das Verzeichnis navigieren, wo sich TrustedSHVconfig befindet. Dort angekommen ist folgender Befehl einzugeben, um die Konfigurationen aus der Registry zu löschen:

TrustedSHVconfig.exe clearreg

8.3 Konfigurieren mit TrustedSHAconfig/TrustedSHVconfig

TrustedSHAconfig.exe bzw. TrustedSHVconfig.exe wird dazu verwendet, um die Trusted NAP-Umgebung einzurichten und zu konfigurieren. Es kann aus dem Command Prompt (cmd) sowie durch ein Batch-File verwendet werden. Voraussetzung für die Ausführung des Tools auf dem Client ein installierter TrouSerS-Service. Dies ist notwendig, da TPM-Funktionalitäten angeboten werden, die auf TrouSerS zurückgreifen.

> *Es ist wichtig, dass die Ausführung des Command Prompt mit Administratorenrechten erfolgt, da sonst die Befehle aus Rechtegründen nicht vollständig durchgeführt werden können. Wie dies funktioniert wird in Kapitel 8.7.2 erläutert.*

Sollten im Umgang mit dem Config-Tool Fehlercodes zurückgegeben werden, so können diese im Anhang in Kapitel G nachgeschlagen werden.

Die angebotenen Funktionalitäten sollen nachfolgend aufgezeigt werden.

Anmerkung: Die Angabe Trusted*config bedeutet, dass für den Befehl TrustedSHAconfig bzw. TrustedSHVconfig eingegeben werden kann.

8.3.1 Allgemeine Konfigurationen

Hilfe in der Konsole

In der Konsole kann die Hilfe ebenfalls abgerufen werden.

Mit dem Befehl

*Trusted*config help*

wird die Kurzhilfe angezeigt.

Mit

*Trusted*config longhelp*

bzw.

*Trusted*config longhelp*

wird die erweiterte Hilfe angezeigt.

Loglevel

*Trusted*config loglevel [grad]*

Mit diesem Befehl wird in der Registry gespeichert, mit welchem Detaillierungsgrad Protokollierungen geschrieben werden sollen. Dabei sind folgende Grade möglich:

- 0: Keine Logs schreiben
- 1: Nur Fehler protokollieren
- 2: Fehler und Infoausgaben protokollieren
- 3: Fehler, Infoausgaben und Debuginformationen ausgeben

Beispielsweise kann der folgende Befehl ausgeführt werden, um lediglich Fehler zu protokollieren:

*Trusted*config loglevel 1*

Konfiguration Logdatei

*Trusted*config logfile "[path]"*

Hiermit kann der Pfad der Logfile eingestellt werden. Die Angabe des Pfades muss mit Anführungszeichen erfolgen.

*Bsp: Trusted*config logfile "C:\TrustedNAP\Logs\TrustedSha.log"*

Pfad zu den SHA-Dateien

TrustedSHAconfig setshapath "[path]"

Wird verwendet, um den Pfad zum SHA-Verzeichnis zu definieren. Die Angabe des Pfades muss mit Anführungszeichen erfolgen.

Bsp: TrustedSHAconfig setshapath "C:\TrustedNAP\"

Registrykonfiguration löschen

*Trusted*config clearreg*

Hiermit werden alle Konfigurationen, welche durch das Trusted*config-Tool erstellt wurden, aus der Registry gelöscht.

Fingerprint aus Zertifikat lesen

TrustedSHAconfig getfp "[path]"

Erzeugt und gibt den Fingerprint des AIK-Zertifikats ([path]) aus. Die Angabe des Pfades muss mit Anführungszeichen erfolgen.

Bsp: TrustedSHAconfig getfp "C:\TrustedNAP\tpm.pem"

SHA1-Hash einer Datei erstellen

TrustedSHAconfig getsha1 "[path]"

Erzeugt Checksumme einer Datei als SHA-1-Hash und gibt diesen aus. Die Angabe des Pfades muss mit Anführungszeichen erfolgen.

Bsp: TrustedSHAconfig getsha1 "C:\Windows\SysWOW64\mssha.dll"

Konfigurationsversion setzen

*Trusted*config confversion [vers]*

Setzt die Version der Messdateikonfiguration. Es ist wichtig, dass auf dem Server und dem Client dieselbe Konfigurationsversion eingetragen ist, da sonst die Verifikation fehlschlägt. Für [vers] wird eine Versionszahl eingegeben.

*Trusted*config confversion 1.36*

8.3.2 Konfigurationen Trusted SHA / Trusted SHV

generell erfolgt die Konfiguration des Trusted SHA und des Trusted SHV nach dem folgenden Muster:

*Trusted*config OPTION OPERATION*

Folgende Parameter können dafür gewählt werden:

OPTION

- pcr
- aik

OPERATION

- list
- add [key] [value]
- remove [key]

Folgend werden die damit möglichen Kommandos aufgezeigt:

Konfigurationen auflisten

*Trusted*config [pcr/aik] list*

Mit diesem Befehl können die in der Registry hinterlegten Konfigurationen im Command Prompt aufgelistet werden. Folgende Befehlskombinationen sind möglich:

Befehl	Funktionalität
TrustedSHAconfig list	Listet SHA-Konfigurationen
TrustedSHAconfig pcr list	Listet die zu messenden Daten und PCRs
TrustedSHVconfig pcr list	Listet die SHA-1-Hashes der Messdateien inkl. PCR
TrustedSHVconfig aik list	Listet die vertrauenswürdigen AIK-Fingerprints

Tabelle 8.1: List-Befehlsübersicht

Key hinzufügen

*Trusted*config [pcr/aik] add [key] [value]*

SHA / PCR Auf dem Trusted SHA kann eingegeben werden, welche Dateien auf welchen PCRs in welcher Reihenfolge gehasht werden. Dabei wird wie folgt vorgegangen:

Bsp: TrustedSHAconfig pcr add 15.2 "C:\Windows\SysWOW64\mssha.dll"

Der Key muss im Format [pcrNo].[order] vorliegen. Auf das Beispiel bezogen bedeutet dies, dass die Messung dieser Datei als Zweites auf das PCR 15 ausgeführt wird. Der Value ist der Pfad auf die Datei, welche gemessen werden soll.

SHV / PCR Mit diesem Befehl können die Hashwerte für die Validierung hinzugefügt werden. Der Key ist gleich aufgebaut wie bei SHA / PCR. Der Value hingegen muss als SHA1-Hash der zu messenden Datei angegeben werden.

Bsp: TrustedSHVconfig pcr add 15.2 c78f346c2b0df02cba03794911050a7-18ca8aa20

SHV / AIK Dieser Befehl ermöglicht das Hinzufügen von validen Clientzertifikat-Fingerprints. Als Name kann ein beliebiger Bezeichner für den Client verwendet werden. Als Value wird der Fingerprint des Zertifikats angegeben.

Bsp: TrustedSHVconfig aik add client1 f1d6b6340bbd40cdc3637cbe10ff7f6-d49754a73

8.3.3 Key löschen

*Trusted*config [pcr/aik] remove [key]*

Mit remove [key] wird eine in der Registry hinterlegte Konfiguration gelöscht.

Um beispielsweise eine gemessene Datei nicht mehr in die Messung einfliessen zu lassen kann folgender Befehl benutzt werden:

Bsp: TrustedSHAconfig pcr remove 15.2

Hinweis: bei Veränderungen der PCR-Messung muss darauf geachtet werden, dass die Änderungen bei Client und Server getätigt werden, da sonst die Messungsberechnung und die Validierungsberechnung nicht die selben Werte ergeben.

8.3.4 Initialisierung

Initialisierung (Level 0)

SHA Der Befehl

TrustedSHAconfig init

bewirkt eine vollständige Initialisierung. Voraussetzung für diesen Befehl ist ein zurückgesetztes und aktiviertes TPM. Auf dem TPM wird ein Besitzer (Take_Ownership) gesetzt. Das User Passwort (Passwort für den SRK) wird in der Registry als SHA1 abgelegt. Danach wird ein AIK-Schlüssel generiert und ein Level 0 AIK-Zertifikat angefordert. Als Nächstes wird die Registrystruktur erstellt und abschliessend der Pfad zu den SHA-Dateien in der Registry abgespeichert. Für diesen Befehl wird ein bereits durch das TrustedNAPconfig-Tool initialisiertes TPM sowie eine Internetverbindung vorausgesetzt.

SHV Auf dem Verifier wird mit

TrustedSHVconfig init

die Registrystruktur erstellt. Jede weitere individuelle Konfiguration ist mit weiteren Befehlen zu erreichen.

Initialisierung (Level 1) - experimentell

Diese Funktion ist nur für experimentelle Zwecke zu verwenden.

Mit

TrustedSHVconfig init realek

wird wie bereits im Abschnitt mit der Level 0 Initialisierung beschrieben, eine vollständige Initialisierung vorgenommen. Der AIK-Zertifikat-Request wird allerdings für ein Level 1 Zertifikat durchgeführt. Es wird das EK-Zertifikat des TPMs verwendet, um eine Verifizierung des TPMs durchzuführen. Dabei muss allerdings beachtet werden, dass nicht alle TPM diese Voraussetzung erfüllen. Derzeit besitzen lediglich TPMs der Firma Infineon solche Zertifikate. Für diesen Befehl wird ein bereits durch das TrustedSHV-config-Tool initialisiertes TPM sowie eine Internetverbindung vorausgesetzt.

Initialisierung einzelner Teile

*Trusted*config init INITOPTION*

Dieser Befehl führt eine Initialisierung einzelner Teile durch. Für INITOPTION können dabei einer oder mehrere der folgenden Werte eingesetzt werden:

INITOPTION

- tpm (nur für sha verfügbar)

- aik (nur für sha verfügbar)
- aik realek (nur für sha verfügbar)
- registry

tpm

TrustedSHAconfig init tpm

Initialisiert das TPM und schreibt den SRK-Zugriffsschlüssel in die Registry. Das TPM muss sich in einem zurückgesetzten und aktivierten Zustand befinden. Diese Funktion ist nur mit dem Keyword sha nutzbar.

aik

TrustedSHAconfig init aik

Erzeugt einen neuen AIK und fordert einen neues Level 0 AIK Zertifikat an. Diese Funktion ist nur mit sha nutzbar. Für diesen Befehl wird ein bereits durch das TrustedNAPconfig-Tool initialisiertes TPM sowie eine Internetverbindung vorausgesetzt.

aik realek - experimentell

TrustedSHAconfig init aik realek

Dasselbe wie aik, nur dass ein neues Level 1 AIK Zertifikat angefordert wird. Diese Funktion ist nur mit sha nutzbar.

registry

*Trusted*config init aik*

Erstellt die Registrystruktur für die weiteren Konfigurationen.

8.4 TPM-Konfiguration im BIOS

Damit das TPM benutzt werden kann, muss es im BIOS aktiviert werden. Denn es ist im Auslieferzustand standardmässig deaktiviert. Um es zu aktivieren muss ins BIOS des Computers gewechselt werden.

Hier wird anhand eines Fujitsu Siemens Computers gezeigt, wie das TPM zurückgesetzt bzw. aktiviert werden kann.

8.4.1 Zurücksetzen des TPMs

Das TPM muss zurückgesetzt werden, falls ein anderer Besitzer (Owner) für das TPM definiert werden soll oder das Benutzungspasswort nicht mehr vorhanden ist.

1. Als Erstes muss ins BIOS gewechselt werden. Dies wird mit F2 erreicht (je nach BIOS erfolgt dies durch eine andere Taste).
2. Danach muss ins Register *Security* gewechselt werden. Dort ist dann *TPM (Security Chip) Setting* auszuwählen (siehe Abbildung 8.7).
3. Die nun neu erscheinende Eigenschaft *Change TPM State* ist auf *Clear* zu setzen (Abbildung 8.5).

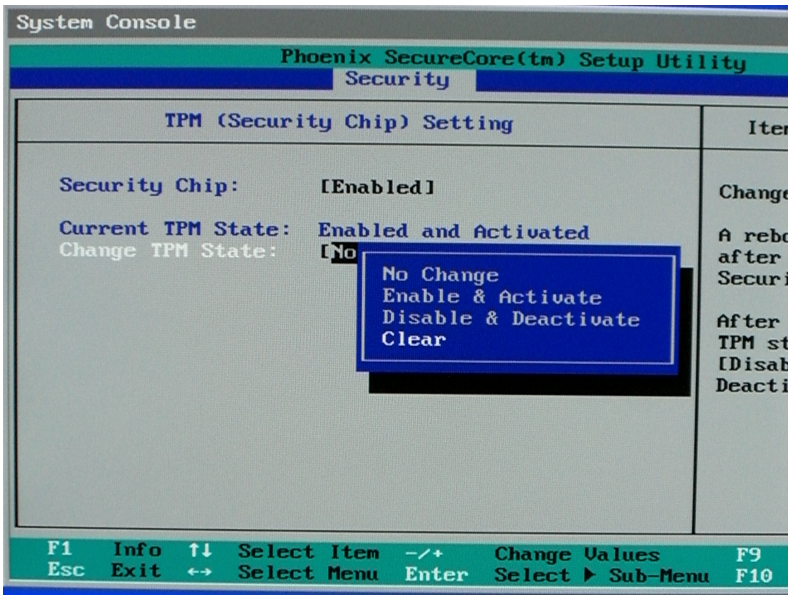


Abbildung 8.5: TPM zurücksetzen auswählen

4. Danach ist die Konfiguration zu speichern und mit F10 das BIOS-Menü zu verlassen.
5. Nach dem automatischen Neustart wird ein BIOS-ähnliches Fenster angezeigt, welches eine Bestätigung zum Rücksetzen des TPMs erfordert (Abb. 8.6). Dies wird mit *Execute* bestätigt.
6. Das TPM ist nun zurückgesetzt, aber standardmässig wieder deaktiviert. Aus diesem Grund muss nun das Kapitel 8.4.2 durchlaufen werden.

Es wird empfohlen, für das BIOS ein Passwort zu setzen, damit unbefugte Personen das TPM nicht zurücksetzen oder deaktivieren können.

