HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

ita Institute for Internet Technologies and Applications

strongSwan Android 4 Client with Endpoint Assessment

Studienarbeit

Abteilung Informatik

Hochschule für Technik Rapperswil

[Herbstsemester 2012]

Autor(en): Christoph Bühler
Patrick Lötscher

Betreuer: Prof. Dr. Andreas Steffen

Abstract

Zielsetzung

„Bring your own device“ (BYOD), ein grosses Thema der modernen Computerzeit. Smartphones sind nicht mehr nur für Businessleute wichtig, vielmehr ist das Smartphone zu einem Alltagsgegenstand geworden. Klar ist auch, dass die Funktionalitäten des intelligenten Telefons genutzt werden wollen. So gibt es viele Firmennetze, welche die Benutzer mit ihren eigenen Geräten ins Netz lassen.

Das Ziel dieser Studienarbeit ist, die bestehende Implementation der strongSwan – Android App so zu erweitern, dass eine Validierung der Endgeräte möglich wird. Zu diesem Zweck sollen die Implementationsteile in C und Java dahingehend erweitert werden, dass eine dynamische Validierung ermöglicht wird. Der Server sendet dem Client sogenannte Attribute-Requests, welche vom Client verarbeitet und beantwortet werden. Sollte ein Gerät gewisse Kriterien nur zum Teil oder gar nicht erfüllen, so wird die Verbindung geschlossen oder in eine Quarantänezone verschoben, wo nur eingeschränkter Zugriff möglich ist.

Dies erlaubt Benutzern den Zugriff auf ein Firmennetzwerk mit ihren eigenen Geräten, ohne die Sicherheit des Netzwerks zu gefährden.

Ergebnis

Mittels des Trusted Network Connect Protokolls (PA-TNC [1]) kommuniziert der Android Client mit einem Server. Der Server selbst dient als Zertifizierungsstelle und hält die „Integrity Measurement Verifier“ (IMV) bereit, welche die Resultate des Clients überprüfen. Der Client selbst hat einen „Integrity Measurement Collector“ (IMC) welcher die Messung im Auftrag des Servers vornimmt. Ein Teil der Implementation erfolgte in Java, ein anderer in C. Die native (C) Implementation ist zuständig für die Kommunikation mit dem Server (IMVs) und die Weiterleitung von Nachrichten an die Java-Implementation.

Mit dem AndroidIMC ist ein generischer Java-Manager entstanden, welcher die einzelnen Messungen im Auftrag des nativen Codes durchführt. Dieser kann nach Belieben erweitert werden. Werden die Messungen durchgeführt, so wird vom Server zum Schluss des Verbindungsvorganges und nach den verschiedensten Abklärungen ein Resultat zum Client gesandt. Ist dieses Positiv, so darf der Client die Verbindung aufbauen.

Ein neues GUI-Element zeigt dem Benutzer den Status der Validierung an. Dieser ändert sich nach jedem Verbindungsversuch und widerspiegelt die letzte Messung. Der Benutzer kann diese Resultate stets anzeigen und sich darüber informieren, wie diese Fehler zu beseitigen sind.

Christoph Bühler
Patrick Lötscher

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:

Christoph Bühler

Ort, Datum:

Patrick Lötscher

Aufgabenstellung

Studienarbeit 2012

strongSwan Android 4 Client with Endpoint Assessment

Studenten: Christoph Bühler & Patrick Lötscher

Betreuer: Prof. Dr. Andreas Steffen

Ausgabe: Montag, 17. September 2012

Abgabe: Freitag, 21. Dezember 2012

Einführung

Im Rahmen einer Bachelorarbeit wurde kürzlich eine strongSwan App für Android 4 entwickelt, welche die Konfiguration, das Aufbauen und das Abbauen von VPN Verbindungen erlaubt. Die App wurde in Java geschrieben und bindet die strongSwan C-Libraries via JNI ein.

Diese Anwendung soll nun mit einer Endpoint-Assessment Fähigkeit ausgestattet werden, also der Möglichkeit, von einem Policy Decision Point aus den Gesundheitszustand eines Android Geräts zu bestimmen. Dazu soll die bestehende Trusted Network Connect Funktionalität von strongSwan in der Form von C-Libraries eingebunden werden. In einem ersten Schritt soll abgeklärt werden, welche Integritätsmessungen ohne "Rooten" des Geräts möglich sind und welche Möglichkeiten sich mit Root-Rechten eröffnen würden.

Als Proof-of-Concept soll ein einfacher Integrity Measurement Collector (IMC) entwickelt werden, der durch die strongSwan Android App geladen oder eingebunden wird, auf dem Gerät einen Gesundheitscheck durchführt, die Messresultate über die bestehende IKEv2 Verbindung an einen Server übermittelt und anschliessend den positiven oder negativen Entscheid des Policy Decision Points grafisch visualisiert..

Aufgabenstellung

- Abklärung der Endpoint Assessment Möglichkeiten unter Android 4 ohne „Rooten“ des Gerätes (Systemdateien, Konfigurationseinstellungen, installierte Apps, offene Netzwerkports, etc).
- Einarbeiten in das Trusted Network Connect Framework der Trusted Computing Group, insbesondere die IETF Standard Attribute, welche im PA-TNC RFC 5792 beschrieben sind.
- Einarbeiten in die Architektur des bestehenden strongSwan Android 4 VPN Clients.
- Bestimmung der optimalen Systemgrenze zwischen Java und C Code im Falle des zu erstellenden Android Integrity Measurement Collectors (IMC).

- Entwurf, Implementation und Testen des Android IMCs, der das Endpoint Assessment vornimmt und den Entscheid und die empfohlenen Massnahmen des Android IMVs via GUI visualisiert.

Links

- P. Sangster and K. Narayan, *PA-TNC: A Posture Attribute (PA) Protocol Compatible with Trusted Network Connect (TNC)*, RFC 5792, March 2010, **http://tools.ietf.org/html/rfc5792**
- P. Sangster et al, *TCG Attestation PTS Protocol: Binding to TNC IF-M*, TCG Specification Version 1.0, Rev. 28, Aug. 24, 2011, **http://www.trustedcomputinggroup.org/resources/tcg_attestation_pts_protocol_binding_to_tnc_ifm**
- A. Steffen, *The Linux Integrity Measurement Architecture and TPM-Based Network Endpoint Assessment*, Linux Security Summit 2012, San Diego, **http://www.strongswan.org/lss2012.pdf**
- TNC Infoseite des strongSwan Projekts **http://www.strongswan.org/tnc/**
- TNC Beispielszenario „Installed Packages“ **http://www.strongswan.org/uml/tnc-os/tnc/tnccs-20-os/**
- strongSwan Software Repository **http://git.strongswan.org/**

Rapperswil, 26. September 2012

Prof. Dr. Andreas Steffen

Christoph Bühler
Patrick Lötscher

Inhalt

Christoph Bühler
Patrick Lötscher

Christoph Bühler
Patrick Lötscher

1. Einleitung

1.1 Ziele

Ziel der Studienarbeit soll sein, das bestehende Android App „strongSwan“ so zu erweitern, dass ein Endpoint Assessment möglich ist. Diese Überprüfung ermöglicht es dem Dienstanbieter sicherzustellen, dass keine verseuchten Geräte in sein Netzwerk gelangen. Es soll möglich sein, die installierte Software auf den Geräten sowie die Systemdateien zu überprüfen, ob diese verändert oder ausgetauscht wurden. So kann sichergestellt werden, dass nur „saubere“ Geräte im Netzwerk sind.

1.2 Gültigkeit

Dieses Dokument und alle enthaltenen Informationen sind für das gesamte Projekt gültig. Diese Arbeit basiert auf einer vergangen Bachelorarbeit, welche zum Ziel hatte, strongSwan auf das Android Betriebssystem zu portieren [2].

2. Grundlagen

2.1 Trusted Network Connect

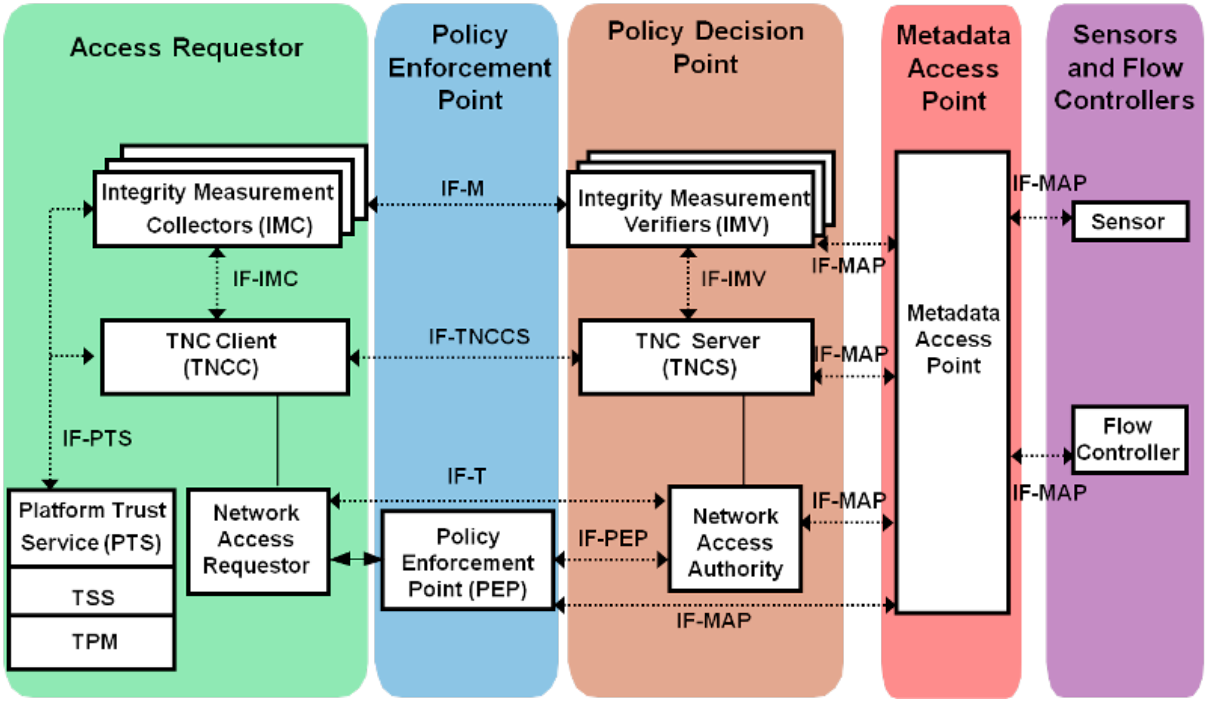

Abbildung 1: TNC Protokoll für die Kommunikation [3][1]

Trusted Network Connect (TNC) ist ein Konzept von der Trusted Computing Group für die Netzwerksicherheit. Mit TNC kann die Integrität und Konformität der Endgeräte sichergestellt werden. Clients können authentifiziert und deren Verbindung zum Server als vertrauenswürdig eingeordnet werden.

[1] http://wiki.strongswan.org/projects/strongswan/wiki/TrustedNetworkConnect

Die Interoperabilität zwischen Endgeräten verschiedener Hersteller wird gewährleistet. Der Server kann die Konfigurationen eines Endgeräts verlangen, anhand derer die Vertrauenswürdigkeit ermitteln und den Zugriff auf die Netzwerkressourcen wenn nötig einschränken.

Diese Arbeit befasst sich nur mit der obersten Schicht der TNC Protokoll-Suite. Die Kommunikation zwischen Client und Server über den VPN Tunnel wird von unteren Schichten beziehungsweise von strongSwan übernommen. Der Integrity Measurement Collector (IMC) wird in Java realisiert. In diesem Fall gibt es genau einen IMC, welcher die verschiedenen Messungen auf dem Endgerät durchführt. Er liefert die Messergebnisse an die C-Schicht, welche ihrerseits die Ergebnisse in das Protokollpaket verpackt und danach an den Server sendet. Der Integrity Measurement Veryfier (IMV) überprüft die Measurements des Endgeräts und erstellt ein Assessment Result. Sofern der IMV Beanstandungen hat, fügt er dem Assessment Result Remediation Instructions bei, die dem Endgerät zur Fehlerbehebung behilflich sind.

2.2 Server / Client Kommunikation

2.2.1 Beschreibung

Die grundlegende Kommunikation zwischen Client und Server in strongSwan ist folgendermassen aufgebaut:

1) Client will Verbindung aufbauen und meldet sich beim Server
2) Server erhält Verbindungs-Request
3) Client authentisiert sich:
   a) Mittels Zertifikat
   b) Mittels Benutzername / Passwort (aktuell bei der HSR so eingerichtet)
4) Serverkonfiguration entscheidet:
   a) Client wird eingelassen
   b) Client muss erst Messungen durchführen (Ziel der Arbeit)
5) Client führt Messungen durch
6) Client übermittelt die Messresultate an den Server
7) Server sendet Assessment Result:
   a) Alles in Ordnung
   b) Kleinere Fehler (z.B. nicht die aktuellsten Virendefinitionen)
   c) Grobe Fehler (z.B. kein Antivirus installiert)

Die Arbeit befasst sich mit den Messungen. Ziel soll es sein, einen (oder mehrere) Integrity Measurement Collector (IMC) zu schreiben, welcher die Messungen vornimmt.

2.2.2 Visualisierung

2.2.2.1 Flowchart

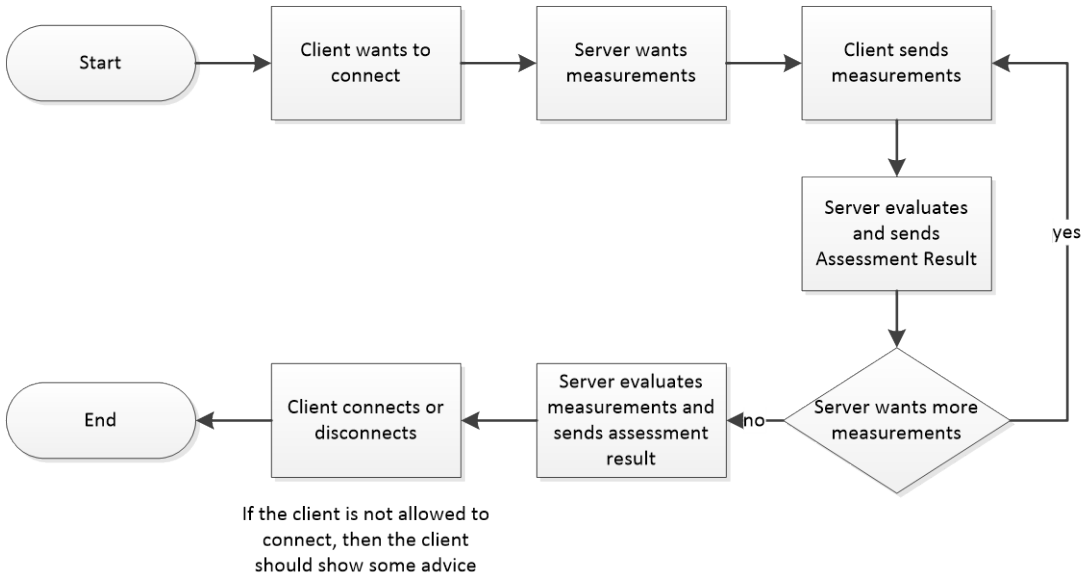

Abbildung 2: Kommunikation zwischen Client und Server

2.2.2.2 Client / Server Diagramm

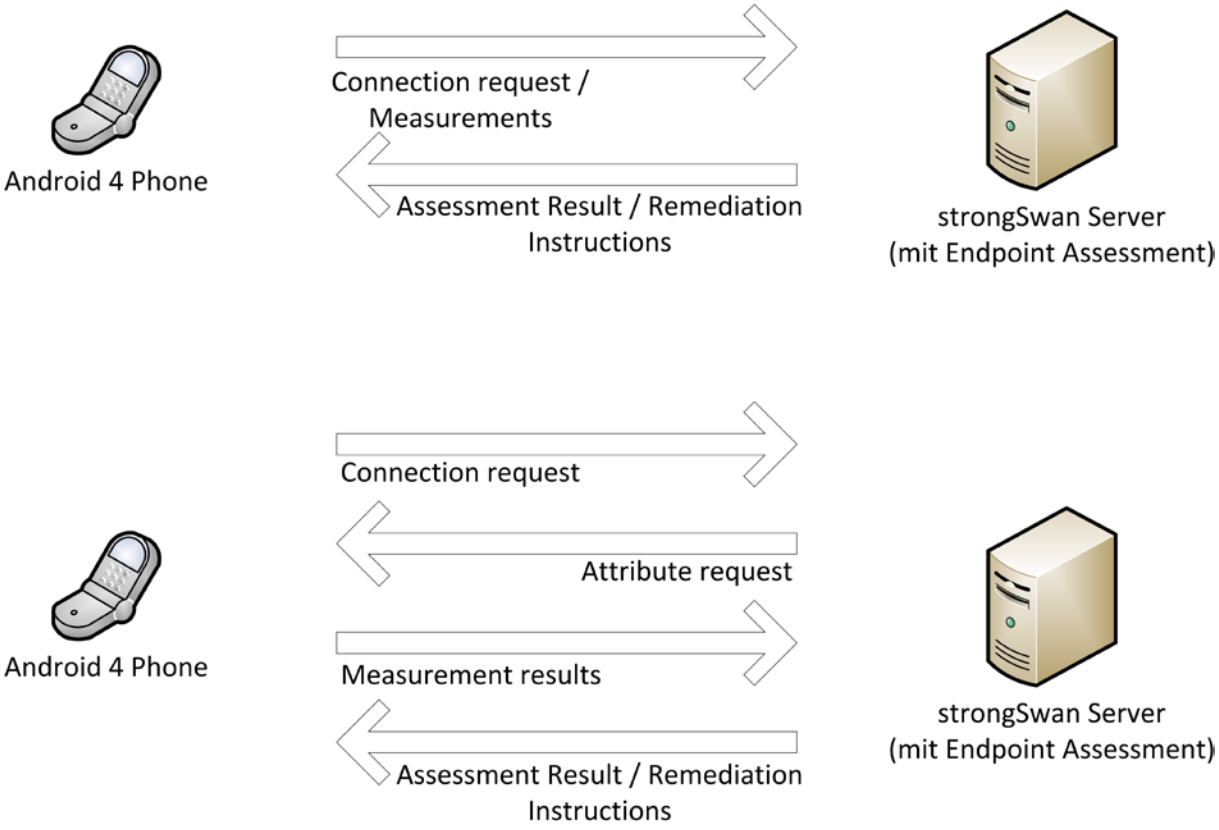

Abbildung 3: Kommunikationsdiagramm Client / Server

2.3 Remediation Instructions

Die Remediation Instructions sind verschiedenste Nachrichten, welche dem Benutzer Anweisung geben, wie er einen Fehler beheben kann. Diese werden vom IMV generiert und zugesandt. Der IMV erstellt diese Instruktionen anhand der Messresultate und der vorhandenen Serverregeln. Ist beispielsweise eine Software auf dem Androidgerät installiert, welche als „Blacklisted“ aufgeführt ist, so erhält der Benutzer eine Nachricht, welche ihn darüber informiert, dass die Software deinstalliert werden muss, um das Netzwerk zu betreten. Sind solche Instruktionen vorhanden, ist entweder ein „minor“ oder ein „major“ Fehler aufgetreten und die Verbindung kam nicht zustande oder ist im Mindesten nur eingeschränkt nutzbar.

