HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Studienarbeit, Abteilung Informatik

App Analysator

Hochschule für Technik Rapperswil

Herbstsemester 2012 / 2013

Autoren:	Pascal Roman Artho & Rebekka Zahler
Betreuer:	Prof. Dr. Peter Heinzmann
Projektpartner:	Philipp Klomp, Nomasis AG
Arbeitsperiode:	17. September 2012 bis 21. Dezember 2012
Arbeitsumfang:	240 Stunden, 8 ECTS pro Student

Abstract

Abteilung	Informatik
Name[n] der Studierenden	Pascal Roman Artho Rebekka Zahler
Studienjahr	Herbstsemester 2012 / 2013
Titel der Studienarbeit	App Analysator
Examinatorin / Examinator	Prof. Dr. Peter Heinzmann
Themengebiet	Internet-Technologien und -Anwendungen
Projektpartner	Philipp Klomp, Nomasis AG
Institut	ITA: Institut für Internet-Technologien und -Anwendungen

Mobile Apps auf Smartphones oder Tablets werden von den meisten Leuten in jeder erdenklichen Lebenslage bedenkenlos installiert und genutzt. Es ist aber nicht immer klar, welche Informationen unter welchen Umständen solche Apps an andere Stellen weitergeben. Es tauchen immer wieder Meldungen auf über die unerwünschte Weitergabe von Daten wie beispielsweise Kontaktdaten, Telefonnummern oder Positionsdaten.

Im Rahmen der Studienarbeit wurde ein Messsystem zur Analyse von Tablet- oder Smartphone-Datenverkehr realisiert. Das zu untersuchende Gerät verbindet man via WLAN-Access Point über einen WebScarab Proxy mit dem Internet. Nach der Installation eines speziellen Zertifikats, kann der Proxy sowohl den HTTP- als auch den HTTPS-Verkehr unverschlüsselt aufzeichnen. Dadurch besteht die Möglichkeit auch die Daten zu untersuchen, die verschlüsselt an Drittpersonen verschickt werden. Zusätzlich werden alle Daten mit einem AirPcap USB-Stick aufgezeichnet, sodass neben den HTTP- und HTTPS- auch TCP- sowie UDP-Konversationen erfasst werden. Der AirPcap Stick erlaubt es, verschiedene Tablets und Smartphones voneinander zu unterscheiden.

Die gesammelten Daten aus Wireshark und WebScarab werden in einer selbst realisierten Java-Applikation, dem AppAnalysator, angezeigt und ausgewertet. Die Ergebnisse werden nach den Verbindungsendpunkten gruppiert und zeigen die mögliche Verursacher-App an.

Die entwickelte Software eignet sich am besten für iOS-Geräte, weil da der gesamte HTTP- und HTTPS-Internetverkehr über den eingegebenen Proxy geleitet wird. Bei Android sind dies meistens nur die Browser-Verbindungen. Andere Apps senden die Daten nicht über den angegebenen Proxy ins Internet. Dieses Phänomen hat sich bei den ersten Analysen gezeigt.

Aufgabenstellung

Studiengang:	Informatik (I)
Semester:	Herbstsemester 2012
	(17. September 2012 bis 21. Dezember 2012)
Verantwortlicher:	Prof. Dr. Peter Heinzmann
Industriepartner:	Philipp Klomp, Nomasis AG

Ausgangslage

Apps für Smartphone und Tablets werden von den meisten Leuten bedenkenlos installiert. Im Rahmen dieser Arbeit soll analysiert werden, welche Informationen unter welchen Umständen von Apps an welche Stellen gesendet werden. Es ist ein System zu realisieren, welches aufzeigt, welche Rechner aufgrund des Betriebs bestimmter Apps kontaktiert werden (Häufigkeit, Inhalte, Datenmengen).

Ziel

Das Ziel dieser Arbeit ist es, die Gefahren dieser Apps aufzuzeigen. Die Benutzer zu sensibilisieren und ihnen zu zeigen, dass ihre Daten auf dem Smartphone nicht ohne weiteres sicher sind, wenn man Apps ohne diese zu kennen, installiert. Mit einem Demonstrator wird gezeigt, welche Apps über das WLAN welche Verbindungen ins Internet tätigen. Ob diese verschlüsselt sind oder nicht und was dabei versendet wird.