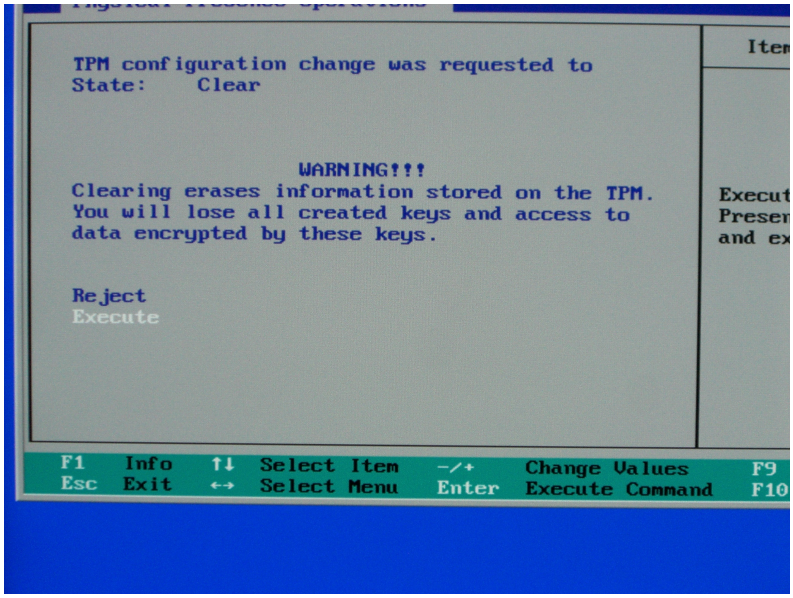


Abbildung 8.6: TPM aktivieren

8.4.2 Einschalten des TPMs im Bios

Um das TPM zu aktivieren müssen folgende Schritte beachtet werden.

1. Als Erstes muss ins BIOS gewechselt werden. Dies wird mit F2 erreicht (je nach BIOS erfolgt dies durch eine andere Taste).
2. Danach muss ins Register *Security* gewechselt werden. Dort ist dann *TPM (Security Chip) Setting* auszuwählen (siehe Abbildung 8.7).
3. Die nun neu erscheinende Eigenschaft *Change TPM State* ist auf *Enabled & Activate* zu setzen (Abbildung 8.8).
4. Danach ist die Konfiguration zu speichern und mit F10 das BIOS-Menü zu verlassen.
5. Das TPM ist nun aktiviert und kann verwendet werden.

8.5 Analysetools von Windows für NAP und TPM

8.5.1 Trusted Platform Module (TPM) Management

Dies ist die Verwaltungszentrale für das TPM im Windows. Allerdings wird dieses TPM-Managementinterface durch Trusted NAP nicht unterstützt, da Trusted NAP direkt via TrouSerS auf das TPM zugreift und dadurch die Einstellungen in dieser Umgebung nicht aktuell sind.

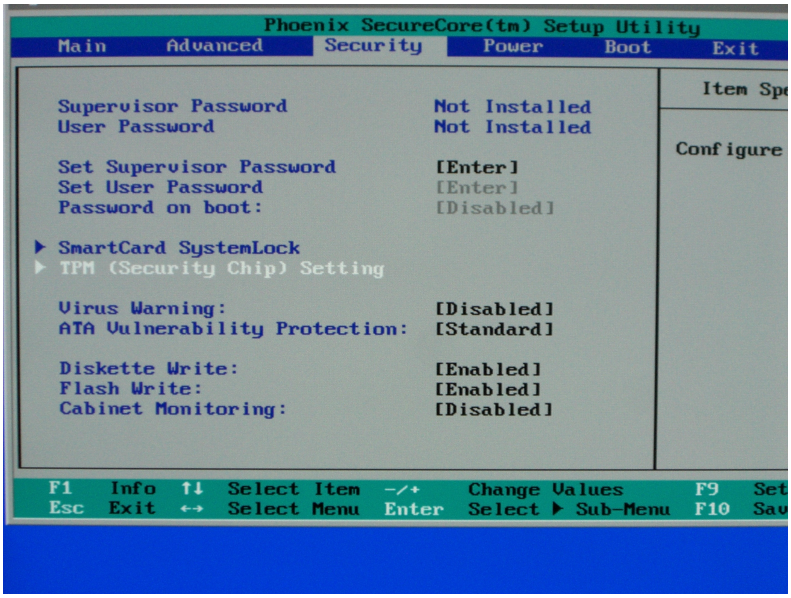


Abbildung 8.7: Wechsel ins Reiter Security des BIOS

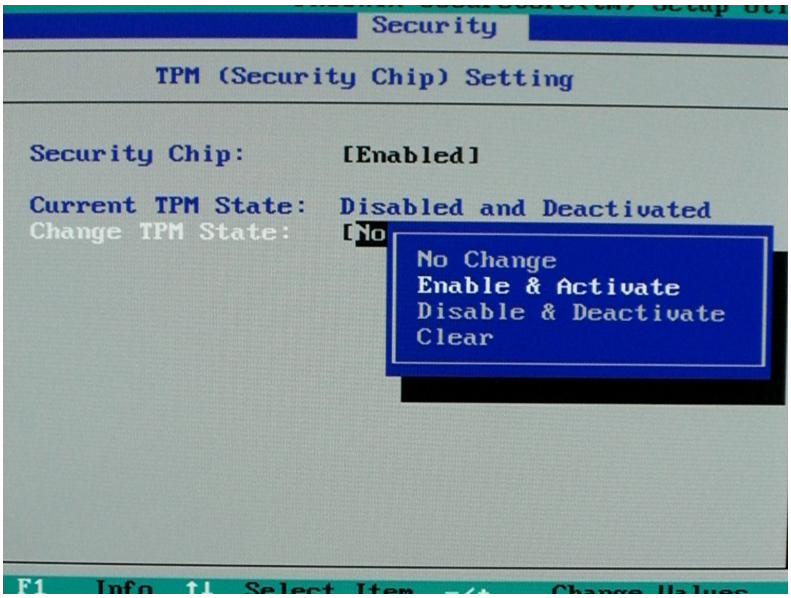


Abbildung 8.8: TPM einschalten und aktivieren

Dieses Kontrollzentrum kann mit dem Drücken von *Win + R* und der Eingabe von *msc.tpm* geöffnet werden. Nun ist es möglich die aktuelle Konfiguration einzusehen und Änderungen anzubringen. In der linken Spalte können unter *Command Management* die Gruppenrichtlinieneinstellungen geöffnet werden (s. Abbildung 8.10). Dort kann überprüft werden, ob alle TPM-Funktionalitäten verfügbar sind, oder ob welche gesperrt wurden.

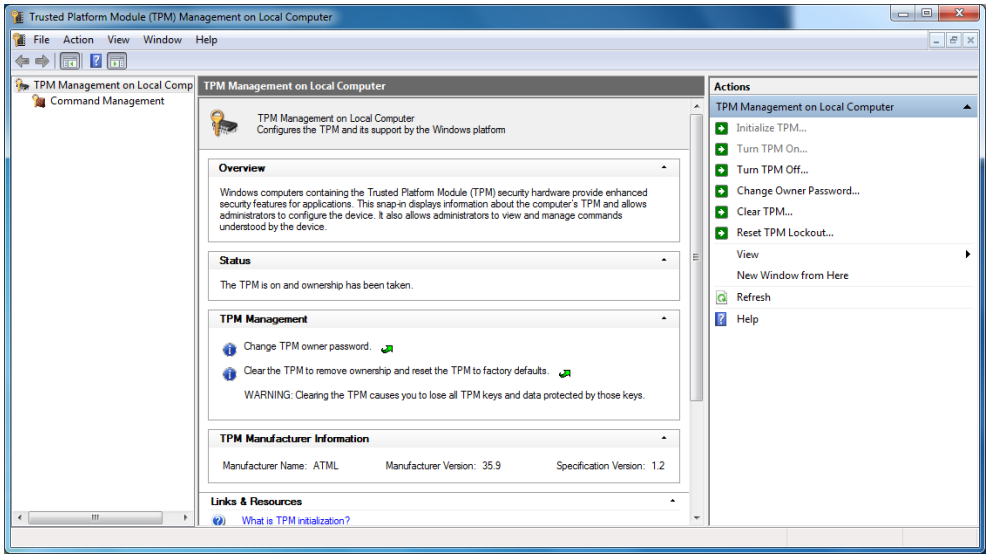


Abbildung 8.9: TPM Managementumgebung

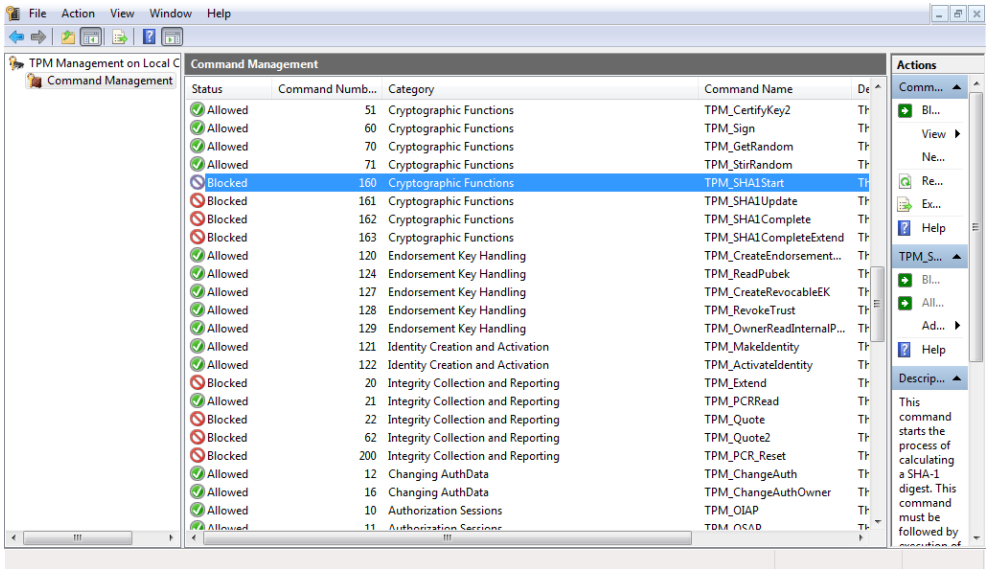


Abbildung 8.10: Von der Gruppenrichtlinie blockierte Befehle

8.5.2 Napstat

Napstat vermittelt dem Benutzer den Konfirmitätsstatus seines Computers. Bei einem nicht konformen Computer wird unten rechts automatisch ein kleines Fenster (Abbildung 8.11) angezeigt. Bei einem konformen Computer kann die Ansicht dieses Fensters durch *Win* + *R* sowie durch die Eingabe von *napstat* erzwungen werden (siehe Abbildung 8.12).

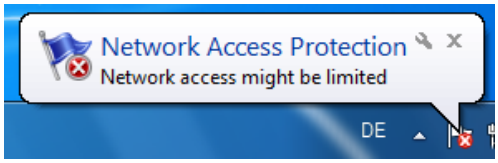


Abbildung 8.11: napstat auf einem *nicht konformen* Rechner

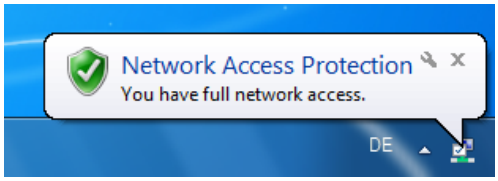


Abbildung 8.12: napstat auf einem *konformen* Rechner

Nach einem Klick auf die kleine Statusmeldung (Abbildung 8.11 und 8.12) wird die Detailansicht eingeblendet. Darauf ist detaillierter ersichtlich, welche SHAs auf dem Client aktiv sind und welche Stati die jeweiligen Agents besitzen.

In Abbildung 8.13 ist der konforme Status eines Clients ersichtlich. In diesem Beispiel sind auf dem Client zwei SHAs aktiv: einerseits der Windows eigene *Windows Security Health Agent* und der durch dieses Projekt entwickelte *Trusted SHA*. In diesem Fenster erhält der Betrachter die Möglichkeit, nachzuvollziehen, welche Versionen installiert sind und welche Zustände die einzelnen System Health Agents besitzen. Zudem wird durch die *Compliance Results* 0x00000000 dargestellt, welcher Zustand der Client besitzt.

In Abbildung 8.14 ist hingegen der Fehlerfall ersichtlich: er zeigt auf, dass der Computer nicht über einen policykonformen Zustand verfügt. Dies wird zusammenfassend durch das *unsuccessful* dargestellt.

8.6 Troubleshooting mit NetShell (netsh)

Mit der NetShell können Konfigurationen sowie Stati des NAPs abgefragt werden. Zwei Befehle, welche während des Projektes oft benutzt wurden, sind die Folgenden:

Aktueller NAP-Status des Computers ausgeben

netsh NAP client show state

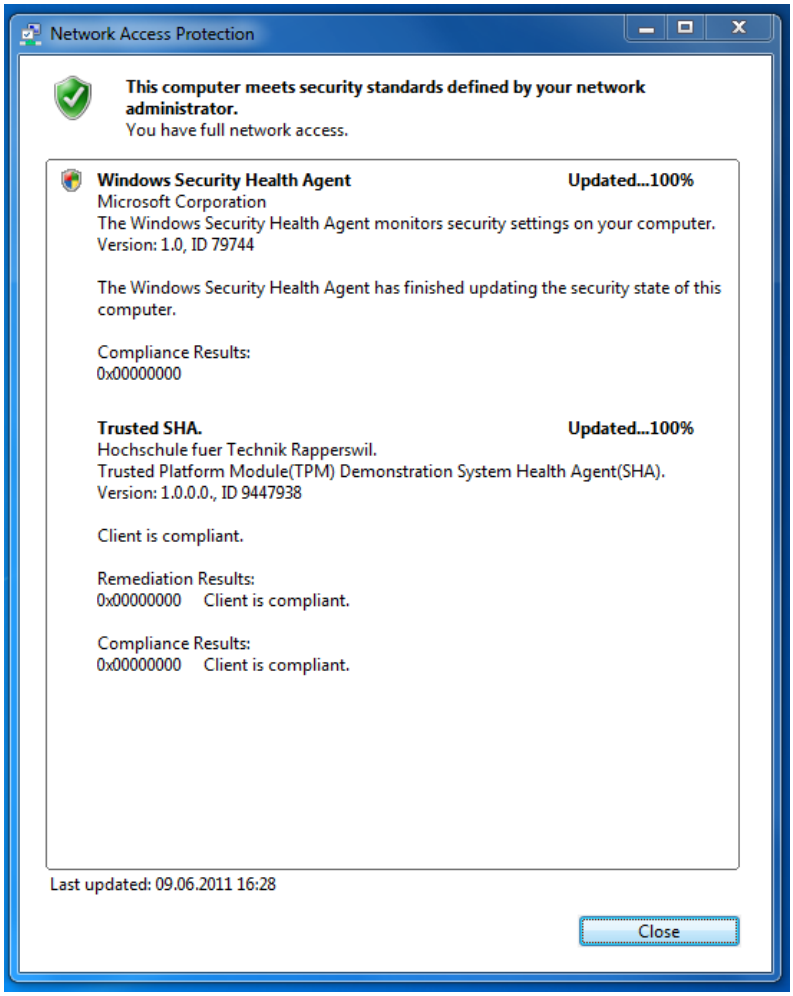


Abbildung 8.13: Detailansicht von napstat auf einem konformem Rechner

Lokale NAP-Konfiguration des Computers ausgeben

netsh NAP client show config

Eine Übersicht über mögliche Kommandos kann unter [5] eingesehen werden.