Die Instruktionen sind in verschiedenen Sprachen realisiert. Die Sprache wird bei der Initialisierung des IMCs ausgelesen und danach an den Server übermittelt. Dieser sendet dann die entsprechenden Instruktionen an den Client zurück. Ist ein Sprachcode nicht verfügbar, so wird automatisch Englisch als Standardsprache verwendet.

2.4 Vorabklärungen

2.4.1 Einführung

2.4.1.1 Beschreibung

Um für strongSwan sinnvolle Endpoint Assessment Messungen durchführen zu können, müssen erst die Möglichkeiten abgeklärt werden. Zu diesem Zweck wurden mehrere Vorabklärungen in Bezug auf die Berechtigungen und andere kritische Punkte durchgeführt. Im Weiteren sind die Ergebnisse und Erkenntnisse aus den Abklärungen dokumentiert.

2.4.2 Auslesen des Dateisystems

2.4.2.1 Kurzbeschreibung

Das Ziel ist, das Dateisystem auszulesen und bestimmte Systemdateien mittels eines SHA1-Hash zu vergleichen. Sollten die Dateien nicht mit denen des Servers übereinstimmen, so besteht die Möglichkeit, dass das Gerät beziehungsweise das Betriebssystem modifiziert wurde. Dies kann durch Schadsoftware oder durch den Benutzer geschehen und ist somit ein wichtiger Punkt des Endpoint Assessments.

2.4.2.2 Auslesen mit Java

Da auf beinahe allen Dateien der „Read“-Zugriff garantiert ist, kann das Dateisystem ausgelesen werden. Wichtig dabei ist, dass Verknüpfungen ausgelassen werden, da es dann zu Endlosschleifen kommen kann. Grundsätzlich ist der Zugriff auf das Dateisystem mittels der Java Klasse „File“ möglich. Sobald der aktuelle Benutzer (nicht-gerootetes Gerät) keine weiteren Zugriffsberechtigungen besitzt, erhält man eine Exception.

2.4.2.3 Code

```
@Override
protected void onStart() {
	super.onStart();
	showFiles(new File("/system"), "");
}

private int depth = 0;

private void showFiles(File f, String prefix) {
	try {
		for (File subFile : f.listFiles()) {
			System.out.println(prefix + " " + subFile.getName());
			if(subFile.isDirectory()){
				depth++;
				if (depth < 10) showFiles(subFile, prefix + "-");
			}
		}
	} catch (Exception e) {
		System.out.println("EXCEPTION !");
	}
}
```

Der Code ist nicht optimiert, sondern dient nur einer ersten Abklärung der Möglichkeiten.

2.4.2.4 Ergebnisse

Das Dateisystem kann ausgelesen werden, solang die „Read“ Berechtigung vorhanden ist. Es wäre möglich, als BinaryStream, eine Datei in einen SHA1-Hash zu verwandeln und zu vergleichen. Abklärungen über den Typ einer Datei sind nur möglich, solange es sich bei der Datei um eine normale Datei oder ein Verzeichnis handelt. Es ist nicht möglich zu eruieren, ob es sich um einen „Link“ handelt.

2.4.3 Anzeigen der aktuell offenen Ports

2.4.3.1 Kurzbeschreibung

Anhand der offenen Ports soll ermittelt werden, welche Verbindungen im Hintergrund geöffnet sind. Da gewisse Verbindungen, wie z.B. eMule auf TCP Port 4662, in einem Netzwerk nicht erwünscht sind, soll bei geöffnetem Port, welcher geblacklistet ist, der Verbindungsaufbau per VPN verweigert werden. Es soll des Weiteren überprüft werden, ob das Auslesen der Ports ohne Root-Rechte möglich ist.

2.4.3.2 Auslesen mit Java

Das Auslesen der Ports ist mit dem Befehl „netstat –n“ möglich. Dazu muss der Befehl per exec(cmd) an die Runtime geschickt werden. Die Ausgabe kann mittels getInputStream() auf den Prozess, der erstellt wurde, in einen BufferedReader geschrieben werden.

2.4.3.3 Code

```
executeCmd("netstat -n");
```

```
public static ArrayList<String> executeCmd(String cmd) {
	ArrayList<String> portNumbers = new ArrayList<String>();
	try {
		Process p = Runtime.getRuntime().exec(cmd);
		BufferedReader stdInput = new BufferedReader(new
InputStreamReader(p.getInputStream()));

		String s;
		int indexOfPort;
		while ((s = stdInput.readLine()) != null) {
			if (s.contains(":") && !s.contains("*")) {
				s = s.substring(43);
				indexOfPort = s.lastIndexOf(":") + 1;
				portNumbers.add(s.substring(indexOfPort, indexOfPort +
5).trim());
			}
		}
		p.destroy();
		return portNumbers;
	} catch (Exception e) {
		e.printStackTrace();
	}
	return portNumbers;
}
```

2.4.3.4 Ergebnisse

Der BufferedReader liefert zeilenweise den Output von „netstat“. Der ganze Output wird in einem String Array abgelegt, welches von der Methode executeCmd() zurückgegeben wird. Ein Output auf dem Emulator sieht folgendermassen aus:

```
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp        0      0 127.0.0.1:5037         0.0.0.0:*              LISTEN
tcp        0      0 0.0.0.0:5555           0.0.0.0:*              LISTEN
tcp        0      0 10.0.2.15:5555         10.0.2.2:13230         ESTABLISHED
```

Abbildung 4: Output von netstat -n

2.4.4 Anzeigen der aktuell installierten Apps / Packages

2.4.4.1 Kurzbeschreibung

Da die „reale“ Bedrohung durch Schadsoftware, wie der Name schon sagt, von Software ausgeht, müssen die aktuell installierten Apps aufgelistet werden können. Somit sind wir in der Lage, eine Policy für blockierte Applikationen zu erstellen. Diese „blacklisted Apps“ werden dann nicht zugelassen und müssen vom Benutzer deinstalliert werden, bevor er das Netzwerk betreten darf.

2.4.4.2 Auslesen mit Java

In Java (unter Android) ist dies mit dem Package-Manager möglich. Der Package-Manager ist eine Schnittstelle, welche es ermöglicht, die installierten Packages aufzulisten. Danach wird mittels der „ApplicationInfo“ das Label der Applikation (der Name beim Icon im Applicationlauncher) eruiert und in die Liste der Applikationen eingefügt. Es besteht auch die Möglichkeit, direkt den Packagenamen anzuzeigen, welcher dann jedoch auch die gesamte Packagehierarchie repräsentiert. Diese Liste enthält auch Systemapplikationen, welche jedoch nicht zwingend überprüft werden müssen, da diese sowieso mitinstalliert werden. So erscheinen in der aktuellen Liste Applikationen wie „Android System“.

2.4.4.3 Code

```
@Override
protected void onStart() {
	super.onStart();
	appList = (ListView) findViewById(R.id.appList);
	ArrayList<String> appArrayList = new ArrayList<String>();
	final PackageManager pm = getPackageManager();
	// get a list of installed apps
	List<ApplicationInfo> packages = pm
			.getInstalledApplications(PackageManager.GET_META_DATA);
	// loadLabel --> get label name for that app package
	for (ApplicationInfo packageInfo : packages) {
		appArrayList.add(packageInfo.loadLabel(getPackageManager())
				.toString());
	}
	appList.setAdapter(new ArrayAdapter<String>(this,
			android.R.layout.simple_expandable_list_item_1, appArrayList));
}
```

2.4.4.4 Ergebnisse

Die Abfrage mit Java liefert über den Package-Manager die installierten Applikationen als „ApplicationInfo“. Die Klasse ApplicationInfo enthält die Metadaten der Applikationen. Es können Informationen wie Packagename, Sourcedirectory und Minimum-SDK-Version ausgelesen werden. Eine vollständige Auflistung der API dazu ist auf der Android SDK Reference zu finden[2]. Eine distanziertere Abfrage liefert „PackageInfo“ als Resultat. Diese PackageInfo enthalten generische Informationen über das Package und zusätzlich dazu noch die ApplicationInfo. In der PackageInfo sind die benötigten Permissions aufgelistet sowie der Name des Packages und weitere Informationen. Die vollständige Liste ist ebenfalls in der Reference zu finden[3].

[2] http://developer.android.com/reference/android/content/pm/ApplicationInfo.html
[3] http://developer.android.com/reference/android/content/pm/PackageInfo.html

2.4.5 Einstellungen auslesen

2.4.5.1 Kurzbeschreibung

Auf Android-Geräten ist es durch das Betriebssystem möglich, nicht signierte und unsichere Inhalte zu installieren. Diese Einstellung wird unter dem Menüpunkt „Anwendungen“ vorgenommen und heisst „Unsichere Quellen“. Die Idee ist, auch Applikationen installieren zu können, welche nicht über den Google-Play Store verteilt wurden. Was grundsätzlich für den Wettbewerb und die Motivation der Entwickler gut ist, kann sich jedoch auch negativ auswirken, da so potentielle Schadsoftware installiert werden kann, welche nicht zwingend das tut, wofür sie zu sein scheint. Um zu verhindern, dass solche Schadsoftware in das Netzwerk eingeschleust wird, soll versucht werden, diese Einstellungen auszulesen und dazu eine Policy zu erstellen, welche es Benutzern nicht erlaubt, das Netzwerk zu betreten, wenn diese Option aktiviert ist.

2.4.5.2 Auslesen mit Java

Um die Settings des Android Betriebssystems auszulesen, bedarf es nicht vieler Zeilen Code. Android stellt einen Provider zur Verfügung, welcher dann mittels Konstanten auf die entsprechenden Settings zugreifen kann. Damit allenfalls Einstellungen zurückgeschrieben werden können, benötigt die App „WRITE_SETTINGS“ Berechtigungen, welche im Manifest festgehalten werden.

2.4.5.3 Code

```
@Override
protected void onStart() {
	super.onStart();
	System.out.println("Setting getString() value: "
		+ android.provider.Settings.System.getString(
		getContentResolver(),
		android.provider.Settings.Secure.INSTALL_NON_MARKET_APPS));
	int setting = Integer.parseInt(android.provider.Settings.System.getString(
		getContentResolver(),
		android.provider.Settings.Secure.INSTALL_NON_MARKET_APPS));
	CheckBox cb = (CheckBox) findViewById(R.id.checkBox1);
	cb.setChecked(setting == 1 ? true : false);
}
```

2.4.5.4 Ergebnisse

Grundsätzlich können alle in der SDK-Referenz [4] [4] enthaltenen Optionen des Androidphones ausgelesen werden. Als Ergebnis der Abfrage erhält man einen String, welchen man dann in den entsprechenden Typen konvertieren muss. Bei den meisten Optionen handelt es sich um Integer-Werte, welche 0 oder 1 enthalten.

2.4.6 Antivirus / Antispyware Software

2.4.6.1 Kurzbeschreibung

Es soll ermittelt werden, ob eine Antiviren-Software installiert und die Virendefinition „up to date“ ist.

[4] http://developer.android.com/reference/android/provider/Settings.System.html

2.4.6.2 These

Ob eine Antiviren-Software installiert ist, lässt sich nur über das Filesystem ermitteln. Dafür müsste eine Liste von Namen der aktuellen Antiviren-Software vorhanden sein. Das Problem liegt darin, dass diese Liste aktualisiert werden müsste, sobald eine neue Antiviren-Software vorhanden ist.

Des Weiteren ist es schwierig abzufragen, ob die Antiviren-Software „up to date“ ist. Entweder wird das Datum der Virendefinition in einem File oder in einer SQLite Datenbank gespeichert. Es müsste also für jede Antiviren-Software das Datum spezifisch ausgelesen werden. Dazu kommt noch die Restriktion, dass auf die Daten einer App nicht zugegriffen werden kann, sofern keine Root-Rechte verfügbar sind.

Eine andere Frage die sich stellt ist, wie aussagekräftig es ist, ob eine Antiviren-Software installiert ist oder eben nicht. Bisher fehlt allen Antiviren-Apps ein heuristisches Verfahren, um Schadsoftware zu erkennen. Neue Schädlinge werden nur durch bereits bekannte Signaturen erkannt.

Nachfolgend eine Grafik von av-test.org vom März 2012, die 41 verschiedene Virentester mit ihrem Erkennungsgrad von Malware auflistet:

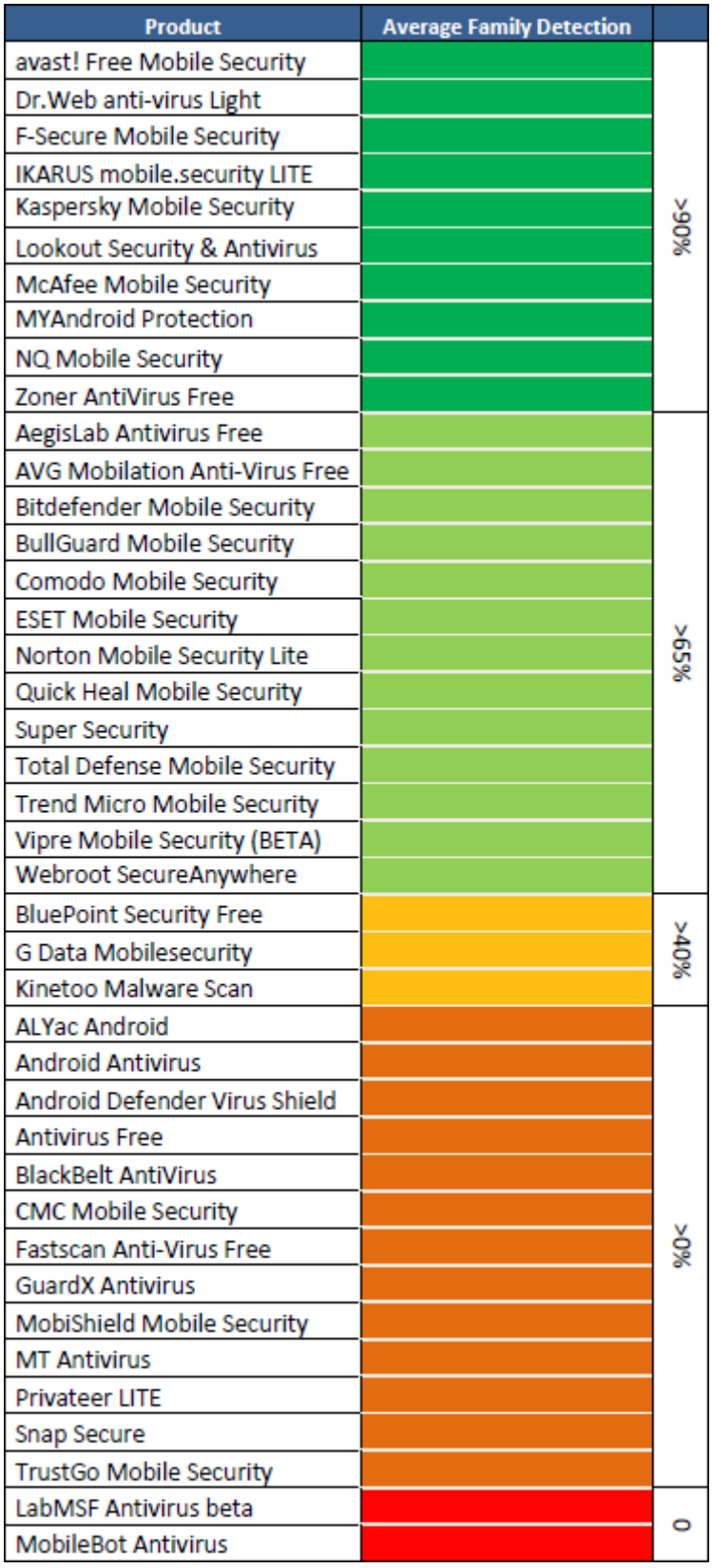

Abbildung 5: Virenscanner mit ihrem Erkennungsgrad für Android

Quelle: [5]

2.4.6.3 Bestätigung

Der Support von „avast!“ hat uns bestätigt, dass es nicht möglich ist, die Virendefinitionsversion als Fremdprogramm auszulesen. Wie angenommen ist es auch nur möglich, über das Package zu ermitteln, ob ein Virenscanner installiert ist (für avast! „com.avast.android.mobilesecurity“). Eine Möglichkeit wie in Windows, um die installierte Antivirus-Software zu ermitteln, ist in Android nicht implementiert.