Rapperswil, den 20.12.12 .....

.....

Betreuer: Prof. Dr. Peter Heinzmann

Erklärung zur Urheberschaft

Die vorliegende Arbeit basiert auf Ideen, Arbeitsleistungen, Hilfestellungen und Beiträgen gemäss folgender Aufstellung:

Gegenstand, Leistung	Person	Funktion
Kapitel 3, 5, 6, 7 sowie Anhänge	Pascal Roman Artho	Autor der Arbeit
Kapitel 1, 2, 4, 7 sowie Anhänge, Abstract und Management Summary	Rebekka Zahler	Autorin der Arbeit
Ideengeber, Betreuung während der Arbeit	Prof. Dr. Peter Heinzmann	Betreuer
Ideengeber	Philipp Klomp, Nomasis AG	Industriepartner
Korrektur	Rahel Zahler	Lektorat
Korrektur	Anna Artho	Lektorat
Korrektur	Michael Schnyder	Lektorat
Infrastruktur	Angestellte der Hochschule Rapperswil	Hochschule
Latex Vorlage	Florian Bentele	Student

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit gemäss obiger Zusammenstellung selber und ohne weitere fremde Hilfe durchgeführt habe,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.

Rapperswil, den 21.12.17 ..... P. Artho .....
Student: Pascal Roman Artho

Rapperswil, den 21.12.17 ..... R. Z. .....
Studentin: Rebekka Zahler

Vereinbarung zur Verwendung und Weiterentwicklung der Arbeit

1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Semesterarbeit «App Analysator» von Pascal Roman Artho und Rebekka Zahler unter der Betreuung von Prof. Dr. Peter Heinzmann (für die Arbeit verantwortlicher Professor) geregelt.

2. Urheberrecht

Die Urheberrechte stehen der Studentin / dem Student zu.

3. Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von allen an der Arbeit beteiligten Parteien, daher von den Studenten, welche die Arbeit verfasst haben, vom verantwortlichen Professor sowie vom Industriepartner verwendet und weiterentwickelt werden. Die Namensnennung der beteiligten Parteien ist bei der Weiterverwendung erwünscht.

Rapperswil, den 20.12.12 ..... P. Artho .....
Student: Pascal Roman Artho

Rapperswil, den 20.12.12 ....
Studentin: Rebekka Zahler

Rapperswil, den 20.12.12 ....
Betreuer: Prof. Dr. Peter Heinzmann

Rapperswil, den ....
Industriepartner: Philipp Klomp, Nomasis AG

Management Summary

Ausgangslage

Die Zahl der Smartphones und deren Applikationen, kurz Apps, nehmen immer weiter zu. Die Benutzer dieser Smartphones installieren deshalb immer wieder Apps. Von diesen Apps kennen sie dabei weder die Quelle, noch wissen die Benutzer, welche Informationen im Hintergrund ausgetauscht werden. Auf den verschiedenen Plattformen wie iOS und Android ist teilweise weder den Laien noch den Profis klar, was wirklich im Hintergrund alles passiert und zu welchem Zeitpunkt welche Daten, vielleicht sogar persönliche, verschickt werden.

Im Internet kursieren Berichte über Apps, die getestet wurden und Sicherheitslücken aufweisen. So zum Beispiel die App «WhatsApp», eine plattformunabhängige Messenger-App für Smartphones, welche alle gespeicherten Telefonnummern an den WhatsApp-Server schickt. [4] Weiter ist aber auch der einfache Zugriff von aussen auf den eigenen Account möglich. [21]

Als Ausgangslage werden die beiden Plattformen iOS und Android verwendet. Das Ziel ist eine Applikation, die aufzeigt, welches App welche Verbindungen über das WLAN nach aussen getätigt hat und welche Informationen übertragen werden. Um die Arbeit einzuschränken, wird lediglich eine Analyse des WLAN-Verkehrs durchgeführt. Die Datenkommunikation via GSM / 3G wird nicht weiter beachtet.