8.7 Fehlerquellen

8.7.1 Gruppenrichtlinien blockieren TPM-Funktionalitäten

Fehlercode: *TPM_E_COMMAND_BLOCKED* 0x80280400

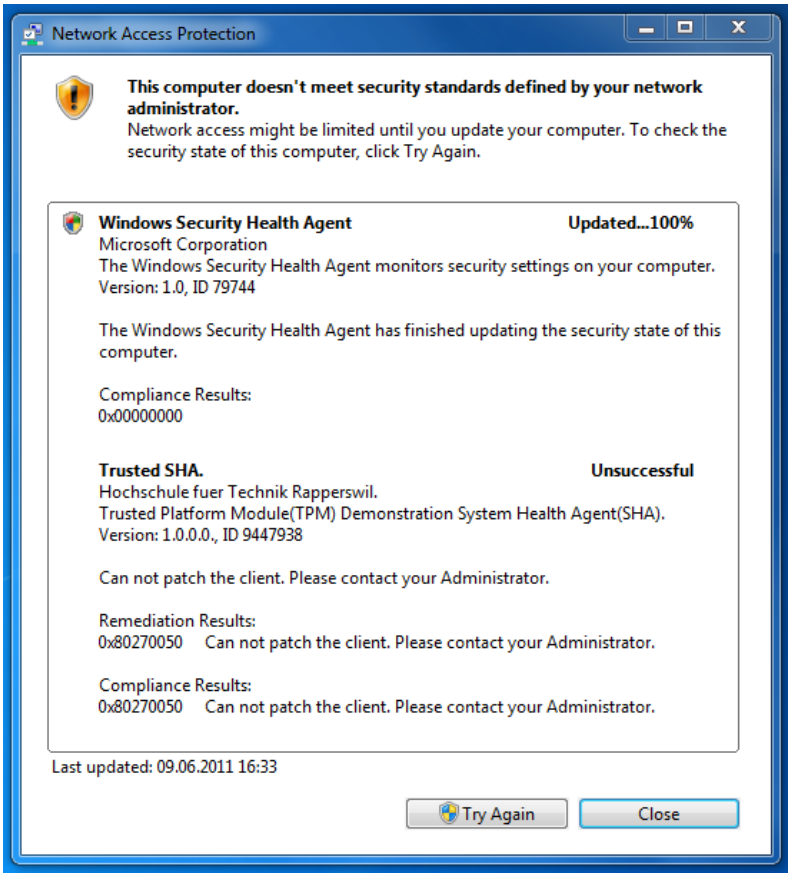


Abbildung 8.14: Detailansicht von napstat auf einem nicht konformem Rechner

Mithilfe der Gruppenrichtlinien können einzelne TPM-Funktionen explizit zugelassen oder blockiert werden. Beim Aufrufen einer blockierten Methode wird der Fehlercode 0x80280400 (*TPM_E_COMMAND_BLOCKED*) zurückgegeben. Damit solche deaktivierten Funktionalitäten detektiert werden können ist es wichtig, dass der Rückgabecode der auf dem TPM auszuführenden Funktion ausgewertet wird und bei entsprechendem Fehlercode *TPM_E_COMMAND_BLOCKED* der Benutzer informiert wird, resp. bekannt gegeben wird, weshalb die Operation nicht weiter ausgeführt werden kann.

Die aktuellen TPM-Restriktionen können mit dem Programm *tpm.msc* angezeigt werden (Abbildung 8.15). Interessanterweise werden in der Standardkonfiguration gewisse Funktionalitäten, wie beispielsweise die Berechnung des SHA1-Hashes, deaktiviert. Zudem werden ebenfalls standardmässig die deprecated Methoden blockiert, also jene Methoden, die durch neue (bessere) Methoden ersetzt wurden. Dazu muss nach dem Starten des *tpm.msc* in der linken Spalte *Command Management* ausgewählt werden.

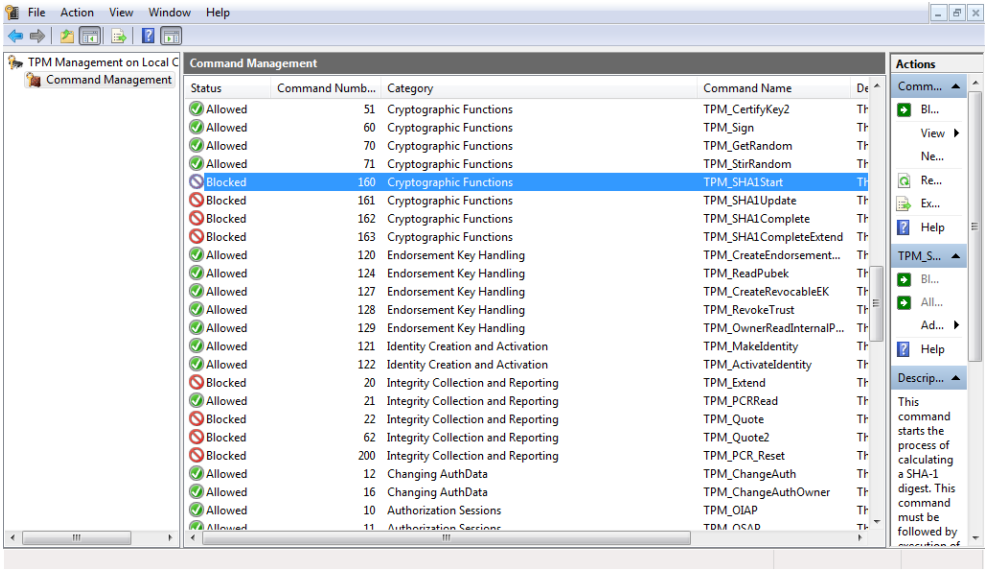


Abbildung 8.15: Durch Gruppenrichtlinien deaktivierte TPM-Befehle (tpm.msc)

8.7.2 Command Prompt mit Administratorrechten starten

Um ein Command Prompt unter Windows mit Administratorrechten zu starten muss in der Desktopsuche, welche mit der Windowstaste geöffnet wird, *cmd* eingegeben werden. Auf die nun erscheinende Verknüpfung muss nun mit einem Rechtsklick auf *als Administrator starten* geklickt werden.

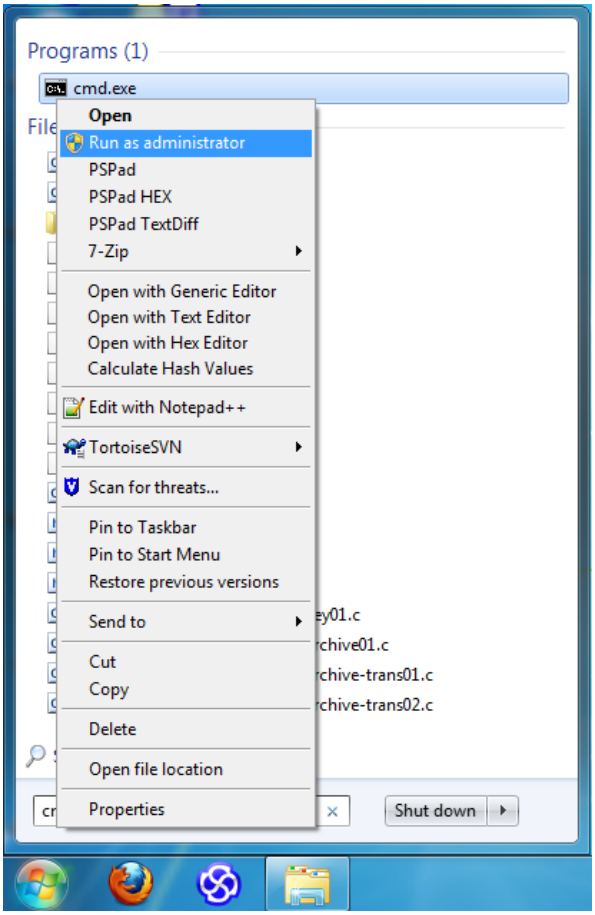


Abbildung 8.16: Vorgehen für eine Command Prompt mit Administratorrechten

Kapitel 9

Projektstand

9.1 Zielerreichung

Folgend wird dargestellt, welche der funktionalen Anforderungen mit Trusted NAP umgesetzt wurden:

Must

- FAnf01: SHA / SHV Paar: **erreicht**
- FAnf02: Clientseitige Manipulationsdetektion: **erreicht**
- FAnf03: Attestation mit TPM-Chip: **erreicht**
- FAnf04: Attestation-Vorgang mit dem quote-Befehl: **erreicht**
- FAnf06: Attestation mit Attestation Identity Key (AIK): **erreicht**
- FAnf07: Trusted by PrivacyCA.com / AIK-Zertifikat: **erreicht**
- FAnf10: Fehlerlogging in Protokolldatei: **erreicht**
- FAnf14: Dynamisches Laden der SHAs auf dem Client überprüfen: **erreicht**

Optional

- FAnf05: Attestation-Vorgang mit dem quote2-Befehl: **nicht umgesetzt**
- FAnf08: Level 1 AIK-Zertifikat (PrivacyCA.com): **teilweise**
- FAnf09: Paralleles Ausführen des SHV: **teilweise**

- FAnf11: Fehlerlogging in Event Viewer: **nicht umgesetzt**
- FAnf12: Platform Configuration Register (PCR) sperren: **entfiel**
- FAnf13: Überprüfung der Dateien im Arbeitsspeicher: **nicht umgesetzt**

Alle als Muss definierten Anforderungsziele wurden **erreicht**. Die als **nicht umgesetzt** dargestellten Anforderungen wurden aus Zeitgründen weggelassen. Folgend wird auf die speziellen Aspekte **teilweise** und **entfiel** detaillierter eingegangen:

Level 1 AIK-Zertifikat teilweise: Die Funktionalität für den Level 1 AIK-Zertifikat-Einbezug wurde vom Beispielcode der Privacy CA übernommen. Er ist aber nicht vollständig lauffähig und bricht mit der Fehlermeldung ab, dass der Request nicht gültig war. Es ist allerdings nicht ganz klar, weshalb die Umsetzung nicht funktioniert. Die vollständige Implementation wurde schliesslich aus Zeitgründen abgebrochen. Die Funktionalität wurde aber noch als mit dem Vermerk *experimentell* im Programm belassen.

Paralleles Ausführen des SHV teilweise: Die Entwicklung des Codes wurde auf das parallele Ausführen mehrerer Trusted SHV-Instanzen ausgelegt. Allerdings kann diese Anforderung erst mit vielen gleichzeitigen Anfragen getestet werden. Dies muss noch erfolgen.

PCR sperren entfiel: Diese Anforderung entfiel, da ein anderer Weg gefunden wurde. Es wird nicht von einem zurückgesetzten PCR ausgegangen, sondern von einem PCR, das bereits beliebige Daten beinhaltet kann. Dafür kann auf dem Server nicht mehr nur der PCR Zustand verglichen werden; es müssen sämtliche Kalkulationen nachgerechnet werden. Der Server geht vom Wert aus, welcher in den PCRs des Clients vor der Messungsnahme vorhanden war. Daraufhin werden die Messwerte dazugerechnet und der nun berechnete Wert mit dem resultierten PCR-Wert des Clients verglichen.

9.2 Ausblick

In einem weiteren Schritt sollte der Trusted SHA und der Trusted SHV in eine produktive NAP-Umgebung eingebunden werden, um das Produkt intensiv testen zu können und weitere Erfahrungen zu sammeln.

Die Privacy CA kann durch eine eigene Trusted Third Party ersetzt werden. Dies ist z.B. mit dem im eigenen Netzwerk vorhandenen Zertifizierungsserver möglich. Somit wird die externe Abhängigkeit minimiert.

Mit der Kopplung des Trusted SHA an die Trusted Execution Technology (TXT) kann die Sicherheit nochmals gesteigert werden. Durch diese Integration wird es möglich, ebenfalls Manipulationen durch Rootkits zu erkennen.

Anhang A

Projektmanagement

A.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.

Der Projektplan konnte aufgrund der Erkenntnisse und Erfahrungen aus der Semesterarbeit präzise geplant werden. Ebenso wurde darauf geachtet, dass für die Erreichung jedes Meilensteins etwas Reservezeit eingeplant wurde, welche jeweils für auftretende Technologie-Probleme (wie beispielsweise mit dem TPM) eingesetzt werden konnte.

A.2 Involvierte Personen

A.2.1 Projektmitglieder

Das Team besteht aus zwei Informatikstudenten Wolfgang Altenhofer und Lukas Studer der HSR, die sich im sechsten Semester befinden (Tabelle A.1).

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

Tabelle A.1: Projektmitglieder

Prof. Dr. Andreas Steffen	as
Prof. Dr. Peter Heinzmann	ph
Dr. Ralph Hauser	rh

Tabelle A.2: Betreuungspersonen

A.2.2 Externe Schnittstellen

Das Projekt wird von Herrn Prof. Dr. Steffen betreut und von Herrn Prof. Dr. Heinzmann gegengelesen. Experte ist Herr Dr. Hauser (Tabelle A.2).

A.3 Management Abläufe

A.3.1 Projekt Aufwand

Die Zeitplanung sieht pro Student 360 Stunden über 17 Wochen vor. Dies entspricht einem Total von 720 Stunden (21.2 Stunden pro Woche und Student).

A.3.2 Zeitplan

Der Zeitplan wurde zu Beginn geplant und aufgrund der externen Einflüsse, aufgetretenen Probleme und Unregelmässigkeiten angepasst. Die Endfassung sowie die Auswertung der benötigten Arbeitsstunden sind im Kapitel B ersichtlich. Alternativ ist die Tabelle der geleisteten Stunden auf der CD zu finden.

Die Zeiterfassung erfolgt über das Tool mite.