2.4.6.4 Mögliche Alternative

Eine Möglichkeit wäre, dass man eine Datenbank führt, bei der man Antiviren-Hersteller eintragen könnte. Die Einträge werden verifiziert und permanent in der Datenbank gespeichert. Vor dem „Endpoint Assessment“ wird die Liste der Antiviren-Hersteller abgerufen und kann dazu verwendet werden um zu überprüfen, ob eine Antiviren-Software installiert ist.

3. Anforderungsspezifikationen

3.1 Allgemeines

3.1.1 Produktfunktion

Das Endprodukt soll die Messungen auf dem Client vornehmen. Der Benutzer merkt davon wenig, ausser im Falle eines Nichtbestehens eines Tests. Es können gewisse Fehler auftreten, welche nur als „Minor“ gewertet sind, dann wird der Client trotzdem in das Netzwerk eingelassen – natürlich abhängig von der Konfiguration auf dem Server – und es können grobe Verstösse gegen die Policies („Major“) auftreten, aufgrund derer dem Client sicherlich die Verbindung zum Netzwerk verweigert wird.

3.1.2 Abhängigkeiten

- Androidgerät muss mindestens Android 4.0 (Ice Cream Sandwich – ICS) vorweisen können, da die VPN API erst mit dieser Version verfügbar ist

3.2 UseCases

3.2.1 Use Case Diagramm

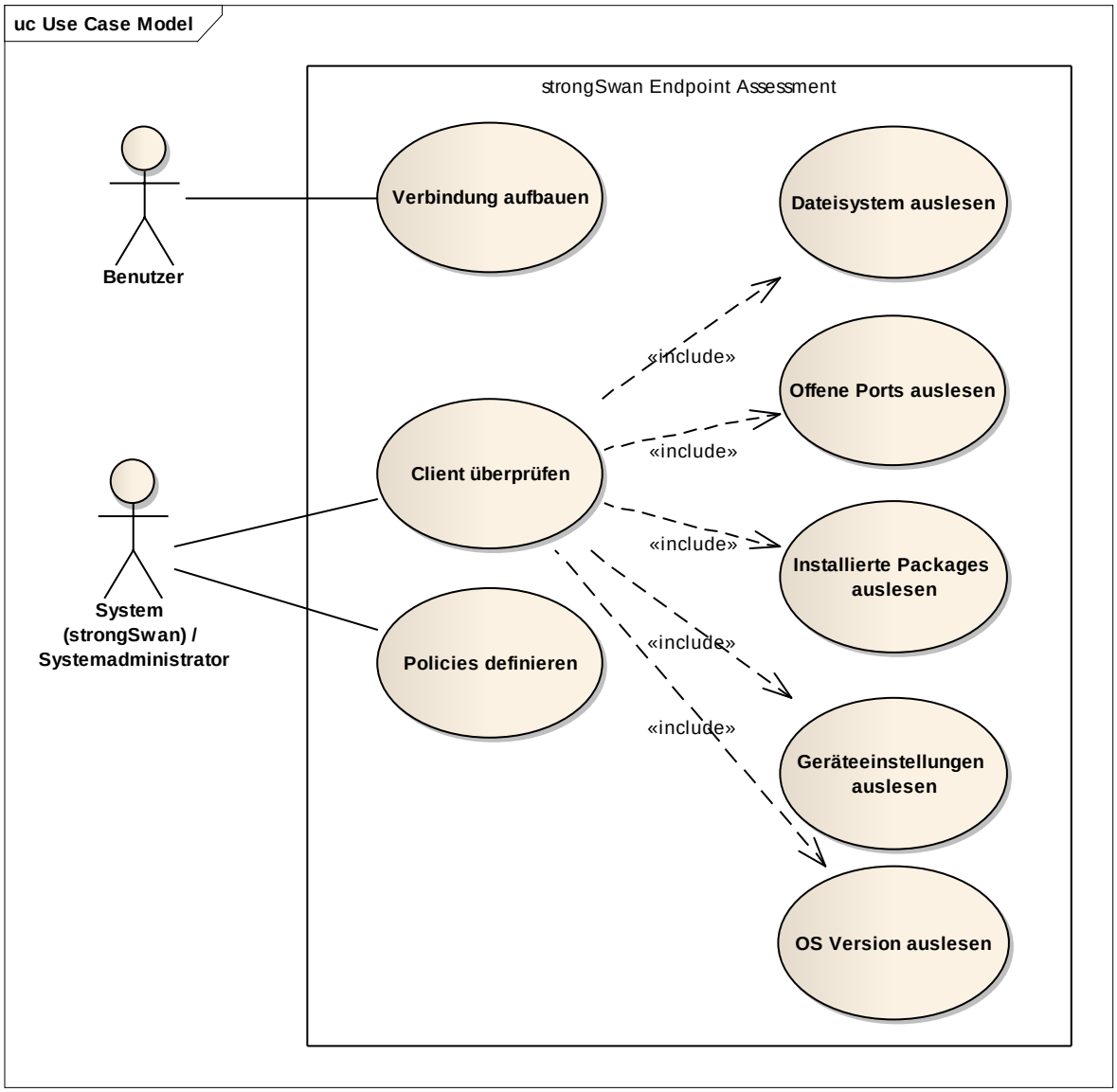

Abbildung 6: Use Case Diagramm strongSwan with Endpoint Assessment

3.2.2 Actors & Stakeholders

3.2.2.1 System (-administrator)

Der Administrator beziehungsweise das System stellt in gewissem Sinne die Interessen des Serveranbieters dar. Der Administrator will keine verseuchten Geräte in seinem Netzwerk und das System selbst möchte den Client überprüfen und die Messungen auswerten. Es ist eine Symbiose der Interessen zweier Actors.

3.2.2.2 Benutzer

Der Benutzer will einzig und allein das Netzwerk betreten. Gelingt dies nicht, so will er wissen, was zu tun ist um Einlass zu erhalten und was das Problem war.

3.2.3 Beschreibung

3.2.3.1 Verbindung aufbauen

Der Benutzer will sich mittels der strongSwan Applikation in ein Netzwerk verbinden. Damit dies geschehen kann, müssen auf der Serverseite die Authentifizierung vorhanden (Benutzername / Passwort; Radius-Server; Zertifikat; etc.) und die eventuellen Policies definiert sein. Falls die Verbindung fehlschlägt, soll dem Benutzer eine GUI-Maske angezeigt werden, auf welcher Hilfestellungen zu finden sind, wie das Problem zu lösen sei. Ist alles in Ordnung, so erhält der Benutzer keine Meldung.

3.2.3.2 Client überprüfen

Das strongSwan System beziehungsweise der Server will mittels TNC-Protokoll den Client überprüfen. Dies kann in zwei Varianten geschehen:

a) Der Client sendet beim „Connection Request“ bereits gewisse Messresultate mit, welche dann vom Server verifiziert werden und danach entweder ein Assessment Result mit „OK“ sendet oder weitere Messungen verlangt. Diese Option benötigt eventuell nur einen Round-Trip.
b) Der Client meldet sich mittels „Connection Request“ und erhält dann vom Server die erforderlichen Messwerte, welche er abzuliefern hat. Der Client sendet diese Messwerte danach zum Server, welcher sie verifiziert und gegebenenfalls weitere Messungen verlangt oder den Client einlässt. Diese Option benötigt mindestens zwei Round-Trips.

3.2.3.2.1 Dateisystem auslesen

Das Dateisystem wird nach den geforderten Dateien durchsucht. Aus jeder dieser Datei wird ein SHA1-Hash erstellt und zurückgegeben.

3.2.3.2.2 Offene Ports auslesen

Alle offenen Ports werden auf dem Smartphone ausgelesen. Ports die auf „localhost“ zeigen, werden entfernt. Die offenen Ports zu einer externen IP-Adresse werden zurückgeliefert.

3.2.3.2.3 Installierte Packages auslesen

Die installierten Packages auf dem Smartphone werden ermittelt, zum Beispiel „com.avast.android.mobilesecurity“ und zurückgegeben.

3.2.3.2.4 Geräteeinstellungen auslesen

Die vom Server geforderten, zu überprüfenden Einstellungen auf dem Smartphone, wie zum Beispiel ob der Download von unsicheren Quellen zugelassen ist, werden ermittelt und zurückgeliefert.

3.2.3.3 Policies definieren

Der Systemadministrator des Netzwerks will bestimmte Policies definieren, welche ein Client erfüllen muss, bevor er in das Netzwerk eingelassen wird. Diese Policies können bestimmte Attribute des Androidphones umfassen wie:

- Version des Betriebssystems
- bestimmte Apps, welche installiert sind (Blacklist / Whitelist)
- bestimmte Ports, welche von Schadsoftware oder nicht erwünschter Software geöffnet wurden

Es können noch weitere Kriterien hinzugefügt werden, für welche dann bei Bedarf ein eigener IMC definiert und implementiert werden muss.

4. Analyse und Design

4.1 Risikoanalyse

4.1.1 Risiken

Nr	Titel	Beschreibung	max. Schaden [h]	Eintrittswahrscheinlichkeit	Gewichteter Schaden	Vorbeugung	Verhalten beim Eintreten
R1	Benötigte Berechtigungen	Es ist nicht bekannt, ob für gewisse Systemzugriffe, wie z.B. die geöffneten Ports auslesen, Root-Rechte benötigt werden.	50	40%	20	Vorabklärungen treffen, ob alle gewünschten Funktionalitäten im Benutzermodus verfügbar sind.	Umgehungsmöglichkeiten studieren oder auf eine Funktionalität verzichten
R2	C Programmierung	Beide Teammitglieder haben sehr wenig Erfahrung in der C Programmierung	60	80%	48	Frühe Einarbeitung in C und den Source Code von strongSwan	Betreuer zu Hilfe ziehen
R3	Übertragungs-geschwindigkeit	Die Übertragung der ermittelten Daten an den Server dauert zu lange für den User	15	30%	5	Nur die nötigsten Daten übermitteln, Komprimierung, Benutzer informieren (Progress Bar)	Gewisse Daten schon beim ersten Round Trip mitschicken
R4	IMC Geschwindigkeit	Das Erheben der Daten dauert zu lange	20	40%	8	Nur die nötigsten Daten erheben	Algorithmen optimieren, Benutzer informieren (Progress Bar)
Summe			**145**		**81**		

Tabelle 1: Risikoanalyse des Projektes

4.1.2 Neubeurteilung

Elaboration 1		
Durchführen bis:	03.10.2012	
Beseitigte Risiken:	R1	Benötigte Berechtigungen
Neue Risiken:	Keine	

Construction 2		
Durchführen bis:	13.11.2012	
Beseitigte Risiken:	R3	Übertragungs-Geschwindigkeit
	R4	IMC Geschwindigkeit
Neue Risiken:	Keine	

Construction 4		
Durchführen bis:	04.12.2012	
Beseitigte Risiken:	R2	C Programmierung
Neue Risiken:	Keine	

4.2 RFC Attribute ↔ Messdaten

4.2.1 Beschreibung

Damit die vom Client gemessenen Daten an den Server übertragen werden können, müssen diese erst in ein für den Server verständliches Format konvertiert werden. Dazu existiert ein Standard (RFC 5792), welcher einige Standardattribute definiert. Im Folgenden soll beschrieben werden, welche Messdaten in welche äquivalenten Standardattribute übersetzt werden und für welche Zusatzinformationen neue Attribute im HSR-Namespace definiert werden müssen.

Zur Kommunikation zwischen Client und Server ist die TNC Protokoll-Suite vorgesehen.

4.2.2 Standardattribute

Die Standardattribute wurden der Webseite für den betroffenen RFC – Standard entnommen [1][5].

Attribut	Bedeutung
Attribute Request	Erlaubt dem Server, gewisse Attribute vom Client anzufordern. Der Client weiss dann, welche Messungen er machen muss.
Product Information	Enthält Informationen über das Produkt selbst. Kann „Subtypes“ enthalten (z.B. Informationen, ob es sich um eine Anti-Virus Software handelt). Im Falle von strongSwan würde es sich anbieten, hier das Android Flavor (z.B. Google, Samsung, HTC, etc.) abzufüllen.
Numeric Version	Ermöglicht dem Client, genauere Versionsangaben zu machen. Enthalten sind Majornumber, Minornumber und Buildnumber. Zusätzliche Informationen sind allfällige Servicepack Major-, und Minornumber.
String Version	Dieses PA-TNC Attribut enthält String Version Informationen des Endgerätes. Dazu gehört die Product Version Number (z.B. 4.1.1), die Internal Build Number (z.B. 3.13.163.3) und die Configuration Version Number.
Operational Status	Enthält Informationen zum aktuellen Status des Geräts oder der Software.
Port Filter	Eigentlich enthält dieses Attribut eine Liste der aktuell erlaubten oder blockierten Ports. Für das Endpoint Assessment könnte hier eine Liste der Ports eingefügt werden, welche aktuell offen sind auf dem Gerät.
Installed Packages	Erlaubt dem Client, seine installierten Applikationen aufzulisten. Für den Androidclient ergibt sich so die Möglichkeit, die installieren Apps zu enumerieren und zu übertragen. Anhand dieser Liste kann dann eine nicht erlaubte Software eruiert werden.
PA-TNC Error	Falls ein Error vorliegt, wird der Errorcode hier abgelegt. Dies kann ein Formatierungsfehler des Protokolls sein oder Kommunikationsprobleme.
Assessment Result	Die Antwort des Servers mit der Entscheidung, ob der Client ins Netzwerk eingelassen wird oder nicht. Die Werte sind: 0: Compliant 1: Non-Compliant with minor difference 2: Non-Compliant with major difference 3: Error
Remediation Instructions	Falls dem Client der Zugang verwehrt wird, können hier Instruktionen angegeben werden, wie der Benutzer sein Gerät bereinigen kann, damit er Zugang erhält.
Forwarding Enabled	Gibt an, ob der Client den Traffic weiterleitet, eine Weiterleitung kann zum Beispiel ein „Hotspot“ des Androidgeräts sein. Ist dies so, wird dies generell als „non-compliant“ angesehen.
Factory Default Password Enabled	Ermöglicht es dem Client anzugeben, ob ein Standardpasswort für das Gerät verwendet werden kann. Ist dies der Fall wäre das eine SIcherheitslücke und deswegen non-compliant.