Vorgehen

In einem ersten Schritt wird versucht herauszufinden, was überhaupt machbar ist. Dafür wird ein Versuch nachgebaut, welcher bereits in einem c't-Heft [4] beschrieben und durchgeführt wurde. In diesem Versuch ging es darum, den Verkehr eines iPhones anhand eines Proxys abzuhören und herauszufinden, welche Daten verschickt werden. Im konkreten Fall wurde entdeckt, dass die App «WhatsApp», die Telefonnummern aus dem Telefon unverschlüsselt an den WhatsApp Server sendet. Der Versuch wurde entsprechend nachgebildet, wobei Burp als Proxy verwendet wurde. Dies ist ein in Java realisierter Proxy, der es zulässt «on-the-fly» Zertifikate dem Client auszustellen. «On-the-fly». bedeutet, dass der Burp Proxy automatisch für jede aufgerufene URL ein Zertifikat mit dem entsprechenden URL-Namen generiert.

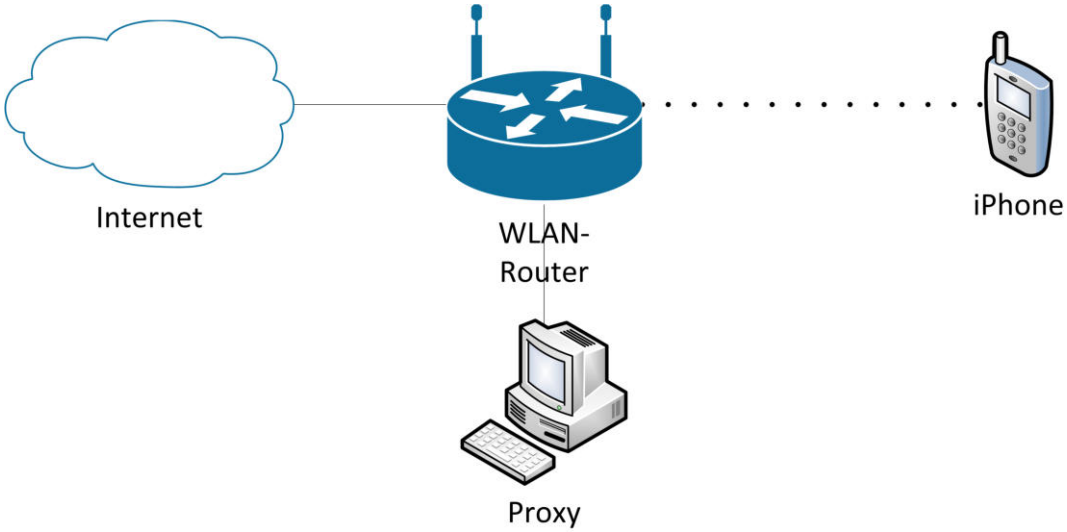


Abbildung 1: Versuchsaufbau Proxy

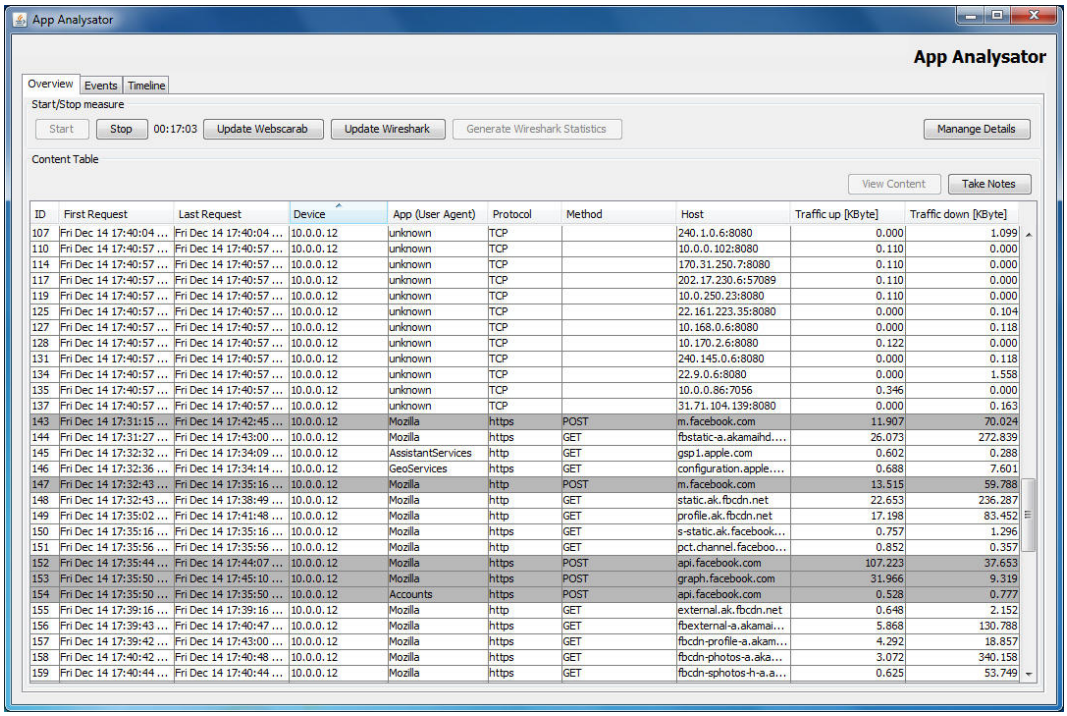
Als Nächstes wird ein weiterer Proxy getestet, der sogenannte WebScarab. Die Funktionen von WebScarab unterscheiden sich im Wesentlichen nicht vom Burp Proxy, ausser dass er immer das gleiche Zertifikat sendet. Der Benutzer wird deutlich mehr von diesem Zertifikat gewarnt, da sowohl die Identität, als auch der falsche Domainname angegeben wird. Zusätzlich wäre es möglich einen transparenten Proxy, sprich ein Proxy, der nicht zuerst im Telefon eingetragen werden muss, zu realisieren. Dadurch kann auch der Verkehr von Android Geräten vollständig abgehört werden, weil bei diesem Betriebssystem nicht so einfach ein Proxy eingetragen werden kann. Diese Funktion erfordert ARP-Spoofing, damit der Verkehr über den Proxy geleitet werden kann.