A.3.3 Risikomanagement

Das Risikomanagement wird in den Abbildungen A.1 und A.2 aufgeführt. Es wurde Wert darauf gelegt, dass die Risiken realistisch eingeschätzt wurden.

Es gilt zu beachten, dass in dieser Tabelle lediglich technologische sowie zu diesem Projekt spezifische Risiken aufgezeigt werden. In Absprache mit dem Betreuer wurde auf eine Auflistung der generischen Risiken (wie beispielsweise ungenaue Einschätzung des Aufwandes, Infrastrukturausfall, Konflikte im Team, Änderung des Aufgabenziels durch Kunden etc.) verzichtet.

Risiko-Nr.	Risikotitel	Risikobeschreibung	Maximaler Schaden [h]	Eintrittswahr-scheinlichkeit	Gewichteter Schaden [h]	Anzeichen / Indikator
R1	Technologie TPM	TPM-Fehler: Es fehlt an Know-How über das TPM, um dieses für Attestation zu verwenden. Ebenso kann die Kommunikation mit dem TPM unterschätzt werden.	100	10.00%	10	Kein oder sehr kleiner Fortschritt bei der Programmierung, es muss viel nachgelesen werden.
R2	Architektur Sicherheit	Die Software ist zuwenig sicher, unbedachte Sicherheitsprobleme treten auf	40	5.00%	2	Während der Programmierung oder beim Testen fallen Schwachstellen auf
R3	Zielerrei-chung	Angestrebtes Ziel kann nicht umgesetzt werden, da das TPM oder das Windows benötigte Funktionalitäten nicht anbieten.	80	10.00%	8	Auftraggeber ist mit zu Beginn definiertem Ziel nicht mehr einverstanden.
R4	Team	Gefahr, dass die Bachelorarbeit nicht bestanden wird.	720	1.00%	7.2	Gesteckte Ziele werden bei weitem verfehlt, Team-Motivation nicht vorhanden
SUMME			**940**	**26.00%**	**27.2**	

Abbildung A.1: Risikomanagement Teil 1

Risiko-Nr.	Risikotitel	Vermeidungsmassnahmen	Aufwand der Massnahmen [h]	Vorgehen, welches bei Eintritt durchzuführen ist
R1	Technologie TPM	Andere Opensource Projekte mit gleichem Thematik-Schwerpunkt betrachten, bereits in das Thema eingearbeitete Personen interviewen. Mögliche Probleme bereits im vornherein abklären.	10	Alternativen suchen, Projekt neu planen, Features kürzen bzw. geziehlt eine Problemlösung entwickeln.
R2	Architektur Sicherheit	Architektur überprüfen und bei Bedarf den aktuellen Kenntnissen anpassen und sauber umsetzen	5	Schwachstellen mit Betreuer diskutieren und Lösungsansatz finden.
R3	Zielerrei-chung	Kunden darauf ansprechen, Lösungen finden.	15	Einen gangbaren Weg gemeinsam mit dem Kunden suchen.
R4	Team	Sitzungen mit dem Betreuer, Protokollierung der Besprechungen, jeweiliger Stand zur Kontrolle durch Betreuer / Gegenleser absegnen lassen.	40	Zusätzliches Semester.
SUMME			**70**	

Abbildung A.2: Risikomanagement Teil 2

A.4 Meilensteine

Die Meilensteine bezeichnen die Zwischenziele, welche während der Entwicklung im Projekt eingesetzt werden. Somit kann jeweils nachverfolgt wer-

den, ob die gesetzten Ziele eingehalten werden , bzw. wie weit das Projekt fortgeschritten ist. In Tabelle A.3 sind die definierten Meilensteine ersichtlich.

Nr.	Datum	Ziele
MS1	13.03.2011 (Woche 3)	• Zeitplan fertiggestellt • Crypto-Einbindung erfolgt • TPM-Wrapper fertigprogrammiert
MS2	27.03.2011 (Woche 5)	• Voraussetzungen für Attestation abgeklärt / wie können Messungen durch das TPM beglaubigt werden. • Dokumentation um Attestationbereich ergänzt
MS3	17.04.2011 (Woche 8)	• Attestation-Funktionalität fertigprogrammiert. Es ist ein SHA- / SHV-Paar vorhanden, der ein AIK und ein dazugehöriges Zertifikat verwendet, um Messungen zu signieren.
MS4	29.05.2011 (Woche 14)	• NAP erweitert: Client kann nicht mehr auf internes Unternehmensnetzwerk zugreifen, wenn dll-Dateien des Windows SHA manipuliert wurden. • Dynamisches Laden der SHA wird überwacht.
MS5	05.06.2011 (Woche 15)	• Codefreeze: Programmierarbeiten fertiggestellt.

MS6	10.06.2011 (Woche 16)	• Abgabe des Abstract- und des Posterdokuments an den Betreuer.
MS7	17.06.2011 (Woche 17)	• Abgabe der Dokumentation, Kurzzusammenfassung und des Berichts.

Tabelle A.3: Meilensteine

A.5 Projektplan

Für die Erreichung der einzelnen Meilensteine wurden detailliertere Arbeitspakete definiert. Die geplante Aufgabenbearbeitung wurde geschätzt und in einen Plan gesetzt. Dieser wurde im Verlaufe des Projektes noch etwas verfeinert. Im Kapitel B wird genauer darauf eingegangen, wo grössere Differenzen aufgetreten sind. Der ausführliche Projektplan ist auf der CD zu finden.

A.6 Infrastruktur

A.6.1 Hardware

Grundsätzlich arbeitet jedes Teammitglied auf seinem privaten Notebook. Neben diesem stehen für das gesamte Projekt drei Computer zur Verfügung. Dabei dienen zwei als Entwicklungsstationen und als Hostrechner für die virtuelle NAP-Infrastruktur, der Dritte ist der NAP-Clientrechner mit Zugriff auf das TPM.

A.6.2 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 resp. Microsoft Office zurückgegriffen.

Für die Zeiterfassung wird mite verwendet. Die Versionsverwaltung wird mit git vorgenommen. Das Projektmanagement und das Bug-Tracking wird mit trac bewältigt. Als Betriebssysteme werden Windows 7, Windows Server

2008 R2 und Linux Ubuntu verwendet. Windows 7 auf dem Client- und den Entwicklungsrechnern, Windows Server 2008 R2 auf den NAP Servern und Linux Ubuntu für TPM-Debugging, Teilentwicklung TPM, Openssl und LaTeX. Für die Visualisierungen wurde auf Microsoft Visio 2010 und Sparx Systems Enterprise Architect 8 zurückgegriffen.

A.7 Qualitätsmassnahmen

A.7.1 Dokumentation

Für das Team ist es wichtig, dass eine hochwertige Software hergestellt wird. Aus diesem Grund wird während der gesamten Projektdauer auf eine verständliche und aktuelle Dokumentation gesetzt.

A.7.2 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 (Kapitel F) eingesehen werden.

A.7.3 Verantwortlichkeiten

Bereich	Verantwortlicher
Einrichtung und Wartung Git, trac	wa
Microsoft Testumgebung	ls
Zeitplanung und Projektmanagement (Excel)	ls
Kommunikation mit TrouSerS (TPM)	wa / ls
Dokumentation	ls
Diagramme und Abläufe	ls
Zeiterfassung	wa / ls
Privacy CA	wa
Registry / Logging	wa
LaTeX	wa / ls

Testing	wa / ls
Code-Architektur	wa / ls
Technologieabklärung	wa / ls
Attestation	wa / ls

Tabelle A.4: Verantwortlichkeiten

A.7.4 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.

A.7.5 Qualitätssicherung

Um die Qualität der Dokumente hoch zu halten werden diese jeweils von der anderen Person gegengelesen. Auch der Code wird von beiden Teammitgliedern durchgesehen, bevor er als final angesehen werden kann. So können allfällige Unklarheiten, Unschönheiten und Fehler gefunden und diskutiert werden.

A.7.6 Versionsverwaltungssystem

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

Anhang B

Projektplanung

Insgesamt wurden durch die beiden Teammitglieder 827.5 Stunden an diesem Projekt gearbeitet, was mehr als 24 Stunden pro Woche pro Person bedeutet (Abbildung B.1).

In Abbildung B.1 kann nachvollzogen werden, welche Person in welcher Woche wieviel gearbeitet hat. Die aufgewendeten Stunden steigen im Verlaufe des Projektes immer mehr an. Der Peak in Woche 8 wurde durch den Meilenstein 3 (Attestation-Funktionalität fertigprogrammiert) verursacht. In Woche 12 fand die Präsentation statt. Deshalb stieg bis zur Woche 12 die aufgewendete Anzahl Stunden, da zusätzlich zum Projekt noch Zeit für die Vorbereitung aufgewendet wurde.

In Abbildung B.2 ist ersichtlich, wie hoch der Aufwand prozentual zur gesamten Arbeitszeit in den einzelnen Themenbereichen war. Mehr als die Hälfte der investierten Zeit wurde für die Implementation genutzt; ein Drittel ist für die Dokumentation angefallen.

Weiter aufgeschlüsselt auf die einzelnen Implementationsteile wird ersichtlich, dass rund 31% der Implementationszeit für Attestation mit dem TPM aufgewendet wurde (siehe Abbildung B.3). Insgesamt wurden 32% für NAP-Implementationen benötigt (NAP-Grundgerüst, NAP-Validator und NAP-Agent).

Wer	Ø	∑
Altenhofer Wolfgang	24.2h	412.0h
Studer Lukas	24.4h	415.5h
Soll	21.2h	360h

Tabelle B.1: Zeitauswertung: Wochenschnitt und Total

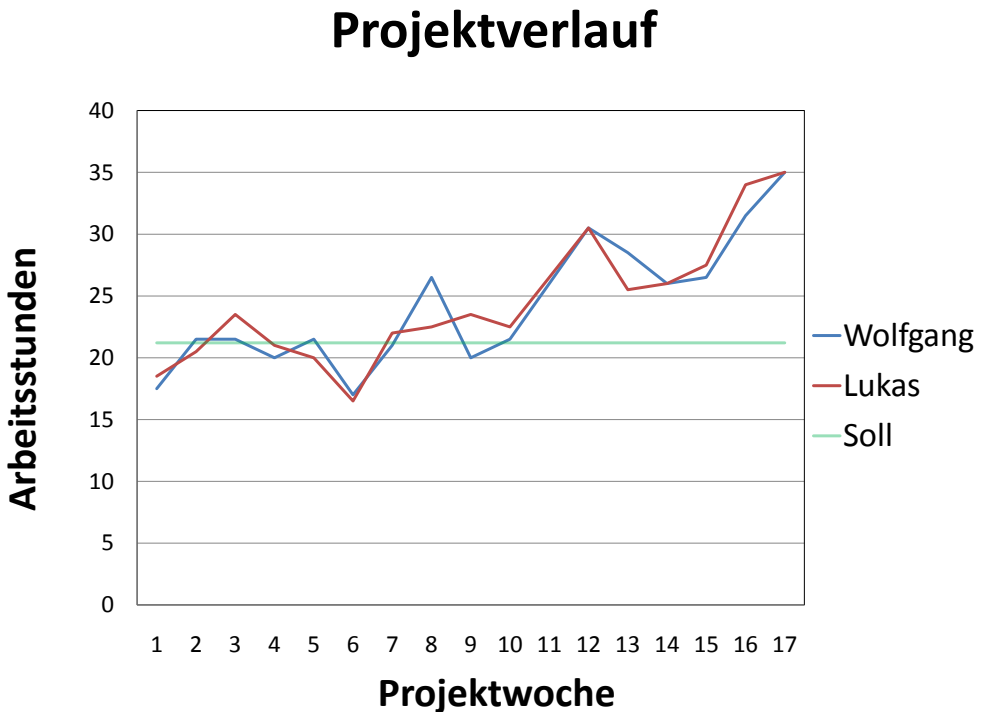


Abbildung B.1: Aufgewendete Stunden pro Person

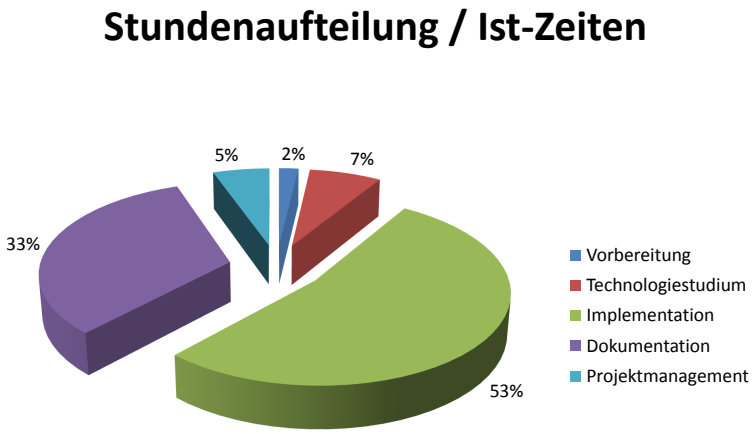


Abbildung B.2: Aufteilung der Aufwände insgesamt

Am Ende dieses Kapitels ist auf zwei Seiten der Vergleich der Soll- und Ist-Werte auf Wochenbasis aufgeführt. Dort ist die jeweils aufgewendete An-

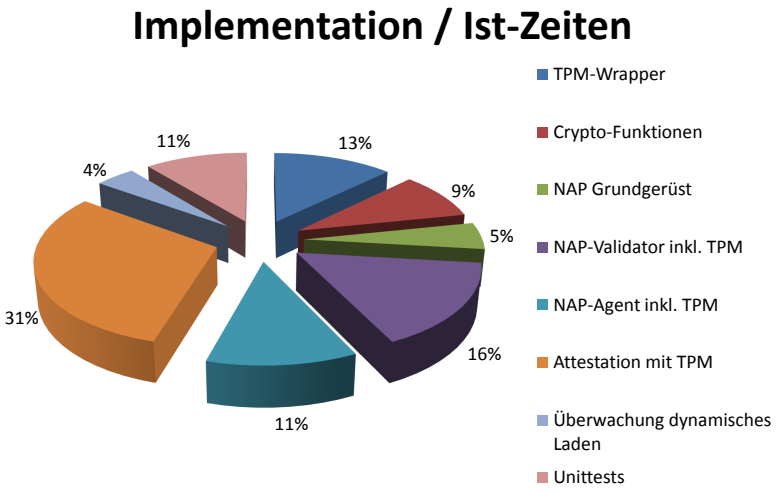


Abbildung B.3: Aufteilung der Aufwände für die Implementation

zahl Stunden des Teams im Soll- und Ist- Bereich ersichtlich. Die Planung stimmte mit der später ausgeführten Arbeit meist überein. Folgend werden jene Situationen beschrieben, an welchen die geplante von der tatsächlichen Zeit erheblich abweicht:

- Woche 4: hier wurde anstatt der 19 eingeplanten insgesamt 34.5 Stunden für die Implementation aufgewendet. Das hatte den Grund, weil direkte Programmierung zum TPM sehr fehleranfällig war, und viele Probleme auftauchten. Ab Woche 5 wurde dann darum TrouSerS einbezogen, damit die Implementierung nicht an solchen Stellen hängen blieb.
- Woche 8: hier wurde 2.5 mal soviel dokumentiert wie geplant (20 Stunden anstatt der 8 geplanten). Dies deshalb, weil die umgesetzten Funktionalitäten im Bezug zu Attestation und TrouSerS in der Dokumentation festgehalten wurden und dies viel Zeit in Anspruch nahm.
- Woche 13 / 15: Für die Implementation wurde mehr Zeit benötigt. Dies ist darauf zurückzuführen, dass der Codefreeze immer näherrückte und die letzte Implementationsphase etwas unterschätzt wurde.
- Woche 16 / 17: Für die Dokumentation wurde einiges mehr an Stunden verwendet. Dadurch, dass die letzten beiden Wochen in die unterrichtsfreie Zeit fielen, wurde mehr Zeit für die Arbeit aufgewendet, obwohl dies gemäss Plan nicht vorgesehen war.