Tabelle 2: Standardattribute RFC 5792 [1]

[5] http://tools.ietf.org/html/rfc5792

4.2.3 Messresultate

4.2.3.1 Dateisystem

Grundsätzlich kann hier vieles ausgelesen werden. Aufgrund der „canRead()“ und „canWrite()“ Funktionen können die Dateiberechtigungen ermittelt werden. Nachfolgend sind sinnvolle Attribute aufgelistet, welche übernommen werden könnten:

- Name der Datei
- Position im Dateisystem
- SHA1 Hashwert

4.2.3.2 Geöffnete Ports

Die Ports können mittels „netstat“ ausgelesen werden. Da auf dem Androidgerät keine vollwertige Version des netstat-Programms vorhanden ist, kann nur eine einfache Abfrage gestartet werden. Diese liefert die verwendeten Ports mit den dazugehörigen Adressen und der Information, ob die Ports im „listening“-Status oder ob sie bereits „established“ sind. Die Kerninformationen, welche übertragen werden sollten, sind somit:

- Protokoll (TCP/UDP)
- Portnummer

4.2.3.3 Installierte Applikationen

Hier können viele Attribute der „ApplicationInfo“ und der noch generischeren „PackageInfo“ ausgelesen werden. Im Hinblick auf die möglichen Standardattribute sollten folgende Informationen verwendet werden:

- Eindeutiger Name des Packages („com.example.testapp“)
- Wenn möglich: Google-ID des Packages / der Applikation (int)
- Version der App (versionString)

4.2.3.4 Geräteeinstellungen

Wie bereits in den Vorabklärungen erwähnt, können alle Einstellungen ausgelesen werden, welche über den SecurityProvider von Android verfügbar sind. Wichtig und relevant für die Sicherheit des Gerätes und des Netzwerks ist:

- Ist die Installation von Applikationen von unsicheren Quellen erlaubt

4.2.3.5 Produktinformationen

Die Attribute „Product Information“ und „String Version“ können mittels einfachen Befehlen ausgelesen werden. Diese Informationen sind über eine statische Klasse im Androidsystem verfügbar. Wichtig für die Attribute sind folgende Informationen:

- Produktname (statisch): „Android“
- Verkäufer-ID (statisch): „11129“
- Version des Betriebssystems

4.2.4 Abbildung der Messungen auf TNC Paket

Im Folgenden sind die verschiedenen Attribute mit einer Abbildung auf die Javaklassen beziehungsweise die Informationen aus dem Javateil dargestellt. **Alle Attribute haben eine Länge (Breite der einzelnen Zeilen) von 4 Byte.**

4.2.4.1 Dateisystem

Um das Dateisystem zu scannen, wird im Folgenden die Struktur der Anfrage und Antwort definiert. Die Angaben zu den Paketen wurden von der Trusted Computing Group (TCG) übernommen.

4.2.4.1.1 File measurement request

Request filemeasurement

Flags	Reserved	Request ID
Delimiter		
Fully Qualified File Pathname (Variable Length)		

Abbildung 7: Attribut „Request Filemeasurement“ (Anfrage des Servers)

Dieses Paket wird vom Server an den Client gesendet und enthält den vollständigen Pfad (fully qualified file pathname) zur zu scannenden Datei. Ist im „Flags“ – Feld das Bit an der Stelle 0 gesetzt, so handelt es sich um ein Verzeichnis und es muss das ganze Verzeichnis (Tiefe 1) gescannt werden. Dieses Attribut kann direkt als Original übernommen werden und bedarf keiner Anpassung. Die Request ID dient dem Wiedererkennen des Requests beim Auswerten der Antwort des Clients. [6]

4.2.4.1.2 File Measurement

File Measurement

Number of Files Included	
Number of Files Included	
Request ID	Measurement Length
Measurement (Variable Length)	
Filename Length	Filename (Variable Length)
Filename (Variable Length)	
Measurement #2 (Variable Length)	
Filename Length #2	Filename #2 (Variable Length)
...	

Abbildung 8: Attribut „File Measurement" (Antwort des Clients)

Die Antwort des Clients auf die „file measurement request“ – Anfrage des Servers ist mit obigem Attribut zu realisieren. Dies ist ein Standardattribut der TCG und kann ebenfalls ohne Ergänzungen übernommen werden. Die Attribute sind im Folgenden kurz erklärt:

Attribut	Bedeutung
Number of files included	Die Anzahl der gescannten Dateien
Request ID	ID des Requests des Servers (dient der Wiedererkennung)
Measurement length	Länge der Messwerte (alle Messwerte sind gleich lang, da nur ein Algorithmus für eine Messung verwendet wird)
Measurement	Messung (z.B. SHA1-Hashwert)
Filename length	Länge des Dateinamens
Filename	Relativer Dateiname

Tabelle 3: Beschreibung der Attribute des „file measurement" Attributes

4.2.4.1.3 Abbildung

Attribut TNC	↔	Attribut JAVA
Filename	↔	**Relativer Dateiname**
Filename length	↔	**Länge des (relativen) Dateinamens**
Measurement	↔	**SHA1-Hash der Datei**
Measurement length	↔	**Länge des SHA1-Hashes**
Number of files included	↔	**Anzahl der gescannten Dateien**

Tabelle 4: Abbildungstabelle des „file measurement“ Attributes

4.2.4.2 Offene Ports

Der Server sendet die Attribute, welche er erwartet, mit dem „measurement request“. Ist der „port filter“ aufgelistet, so müssen die offenen Ports zurückgesendet werden.

4.2.4.2.1 Port Filter

Port Filter

Reserved |B|	Protocol	Port Number
Reserved |B|	Protocol	Port Number
...		

Abbildung 9: Attribut „Port Filter“ (Antwort des Clients)

Die Antwort des Clients beinhaltet alle offenen Ports der Protokolle. Wobei dieses Attribut ursprünglich für die Abfrage einer Firewall gedacht war, weswegen auch das „|B|“ (für Blocked Port) noch im Attribut definiert ist. Im Falle von strongSwan wird das „Port Filter“ – Attribut genutzt, um die offenen Ports des mobilen Geräts zu übertragen.

Attribut	Bedeutung
Reserved |B|	Reserviertes Byte, welches auf 0 gesetzt werden muss. |B| ist für die Firewallabfrage, falls die „blocked ports“ gesucht werden.
Protocol	Angabe des Protokolls (UDP / TCP)
Port Number	Die Nummer des offenen Ports

Tabelle 5: Beschreibung der Attribute des „Port Filter“ Attributes

4.2.4.2.2 Abbildung

Attribut TNC	↔	Attribut JAVA
Reserved |B|	↔	**Wird nicht benötigt**
Protocol	↔	**Protokoll (UDP / TCP)**
Port Number	↔	**Portnummer**

Tabelle 6: Abbildungstabelle des „Port Filter“ Attributes

4.2.4.3 Installierte Applikationen

Ist das „packages“ Attribut vom Server gefordert, so muss der Client die installierten Applikationen auflisten und zurücksenden. Dies geschieht, um eventuelle Schadsoftware zu blockieren.

4.2.4.3.1 Installed Packages

Installed Packages

Reserved	Package Count
Pkg Name Len	Package Name (Variable Length)
Version Len	Package Version Number (Variable Length)
...	

Abbildung 10: Attribut „Installed Packages“ (Antwort des Clients)

Die installierten Applikationen werden in diesem Attribut zurückgesendet. Da die Anzahl „Packages“ stark variieren kann und es auch zu Extremwerten kommen kann, werden für diese Messung zwei bestimmte Varianten diskutiert:

a) Der Server sendet eine Liste mit „unerlaubten“ Applikationen beziehungsweise Paketen, welche dann vom Client mit der Liste der installierten Apps abgeglichen wird. Der Client sendet dem Server eine Liste mit „matches“ zu, also mit Paketen, welche unerlaubt sind und auf dem Gerät installiert sind.
b) Der Client sendet alle installierten Pakete zum Server.

Attribut	Bedeutung
Package Count	Anzahl der übermittelten Pakete
Pkg Name Len	Länge des „Package Name“ Feldes
Package Name	Der Name des installierten Pakets (z.B. org.strongswan.android)
Version Len	Länge des „Package Version“ Feldes
Package Version Number	In diesem Falle gibt es zwei Alternativen: - String-Version des installierten Pakets (z.B. „1.5“) - Eindeutige inkrementierte Integer-Version (z.B. 15) Die Stringversion kann vom Programmierer frei wählbar gesetzt werden, die inkrementierte Integerversion muss höher sein als die letzte Versionsnummer, ansonsten kann die App nicht im Google-Play-Store publiziert werden.

Tabelle 7: Beschreibung der Attribute des „Installed Packages“ Attributes

4.2.4.3.2 Abbildung

Attribut TNC	↔	Attribut JAVA
Package Count	↔	**Anzahl der Pakete**
Pkg Name Len	↔	**Länge des Namens**
Package Name	↔	**Name des Pakets (packageName)**
Version Len	↔	**Länge der Versionsnummer**
Package Version	↔	**Version des Pakets (versionCode)**

Tabelle 8: Abbildungstabelle des „Installed Packages" Attributes

4.2.4.4 Geräteeinstellungen

Für die Geräteeinstellungen existiert aktuell kein Standardattribut. Nachfolgend ist ein Designvorschlag dargestellt, welcher den Anforderungen genügen sollte. Das Attribut ist so geplant, dass auch mehrere Einstellungswerte übertragen werden können. Der Name ist variabel, ein möglicher Name in diesem Kontext wäre „Android Settings".

4.2.4.4.1 Request Settings

Request Settings

Settings Count	
Set Name Len	Settings Name (Variable Length)
...	

Abbildung 11: Attribut „Request Settings" (Anfrage des Servers)

Der Server sendet hier dem Client die Anfrage, in welcher die verlangten Settings aufgelistet sind. Dies geschieht, damit der Client weiss, welche Einstellungen der Client abfragen und zurücksenden muss.

Attribut	Bedeutung
Settings Count	Anzahl der angefragten Settings
Set Name Len	Länge des Namens der Einstellung
Settings Name	Der Name der Einstellung (basierend auf den Android Einstellungen und der Enumeration derselben)

Tabelle 9: Beschreibung der Attribute des „Request Settings" Attributes

4.2.4.4.2 Device Settings

Device Settings

Settings Count	
Set Name Len	Settings Name (Variable Length)
Value Len	Settings Value (Variable Length)
...	

Abbildung 12: Attribut „Device Settings" (Antwort des Clients)

Attribut	Bedeutung
Settings Count	Anzahl der übermittelten Einstellungen
Set Name Len	Länge des „Settings Name“ Feldes
Settings Name	Der Name des installierten Pakets (z.B. org.strongswan.android)
Value Len	Länge des „Settings Value“ Feldes
Settings Value	Der Wert der Einstellung (Ein / Aus; String-Value; Mehrere String-Values; Numerischer Wert)

Tabelle 10: Beschreibung der Attribute des „Device Settings“ Attributes

4.2.4.4.3 Abbildung

Attribut TNC	↔	Attribut JAVA
Package Count	↔	**Anzahl der Pakete**
Pkg Name Len	↔	**Länge des Namens**
Package Name	↔	**Name des Pakets (packageName)**
Version Len	↔	**Länge der Versionsnummer**
Package Version	↔	**Version des Pakets (versionCode)**

Tabelle 11: Abbildungstabelle des „Device Settings" Attributes

4.2.4.5 Produktinformationen

Für die Produktinformationen (im Allgemeinen) gibt es zwei verschiedene Attribute, welche verwendet werden müssen. Zum einen ist dies das Attribut „Product Information“, welches die Verkäufer ID und den Produktnamen enthält. Diese beiden Werte sind statisch und werden so zurückgegeben. Das zweite Attribut ist „String Version“ und enthält die Versionsnummer des Android Betriebssystems.

4.2.4.5.1 Product Information

Product Information

Product Vendor ID	Product ID
Product ID	Product Name (Variable Length)

Abbildung 13: Attribut "Product Information" (Antwort des Clients)

Attribut	Bedeutung
Product Vendor ID	Private Enterprise Number (PEN) des Herstellers. Falls für den Hersteller keine PEN existiert, so muss 0 eingetragen werden.
Product ID	Ein numerischer Wert mit der Produktnummer des Herstellers. Falls keine Produktnummer eruiert werden kann, muss 0 eingetragen werden.
Product Name	Name des Produktes. Im Falle unserer Arbeit wird hier statisch „Android“ eingetragen, da es sich nur um dieses Produkt handelt.

Tabelle 12: Beschreibung der Attribute des "Product Information" Attributes

4.2.4.5.2 String Version

String Version

Version Len	Product Version Number (Variable Length)
Build Num Len	Internal Build Number (Variable Length)
Config. Len	Configuration Version Number (Variable Length)

Abbildung 14: Attribut "String Version" (Antwort des Clients)

Attribut	Bedeutung
Version Len	Länge des „Product Version Number“ Feldes
Product Version Number	Versionsnummer des Produkts (Beispielsweise: 4.1.2)
Build Num Len	Länge des „Internal Build Number“ Feldes
Internal Build Number	Interne Build Nummer des Geräts (Beispielsweise: 3.13.163.3)
Config. Len	Länge des „Configuration Version Number“ Feldes
Configuration Version Number	Eine Konfigurationsnummer, welche als allgemeingültige Version gilt. Im Falle von Android gibt es diese nicht. Somit wird dieses Feld nicht befüllt.

Tabelle 13: Beschreibung der Attribute des "String Version" Attributes

4.2.4.5.3 Abbildung Product Information

Attribut TNC	↔	Attribut JAVA
Product Vendor ID	↔	**11129**
Product ID	↔	**0**
Product Name	↔	**„Android“**

Tabelle 14: Abbildungstabelle des "Product Information" Attributes

4.2.4.5.4 Abbildung String Version

Attribut TNC	↔	Attribut JAVA
Version Len	↔	**Länge des Versionsnamens**
Product Version Number	↔	**Versionsnummer des Android Betriebssystems**
Build Num Len	↔	**Länge der Buildnummer**
Internal Build Number	↔	**Buildnummer des Android Betriebssystems (falls auslesbar)**
Config. Len	↔	**0**
Configuration Version Number	↔	**„“**

Tabelle 15: Abbildungstabelle des "String Version" Attributes

4.3 Wireframes Benutzerfeedback

Remediation Instructions | Detaillierte Anweisung | Einstellungen

Sobald ein Assessment Result „non-compliant minor“ oder „non-compliant major“ ist, sind Remediation Instructions vorhanden. Für jedes Measurement, dessen Resultat für den IMV nicht zufriedenstellend ist, gibt es genau eine Remediation Instruction, welche in der Übersicht mit einer Beschreibung dargestellt wird. Durch einen Klick auf die jeweilige Remediation Instruction wird eine neue Activity gestartet, auf der weitere Details in Form einer Liste dargestellt werden.

4.4 Definition der Schnittstelle Java ↔ C

Damit der IMC die Daten abholen kann, benötigt es eine definierte Schnittstelle. Diese wird mithilfe des Java Native Interface (JNI) realisiert. Eine Dokumentation zu JNI ist von Matthew Mead verfügbar [7][6]. Die Schnittstelle wurde so designed, dass möglichst viel im Javacode erledigt werden kann. Der Programmieraufwand in C soll so klein wie möglich gehalten werden, da die Verwaltung und Umsetzung in der Programmiersprache C mit dem JNI ein sehr komplexes Unterfangen ist. Die Implementation in Java ermöglicht es uns, mit den Techniken einer objektorientierten Sprache eine möglichst generische Lösung zu produzieren.

Die native Implementation soll für die Verwaltung und das Handling der Nachrichten vom IMV zuständig sein, und diese Messanfragen an die Java-Implementation weiterleiten.