Die Entscheidung, welcher Proxy verwendet werden soll, fiel bei der Entwicklung der Applikation. Im WebScarab gibt es die Möglichkeit, die Requests und Responses direkt und automatisch in einzelne Dateien zu speichern. Dabei wird zu jedem Request angegeben, welches Gerät beziehungsweise welche IP-Adresse diesen Request erstellt hat. Dies vereinfacht das Auslesen und die Automatisierung unserer entwickelten Software, dem AppAnalysator, um Einiges. Da sich die Request- und Response-Inhalte sonst nicht weiter unterscheiden, wurde WebScarab verwendet.

Ergebnisse

Die Entwicklung des AppAnalysators gibt die Möglichkeit, Apps auf Smartphones und Tablets zu testen. Als Ergebnis sieht man, wie die Hosts zu denen Daten geschickt werden heissen und wie viele Daten dies waren. Der Inhalt, der gesendet oder empfangen wird, ist ebenfalls ersichtlich. Zusätzlich gibt es eine Ansicht, die chronologisch aufzeigt, welche Requests zu welchem Zeitpunkt gemacht werden und wie lange diese gedauert haben. Als Vorlage für die Darstellung wird das Firefox-Plugin «Firebug» betrachtet.

Das Ziel der Arbeit ist, zu sehen welche Apps zu welchen IP-Adressen / Domain-Namen einen Datenaustausch tätigen und welche Datenmengen und Protokolle dabei verwendet werden. Dies ist mit dem AppAnalysator möglich.