Der vollständige Projektplan mit den detailliert aufgewendeten Stunden pro

Teilnehmer ist auf der Projekt-CD zu finden.

Balkendiagramm **Trusted NAP**

Meilensteine / Task	Total SOLL	Total IST	Woche 1	Woche 2	Woche 3	Woche 4	Woche 5	Woche 6	Woche 7	Woche 8	Woche 9
Vorbereitung	**22.0**	**15.5**	20 / 15.5	2							
Technologiestudium	**63.0**	**57.5**	18 / 18	9 / 10	2 / 2	13 / 3	18 / 19.5		1	4	
Implementation	**374.0**	**439.5**		25 / 23.5	30 / 27.5	19 / 34.5	10 / 14.5	33 / 31.5	26 / 29	30 / 23.5	28 / 33
TPM-Wrapper um TPM-Funktional	29.0	55.5		13 / 15.5	4	8 / 20				4 / 14	6
Implementation Crypto-Funktionen	38.0	40.0		9 / 7.5	24 / 24	5 / 8.5					
NAP Grundgerüst	28.0	22.5								2	2
NAP-Validator (inkl. TPM-Funktion	31.0	70.5								5	
NAP-Agent (inkl. TPM-Funktionalit	38.0	50.5								7	6
Attestation mit TPM	135.0	134.5				3	7 / 11.5	29 / 28	23 / 25	8 / 6	18 / 22.5
Überwachung des dynamischen L	22.0	18.5									18
Unittests schreiben	53.0	47.5		3 / 0.5	2 / 3.5	6 / 3	3 / 3	4 / 3.5	3 / 4	4 / 3.5	4 / 2.5
Dokumentation	**212.5**	**270.5**	2	2	8 / 14	6	8 / 6	7	12 / 11	8 / 20	10 / 9
Projektmanagement	**43.5**	**45.0**	2.5 / 2.5	5.5 / 6.5	4.5 / 1.5	2 / 3.5	3.5 / 1.5	1.5 / 2	3.5 / 2	2.5 / 1.5	2 / 1.5
Total	**715.0**	**828.0**									

Legende

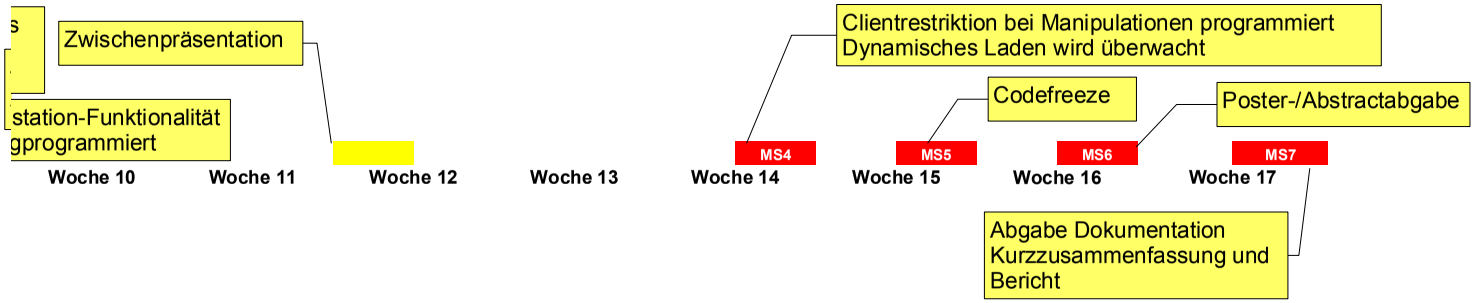
SOLL [Anzahl Stunden]:

	< 8 Stunden	0.001	7.999
	zwischen 8 und 16 Std.	8	16
	> 16 Stunden	16.001	999

IST [Anzahl Stunden]:

	< 8 Stunden	0.001	7.999
	zwischen 8 und 16 Std.	8	16
	> 16 Stunden	16.001	999

Balkendiagramm **Trusted NAP**



Meilensteine / Task	Total SOLL	Total IST	Woche 10	Woche 11	Woche 12	Woche 13	Woche 14	Woche 15	Woche 16	Woche 17
Vorbereitung	**22.0**	**15.5**								
Technologiestudium	**63.0**	**57.5**		3						
Implementation	**374.0**	**439.5**	32 / 32.5	32 / 42	38 / 41.5	27 / 40	28 / 32	16 / 34.5		
TPM-Wrapper um TPM-Funktional	29.0	55.5								
Implementation Crypto-Funktionen	38.0	40.0								
NAP Grundgerüst	28.0	22.5	5 / 6	9 / 11.5	8 / 3			4		
NAP-Validator (inkl. TPM-Funktion	31.0	70.5		3 / 2.5	6 / 12.5	10 / 23	7 / 10.5	22		
NAP-Agent (inkl. TPM-Funktionalit	38.0	50.5	4.5	5 / 10.5	8 / 13.5	4 / 3.5	8 / 12.5	6		
Attestation mit TPM	135.0	134.5	21 / 17.5	11 / 12	14 / 9			4		
Überwachung des dynamischen La	22.0	18.5	21			9 / 10.5	9 / 5	4 / 3		
Unittests schreiben	53.0	47.5	6 / 4.5	4 / 5.5	2 / 3.5	4 / 3	4 / 4	4 / 3.5		
Dokumentation	**212.5**	**270.5**	8 / 11.5	5 / 6	3 / 16	12 / 14	11 / 19	22 / 15	46 / 60	44.5 / 67
Projektmanagement	**43.5**	**45.0**	2	3 / 4.5	1.5 / 3.5	2.5	1.5 / 1	2.5 / 4.5	1.5 / 5.5	1.5 / 3
Total	**715.0**	**828.0**								

Legende

SOLL [Anzahl Stunden]:

	< 8 Stunden	0.001	7.999
	zwischen 8 und 16 Std.	8	16
	> 16 Stunden	16.001	999

IST [Anzahl Stunden]:

	< 8 Stunden	0.001	7.999
	zwischen 8 und 16 Std.	8	16
	> 16 Stunden	16.001	999

Anhang C

Erfahrungsberichte

C.1 Wolfgang Altenhofer

Das Arbeiten mit den Technologien NAP und TPM gestaltete sich als sehr interessant und fordernd. Speziell mit dem TPM und dem damit verbundenen TSS-Stack kamen viele Herausforderungen auf uns zu. So musste schon früh in der Arbeit auf eine eigene Implementation der TPM Funktionalität verzichtet werden, da sich dies als zu komplex herausstellte. So wurde nach etwa drei Wochen auf TrouSerS gesetzt, was wiederum Einarbeitungszeit beanspruchte. Jedoch stellte sich das Arbeiten mit TrouSerS als sehr angenehm heraus und ich würde jederzeit wieder mit dieser TSS-Implementation arbeiten.

Nach den Schwierigkeiten zu Beginn und dem erfolgreichen Einarbeiten ins TrouSerS verlief die Bachelorarbeit produktiv und wir kamen gut voran. Dies sicherlich auch dadurch, dass von der Semesterarbeit her bereits viel Wissen über die verwendeten Technologien bestand. Teilweise machte uns das Debuggen im Kryptobereich etwas sorgen und verschlang einige Zeit. Es kann sehr umständlich werden, bei einer verschlüsselten Ausgabe die genaue Fehlerquelle zu finden.

Die Zusammenarbeit mit Lukas verlief sehr positiv, wir konnten uns gegenseitig gut ergänzen, wodurch die Arbeit zügig vorangetrieben werden konnte. Auch entstanden immer wieder interessante Diskussionen zu auftretenden Hindernissen.

Die Betreuung durch Andreas Steffen hab ich sehr geschätzt. Er hatte in den Sitzungen sehr gute Inputs zu auftretenden Problemen geliefert. Auch habe ich es als sehr angenehm empfunden, dass er uns viele Freiheiten gewährte.

C.2 Lukas Studer

Sicherheit ist ein interessanter und doch auch anspruchsvoller Bereich, in welchem wir während des letzten halben Jahres tätig waren. Mit viel bereits in der Semesterarbeit angeeignetem Know-How gelangten wir in die Bachelorarbeit, welche sich um die Technologien TPM und NAP drehte. Dadurch, dass wir die Arbeit als Fortsetzung weiterführten, ersparten wir uns einiges an Einarbeitungszeit und konnten bereits nach kurzer Zeit mit der detaillierten Planung und Umsetzung des Ziels beginnen.

Das Wissen, welches wir bereits mitbrachten, konnte uns aber nicht ganz vor Fehlentscheidungen schützen. Beispielsweise mussten wir aufgrund der Komplexität des TPM-Zugriffs nach den ersten Wochen einsehen, dass für die Kommunikation zum TPM ein TSS eingesetzt werden muss, weil dies sonst zusätzlich zu viel Zeit verschlungen hätte. Wolfgang und mir wurde zudem einige Male vor Augen geführt, dass für Entwicklungen im Kryptobereich auch bereits kleinste Probleme viel Zeit verschlingen können.

Die Betreuung durch Herrn Steffen war aus meiner Sicht genau richtig: als Experte im Sicherheitsgebiet konnten wir interessante Diskussionen über Knacknüsse führen und so schnell zu einem Weg finden. Andererseits wurde auf viel Eigeninitiative gesetzt, was ich als sehr angenehm empfunden habe.

Mit Wolfgang hatte ich einen zuverlässigen Partner im Team. Wir führten viele fachliche Diskussionen, um Probleme zu lösen und gute Lösungsansätze zu finden.

Ein wenig schade finde ich, dass die Arbeit mit dem jetzigen Stand beendet werden muss. Klar, die Hauptziele wurden voll und ganz erreicht, doch für weitere Funktionalitäten müsste nicht mehr soviel Zeit aufgewendet werden, um sie zu erlangen. Alles in allem ist die Erreichung der Ziele und das dadurch entstandene Endprodukt meines Erachtens ein voller Erfolg, der aufzeigt, dass das TPM in der Umgebung mit NAP sinnvoll eingebunden werden kann.

Anhang D

Abkürzungsverzeichnis

AIK Attestation Identity Key

API Application Programming Interface

AR Access Requestor

ASCII American Standard Code for Information Interchange

BIOS Basic Input Output System

CA Certification Authority

Cisco NAC Cisco Network Admission Control

CNG Cryptography API: Next Generation

COM Component Object Model

CSP Cryptographic Service Provider

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

MS-HCEP Microsoft Health Certificate Enrollment Protocol

MS-SoH Microsoft Statement of Health for Network Access Protection

NAC Network Access Control

NAP Network Access Protection

NPS Network Policy Server

NSA National Security Agency

PCR Platform Configuration Register

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

TCS TSS Core Services

TCSi TCS interface

TDD TPM Device Driver

TDDL TSS Device Driver Library

TDDLi TDDL interface

TLV Type Length Value

TNAC Trusted Network Access Control

TNC Trusted Network Connect

TPM Trusted Platform Module

TSP TSS Service Provider

TSPi TSP interface

TSS TCG Software Stack

TXT Trusted Execution Technology

URL Uniform Resource Locator

VPN Virtual Private Network

Anhang E

Inhalt der CD

Auf der CD ist folgender Inhalt gespeichert:

Dokumente

- Dokumentation
- Poster
- Projektplan
- Demonstrations-Video

Trusted NAP Programm

- Programm Source Code
- Kompilierte Programme
- Sample Code Microsoft NAP
- Sample Code Privacy CA

Diverses

- TrouSerS-Installer

Anhang F

Sitzungsprotokolle

F.1 Protokoll vom 21.02.2011 - Woche 1 - Kickoff

Sitzungsinhalt

- Besprechung Semesterarbeit / Bewertung / Kritikpunkte
- Offene Diskussion über Fortsetzung der Arbeit als Bachelorarbeit: welche Richtung soll weiter verfolgt werden? Wo liegen Angriffspunkte des NAP-Systems, welches in der Bachelorarbeit mit TNAC beseitigt werden soll?
- Abklären, wie Ladefunktion (DLL, etc.) für NAP funktioniert

Pendenzen

- s. Beschlüsse

Termine

- Nächste Sitzung: Mo, 28.02.2011 - 10:00 Uhr

Beschlüsse

- Ziel-Thematik genauer bis Woche 2 / 3 bestimmen
- TPM-Wrapper fertigprogrammieren
- Zeitplanung bis Woche 3 erstellen

- Ungefähre Projektvorgehensweise:
  1. Abklären, wie es mit der Technologie steht (fertigprogrammieren TPM-Wrapper)
  2. Ganzer Startablauf sicher gestalten (Fachwissen erarbeiten)
  3. Prüfen, ob Collector nicht verändert wurde (Signatur)
  4. Sinnvolle Prüfung des Health Status des Clients

Aufgetretene Probleme

- -

F.2 Protokoll vom 28.02.2011 - Woche 2

Sitzungsinhalt

- SHA1-Hashing auf dem TPM funktioniert.
- Für RSA-Algorithmen sollte auf das Microsoft Crypto-API zugegriffen werden. Dieser ist sehr umfangreich ist unsere Zwecke geeignet.
- Measurements (Attestation): Projekt der FH Hannover analysieren - nachvollziehen, wie sie vorgegangen sind. Wie werden die Platform Configuration Register (PCR) signiert?
- TPM_Quote: ermöglicht die Signatur über die Register des TPMs
- Entwurf der Aufgabenstellung besprochen
- Zertifikaterzeugung aus dem Public Key des TPMs
- Quelle für Fachwissen: Joanna Rutkowska - befasst sich mit Malware, die sich in virtuelles System während des Starts einbinden kann.