[6] http://home.pacifier.com/~mmead/jni/cs510ajp/index.html

4.4.1 Kommunikation Java ↔ C

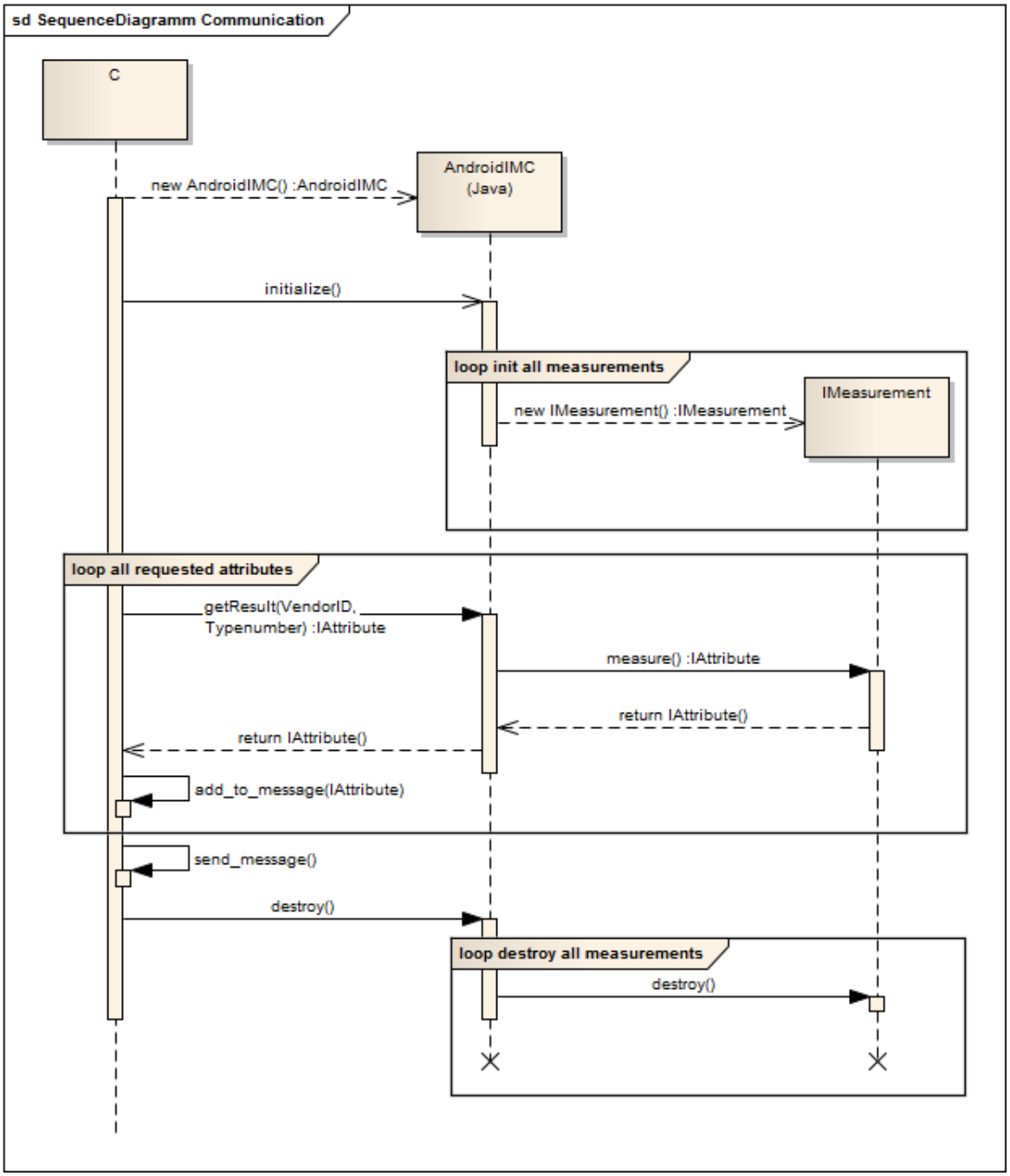

Abbildung 15: Sequenzdiagramm der Kommunikation C ↔ java

Das obige Bild stellt den normalen Kommunikationsablauf zwischen dem Client und dem Server dar. Will der Client eine Verbindung aufbauen, so wird lokal der IMC initialisiert. Dieser native IMC initialisiert seinerseits den IMC, welcher in Java implementiert ist. Sobald der IMV auf der Serverseite einen Attributrequest absendet, wird dieser vom nativen C empfangen und mittels „VendorID“ und „Typenumber“ an die Java-Schnittstelle weitergeleitet. Die Java-Implementation verarbeitet den Request und führt die Messung aus. Sobald die Messung fertiggestellt wurde, wird das komplette Attribut an den nativen Teil ausgeliefert. Der native Code holt sich nun das Encoding (ohne Header) des Attributes und fügt dieses der Message hinzu. Zum Schluss wird die Message an den IMV versendet und sobald der Verbindungsaufbau – erfolgreich oder nicht – beendet ist, werden die einzelnen Elemente abgeräumt.

4.5 State-Diagramm Assessmentstatus

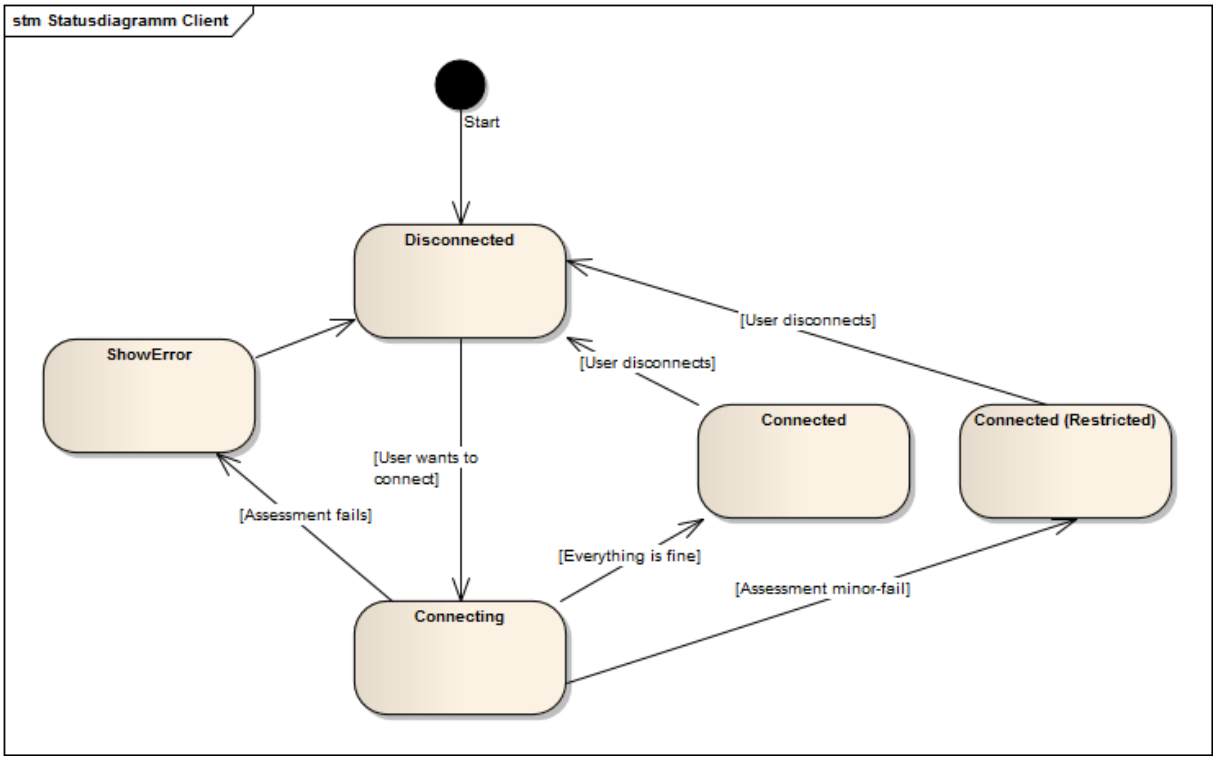

Abbildung 16: Statusdiagramm Client

In der obigen Abbildung wird der Status des Assessments dargestellt. Gestartet wird mit einer getrennten Verbindung. Sobald der Benutzer ein VPN Profil auswählt, geschieht ein Übergang in den Status „Connecting“. Dieser Status beinhaltet das Auslesen und Verpacken der Daten auf dem Client und das Zusenden derselben an den Server. Sobald vom Server eine Antwort (Assessment Result) geschickt wird, geht der „Connecting“ Zustand in den korrespondierenden Zustand über. Diese können sein:

- ShowError (falls Assessment fehlschlägt und major Fehler zurückgegeben wird)
- Connected (falls alles in Ordnung ist)
- Connected Restricted (falls ein minor Fehler vorliegt und der Client in eine Quarantäneverbindung geschoben wird)

5. Realisierung

5.1 Designmodell

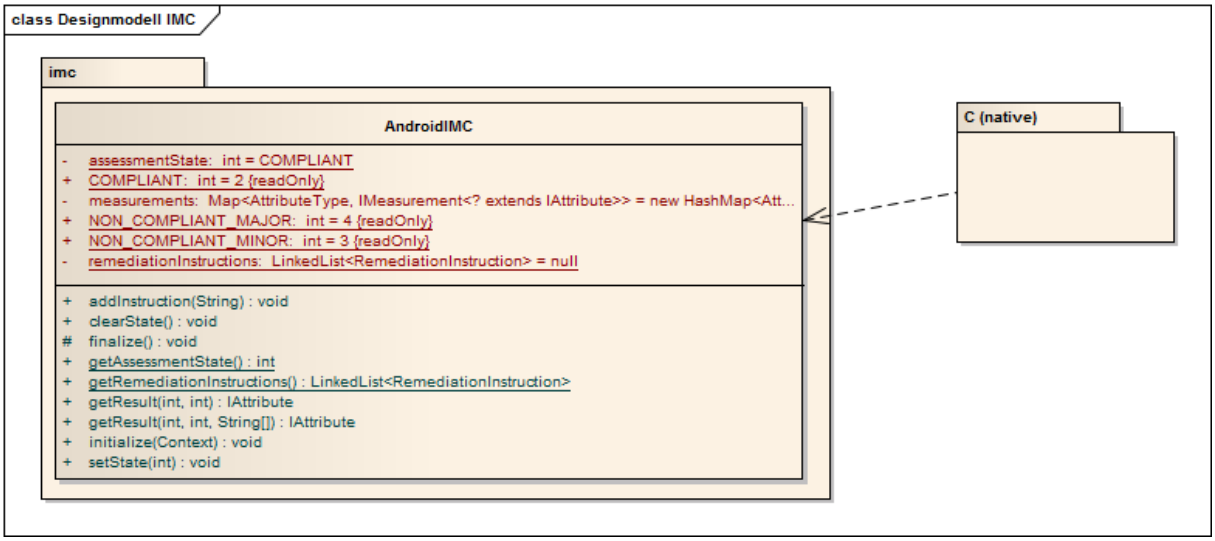

Abbildung 17: Designmodell IMC

Die Basis stellt die native C Implementation dar, welche für die Kommunikation zwischen Java und dem Server zuständig ist. Der native Code nimmt die Nachrichten des IMVs entgegen und sendet diese entsprechend an den AndroidIMC weiter. Dieser ist mit einigen Methoden ausgestattet. Für die Messung sind die Methoden „getResult(int, int)“ und „getResult(int, int, String[])“ zuständig. Diese führen auf dem Interface die Messung durch. Um den Status des Assessments anzuzeigen, werden die Methoden „getAssessmentState()“ und „getRemediationInstructions()“ verwendet. Liefern diese ein entsprechendes Ergebnis, so ist die Verbindung eingeschränkt oder fehlgeschlagen.

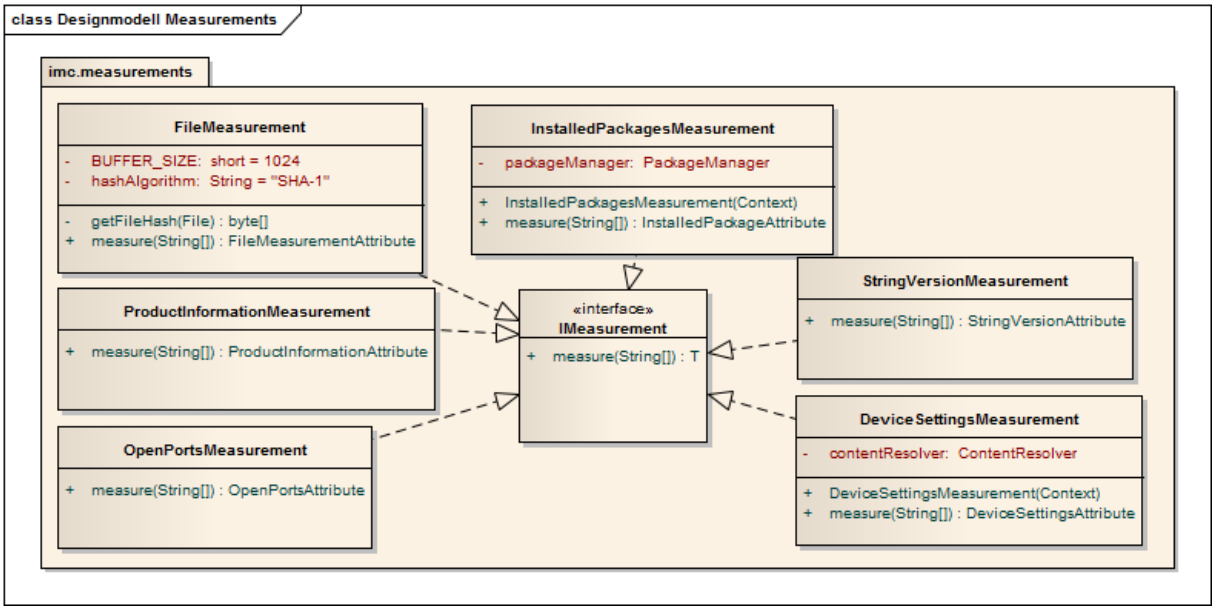

Abbildung 18: Designmodell Measurements

Die Measurements sind mittels eines Interfaces aufgebaut. Das Interface implementiert die „measure(String[])“ Methode, welche die effektive Messung durchführt und danach ein Attribut zurückliefert. Dieses Attribut wird mittels des generischen Typenparameters „T“ definiert. Die einzelnen Messungsklassen haben eventuelle spezifische Variablen, wie beispielsweise den definierten Hash-Algorithmus „SHA-1“.

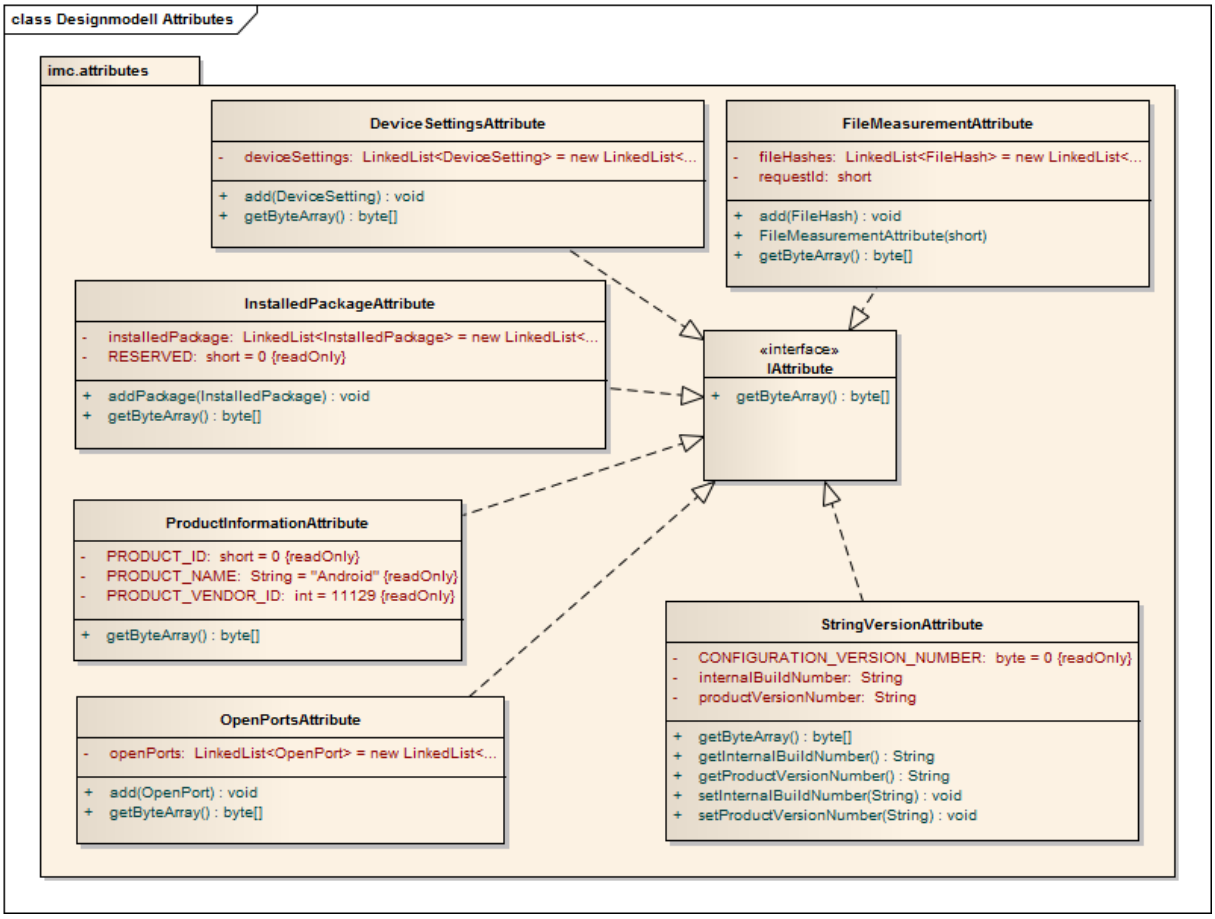

Abbildung 19: Designmodell Attributes

Die Attribute wurden mittels eines Interfaces „IAttribute“ implementiert. Dieses Interface enthält lediglich die „getByteArray()“ Methode, welche das vollständig encodierte Attribut zurückliefert. Der Header der Nachricht muss vom nativen Code erstellt werden, die Attribute liefern nur den „Chunk“, welcher die attributspezifischen Daten enthält. Die einzelnen, realisierten Attribute enthalten eventuelle, lokale Variablen welche für das spezifische Attribut verwendet werden (beispielsweise beim Attribut „ProductInformationAttribute“ ist eine lokale Konstante „PRODUCT_ID“ enthalten).