ID	First Request	Last Request	Device	App (User Agent)	Protocol	Method	Host	Traffic up [KByte]	Traffic down [KByte]
107	Fri Dec 14 17:40:04 ...	Fri Dec 14 17:40:04 ...	10.0.0.12	unknown	TCP		240.1.0.6:8080	0.000	1.099
110	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		10.0.0.102:8080	0.110	0.000
114	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		170.31.250.7:8080	0.110	0.000
117	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		202.17.230.6:57089	0.110	0.000
119	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		10.0.250.23:8080	0.110	0.000
125	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		22.161.223.35:8080	0.000	0.104
127	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		10.168.0.6:8080	0.000	0.118
128	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		10.170.2.6:8080	0.122	0.000
131	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		240.145.0.6:8080	0.000	0.118
134	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		22.9.0.6:8080	0.000	1.558
135	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		10.0.0.86:7056	0.346	0.000
137	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		31.71.104.139:8080	0.000	0.163
143	Fri Dec 14 17:31:15 ...	Fri Dec 14 17:42:45 ...	10.0.0.12	Mozilla	https	POST	m.facebook.com	11.907	70.024
144	Fri Dec 14 17:31:27 ...	Fri Dec 14 17:43:00 ...	10.0.0.12	Mozilla	https	GET	fbstatic-a.akamaihd....	26.073	272.839
145	Fri Dec 14 17:32:32 ...	Fri Dec 14 17:34:09 ...	10.0.0.12	AssistantServices	http	GET	gsp1.apple.com	0.602	0.288
146	Fri Dec 14 17:32:36 ...	Fri Dec 14 17:34:14 ...	10.0.0.12	GeoServices	https	GET	configuration.apple....	0.688	7.601
147	Fri Dec 14 17:32:43 ...	Fri Dec 14 17:35:16 ...	10.0.0.12	Mozilla	http	POST	m.facebook.com	13.515	59.788
148	Fri Dec 14 17:32:43 ...	Fri Dec 14 17:38:49 ...	10.0.0.12	Mozilla	http	GET	static.ak.fbcdn.net	22.653	236.287
149	Fri Dec 14 17:35:02 ...	Fri Dec 14 17:41:48 ...	10.0.0.12	Mozilla	http	GET	profile.ak.fbcdn.net	17.198	83.452
150	Fri Dec 14 17:35:16 ...	Fri Dec 14 17:35:16 ...	10.0.0.12	Mozilla	https	GET	s-static.ak.facebook...	0.757	1.296
151	Fri Dec 14 17:35:56 ...	Fri Dec 14 17:35:56 ...	10.0.0.12	Mozilla	http	GET	pct.channel.faceboo...	0.852	0.357
152	Fri Dec 14 17:35:44 ...	Fri Dec 14 17:44:07 ...	10.0.0.12	Mozilla	https	POST	api.facebook.com	107.223	37.653
153	Fri Dec 14 17:35:50 ...	Fri Dec 14 17:45:10 ...	10.0.0.12	Mozilla	https	POST	graph.facebook.com	31.966	9.319
154	Fri Dec 14 17:35:50 ...	Fri Dec 14 17:35:50 ...	10.0.0.12	Accounts	https	POST	api.facebook.com	0.528	0.777
155	Fri Dec 14 17:39:16 ...	Fri Dec 14 17:39:16 ...	10.0.0.12	Mozilla	http	GET	external.ak.fbcdn.net	0.648	2.152
156	Fri Dec 14 17:39:43 ...	Fri Dec 14 17:40:47 ...	10.0.0.12	Mozilla	https	GET	fbexternal-a.akamai...	5.868	130.788
157	Fri Dec 14 17:39:42 ...	Fri Dec 14 17:43:00 ...	10.0.0.12	Mozilla	https	GET	fbcdn-profile-a.akam...	4.292	18.857
158	Fri Dec 14 17:40:42 ...	Fri Dec 14 17:40:48 ...	10.0.0.12	Mozilla	https	GET	fbcdn-photos-a.aka...	3.072	340.158
159	Fri Dec 14 17:40:44 ...	Fri Dec 14 17:40:44 ...	10.0.0.12	Mozilla	https	GET	fbcdn-sphotos-h-a.a...	0.625	53.749

Abbildung 2: AppAnalysator

Zu Beginn der Arbeit war festgelegt worden, dass iOS- und Android-Geräte getestet werden sollen. Der AppAnalysator eignet sich für beide Betriebssysteme, allerdings gibt es in Android nicht die Möglichkeit, einen globalen Proxy anzugeben, welcher für das ganze Smartphone, also nicht nur für den Browser sondern auch für die Apps gilt. Deshalb können nicht alle HTTP- und HTTPS-Verbindungen von Android abgefangen werden während dies für iOS-Geräte problemlos funktioniert.

Die Systemgrenzen entsprechen der Abbildung 3. Eine grössere Ansicht ist im Anhang A beigelegt.