Pendenzen

- Terminplanung abgeben bis nächsten Montag.

Termine

- Nächste Sitzung: Mo, 07.03.2011 - 10:00 Uhr

Beschlüsse

- -

Aufgetretene Probleme

- -

F.3 Protokoll vom 07.03.2011 - Woche 3

Sitzungsinhalt

- Besprechung des Terminplanes
- CryptoAPI oder OpenSSL: Verwenden für die RSA-Verschlüsselungsfunktionen
- Dokumentation: wichtigste Erkenntnisse aus der Semesterarbeit sollen kurz aufgezeigt werden.

Pendenzen

- -

Termine

- Nächste Sitzung: Mo, 14.03.2011 - 10:00 Uhr

Beschlüsse

- -

Aufgetretene Probleme

- Kryptographische Berechnungen benötigt: Abstützung durch Crypto-Library

F.4 Protokoll vom 14.03.2011 - Woche 4

Sitzungsinhalt

- Attestation: Besprechung der Funktionsweise
- Möglicher Einbezug einer Trusted Third Party bei Zertifikatsgenerierung (privacyCA.com)

Pendenzen

- -

Termine

- Nächste Sitzung: Mo, 21.03.2011 - 10:00 Uhr

Beschlüsse

- -

Aufgetretene Probleme

- Mögliche Problemstellung für Tests unter Linux: Windows und Linux verwenden standardmässig verschiedene Encodings und deshalb werden dann Passwörter im jeweils anderen System nicht akzeptiert.

F.5 Protokoll vom 21.03.2011 - Woche 5

Sitzungsinhalt

- Die Entwicklung der TPM-Grundfunktionen ist sehr aufwendig und führte zu grossen Verzögerungen. Es soll abgeklärt werden, ob für Windows nicht auch eine TSS-Library zur Verfügung steht.
- Mögliche TSS: Infineon TPM / TrouSerS Portierung für Windows
- Nächste Schritte: Abklären, wie Attestation genau funktioniert.

Pendenzen

- -

Termine

- Nächste Sitzung: Mo, 28.03.2011 - 10:00 Uhr

Beschlüsse

- TSS-Library wenn möglich einsetzen.

Aufgetretene Probleme

- -

F.6 Protokoll vom 28.03.2011 - Woche 6

Sitzungsinhalt

- Die TSS-Library TrouSerS ist unter Windows einsetzbar. Die weitere Entwicklung soll deshalb aufbauend auf dieser Library erfolgen.
- Messresultate (Measurements) werden in Registern im TPM abgelegt. Für Entwicklungszwecke wurde im Standard das Register 16 definiert. Dieses ist jederzeit beschreibbar, bzw. rücksetzbar. Für den finalen Betrieb sollen dann mehrere Register verwendet werden, um feststellen zu können, in welchem Bereich das Measurement fehlschlug.
- Beim NAP Challenge soll das öffentliche Zertifikat des TPMs mitgesendet werden (falls eine Grössenbeschränkung im Protokoll kein Hindernis darstellt).
- Infineon TPM besitzen Level 1, alle TPM anderer Marken Level 0 Zertifikate (zum jetzigen Zeitpunkt).

Pendenzen

- -

Termine

- Nächste Sitzung: Mo, 04.04.2011 - 10:00 Uhr

Beschlüsse

- Die weitere Entwicklung soll auf TrouSerS aufbauen.
- Da TrouSerS mit OpenSSL arbeitet soll für die weitere Entwicklung auf OpenSSL aufgesetzt werden.

Aufgetretene Probleme

- -

F.7 Protokoll vom 04.04.2011 - Woche 7

Sitzungsinhalt

- Die Befehle TakeOwnership, ClearOwner wurden mit TSS umgesetzt; PcrExtend wurde bereits mit dem Debug PCR getestet. Nun muss noch abgeklärt werden, wie auf die anderen PCR zugegriffen werden muss, bzw. ob spezielle Zugriffe nötig sind.
- Die Kommunikation mit der Privacy CA wird momentan implementiert.
- Der Gegenleser der Arbeit wurde bekanntgegeben.
- Es soll in den nächsten Wochen eine Kurzpräsentation vor dem Gegenleser gehalten werden. Ebenfalls sollen die Dokumentations-Anforderungen des Gegenlesers notiert werden.

Pendenzen

- -

Termine

- Nächste Sitzung: Mo, 12.04.2011 - 15:00 Uhr

Beschlüsse

-

Aufgetretene Probleme

- -

F.8 Protokoll vom 12.04.2011 - Woche 8

Sitzungsinhalt

- Präsentation des Zwischenstandes am 9. Mai gemeinsam mit Gegenleser (Präsentation Demo, momentaner Stand, was wird noch umgesetzt, Besprechen der Formalitäten, Erwartungen des Gegenlesers einbeziehen)

Pendenzen

- -

Termine

- Nächste Sitzung: Mo, 18.04.2011 - 10:00 Uhr

Beschlüsse

- -

Aufgetretene Probleme

- Unter Windows gibt es Probleme mit dem Laden von Key-Blobs - die Linuxversion ist lauffähig. Requestlänge ist zu kurz.

F.9 Protokoll vom 18.04.2011 - Woche 9

Sitzungsinhalt

- Die quote-Funktionalität ist nun implementiert.
- Nächste Schritte: Funktionalität in NAP-Umgebung integrieren.
- An der nächsten Sitzung wird ein Prototyp gezeigt, der die jetzigen Funktionalitäten zeigt.
- Problem des Keyblob-Ladens wurde behoben: der SRK-Schlüssel muss während des TakeOwnerShips explizit mit dem Flag TSS_KEY_AUTHORIZATION ausgestattet werden.

Pendenzen

- -

Termine

- Nächste Sitzung: Mo, 02.05.2011 - 10:00 Uhr

Beschlüsse

- -

Aufgetretene Probleme

- -

F.10 Protokoll vom 02.05.2011 - Woche 11

Sitzungsinhalt

- Momentaner Projektstand in der Testumgebung betrachtet.
- Konfiguration (wie beispielsweise die PCR-Wahl) in Registry oder Datei ablegbar.
- Übertragung der Informationen zwischen Trusted SHA und Trusted SHV mit TLV.

Pendenzen

- -

Termine

- Präsentation für Gegenleser: Mo, 09.05.2011 - 10:00 Uhr

Beschlüsse

- -

Aufgetretene Probleme

- -

F.11 Protokoll vom 09.05.2011 - Woche 12

Präsentation - Ablauf

- Vortragen der Präsentation
- Besprechung der Präsentation
- Besprechung Dokumentation

Pendenzen

- Besprochene Punkte in Präsentation und Dokumentation in die Endfassung einfliessen lassen (werden hier nicht aufgeführt).

Termine

- Sitzung mit Betreuer: Mo, 23.05.2011 - 10:00 Uhr

Beschlüsse

- Dokumentationsteile jeweils mit Betreuer besprechen.

F.12 Protokoll vom 23.05.2011 - Woche 14

Sitzungsinhalt

- Besprechung der erarbeiteten Anforderungsspezifikation
- Testumgebung wird wenn möglich auf Wunsch des Gegenlesers als virtuelle Maschinen zur Verfügung gestellt. Testumgebung kann mit einem Clientrechner nachgebaut werden.
- Demonstration: Prototyp mit Registry und Loggingfunktionalität
- SHV und Multithreading: ist es darauf ausgelegt? Austesten.

Pendenzen

- -

Termine

- Besprechung mit Betreuer: Mo, 30.05.2011 - 10:00 Uhr

Beschlüsse

- -

Aufgetretene Probleme

- -

F.13 Protokoll vom 30.05.2011 - Woche 15

Sitzungsinhalt

- Anforderungsanalyse und Analyse
- Ablauf Attestation
- Bezeichnung der eigenen SHA / SHV-Lösung: Trusted NAP mit Trusted SHA und Trusted SHV
- Für Nonce selben Timeshift wie in Kerberos verwenden.
- Dritte Aufgabenstellung "Überprüfung des korrekten dynamischen Ladens des SHAs"wird mit Hashing der Dateien erreicht.
- TLV: Version der Konfiguration soll mitgesendet werden, damit überprüft werden kann, ob SHA und SHV die gleiche Konfiguration beinhalten.

Pendenzen

- -

Termine

- Besprechung mit Betreuer: Mo, 06.06.2011 - 10:00 Uhr

Beschlüsse

- s. Sitzungsinhalt

Aufgetretene Probleme

- -

F.14 Protokoll vom 06.06.2011 - Woche 16

Sitzungsinhalt

- Implementationsstand: Funktionalitätsprogrammierung abgeschlossen
- Abstract und Postervorschlag besprochen
- Dokumentation komplettieren
- Testen der Level 1 Funktionalität mit Infineon-TPM

Pendenzen

- -

Termine

- Besprechung mit Betreuer: Fr, 10.06.2011 - 10:00 Uhr

Beschlüsse

- s. Sitzungsinhalt

Aufgetretene Probleme

- -

F.15 Protokoll vom 10.06.2011 - Woche 16

Sitzungsinhalt

- Dokumentation geprüft
- Bessere Struktur in Doku hineinbringen
- Dokumentation komplettieren
- Beim Projektplan / Management nicht zu sehr ins Detail gehen

Pendenzen

- -

Termine

- Abgabe Bachelorarbeit: Fr,17.06.2011 - 12:00 Uhr

Beschlüsse

- Dokumentationsstruktur ändern

Aufgetretene Probleme

- -

Anhang G

Errorcodes

In diesem Kapitel werden drei Übersichten über die Fehlercodes und deren Bedeutung folgender Gebiete gegeben:

- TPM[10]
- TSS[11]
- NAP[4]