5.2 Benutzeranzeige der Remediation Instructions

Sobald die Applikation die Verbindung hergestellt hat, oder diese vom Server zurückgewiesen wurde, muss dem Benutzer angezeigt werden, welchen Status seine Verbindung hat. Zu diesem Zweck wurde ein weiteres sogenanntes „Fragment“ unterhalb des aktuellen Status-Fragments eingefügt. Dieses wird im Falle einer normalen, fehlerlosen Verbindung ausgeblendet. Ist jedoch eine Messung nicht zur Zufriedenheit des Servers, so wird ein Status gesetzt. Dieser kann „non-compliant minor“ oder „non-compliant major“ sein. Sollte die Verbindung „minor“-Status haben, so hat der Benutzer eine eingeschränkte Verbindung, welche beispielsweise nur den Zugriff auf das Internet zulässt, um Updates zu machen oder Ähnliches. Wenn die Verbindung „major“-Status hat, so wird die Verbindung vom Server beendet und die Massnahmen, welche zu treffen sind, werden angezeigt.

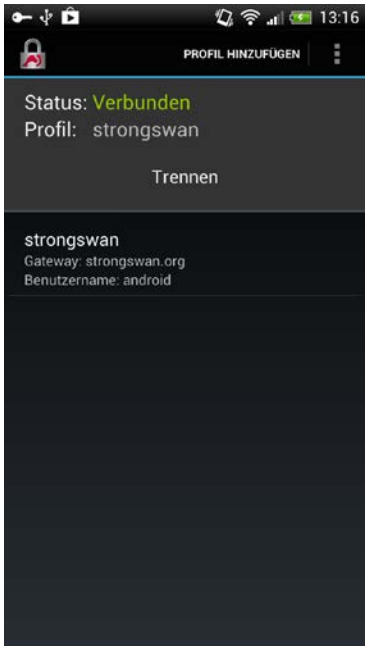

Abbildung 20: Compliant

Abbildung 21: Non-compliant minor

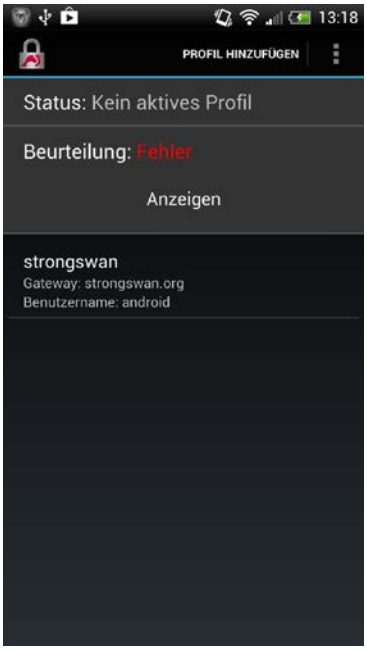

Abbildung 22: Non-compliant major

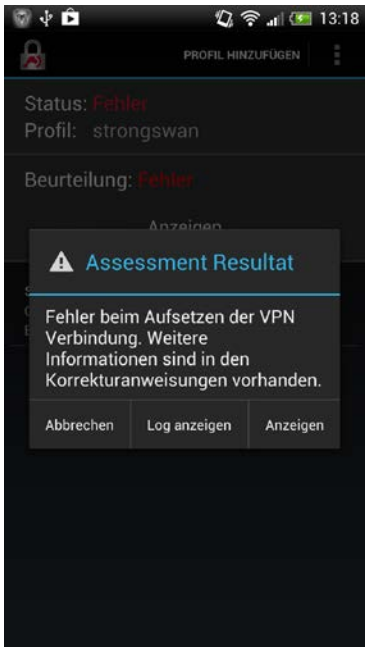

Abbildung 23: Non-compliant major Dialog

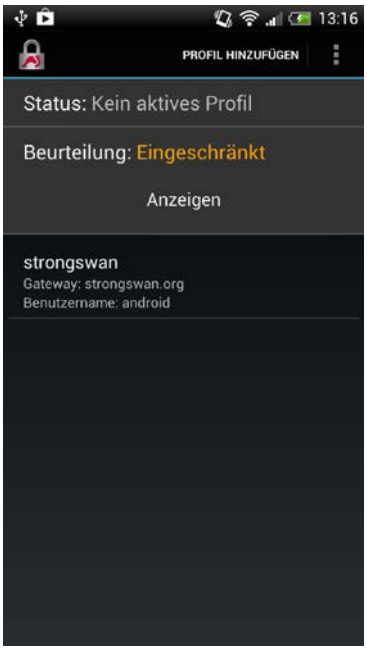

Abbildung 24: Non-compliant minor nachdem die Verbindung getrennt wurde

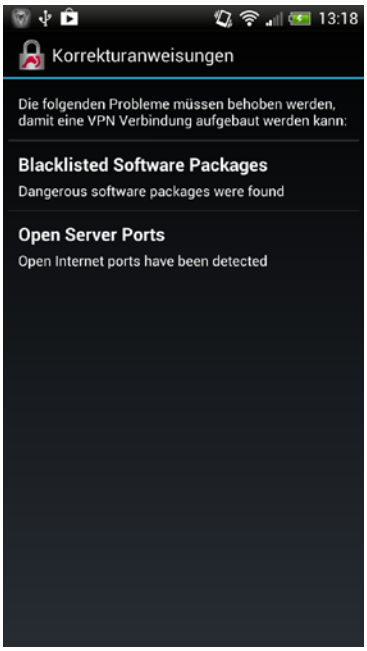

Abbildung 25: Remediation Instructions

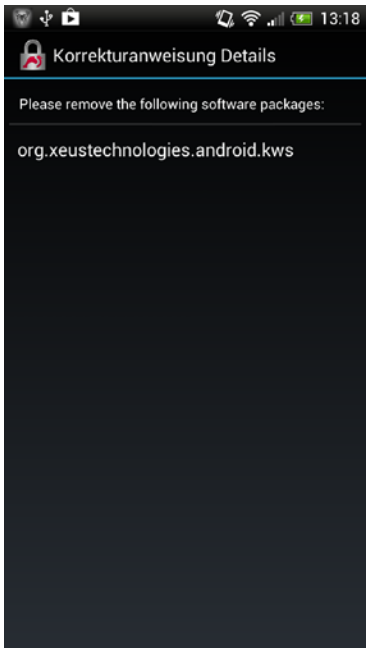

Abbildung 26: Remediation Instruction Package Detail

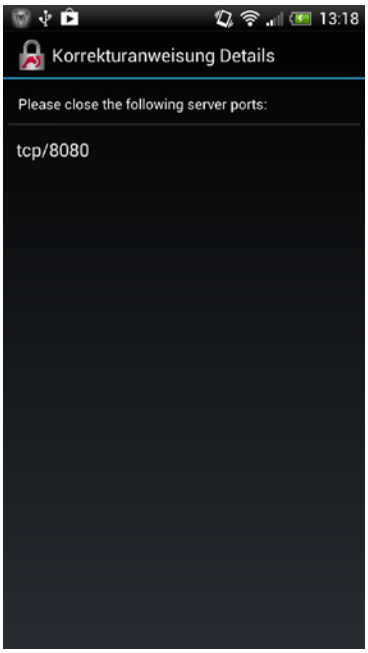

Abbildung 27: Remediation Instruction Port Detail

5.3 Designentscheide

5.3.1 Verhältnis Java Code zu C Code

Da beiden Projektmitgliedern die Kenntnisse der C-Programmierung nicht gegeben sind, wurde ein Java gestützter Programmansatz gewählt. Dies vermindert das Risiko, zuerst grosse Aufwände in das Erlernen der Programmiersprache C stecken zu müssen. Zusätzlich zur Programmiersprache müsste ebenfalls der Ansatz der objektorientierten Programmierung in C erlernt werden.

In C wurden nur die wichtigsten Elemente implementiert. Diese beinhalten:

- Management der empfangenen Nachrichten
- Weiterleiten an eine bestimmte „Handle“-Funktion
- Weiterleitung des Attributrequests an den Java Teil
- Erstellung des zu versendenden Pakets aus den Rohdaten des Java Teils
- Benachrichtigung des Java IMC am Ende des Assessments über den Ausgang der Bewertung

In Java wurden folgende Elemente realisiert:

- „AndroidIMC“: Allgemeiner Manager und Ansprechstelle für den nativen Code (C)
- die verschiedenen Datenkollektoren, welche die verschiedenen Messungen durchführen
- die verschiedenen Attribute, welche als Datenhalter dienen
- Übersetzung der Attribute in einen Byte-Array, welcher vom nativen Code übernommen wird und direkt in ein Paket verwandelt werden kann, ohne gross Übersetzungsarbeit zu leisten
- Anzeigen der Fehlermeldungen und Resultate für den Benutzer

5.3.2 Konzeptionelles Design Java Code

Der Javacode soll möglichst generisch sein. Es muss sichergestellt sein, dass ohne grosse Mühe neue Datenkollektoren hinzugefügt werden können. Damit dies möglich ist, wurden ein generisches Attribut-Interface und ein dazugehöriges „Measurement“-Interface definiert. Diese zwei Interfaces decken alle Funktionen ab, welche für die Messung und deren Auswertung notwendig sind. Dieses Design gestattet es, im Falle einer zusätzlichen Messung, diese zu erstellen und im AndroidIMC zu laden. Ist diese vorhanden, so kann die Messung vom nativen Code her aufgerufen werden.

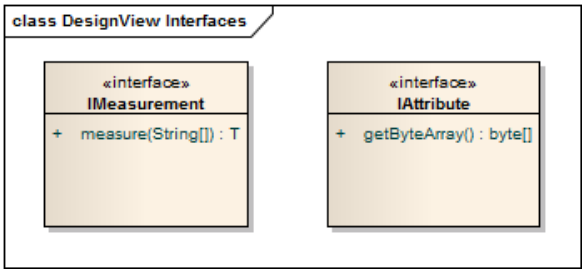

Abbildung 28: DesignView der Interfaces (Messung, Attribut)

Im nativen Code wird eine Messung folgendermassen aufgerufen:

```
method_id = (*env)->GetMethodID(env, android_imc_class,
                                "getResult",
                                "(II[Ljava/lang/String;)"
                                L"JNI_IMC_PATH"/attributes/IAttribute;");
meas_params = get_param_string_array(env, params);
iAttribute = (*env)->CallObjectMethod(env, android_imc_object,
                                      method_id,
                                      attr_type.vendor_id,
                                      _type.type,
                                      meas_params);
method_id = (*env)->GetMethodID(env, android_imc_attribute_class,
                                "getByteArray", "()[B");
byteArray = (*env)->CallObjectMethod(env, iAttribute, method_id);
```

Aus Gründen der Leserlichkeit wurden die Fehlerüberprüfungen hier in diesem Beispiel weggelassen. Im Wesentlichen wird die Methode „getResult(int,int,string[])“ auf dem AndroidIMC gesucht. Ist diese vorhanden, so werden die optionalen String-Argumente generiert. Diese werden in ein JNI-String-Array „meas_params“ geladen. Wurde auch hier kein Fehler verursacht, so wird mit dem Abholen des Resultats weitergefahren. Die Methode „getByteArray()“ liefert hier das gesamte Byte-Encoding, somit muss keine Übersetzung in C gemacht werden. Der Aufruf von „CallObjectMethod(…)“ startet die Methode auf dem AndroidIMC. Das erhaltene ByteArray wird nun mittels eines generischen C-Konstruktors in ein Attribut verwandelt und dieses Attribut wird dann an die Nachricht angehängt, welche dem Server zurückgesandt wird.

Variable / Funktion	Bedeutung
method_id	ID, mit welcher eine Java Methode über JNI aufgerufen werden kann
env	„Environment“: Stellt den angehängten Java-Thread dar
meas_params	Enthält die Parameter für die Messung oder NULL, falls keine Parameter vorhanden sind
iAttribute	Wird von der „getResult“ Methode befüllt. Der retournierte Wert der Methode enthält das entsprechend encodierte Attribut (ohne Messageheader)
byteArray	Der effektive „Chunk“ des Attributes. Dieser wird danach an den Konstruktor des Attributes weitergeleitet und so übernommen, es ist kein encoding durch C nötig

Tabelle 16: Wichtige Elemente der nativen Implementierung

Das Byte-Encoding in Java wird im Folgenden am Beispiel des „ProductInformationAttribute“ erklärt:

```
@Override
public byte[] getByteArray() {
	ByteBuffer vendorID = ByteBuffer.allocate(3);
	ByteBuffer productID = ByteBuffer.allocate(2);
	ByteBuffer productName = ByteBuffer.wrap(PRODUCT_NAME.getBytes());
	vendorID.put(ByteBuffer.allocate(4).putInt(PRODUCT_VENDOR_ID).array(),1,3);
	productID.putShort(PRODUCT_ID);
	ByteBuffer attribute = ByteBuffer.allocate(vendorID.capacity() +
                                                   productID.capacity() +
                                                   productName.capacity());
	attribute.put(vendorID.array())
		 .put(productID.array())
		 .put(productName.array());

	return attribute.array();
}
```

Erst werden die einzelnen Attributsegmente initialisiert und mit den entsprechenden Werten gefüllt. Im Falle eines Integers, welcher statt in 4 Byte in 3 Byte gespeichert werden muss, wird das höchste Byte (0x00) abgeschnitten. Zum Schluss wird der Attribut-Array mit den einzelnen Bytes gefüllt und danach als komplettes ByteArray dem nativen Code retourniert.

Variable / Funktion	Bedeutung
ByteBuffer	Javaklasse, welche die Möglichkeiten bietet, ein Byteencoding zu erstellen
.array()	Liefert den vollständigen Bytearray des Attributes

Tabelle 17: Wichtige Elemente des Java-Attributes

5.3.3 Remediation Instructions

Die „Remediation Instructions“ sind Massnahmen, welche zu treffen sind, wenn keine Verbindung aufgrund des Assessments zustande kommt. Damit diese dem Benutzer angezeigt werden können, muss in Android eine Activity definiert werden. Damit diese Daten gegliedert werden können, wurde eine XML-Formatierung verwendet. Die Formatierung ist anhand eines Beispiels definiert:

```
<?xml version="1.0" encoding="UTF-8"?>
<remediationinstructions>
	<instruction>
		<title>Open Ports</title>
		<description>There are open ports you have to close.</description>
		<itemsheader>The following ports have to be closed for a successfull
			connection:</itemsheader>
		<items>
			<item>8080</item>
			<item>8000</item>
			<item>9999</item>
		</items>
	</instruction>
</remediationinstructions>
```

Falls eines der Elemente „itemsheader“ oder „items“ leer bzw. nicht vorhanden ist, werden diese ausgelassen. Die restlichen Elemente sind jedoch Pflicht. In C wird dieser String nur gelesen und dem AndroidIMC weitergeleitet. Dieser parst das XML in ein Objekt und verwendet dieses dann für die weitere Darstellung.

5.4 Implementationsentscheide

5.4.1 „File-Measurement“ in Java statt in C

Während der Implementierung des File-Measurements stand die Frage offen, in welchem Programmteil das „Hashing“ des Dateisystems vorgenommen werden soll. Da Java eine hohe Anzahl von Überprüfungsabfragen an die virtuelle Java-Maschine sendet, könnte dies zu einem Performanceproblem führen. Grundsätzlich bestand die Möglichkeit, das Auslesen und hashen der Dateien im nativen Code durchzuführen, da jedoch alle anderen Messungen in Java durchgeführt wurden, musste hier ein Kompromiss gemacht werden. Falls der Faktor Zeit sich extrem unterscheiden würde zwischen der Implementation in C und in Java, so würde der schnellere genutzt werden. Nach einigen Messungen mit beiden Implementationen hat sich herausgestellt, dass auch bei grösseren Dateimengen (für Android jedenfalls grössere Dateimengen: Anzahl > 130 Dateien) maximal ein Unterschied von 0.5 Sekunden auftritt. Da dieser Unterschied sehr trivial ist, wurde die Java Implementation bevorzugt.

5.5 Verwendete Technologien

Im Folgenden sind die benutzten Technologien und Programme kurz aufgelistet.

	Name	Beschreibung
	Eclipse	(http://www.eclipse.org) OpenSource Java-Entwicklungsumgebung.
	GitHub	(http://www.github.com) Versionsverwaltung, basierend auf Git.
	Sublime Text	(https://www.sublimetext.com) Eleganter Editor, welcher zahlreiche Sprachen unterstützt. Genutzt für die C Programmierung.
	Android SDK	(http://developer.android.com/sdk/index.html) Software Development Kit für das Android Betriebssystem. (Java Programmierung)
	Android NDK	(http://developer.android.com/tools/sdk/ndk/index.html) Native Development Kit für das Android Betriebssystem. (C Programmierung)
	Skydrive	(http://skydrive.live.com) Dokumentenverwaltung von Microsoft. Genutzt für die Dokumentation und Versionsverwaltung der Dokumente.
	Enterprise Architect	(http://www.sparxsystems.com) Projektdesigner. Genutzt für die verschiedenen Diagramme wie „UseCase Diagram“ oder „Sequenzdiagramm“.

Tabelle 18: Verwendete Technologien

6. Tests

Da für das Android-Betriebssystem keine Unit-Tests vorhanden sind, kann nur „von Hand“ getestet werden. Ebenfalls ist es nicht möglich, Unit-Tests im nativen C Code zu starten.