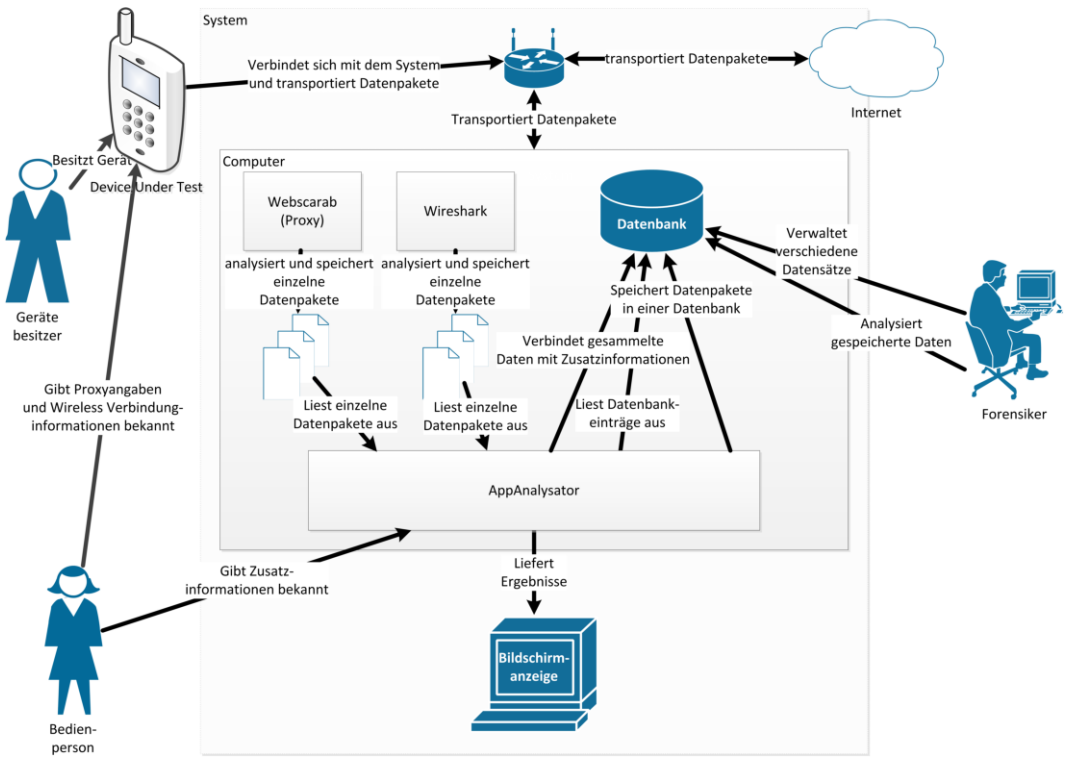


Abbildung 3: Systemgrenzen

Ausblick

Im AppAnalysator ist es, auf dem jetzigen Stand, nicht möglich IPv6-Pakete zu analysieren. Zusätzlich ist nicht gewährleistet, ob die Konfiguration auch nach einem Update von iOS lauffähig ist. Als Weiterentwicklung wäre es denkbar für Android-Smartphones eine andere Lösung zu entwickeln, als das Eintragen des Proxys. Als Lösungsansatz könnte man prüfen, ob ein transparenter Proxy verwendet werden kann, der mittels ARP-Spoofing die HTTP- und HTTPS-Verbindungen über einen «Man-in-the-Middle» leitet.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Weltweit gibt es eine Milliarde Smartphone-Nutzer. [11] Für die iOS- und Android-Plattform gibt es zusammen über eine Million Apps. [2] Die Wahrscheinlichkeit, dass sich unter diesen Apps eine befindet, welche auf einem der Smartphones persönliche Daten ausliest und an Drittanbieter schickt, ist riesig. Ebenso ist die Eintrittswahrscheinlichkeit gross, dass ein App ohne das Wissen des Benutzers unverschlüsselt wichtige Daten versendet.

Eine Applikation mit welcher man Testen kann, ob die oben genannten Fälle eintreffen, ist das Ziel dieser Studienarbeit. Um solch eine Applikation zu entwickeln, mussten zuerst diverse Versuche durchgeführt werden, die im Kapitel «Versuche» genauer beschrieben werden. Durch Versuche wurde eruiert, wie man am Besten den Verkehr aufzeichnen, beziehungsweise analysieren kann und mit welchen Methoden diese weiter verarbeitet werden können. Um die Vorgänge zu verstehen, mussten zuerst die Grundlagen erarbeitet werden. Auf diesen Bereich wird im Abschnitt «Grundlagen» weiter eingegangen.