TPM V1.2 Error Code Cheat Sheet

Error	V1.1B	(Dec)	(Hex)	Description
TPM_SUCCESS	O	0	0x 00000000	Successful completion of the operation
TPM_AUTHFAIL	O	1	0x 00000001	Authentication failed
TPM_BADINDEX	O	2	0x 00000002	The index to a PCR, DIR or other register is incorrect
TPM_BAD_PARAMETER	O	3	0x 00000003	One or more parameter is bad
TPM_AUDITFAILURE	O	4	0x 00000004	An operation completed successfully but the auditing of that operation failed.
TPM_CLEAR_DISABLED	O	5	0x 00000005	The clear disable flag is set and all clear operations now require physical access
TPM_DEACTIVATED	O	6	0x 00000006	The TPM is deactivated
TPM_DISABLED	O	7	0x 00000007	The TPM is disabled
TPM_DISABLED_CMD	O	8	0x 00000008	The target command has been disabled
TPM_FAIL	O	9	0x 00000009	The operation failed
TPM_BAD_ORDINAL	O	10	0x 0000000A	The ordinal was unknown or inconsistent
TPM_INSTALL_DISABLED	O	11	0x 0000000B	The ability to install an owner is disabled
TPM_INVALID_KEYHANDLE	O	12	0x 0000000C	The key handle can not be interpreted
TPM_KEYNOTFOUND	O	13	0x 0000000D	The key handle points to an invalid key
TPM_INAPPROPRIATE_ENC	O	14	0x 0000000E	Unacceptable encryption scheme
TPM_MIGRATEFAIL	O	15	0x 0000000F	Migration authorization failed
TPM_INVALID_PCR_INFO	O	16	0x 00000010	PCR information could not be interpreted
TPM_NOSPACE	O	17	0x 00000011	No room to load key.
TPM_NOSRK	O	18	0x 00000012	There is no SRK set
TPM_NOTSEALED_BLOB	O	19	0x 00000013	An encrypted blob is invalid or was not created by this TPM
TPM_OWNER_SET	O	20	0x 00000014	There is already an Owner
TPM_RESOURCES	O	21	0x 00000015	The TPM has insufficient internal resources to perform the requested action.
TPM_SHORTRANDOM	O	22	0x 00000016	A random string was too short
TPM_SIZE	O	23	0x 00000017	The TPM does not have the space to perform the operation.
TPM_WRONGPCRVAL	O	24	0x 00000018	The named PCR value does not match the current PCR value.
TPM_BAD_PARAM_SIZE	O	25	0x 00000019	The paramSize argument to the command has the incorrect value
TPM_SHA_THREAD	O	26	0x 0000001A	There is no existing SHA-1 thread.
TPM_SHA_ERROR	O	27	0x 0000001B	The calculation is unable to proceed because the existing SHA-1 thread has already encountered an error.
TPM_FAILEDSELFTEST	O	28	0x 0000001C	Self-test has failed and the TPM has shutdown.
TPM_AUTH2FAIL	O	29	0x 0000001D	The authorization for the second key in a 2 key function failed authorization
TPM_BADTAG	O	30	0x 0000001E	The tag value sent to for a command is invalid
TPM_IOERROR	O	31	0x 0000001F	An IO error occurred transmitting information to the TPM
TPM_ENCRYPT_ERROR	O	32	0x 00000020	The encryption process had a problem.
TPM_DECRYPT_ERROR	O	33	0x 00000021	The decryption process did not complete.
TPM_INVALID_AUTHHANDLE	O	34	0x 00000022	An invalid handle was used.
TPM_NO_ENDORSEMENT	O	35	0x 00000023	The TPM does not a EK installed
TPM_INVALID_KEYUSAGE	O	36	0x 00000024	The usage of a key is not allowed
TPM_WRONG_ENTITYTYPE	O	37	0x 00000025	The submitted entity type is not allowed
TPM_INVALID_POSTINIT	O	38	0x 00000026	The command was received in the wrong sequence relative to TPM_Init and a subsequent TPM_Startup
TPM_INAPPROPRIATE_SIG	O	39	0x 00000027	Signed data cannot include additional DER information
TPM_BAD_KEY_PROPERTY	O	40	0x 00000028	The key properties in TPM_KEY_PARMs are not supported by this TPM
TPM_BAD_MIGRATION	O	41	0x 00000029	The migration properties of this key are incorrect.
TPM_BAD_SCHEME	O	42	0x 0000002A	The signature or encryption scheme for this key is incorrect or not permitted in this situation.
TPM_BAD_DATASIZE	O	43	0x 0000002B	The size of the data (or blob) parameter is bad or inconsistent with the referenced key
TPM_BAD_MODE	O	44	0x 0000002C	A mode parameter is bad, such as capArea or subCapArea for TPM_GetCapability, physicalPresence parameter for TPM_PhysicalPresence, or migrationType for TPM_CreateMigrationBlob.
TPM_BAD_PRESENCE	O	45	0x 0000002D	Either the physicalPresence or physicalPresenceLock bits have the wrong value
TPM_BAD_VERSION	O	46	0x 0000002E	The TPM cannot perform this version of the capability
TPM_NO_WRAP_TRANSPORT		47	0x 0000002F	The TPM does not allow for wrapped transport sessions
TPM_AUDITFAIL_UNSUCCESSFUL		48	0x 00000030	TPM audit construction failed and the underlying command was returning a failure code also
TPM_AUDITFAIL_SUCCESSFUL		49	0x 00000031	TPM audit construction failed and the underlying command was returning success
TPM_NOTRESETABLE		50	0x 00000032	Attempt to reset a PCR register that does not have the resettable attribute
TPM_NOTLOCAL		51	0x 00000033	Attempt to reset a PCR register that requires locality and locality modifier not part of command transport
TPM_BAD_TYPE		52	0x 00000034	Make identity blob not properly typed
TPM_INVALID_RESOURCE		53	0x 00000035	When saving context identified resource type does not match actual resource
TPM_NOTFIPS		54	0x 00000036	The TPM is attempting to execute a command only available when in FIPS mode
TPM_INVALID_FAMILY		55	0x 00000037	The command is attempting to use an invalid family ID
TPM_NO_NV_PERMISSION		56	0x 00000038	The permission to manipulate the NV storage is not available
TPM_REQUIRES_SIGN		57	0x 00000039	The operation requires a signed command
TPM_KEY_NOTSUPPORTED		58	0x 0000003A	Wrong operation to load an NV key
TPM_AUTH_CONFLICT		59	0x 0000003B	NV_LoadKey blob requires both owner and blob authorization
TPM_AREA_LOCKED		60	0x 0000003C	The NV area is locked and not writable
TPM_BAD_LOCALITY		61	0x 0000003D	The locality is incorrect for the attempted operation
TPM_READ_ONLY		62	0x 0000003E	The NV area is read only and can't be written to
TPM_PER_NOWRITE		63	0x 0000003F	There is no protection on the write to the NV area
TPM_FAMILYCOUNT		64	0x 00000040	The family count value does not match
TPM_WRITE_LOCKED		65	0x 00000041	The NV area has already been written to
TPM_BAD_ATTRIBUTES		66	0x 00000042	The NV area attributes conflict
TPM_INVALID_STRUCTURE		67	0x 00000043	The structure tag and version are invalid or inconsistent
TPM_KEY_OWNER_CONTROL		68	0x 00000044	The key is under control of the TPM Owner and can only be evicted by the TPM Owner.
TPM_BAD_COUNTER		69	0x 00000045	The counter handle is incorrect
TPM_NOT_FULLWRITE		70	0x 00000046	The write is not a complete write of the area
TPM_CONTEXT_GAP		71	0x 00000047	The gap between saved context counts is too large
TPM_MAXNVWRITES		72	0x 00000048	The maximum number of NV writes without an owner has been exceeded
TPM_NOOPERATOR		73	0x 00000049	No operator AuthData value is set
TPM_RESOURCEMISSING		74	0x 0000004A	The resource pointed to by context is not loaded
TPM_DELEGATE_LOCK		75	0x 0000004B	The delegate administration is locked
TPM_DELEGATE_FAMILY		76	0x 0000004C	Attempt to manage a family other then the delegated family
TPM_DELEGATE_ADMIN		77	0x 0000004D	Delegation table management not enabled
TPM_TRANSPORT_NOTEXCLUSIVE		78	0x 0000004E	There was a command executed outside of an exclusive transport session
TPM_OWNER_CONTROL		79	0x 0000004F	Attempt to context save a owner evict controlled key
TPM_DAA_RESOURCES		80	0x 00000050	The DAA command has no resources available to execute the command
TPM_DAA_INPUT_DATA0		81	0x 00000051	The consistency check on DAA parameter inputData0 has failed.
TPM_DAA_INPUT_DATA1		82	0x 00000052	The consistency check on DAA parameter inputData1 has failed.
TPM_DAA_ISSUER_SETTINGS		83	0x 00000053	The consistency check on DAA_issuerSettings has failed.
TPM_DAA_TPM_SETTINGS		84	0x 00000054	The consistency check on DAA_tpmSpecific has failed.
TPM_DAA_STAGE		85	0x 00000055	The atomic process indicated by the submitted DAA command is not the expected process.
TPM_DAA_ISSUER_VALIDITY		86	0x 00000056	The issuer's validity check has detected an inconsistency
TPM_DAA_WRONG_W		87	0x 00000057	The consistency check on w has failed.
TPM_BAD_HANDLE		88	0x 00000058	The handle is incorrect
TPM_BAD_DELEGATE		89	0x 00000059	Delegation is not correct
TPM_BADCONTEXT		90	0x 0000005A	The context blob is invalid
TPM_TOOMANYCONTEXTS		91	0x 0000005B	Too many contexts held by the TPM
TPM_MA_TICKET_SIGNATURE		92	0x 0000005C	Migration authority signature validation failure
TPM_MA_DESTINATION		93	0x 0000005D	Migration destination not authenticated
TPM_MA_SOURCE		94	0x 0000005E	Migration source incorrect
TPM_MA_AUTHORITY		95	0x 0000005F	Incorrect migration authority
TPM_PERMANENTEK		97	0x 00000061	Attempt to revoke the EK and the EK is not revocable
TPM_BAD_SIGNATURE		98	0x 00000062	Bad signature of CMK ticket
TPM_NOCONTEXTSPACE		99	0x 00000063	There is no room in the context list for additional contexts
TPM_RETRY	O	2048	0x 00000800	The TPM is too busy to respond to the command immediately, but the command could be resubmitted at a later time.The TPM MAY return TPM_RETRY for any command at any time.
TPM_NEEDS_SELFTEST		2049	0x 00000801	TPM_ContinueSelfTest has not been run.
TPM_DOING_SELFTEST		2050	0x 00000802	The TPM is currently executing the actions of TPM_ContinueSelfTest because the ordinal required resources that have not been tested.
TPM_DEFEND_LOCK_RUNNING		2051	0x 00000803	The TPM is defending against dictionary attacks and is in some time-out period.

TSS V1.2 Error Code Cheat Sheet

Error		TDDL		TCS		TSP		Description
	offset	Dec	Hex	Dec	Hex	Dec	Hex	
TDDL/TSS_SUCCESS	0	0	0x 00000000	0	0x 00000000	0	0x 00000000	Successful completion of the operation.
TDDL/TSS_E_FAIL	2	4098	0x 00001002	8194	0x 00002002	12290	0x 00003002	An internal error has been detected, but the source is unknown.
TDDL/TSS_E_BAD_PARAMETER	3	4099	0x 00001003	8195	0x 00002003	12291	0x 00003003	One or more parameter is bad.
TSS_E_INTERNAL_ERROR	4			8196	0x 00002004	12292	0x 00003004	An internal SW error has been detected.
TDDL/TSS_E_OUTOFMEMORY	5	4101	0x 00001005	8197	0x 00002005	12293	0x 00003005	Ran out of memory.
TSS_E_NOTIMPL	6			8198	0x 00002006	12294	0x 00003006	Not implemented.
TSS_E_KEY_ALREADY_REGISTERED	8			8200	0x 00002008	12296	0x 00003008	Key is already registered
TSS_E_TPM_UNEXPECTED	16			8208	0x 00002010	12304	0x 00003010	An unexpected TPM error has occurred.
TSS_E_COMM_FAILURE	17			8209	0x 00002011	12305	0x 00003011	A communications error with the TPM has been detected.
TSS_E_TIMEOUT	18			8210	0x 00002012	12306	0x 00003012	The operation has timed out.
TSS_E_TPM_UNSUPPORTED_FEATURE	20			8212	0x 00002014	12308	0x 00003014	The TPM does not support the requested feature.
TSS_E_CANCELED	22			8214	0x 00002016	12310	0x 00003016	The action was canceled by request.
TSS_E_PS_KEY_NOTFOUND	32			8224	0x 00002020	12320	0x 00003020	The key cannot be found in the persistent storage database.
TSS_E_PS_KEY_EXISTS	33			8225	0x 00002021	12321	0x 00003021	The key already exists in the persistent storage database.
TSS_E_PS_BAD_KEY_STATE	34			8226	0x 00002022	12322	0x 00003022	The key data set not valid in the persistent storage database.
TDDL_E_ALREADY_OPENED	129	4225	0x 00001081					The connection was already established.
TDDL_E_ALREADY_CLOSED	130	4226	0x 00001082					The device was not connected.
TDDL_E_INSUFFICIENT_BUFFER	131	4227	0x 00001083					The receive buffer is too small.
TDDL_E_COMMAND_COMPLETED	132	4228	0x 00001084					The command has already completed.
TDDL_E_COMMAND_ABORTED	133	4229	0x 00001085					TPM aborted processing of command.
TDDL_E_IOERROR	135	4231	0x 00001087					The request could not be performed because of an I/O device
TDDL_E_BADTAG	136	4232	0x 00001088					Unsupported TAG is requested
TDDL_E_COMPONENT_NOT_FOUND	137	4233	0x 00001089					the requested TPM component was not found
TSS_E_INVALID_OBJECT_TYPE	257			8449	0x 00002101	12545	0x 00003101	Object type not valid for this operation.
TSS_E_NO_CONNECTION	258			8450	0x 00002102	12546	0x 00003102	Core Service connection doesn't exist.
TSS_E_CONNECTION_FAILED	259			8451	0x 00002103	12547	0x 00003103	Core Service connection failed.
TSS_E_CONNECTION_BROKEN	260			8452	0x 00002104	12548	0x 00003104	Communication with Core Service failed.
TSS_E_HASH_INVALID_ALG	261			8453	0x 00002105	12549	0x 00003105	Invalid hash algorithm.
TSS_E_HASH_INVALID_LENGTH	262			8454	0x 00002106	12550	0x 00003106	Hash length is inconsistent with hash algorithm.
TSS_E_HASH_NO_DATA	263			8455	0x 00002107	12551	0x 00003107	Hash object has no internal hash value.
TSS_E_INVALID_ATTRIB_FLAG	265			8457	0x 00002109	12553	0x 00003109	Flag value for attrib-functions inconsistent.
TSS_E_INVALID_ATTRIB_SUBFLAG	266			8458	0x 0000210A	12554	0x 0000310A	Subflag value for attrib-functions inconsistent.
TSS_E_INVALID_ATTRIB_DATA	267			8459	0x 0000210B	12555	0x 0000310B	Data for attrib-functions invalid.
TSS_E_INVALID_OBJECT_INITFLAG	268			8460	0x 0000210C	12556	0x 0000310C	Wrong flag information for object creation.
TSS_E_NO_PCRS_SET	269			8461	0x 0000210D	12557	0x 0000310D	No PCR register are selected or set.
TSS_E_KEY_NOT_LOADED	270			8462	0x 0000210E	12558	0x 0000310E	The addressed key is currently not loaded.
TSS_E_KEY_NOT_SET	271			8463	0x 0000210F	12559	0x 0000310F	No key information is currently available.
TSS_E_VALIDATION_FAILED	272			8464	0x 00002110	12560	0x 00003110	Internal validation of data failed.
TSS_E_TSP_AUTHREQUIRED	273					12561	0x 00003111	Authorization is required.
TSS_E_TSP_AUTH2REQUIRED	274					12562	0x 00003112	Multiple authorization is required.
TSS_E_TSP_AUTHFAIL	275					12563	0x 00003113	Authorization failed.
TSS_E_TSP_AUTH2FAIL	276					12564	0x 00003114	Multiple authorization failed.
TSS_E_KEY_NO_MIGRATION_POLICY	277			8469	0x 00002115	12565	0x 00003115	There's no migration policy object set for the addressed key.
TSS_E_POLICY_NO_SECRET	278			8470	0x 00002116	12566	0x 00003116	No secret information is currently available for the addressed policy object.
TSS_E_INVALID_OBJ_ACCESS	279			8471	0x 00002117	12567	0x 00003117	The operation failed due to an invalid object status.
TSS_E_INVALID_ENCSCHEME	280			8472	0x 00002118	12568	0x 00003118	Invalid encryption scheme.
TSS_E_INVALID_SIGSCHEME	281			8473	0x 00002119	12569	0x 00003119	Invalid signature scheme.
TSS_E_ENC_INVALID_LENGTH	288			8480	0x 00002120	12576	0x 00003120	Invalid length of data to be encrypted.
TSS_E_ENC_NO_DATA	289			8481	0x 00002121	12577	0x 00003121	No data to encrypt.
TSS_E_ENC_INVALID_TYPE	290			8482	0x 00002122	12578	0x 00003122	Invalid encryption type.
TSS_E_INVALID_KEYUSAGE	291			8483	0x 00002123	12579	0x 00003123	Invalid key usage.
TSS_E_VERIFICATION_FAILED	292			8484	0x 00002124	12580	0x 00003124	Verification of signature failed.
TSS_E_HASH_NO_IDENTIFIER	293			8485	0x 00002125	12581	0x 00003125	Hash algorithm identifier is not set.
TSS_E_INVALID_HANDLE	294			8486	0x 00002126	12582	0x 00003126	An invalid handle
TSS_E_SILENT_CONTEXT	295			8487	0x 00002127	12583	0x 00003127	A silent context requires user input
TSS_E_EK_CHECKSUM	296			8488	0x 00002128	12584	0x 00003128	TSP is instructed to verify the EK checksum and it does not
TSS_E_DELGATION_NOTSET	297			8489	0x 00002129	12585	0x 00003129	The Policy object does not have a delegation blob set.
TSS_E_DELFAMILY_NOTFOUND	304			8496	0x 00002130	12592	0x 00003130	The specified delegation family was not found
TSS_E_DELFAMILY_ROWEXISTS	305			8497	0x 00002131	12593	0x 00003131	The specified delegation family table row is already in use and the command flags does not allow the TSS to overwrite the
TSS_E_VERSION_MISMATCH	306			8498	0x 00002132	12594	0x 00003132	The specified delegation family table row is already in use and the command flags does not allow the TSS to overwrite the
TSS_E_DAA_AR_DECRYPTION_ERROR	307			8499	0x 00002133	12595	0x 00003133	Decryption of the encrypted pseudonym has failed, due to either a wrong secret key or a wrong decryption condition.
TSS_E_DAA_AUTHENTICATION_ERROR	308			8500	0x 00002134	12596	0x 00003134	The TPM could not be authenticated by the DAA Issuer.
TSS_E_DAA_CHALLENGE_RESPONSE_ERROR	309			8501	0x 00002135	12597	0x 00003135	DAA Challenge response error.
TSS_E_DAA_CREDENTIAL_PROOF_ERROR	310			8502	0x 00002136	12598	0x 00003136	Verification of the credential TSS_DAA_CRED_ISSUER issued by the DAA Issuer has failed.
TSS_E_DAA_CREDENTIAL_REQUEST_PROOF_E	311			8503	0x 00002137	12599	0x 00003137	Verification of the platform's credential request
TSS_E_DAA_ISSUER_KEY_ERROR	312			8504	0x 00002138	12600	0x 00003138	DAA Issuer's authentication key chain could not be verified or is not correct.
TSS_E_DAA_PSEUDONYM_ERROR	313			8505	0x 00002139	12601	0x 00003139	While verifying the pseudonym of the TPM, the private key of the TPM was found on the rogue list.
TSS_E_INVALID_RESOURCE	314			8506	0x 0000213A	12602	0x 0000313A	Pointer to memory wrong.
TSS_E_NV_AREA_EXIST	315			8507	0x 0000213B	12603	0x 0000313B	The NV area referenced already exists
TSS_E_NV_AREA_NOT_EXIST	316			8508	0x 0000213C	12604	0x 0000313C	The NV area referenced doesn't exist
TSS_E_TSP_TRANS_AUTHFAIL	317					12605	0x 0000313D	The transport session authorization failed
TSS_E_TSP_TRANS_AUTHREQUIRED	318					12606	0x 0000313E	Authorization for transport is required
TSS_E_TSP_TRANS_NOT_EXCLUSIVE	319					12607	0x 0000313F	A command was executed outside of an exclusive transport
TSS_E_TSP_TRANS_FAIL	320					12608	0x 00003140	Generic transport protection error.
TSS_E_TSP_TRANS_NO_PUBKEY	321					12609	0x 00003141	A command could not be executed through a logged transport session because the command used a key and the key's public key is not known to the TSP.
TSS_E_NO_ACTIVE_COUNTER	322			8514	0x 00002142	12610	0x 00003142	The TPM active counter has not been set yet.