Damit wir das korrekte Verhalten der Measurements garantieren können, wurden diese mehrfach geprüft. In nachfolgender Tabelle wird aufgezeigt, welches Measurement beim Auftreten eines Problems zu welchem Assessment Result geführt hat.

Measurement	Problem	Erwartetes Verhalten	Tatsächliche Verhalten	Erfolgreich?
Dateisystem*	-	-	-	-
Geöffnete Ports	Open Server Port: 8080	Non-compliant major	Non-compliant major	✓
Installierte Applikationen	Blacklisted Software Package: kWs – android web server	Non-compliant minor	Non-compliant minor	✓
Geräteeinstellungen	Installation von Apps aus unbekannten Quellen zugelassen	Non-compliant minor	Non-compliant minor	✓
Produktinformationen*	-	-	-	-

Tabelle 19: Erwartete Assessment Results

*) Für die Measurements Dateisystem und Produktinformationen sind keine Tests vorhanden, da auf dem Server momentan keine Referenzwerte in der Datenbank vorhanden sind.

Beim Measurement Dateisystem könnte das erwartete Verhalten ein „non-compliant major“ sein, sobald ein Hash-Wert nicht übereinstimmt. Man geht dann davon aus, dass etwas manipuliert wurde. Das erwartete Verhalten beim Measurement Produktinformationen wäre „non-compliant minor“, wenn die Version des Betriebssystems nicht aktuell ist. Folglich würde man in der Quarantäne landen, wo man nur Internet-Zugriff hat, damit man das Gerät auf die neuste Version aktualisieren kann.

7. Erfahrungsberichte

7.1 Christoph Bühler

Diese Studienarbeit hat mich sehr positiv überrascht. Als ich zu Beginn der Arbeit erfuhr, dass wir auch mit C-Code konfrontiert werden, hatte ich ein eher mulmiges Gefühl. Wir haben beide keine Ahnung von C-Programmierung und hatten uns auch nie sehr dafür interessiert. Glücklicherweise konnten wir die Implementierung so aufbauen, dass nur sehr wenig in C geschrieben werden musste. Dies ist auch darauf zurückzuführen, dass eine Implementation in C mittels JNI ein sehr komplexes Unterfangen gewesen wäre. Ein grosser Teil wird nun von Java erledigt, was für uns bedeutet, dass wir eine „schöne“ Implementation realisieren konnten.

Als ich dann jedoch mit dem C Code in Kontakt kam und auch verstanden hatte, was wo genau passiert, war der C Code gar nicht mehr so abschreckend wie zu Beginn der Arbeit. Dies ist sicherlich auch darauf zurückzuführen, dass es ein sehr grosses Projekt ist und sehr viel in C realisiert ist, nicht zuletzt wegen den Performancegründen. Mittlerweile habe ich eine genaue Vorstellung, wie das Projekt aufgebaut ist und kann mich darin zurechtfinden. Ich habe einige gute Eindrücke in die Programmierung in C erhalten und finde mich auch hier mit einiger Hilfe zurecht. Auch die „objektorientierte Programmierung“, welche man in C mithilfe von „structs“ realisieren kann, ist nach einer gewissen Angewöhnungsphase gar nicht mehr so befremdlich. Im Grossen und Ganzen war dies ein grosser Erfolg und beinhaltet einen noch grösseren Lerneffekt für mich persönlich.

Das Projekt hatte natürlich seine Höhen und Tiefen, wie das in jedem Projekt der Fall ist. Zu Beginn wussten wir nicht, ob wir die Arbeit überhaupt realisieren können, wie wir das gerne gehabt hätten. Für gewisse Messungen bestand die Gefahr, dass sie nur mit root-Rechten durchgeführt werden konnten. Glücklicherweise wurde dieses Risiko schon in den ersten zwei Wochen, in welchen wir unsere Vorabklärungen getroffen hatten, beseitigt. Die Zeitplanung unseres Projektes war eigentlich immer im Lot, wir hatten genügen Pufferzeiten und hatten das Glück, diese nie auszureizen. Das Projekt lief erstaunlich gut, weswegen wir beinahe „zu früh“ fertig waren. Dies bot uns zeitliche Polster, um unsere Codes noch einmal zu überarbeiten und zu verschönern.Es war eine gute und interessante Studienarbeit, in welcher ich vieles gelernt habe. Auch dank der höchst kompetenten Betreuung durch Prof. Dr. Steffen und Tobias Brunner konnten wir viele Probleme bereits im Ansatz und in kürzester Zeit beseitigen. Grössere Probleme traten dadurch gar nie auf. Die grössten Differenzen hatten wir wahrscheinlich mit der Versionskontrolle „GitHub“, da das Repository von strongSwan auf einem automatischen, sogenannten „rebasing“ basiert. Leider hatte unser Klon des Repositories auf das „merging“ gesetzt. Dies führte beim Laden der neuesten Änderungen von der Originalquelle zu einigen Problemen, welche dann aber durch die Hilfe von Tobias Brunner immer wieder behoben werden konnten. Nach der Hälfte des Projektes hatten wir git dann auch dahingehend im Griff, dass diese Fehler nicht mehr auftraten. Schlussendlich ist zu erwähnen, dass diese Arbeit eine sehr angenehme Symbiose zwischen Netzwerktechnologie und Softwareentwicklung darstellt. Auf der einen Seite steht die Verschlüsselung und Netzwerkproblematik und auf der anderen die Umsetzung der selbigen. Eine gute Mischung, um vieles zu lernen und Erfahrungen zu machen.

7.2 Patrick Lötscher

Für mich war die Studienarbeit eine interessante Herausforderung. Besonders motiviert hat mich die Tatsache, dass wir etwas erarbeiten konnten, was später auch eingesetzt wird. Das Entwickeln einer Erweiterung für eine bestehende Software war für mich ebenfalls eine Bereicherung, da wir bis jetzt nur neue Lösungen implementiert haben. Im Berufsleben wird es jedoch oftmals auch vorkommen, dass man den bestehenden Code erweitern muss und man nicht immer von vorne beginnen kann.

Am Anfang war ich mir etwas unsicher, wie viel man effektiv in C programmieren muss, da Christoph und ich nur rudimentäre Fähigkeiten in der C-Programmierung vorweisen konnten. Einen Teil zur Unsicherheit steuerte auch die Tatsache bei, dass wir nicht wussten, welche Messungen wir effektiv ohne root-Rechte durchführen konnten. Zum Glück konnten wir letztere bereits in den ersten 2 Wochen durch die Entwicklung eines Prototyps beseitigen. Damit wir nur so wenig wie möglich in C programmieren mussten, haben wir uns entschieden, soviel wie möglich in Java zu implementieren.

Die Implementation des GUIs und das Bearbeiten des Assessment Results boten mir einen tieferen Einblick in die Android/Java Programmierung. Es war auch sehr interessant zu sehen, wie bereits vorhandene Elemente im Android App implementiert wurden. So ist es möglich, seinen eigenen Horizont zu erweitern, indem man neue/andere Möglichkeiten einer Implementation sieht. Ein anderer interessanter Aspekt für mich war, einige Attribute des TNC Protokolls in Java zu implementieren und in geeigneter Form weiter zu reichen, damit sie im C-Code an den Server übermittelt werden konnten.

Die Zusammenarbeit der Gruppe war ausserordentlich gut. Bereits zu Beginn des Projekts konnten wir uns über die Zuständigkeiten einigen und diese somit klar aufteilen. Damit wurde eine perfekte Grundlage für das Weiterführen des Projekts gelegt. Erfreulich war für mich, dass wir bereits von Anfang an sehr engagiert gestartet sind. Aus diesem Grund konnten wir den Terminplan, den wir im Projektplan festlegten, stets einhalten.

Allgemein muss ich vor allem auch die sehr kompetente und hilfsbereite Betreuung durch Professor Dr. Steffen und Tobias Brunner loben. Bei Problemen wurde uns immer sehr schnell geholfen und auch die Implementationen, die am Server getätigt werden mussten, wurden innert kürzester Zeit realisiert, womit wir nie wirklich ins Stocken geraten sind.

Die Studienarbeit war für mich auf jeden Fall sehr interessant und ich konnte einige neue Erfahrungen sammeln, die sich später sicherlich als nützlich erweisen werden.

8. Appendix

8.1 Projektplanung

8.1.1 Organigramm

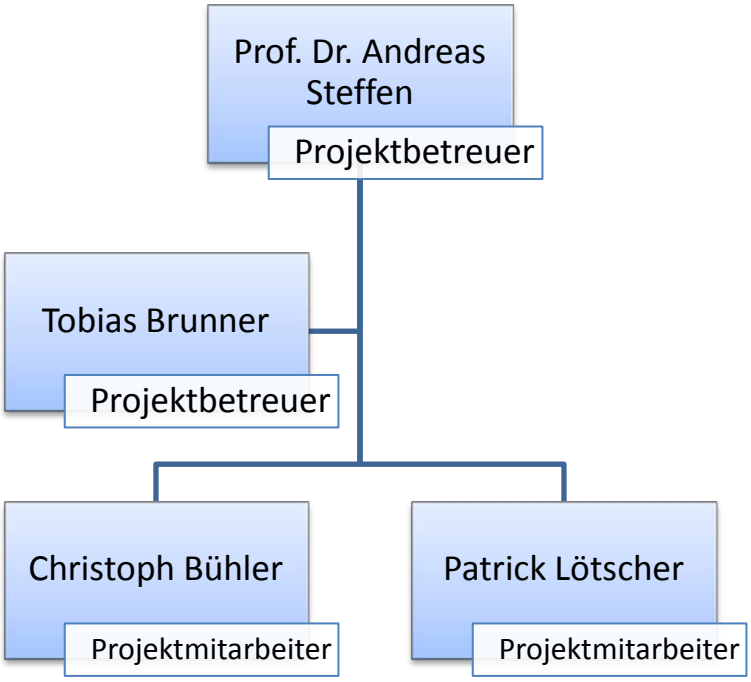

Abbildung 29: Projektorganigramm

8.1.2 Meilensteine

Meilen-stein	Name	Beschreibung	Arbeitsprodukte für Review	Ende
MS1	Vorabklärungen	Abklärungen zu Android getroffen, bereit für Projektarbeit	• Vorabklärungen	02.10.2012
MS2	Design	Designvorstellungen zur Implementation getroffen und beschrieben	• Designmodell • Schnittstellendefintion	16.10.2012
MS3	Implementierung IMCs 1	Implementierung der Kollektoren für: - String Version - Product information - Installed Packages	• Kollektoren	30.10.2012
MS4:	Implementierung IMCs 2	Implementierung der Kollektoren für: - Open Ports - File Measurements - Device Settings	• Kollektoren	13.11.2012
MS5:	GUI Vorschlag	Vorschlag für das Format der remediation instructions	• Vorschlag GUI	20.11.2012
MS6:	GUI implementiert	GUI für die remediation instructions ist implementiert und läuft	• Activity für Anzeige	27.11.2012
MS7:	Refactoring und Testing	App fertiggestellt, bereit zum Test durch Team	• Fertiger Code	04.12.2012

Tabelle 20: Meilensteintabelle

8.1.3 Projektplan

	Task Mode	Task Name	Duration	Start	Finish	Resource Initials
1		**Inception**	**5 days**	**Wed 19.09.12**	**Tue 25.09.12**	
2		Kick off meeting	2 hrs	Wed 19.09.12	Wed 19.09.12	cb;pl
3		Technischer Bericht vorbereiten	4 hrs	Wed 19.09.12	Fri 21.09.12	cb;pl
4		Projektplan erstellen	1 day	Fri 21.09.12	Tue 25.09.12	pl
5		**Elaboration 1**	**5 days**	**Wed 26.09.12**	**Tue 02.10.12**	
6		Git konfigurieren	1 hr	Wed 26.09.12	Wed 26.09.12	cb
7		**Abklärung Android**	**5 days**	**Wed 26.09.12**	**Tue 02.10.12**	
8		Offene Ports Scannen	0.5 days	Wed 26.09.12	Fri 28.09.12	pl
9		Antivirus / Antispyware Android	0.5 days	Fri 28.09.12	Fri 28.09.12	pl
10		Welche Apps sind Installiert (auslesen)	0.5 days	Wed 26.09.12	Fri 28.09.12	cb
11		Filesystem auslesen	0.5 days	Fri 28.09.12	Fri 28.09.12	cb
12		Settings auslesen von Android	0.5 days	Fri 28.09.12	Tue 02.10.12	cb
13		MS1: Abklärungen zu Android getroffen, bereit für Projektarbeit	0 days	Tue 02.10.12	Tue 02.10.12	
14		**Elaboration 2**	**10 days**	**Wed 03.10.12**	**Tue 16.10.12**	
15		Risikoanalyse durchführen	2 hrs	Wed 03.10.12	Wed 03.10.12	pl
16		Use Case Diagramm erstellen	2 hrs	Wed 03.10.12	Wed 03.10.12	cb;pl
17		Use Case Brief Beschreibung	2 hrs	Fri 05.10.12	Fri 05.10.12	cb
18		Analyse von strongSwan, Entwicklungsumgebung einrichten	2.5 days	Fri 05.10.12	Fri 12.10.12	cb;pl
19		RFC Attribute beschreiben	8 hrs	Fri 12.10.12	Tue 16.10.12	cb
20		GUI - Mockups erstellen	4 hrs	Fri 12.10.12	Fri 12.10.12	pl
21		Design Modell erstellen	4 hrs	Mon 15.10.12	Tue 16.10.12	pl
22		MS2: Designvorstellungen zur Implmementation	0 days	Tue 16.10.12	Tue 16.10.12	
23		**Construction 1**	**10 days**	**Wed 17.10.12**	**Tue 30.10.12**	
24		Android: String Version, Product Information, Installed Packages implementieren	8 hrs	Wed 17.10.12	Fri 19.10.12	cb;pl
25		Schnittstelle zum Android App in C entwickeln	16 hrs	Fri 26.10.12	Tue 30.10.12	cb;pl
26		MS3: IMCs String Version, Product Information, Installed Packages implementiert	0 days	Tue 30.10.12	Tue 30.10.12	
27		**Construction 2**	**10 days**	**Wed 31.10.12**	**Tue 13.11.12**	
28		IMC in der Android App implementieren	8 hrs	Wed 31.10.12	Fri 02.11.12	cb;pl
29		Schnittstelle zum Android App in C entwickeln	16 hrs	Fri 02.11.12	Wed 07.11.12	cb;pl
30		GUI für Android erstellen	8 hrs	Fri 09.11.12	Fri 09.11.12	cb;pl
31		MS4: Open Ports, File Measurement, Device Settings Attribute Implementiert	0 days	Tue 13.11.12	Tue 13.11.12	
32		**Construction 3**	**5 days**	**Wed 14.11.12**	**Tue 20.11.12**	
33		C-IMC: Refactoring	12 hrs	Wed 14.11.12	Fri 16.11.12	cb
34		Java: GUI für Anzeige an User erstellen	12 hrs	Wed 14.11.12	Fri 16.11.12	pl
35		MS5: GUI-Vorschlag für Remediation Instructions erstellt	0 days	Tue 20.11.12	Tue 20.11.12	
36		**Construction 4**	**10 days**	**Wed 21.11.12**	**Tue 04.12.12**	
37		C: Kommunikation für GUI-Anzeige implementieren	1.5 days	Wed 21.11.12	Fri 23.11.12	cb
38		Java: Implementation des definitiven GUI	1.5 days	Wed 21.11.12	Fri 23.11.12	pl
39		MS6: GUI für Remediation Instructions implementiert und lauffähig	0 days	Tue 27.11.12	Tue 27.11.12	
40		C: Refactoring	0.5 days	Tue 27.11.12	Tue 27.11.12	cb
41		Java: Refactoring	0.5 days	Wed 28.11.12	Wed 28.11.12	pl
42		Java: Unit Tests für Kollektoren und Attribute erstellen	1.5 days	Fri 30.11.12	Mon 03.12.12	cb;pl
43		MS7: Refactoring und Testing abgeschlossen	0 days	Tue 04.12.12	Tue 04.12.12	
44		**Transition**	**5 days**	**Wed 05.12.12**	**Tue 11.12.12**	
45		Dokumentationen vervollständigen	2 days	Wed 05.12.12	Mon 10.12.12	cb;pl
46		**Puffer**	**5 days**	**Wed 12.12.12**	**Tue 18.12.12**	
47		Projektabgabe	0 days	Fri 21.12.12	Fri 21.12.12	