Die entwickelte Applikation «AppAnalysator» ist fähig den HTTP- und HTTPS-Verkehr im Klartext zu analysieren. Weiter können die Verbindungen über TCP sowie UDP angezeigt werden. Die Applikation lässt zu, verschiedene Smartphones und Apps gleichzeitig zu analysieren. Dabei können diese dennoch voneinander unterschieden werden. Zusätzlich kann die Menge der ausgetauschten Daten sowie der Kommunikationspartner identifiziert und der Inhalt der Kommunikation angeschaut werden.

Der erste Hauptteil dieser Arbeit, das Kapitel 2 «Allgemeine Grundlagen», richtet sich an Denjenigen, der versuchen will, die Hintergründe der Arbeiten zu verstehen und die Grundlagen dazu zu erkennen. Das Kapitel 3 «Versuche» ist für den Leser von Interesse, der wissen will, wie vorgegangen wurde um den AppAnalysator zu entwickeln. Der letzte Hauptteil der Arbeit (Kapitel 4) handelt von der Softwareentwicklung und wird für Denjenigen wichtig, der weiterentwickeln will.

Kapitel 2

Allgemeine Grundlagen

In diesem Kapitel werden die grundsätzlichen Kenntnisse vermittelt, um die weiteren Kapitel verstehen zu können. Zum einen ist das die Man-in-the-Middle-Attacke, die genauer beschrieben wird und aufgezeigt was ARP-Spoofing ist und wie es eingesetzt wird. Zum anderen wie eine verschlüsselte Verbindung aufgebaut wird, was ein Zertifikat ist und wie dieses verwendet wird.

2.1 Man-in-the-Middle

Man-in-the-Middle wird eine Attacke genannt, bei welcher der Angreifer entweder physikalisch oder meist logisch in einem Computernetz zwischen zwei kommunizierenden Endgeräten steht. Dieser kann den Verkehr mithören sowie Manipulationen vornehmen. Er gibt den beiden Endgeräten jeweils vor, dass er das erwartete Endgerät ist. Meist merkt man nicht, dass sich jemand dazwischen geschaltet hat. Den Man-in-the-Middle gibt es in verschiedenen Formen, zum Beispiel als ARP- oder DNS-Spoofing sowie Phishing.

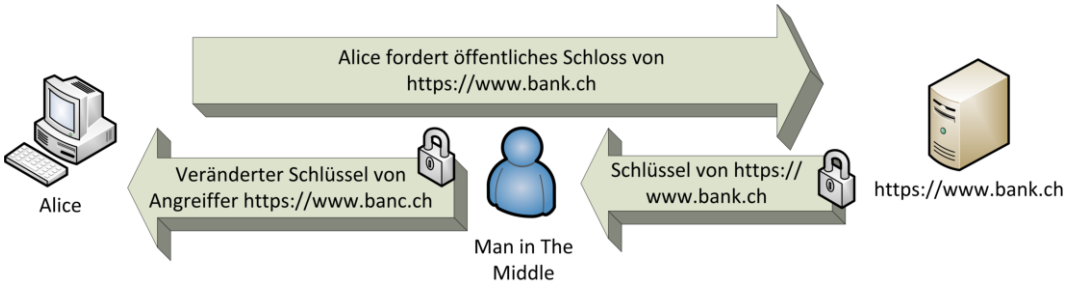


Abbildung 2.1: Man-in-the-Middle

2.1.1 Funktionsweise

In der Abbildung 2.1 ist das Beispiel einer Man-in-the-Middle-Attacke dargestellt. In diesem Beispiel fängt der Angreifer die Nachrichten zwischen dem Kunden und dem Bankserver ab und ändert das Zertifikat von https://www.bank.ch. Somit kommuniziert «Alice»

von nun an mit dem Man-in-the-Middle, denkt aber, sie sei immer noch direkt mit der Bank verbunden. Der Angreifer kann so als Man-in-the-Middle die Daten, die «Alice» eigentlich der Bank schicken will, abfangen. Als mögliche, besonders interessante Daten sind Kontonummern, Passwörter oder Kreditkarteninformationen denkbar.