Reference: TCG Software Stack (TSS) Specification Version 1.2 Level 1Errata A

Seiji Munetoh, 2008-02-18

NAP Error Constants

The following NAP error constants are defined in naperror.h .

NAP_E_INVALID_PACKET
_HRESULT_TYPEDEF_(0x80270001L)

The NAP **SoH** packet is invalid.

NAP_E_MISSING_SOH
_HRESULT_TYPEDEF_(0x80270002L)

An **SoH** was missing from the NAP packet.

NAP_E_CONFLICTING_ID
_HRESULT_TYPEDEF_(0x80270003L)

The entity ID conflicts with an already-registered entity ID.

NAP_E_NO_CACHED_SOH
_HRESULT_TYPEDEF_(0x80270004L)

No cached **SoH** is present.

NAP_E_STILL_BOUND
_HRESULT_TYPEDEF_(0x80270005L)

The entity is still bound to the NAP system.

NAP_E_NOT_REGISTERED
_HRESULT_TYPEDEF_(0x80270006L)

The entity is not registered with the NAP system.

NAP_E_NOT_INITIALIZED
_HRESULT_TYPEDEF_(0x80270007L)

The entity is not initialized with the NAP system.

NAP_E_MISMATCHED_ID
_HRESULT_TYPEDEF_(0x80270008L)

The correlation IDs in the **SoH** request and **SoH** response do not match up.

NAP_E_NOT_PENDING
_HRESULT_TYPEDEF_(0x80270009L)

Completion was indicated on a request that is not currently pending.

NAP_E_ID_NOT_FOUND
_HRESULT_TYPEDEF_(0x8027000AL)

The NAP component's ID was not found.

NAP_E_MAXSIZE_TOO_SMALL
_HRESULT_TYPEDEF_(0x8027000BL)

The maximum size of the connection is too small for an SoH packet.

NAP_E_SERVICE_NOT_RUNNING
_HRESULT_TYPEDEF_(0x8027000CL)

The NapAgent service is not running.

NAP_S_CERT_ALREADY_PRESENT
_HRESULT_TYPEDEF_(0x0027000DL)

A certificate is already present in the certificate store.

NAP_E_ENTITY_DISABLED
_HRESULT_TYPEDEF_(0x8027000EL)

The entity is disabled with the NapAgent service.

NAP_E_NETSH_GROUPPOLICY_ERROR
_HRESULT_TYPEDEF_(0x8027000FL)

Group Policy is not configured.

NAP_E_TOO_MANY_CALLS
_HRESULT_TYPEDEF_(0x80270010L)

Too many simultaneous calls.

NAP_E_SHV_CONFIG_EXISTED
_HRESULT_TYPEDEF_(0x80270011L)

SHV configuration already exists.

Note Supported in Windows 7 or later.

NAP_E_SHV_CONFIG_NOT_FOUND
_HRESULT_TYPEDEF_(0x80270012L)

SHV configuration is not found.

Note Supported in Windows 7 or later.

NAP_E_SHV_TIMEOUT
_HRESULT_TYPEDEF_(0x80270013L)

SHV timed out on the request.

Note Supported in Windows 7 or later.

Requirements

Minimum supported client	Windows Vista
Minimum supported server	Windows Server 2008
Header	NapError.h

See Also

NAP Constants

Send comments about this topic to Microsoft

Build date: 7/2/2010

Community Content

Literaturverzeichnis

[1] Microsoft Corporation. Clock Skew - Kerberos V5 System Administrator's Guide. http://web.mit.edu/kerberos/krb5-1.5/krb5-1.5.4/doc/krb5-admin/Clock-Skew.html. Beschreibung der Zeitdifferenz in Kerberos, Letzte Konsultation: 13. Juni 2011.

[2] Microsoft Corporation. [MS-RNAP]: Vendor-Specific RADIUS Attributes for Network Access Protection (NAP) Data Structure. http://msdn.microsoft.com/en-us/library/aa369721%28v=vs.85%29.aspx. Informationen über die Vendor Specific Data Structure, Letzte Konsultation: 13. Juni 2011.

[3] Microsoft Corporation. [MS-SOH]: Statement of Health for Network Access Protection (NAP) Protocol Specification. http://download.microsoft.com/download/9/5/E/95EF66AF-9026-4BB0-A41D-A4F81802D92C/%5BMS-SOH%5D.pdf. Release: 4. Mai 2011, Letzte Konsultation: 13. Juni 2011.

[4] Microsoft Corporation. NAP Error Constants. http://msdn.microsoft.com/en-us/library/bb427404%28v=vs.85%29.aspx. Auflistung der NAP-Fehlercodes und deren Bedeutung, Letzte Konsultation: 13. Juni 2011.

[5] Microsoft Corporation. Tools for Troubleshooting NAP. http://msdn.microsoft.com/en-us/library/ms708353%28v=VS.85%29.aspx. Letzte Konsultation: 13. Juni 2011.

[6] 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: 13. Juni 2011.

[7] Trusted Computing Group. TCG Architecture Overview. http://www.trustedcomputinggroup.org/resources/tcg_architecture_overview_version_14. Version 1.4, Letzte Konsultation: 13. Juni 2011.

[8] Trusted Computing Group. TNC IF-TNCCS Specification. http://www.trustedcomputinggroup.org/resources/tnc_iftnccs_specification. TLV Binding Version 2.0, Revision 16, Letzte Konsultation: 13. Juni 2011.

[9] Trusted Computing Group. TPM Main Specification. http://www.trustedcomputinggroup.org/resources/tpm_main_specification. Letzte Konsultation: 13. Juni 2011.

[10] Seiji Munetoh. TPM V1.2 Error Code Cheat Sheet. http://sourceforge.jp/projects/openpts/wiki/FrontPage/attach/20080218-TPMv12ErrorCodeCheatSheet.pdf. Auflistung der TPM-Fehlercodes und deren Bedeutung, Letzte Konsultation: 13. Juni 2011.

[11] Seiji Munetoh. TSS V1.2 Error Code Cheat Sheet. http://sourceforge.jp/projects/openpts/wiki/FrontPage/attach/20080218-TSSv12ErrorCodeCheatSheet.pdf. Auflistung der TSS-Fehlercodes und deren Bedeutung, Letzte Konsultation: 13. Juni 2011.

[12] PrivacyCA. PrivacyCA API. http://www.privacyca.com/api.html. API-Beschreibung, Letzte Konsultation: 13. Juni 2011.

[13] PrivacyCA. PrivacyCA Sample Code. http://www.privacyca.com/code.html. API-Beschreibung, Letzte Konsultation: 13. Juni 2011.

[14] Infineon Technologies. Trusted Platform Module. http://www.infineon.com/cms/en/product/chip-card-and-security-ics/embedded-security/trusted-platform-management/trusted-platform-module-%28tpm1.2%29/channel.html?channel=ff80808112ab681d0112ab6921ae011f. Informationen über das TPM mit Auflistung der Computerhersteller, Letzte Konsultation: 13. Juni 2011.

[15] Koordinationsstelle zur Bekämpfung der Internet-Kriminalität KOBIK. Jahresbericht 2009. http://www.kobik.ch/report/Rechenschaftsbericht_2009_DE.pdf. Letzte Konsultation: 13. Juni 2011.

Tabellenverzeichnis

3.1 Gegenüberstellung der Begriffe von NAP und TNC 19

4.1 Anforderungen an die Hardware 35

7.1 Verfügbare Datentypen für das TLV 72
7.2 Konfigurationswerte in der Registry 85
7.3 Rückgabewerte der get*Key Funktionen 86

8.1 List-Befehlsübersicht 108

A.1 Projektmitglieder 125
A.2 Betreuungspersonen 126
A.3 Meilensteine 129
A.4 Verantwortlichkeiten 131

B.1 Zeitauswertung: Wochenschnitt und Total 133

Abbildungsverzeichnis

1 Logo des resultierten Produktes *Trusted NAP* iv

2.1 Übersicht Umfeldanalyse 7
2.2 Mind-Map Umfeldanalyse 9

3.1 TPM: Aufbau und Funktionalitäten[7] 13
3.2 TPM: Schlüsselhierarchie[7] 16
3.3 Architektur-Übersicht: Trusted Network Connect[8] 17
3.4 Verbinden in das Unternehmensnetz mit einem policykonformen Client 20
3.5 Verbinden in das Unternehmensnetz mit einem *nicht* policykonformen Client 21
3.6 Der TCG Software Stack 23
3.7 Netzwerkübersicht Testumgebung 27
3.8 Übersicht virtuelle Netzwerke des NAP [6] 27

4.1 Systemschnittstellen 35
4.2 Programm *napstat* für die Ausgabe von NAP-Stati 36
4.3 Use Cases 41

5.1 Ablauf Trusted NAP 46
5.2 Übersicht über den Challenge- / Responseablauf 48
5.3 Messwerte der NAP-Dateien 50
5.4 Messwerte für Trusted SHA und TrouSerS 51
5.5 Challenge-/Response-Vorgang mit Zeit und Datum als Nonce . . 52
5.6 Ablauf Attestation 52
5.7 Initialisierungsablauf des TPMs 54
5.8 Übersicht über die Komponenten 55
5.9 Trusted SHA Domain Model 56
5.10 Trusted SHV Domain Model 59

6.1 Architekturübersicht 62
6.2 Projekt in der TrustedNAP Solution 63
6.3 Übersicht TPM-Klassen 64
6.4 Übersicht Registry-Klassen 64
6.5 Übersicht Logging-Klassen 65

7.1 Aufbau eines Feldes einer TLV-Nachricht 71
7.2 Ablauf openNewConnection() 75
7.3 PCR Maske für 24 Register über 3 Byte 76
7.4 Ablauf Zertifikatsanforderung 80
7.5 Registry Struktur des Trusted SHVs und Trusted SHAs 85
7.6 Ablauf TakeOwnership 92

8.1 HSR-Schlüssel in der Registry 101
8.2 Vergabe der Leserechte für NETWORK SERVICE am Registry-Key HSR 102
8.3 Erfolgreiches Registrieren der TrustedShv.dll mit Administratorrechten 103
8.4 Aktivieren des Trusted SHV innerhalb der Network Policy Server Health Policies 104
8.5 TPM zurücksetzen auswählen 112
8.6 TPM aktivieren 113
8.7 Wechsel ins Reiter Security des BIOS 114
8.8 TPM einschalten und aktivieren 114
8.9 TPM Managementumgebung 115
8.10 Von der Gruppenrichtlinie blockierte Befehle 115
8.11 napstat auf einem *nicht konformen* Rechner 116
8.12 napstat auf einem *konformen* Rechner 116
8.13 Detailansicht von napstat auf einem konformem Rechner 117
8.14 Detailansicht von napstat auf einem nicht konformem Rechner . 118
8.15 Durch Gruppenrichtlinien deaktivierte TPM-Befehle (tpm.msc) . 119
8.16 Vorgehen für eine Command Prompt mit Administratorrechten . 120

A.1 Risikomanagement Teil 1 127
A.2 Risikomanagement Teil 2 127

B.1 Aufgewendete Stunden pro Person 134
B.2 Aufteilung der Aufwände insgesamt 134
B.3 Aufteilung der Aufwände für die Implementation 135

Listings

7.1 Registrierung des Windows Services 71
7.2 Pseudocode für das Auswerten einer Nachricht 73
7.3 Generierung der PCR-Maske (Hinzufügen eines PCR) 76
7.4 Speichern eines Wertes in einem PCR 77
7.5 Pseudocode für die Funktion *extendMeasurementsToPcrs()* . . . 77
7.6 Definition des *TPM_QUOTE_INFO* nach der TCG[9] 78
7.7 Vorbereiten einer Verbindung mit WinHTTP 82
7.8 Generierung des Fingerprints mit Hilfe von OpenSSL 83
7.9 Aufbau der wichtigsten Funktionen des BaseLogger 89
7.10 Beispielnachricht in der Log-Datei 89
7.11 Code zu takeOwnershipSha1 93