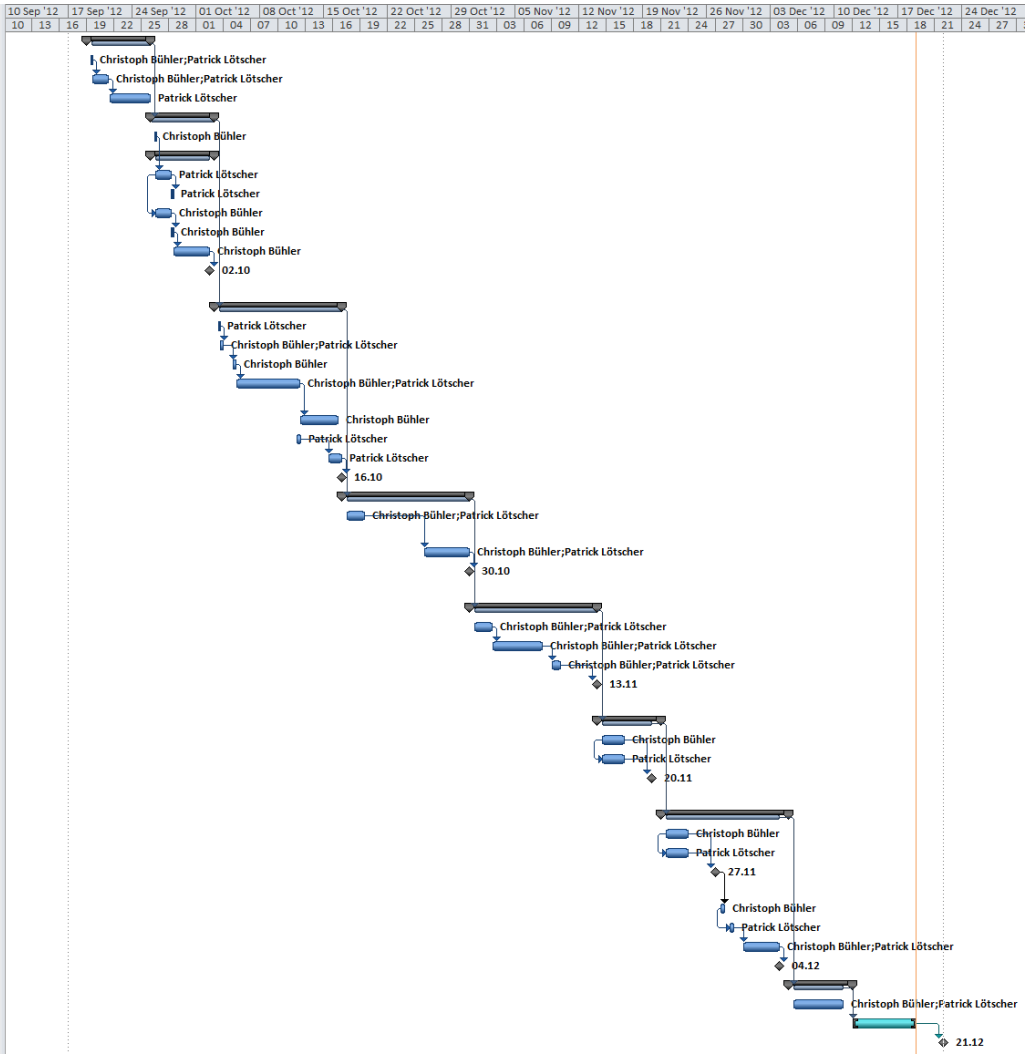

Abbildung 30: Projektplan

8.2 Abkürzungsverzeichnis

Abkürzung	Bedeutung
IMC	Integrity Measurement Collector
IMV	Integrity Measurement Verifier
TNC	Trusted Network Connect
TNCC	TNC Client
TNCS	TNC Server
TCG	Trusted Computing Group
PEN	Private Enterprise Number
NDK	Native Development Kit
SDK	Software Development Kit

Tabelle 21: Abkürzungsverzeichnis

8.3 Abbildungsverzeichnis

8.4 Tabellenverzeichnis

8.5 Literaturverzeichnis

[1] IETF, „IETF RFC 5792,“ [Online]. Available: http://tools.ietf.org/html/rfc5792. [Zugriff am 28 09 2012].

[2] R. Sager und G. Grassi, „Userland IPsec fur Android 4.0,“ 2012. [Online]. Available: http://security.hsr.ch/theses/BA_2012_Userland_IPsec_for_Android_4_Report.pdf. [Zugriff am 13 12 2012].

[3] Strongswan.org, „TNC HowTo,“ [Online]. Available: http://wiki.strongswan.org/projects/strongswan/wiki/TrustedNetworkConnect. [Zugriff am 28 09 2012].

[4] „Android Developers SDK Reference,“ 27 09 2012. [Online]. Available: http://developer.android.com/reference/packages.html. [Zugriff am 27 09 2012].

[5] av-test.org, „av-test.org,“ 03 2012. [Online]. Available: http://www.av-test.org/fileadmin/pdf/avtest_2012-02_android_anti-malware_report_english.pdf. [Zugriff am 27 09 2012].

[6] Trusted Computing Group, „PTS Protocol: Binding to TNC IF-M,“ 2011.

[7] M. Mead, „Programming in C/C++ with the Java Native Interface,“ 1998. [Online]. Available: http://home.pacifier.com/~mmead/jni/cs510ajp/index.html.

[8] «Android Developers SDK,» 28 02 2012. [Online]. Available: http://developer.android.com/sdk/index.html. [Zugriff am 27 09 2012].

[9] Strongswan.org, „Strongswan.org,“ 15 08 2012. [Online]. Available: http://strongswan.org/. [Zugriff am 28 09 2012].

8.6 Sitzungsprotokolle

8.6.1 KW 38

8.6.1.1 Traktanden

- Kick-Off
- StrongSwan
- TNC Protokoll (RFC 5792)

8.6.1.2 Kurzfassung

Kick-Off Meeting mit den Teilnehmern (Christoph Bühler; Patrick Lötscher) und den Betreuern (Prof. Dr. Andreas Steffen; Tobias Brunner). Behandelt wurden die Grundsätze der Funktion „Trusted Network Connect“ (TNC), wie die Projektdokumentation auszusehen hat, welche Elemente im Vorfeld bekannt sein müssen beziehungsweise eruiert werden müssen und welchen Umfang die Arbeit hat. Des Weiteren wurde ein Einblick in strongSwan gewährt, um das Prinzip der Software in Ansätzen zu verstehen.

8.6.1.3 Ergebnis

- Dokumentation kann anhand der HSR Vorgaben und vergangenen Arbeiten gestaltet werden.
- Vorabklärungen zu Android:
    - Auslesen der installierten Apps
    - Auslesen des Dateisystems
    - Bestimmen der Möglichkeiten von Antivirussoftware
    - Anzeigen der offenen Ports

8.6.2 KW 39

8.6.2.1 Traktanden

- Vorabklärungen
- Vorgang strongSwan Connect
- Aufgabenstellung
- Genauere Einblicke in den RFC Standard

8.6.2.2 Kurzfassung

Die Auswertung und Besprechung der Vorabklärungen wurde mit den Einblicken in den RFC Standard kombiniert. Die Vorabklärungen wurden als gut und ausführlich empfunden und dienen so sicherlich der weiteren Arbeit. Des Weiteren wurden die Attribute des RFC genauer unter die Lupe genommen und der C-Code von strongSwan erklärt (beziehungsweise der relevante Teil für die Arbeit). Die wichtigste Funktion, welche eventuell auch vom Projektteam realisiert wird, ist die C-Schnittstelle, welche mit den tieferen Ebenen kommuniziert. Wichtig dafür ist, dass die Schnittstelle zwischen Java und C gut und sauber definiert ist. Zu diesem Zweck wird im späteren Verlauf des Projektes noch ein Termin mit Tobias Brunner vereinbart, um die Schnittstelle zu definieren. Im Wesentlichen muss dann – je nach Design – mehr oder weniger in der Programmiersprache C geschrieben werden.

8.6.2.3 Ergebnis

- Schnittstelle zwischen C und Java in einem späteren Schritt definieren.
- Vorabklärungen ausbauen auf die verwendeten Attribute des Standards. Es müssen Abklärungen getroffen werden, wie die gesammelten Informationen abgebildet werden und in die schon vorhandenen Attribute des Standards passen. Unter gewissen Umständen können im HSR – Namespace auch neue Attribute definiert werden.
- Dokumentation als ein Dokument (Technischer Bericht) anfertigen und so ausbauen.
- Projektplanung (MS-Project) im Wochenintervall grob erstellen bis KW 40/41.
- Google Nexus 7 erhalten.
- Aufgabenstellung wird elektronisch zugesandt durch Prof. Dr. Steffen.

8.6.3 KW 40

8.6.3.1 Traktanden

- Projekt- und Zeitplan Review
- Erweiterte Abklärungen und Attributdefinitionen
- Zeiterfassung pro Aufgabe pro Person
- Use Cases Fully Dressed
- strongSwan Sourcecode (compiling)

8.6.3.2 Kurzfassung

Es wurde definiert, wie der Zeitplan in etwa aussehen muss. Des Weiteren wurden die definierten Standardattribute begutachtet und diskutiert, welche eventuell neuen Attribute erstellt werden müssen. Das Ziel ist, so wenige Zusatzattribute wie möglich zu verwenden. Die UseCases, welche definiert wurden, müssen noch überarbeitet und ein wenig feiner dargestellt werden. Fully Dressed – Beschreibung muss keine erstellt werden, da die UseCases zu wenig komplex sind. Nach der offiziellen Sitzung wurde mit Tobias Brunner der Sourcecode von strongSwan angeschaut. Ziel war es, den Code (C Teil) zu kompilieren, was jedoch bis zum Ende der Sitzung nicht gelang. Es wurde nun beschlossen, dass die Entwicklung auf einer virtuellen Maschine mit Linux stattfindet, da die benötigten Tools einfacher verfügbar sind.

8.6.3.3 Ergebnis

- UseCase-Brief Beschreibung reicht aus
- Projektplan auf Wochenbasis erstellen
- Kompilieren des Sourcecodes ist komplex (mittels VMWare Workstation arbeiten)
- UseCase „Client überprüfen“ muss verfeinert werden

8.6.4 KW 41

8.6.4.1 Traktanden

- Attributzuweisungen
- Aufgabenstellung

8.6.4.2 Kurzfassung

Im Grunde wurden die Attribute begutachtet und die Mappings besprochen. Es wurde festgelegt wie der Projektplan zu planen ist und welche Dokumente relevant sind. Die Aufgabenstellung wird in elektronischer Form zugesandt.

8.6.4.3 Ergebnis

- Projektplan muss auf Wochenbasis geplant werden

8.6.5 KW 42

8.6.5.1 Traktanden

- Projektplan / Zeitplan
- Attributdefinition
- Designmodell Java

8.6.5.2 Kurzfassung

Herr Steffen musste die Sitzung leider wegen eines Termins früher verlassen. Mit Tobias Brunner wurden das Designmodell diskutiert und die grundsätzlichen Design-, und Programmierideen besprochen. Die Übersetzung des Javaobjektes in das C-Attribut erfolgt über ein Interface, welches eine Methode „getByteArray“ implementiert. Mittels dieser Funktion erhält der native Code das gesamte Attribut vordefiniert als Byte-Array. Dieser wird mittels einer Funktion in einen Chunk übersetzt und danach von den vordefinierten Attribut-Konstruktoren in das Attribut übersetzt. Dieses Attribut kann danach an die Nachricht angehängt werden.

8.6.5.3 Ergebnis

- Designmodell ist ok
- Ideen sind in Ordnung, Programmierabläufe sind ideal so, da in Java mehr mit Objekten gehandelt werden kann

8.6.6 KW 43

8.6.6.1 Traktanden

- Attribute für „Stringversion“ und „Productinformation“ zusammenschliessen
- Debuglevel erhöhen um Paketinformationen zu bekommen (How-To)
- Eigenes Attribut begutachten
- Projektplan begutachten

8.6.6.2 Kurzfassung

Die Frage, ob die Attribute „Stringversion“ und „Productinformationen“ zusammengeschlossen werden können, wurde verworfen, da das Protokoll beziehungsweise der Standard dies so vorschreibt. Es wird nun für beide Informationen je ein Measurement und ein Attribut definiert. Es wurden der Code-Stil für C angeschaut und einige „Guidelines“ festgelegt, was das Schreiben von C-Code angeht. Der Projektplan wurde begutachtet und danach noch als PDF Ausdruck an Herrn Steffen übergeben. Das spezielle Attribut wurde begutachtet und wird durch Herrn Steffen noch genauer geprüft und dann gegebenenfalls implementiert.

8.6.6.3 Ergebnis

- Attribute „Stringversion“ und „Productinformation“ müssen eigenständig sein
- Debuglevel wird im „android_logger.c“ geändert
- Eigenes Attribut wird geprüft

8.6.7 KW 44

8.6.7.1 Traktanden

- Spezieller Useraccount für Zugriff auf VPN Services
- Nächste zu implementierende Kollektoren definieren
- Review der implementierten Kollektoren
- Rebase des Branches

8.6.7.2 Kurzfassung

Nach einem kurzen Code-Review der implementierten Kollektoren wurden die weiteren Schritte diskutiert. Diese wurden danach als Meilensteine in den Projektplan eingetragen. Bezüglich des speziellen Useraccounts wurde seitens der HSR der StrongSwan Port im internen WLAN freigeschaltet. Somit kann nun auch aus dem eigenen WLAN probiert werden.

8.6.7.3 Ergebnis

- Implementation neuer Kollektoren

8.6.8 KW 45

8.6.8.1 Traktanden

- Review der Kollektoren
- Definition der nächsten Schritte

8.6.8.2 Kurzfassung

Die Besprechung beinhaltete primär die Implementation der neuen Kollektoren Open Ports und Installed Packages und die daraus resultierende Implementation, die beim Server getätigt werden muss, damit Tests durchgeführt werden können. Am Schluss wurde auf die ausstehenden Kollektoren File Measurement und Device Settings eingegangen, die bis zur nächsten Sitzung implementiert werden.

8.6.8.3 Ergebnis

- Implementation der letzten Kollektoren

8.6.9 KW 46

8.6.9.1 Traktanden

- Review der Kollektoren (File Measurement, Device Settings)
- Wireframes

8.6.9.2 Kurzfassung

Die neu implementierten Kollektoren File Measurement und Device Settings wurden besprochen. Danach wurden die Wireframes zur Darstellung der Remediation Instructions begutachtet.

8.6.9.3 Ergebnis

- Performance Vergleich für das Hashing im File Measurement zwischen Java und C muss durchgeführt werden

8.6.10 KW 47

8.6.10.1 Traktanden

- Struktur für die Übertragung der Remediation Instructions definieren

8.6.10.2 Kurzfassung

Vorgängig wurde ein Beispiel für die Übertragung der Remediation Instructions in XML definiert. Aufgrund des Beispiels wurde die endgültige XML-Formatierung festgelegt. Aus Gründen der Klarheit mussten die Details auf Items umbenannt werden und die Items wurden als optional definiert, da es auch Remediation Instructions ohne zusätzliche Information gibt.

8.6.10.3 Ergebnis

- Endgültige Spezifikation der Struktur (XML) zur Übertragung der Remediation Instructions

8.6.11 KW 48

8.6.11.1 Traktanden

- Implementation des Assessment Results im bestehenden GUI

8.6.11.2 Kurzfassung

Die Anzeige des Asssessment Results im bestehenden Status Fragment und im Menu wurde verworfen. Stattdessen wird ein neues Fragment unterhalb des Status Fragments angezeigt, sobald Remediation Instructions vorhanden sind. Durch diese Lösung ist die Kopplung der Endpoint Assessment Erweiterung mit der bestehenden App geringer.

8.6.11.3 Ergebnis

- Finale Version des GUIs spezifiziert

8.6.12 KW 49

8.6.12.1 Traktanden

- Präsentation des GUIs

8.6.12.2 Kurzfassung

Das GUI wurde Herrn Steffen und Tobias Brunner vorgeführt. Dabei wurden alle möglichen Meldungen (compliant, non-compliant minor, non-compliant major) des Assessment Results getestet, um die verschiedenen Ansichten des Fragments anzuzeigen.

Des Weiteren wurde von Herrn Steffen festgelegt, dass Quellenangaben im technischen Bericht mit eckigen Klammern und einer fortlaufenden Zahl direkt nach dem entsprechenden Element (Beispiel: [1]) auszuweisen sind. Im Literaturverzeichnis sind dann die effektiven Quellen dokumentiert.

8.6.12.3 Ergebnis

- Testen ob der Name des Packages geändert werden kann
- Der Text „Beurteilung“ im Assessment Result Fragment muss auf „Letzte Beurteilung“ geändert werden

8.6.13 KW 50

8.6.13.1 Traktanden

- Management Summary
- Testing
- Struktur des technischen Berichts

8.6.13.2 Kurzfassung

Es wurde besprochen, ob ein Management Summary nötig ist und ob irgendwelche Tests erwartet werden, da unter Android ein sinnvolles, funktionales Testing nicht möglich ist. Danach gab es ein Feedback zur Struktur des Dokuments und zu gewissen Formulierungen. Zuletzt wurde das A0 Plakat besprochen.

8.6.13.3 Ergebnis

- Internationalisierung der Remediation Instructions
- A0 Plakat neu strukturieren
- Management Summary ist nicht nötig
- Es sind keine Tests nötig
- Dokument überarbeiten
    - Abstract
    - Saloppe Formulierungen eliminieren
    - Logische Struktur erstellen