2.2 Address Resolution Protocol (ARP)

Um IP-Pakete über das Netzwerk zu verschicken, werden sie auf der zweiten Schicht des OSI-Modells (siehe Abbildung 2.2), der Sicherungsschicht, in Frames gekapselt. Die Frames werden dabei mit einer MAC-Adresse adressiert. Das Address Resolution Protocol wird gebraucht um die physikalische Adresse (MAC-Adresse) in eine Adresse der Internetschicht aufzulösen. Der Hauptanwendungsfall ist die Ermittlung von der MAC-Adresse zur entsprechenden IP-Adresse. [7] [12]

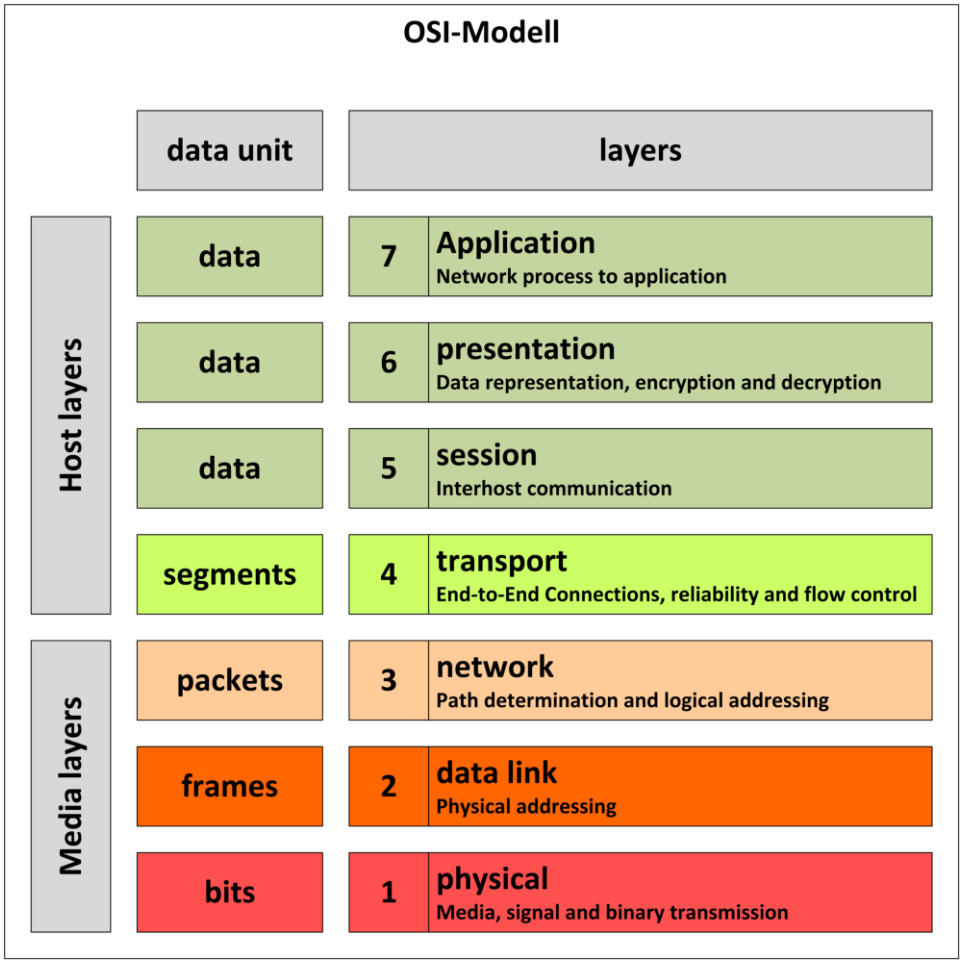


Abbildung 2.2: OSI-Modell [16]

2.2.1 ARP Spoofing

ARP-Spoofing ist eine Möglichkeit, eine Man-in-the-Middle-Attacke zu realisieren. Bei dieser Variante sind keinerlei Konfigurationen auf dem Endgerät notwendig. Ein Man-in-the-Middle gaukelt dem einen Endgerät, «Alice», vor, dass er das andere Endgerät, «Bob», sei und umgekehrt. Dabei verändert man die ARP-Tabellen der einzelnen Endgeräte in einem Netzwerk so, dass der Verkehr umgeleitet wird. Die Möglichkeit, Verkehr zu manipulieren und abzuhören, wird so sichergestellt. So gesehen senden «Alice» und «Bob» ihre Pakete an den Man-in-the-Middle, «Eve» (2.3). Dieser kann also den ganzen Verkehr mithören. Der Benutzer des Endgerätes merkt von dieser Attacke nicht wirklich etwas, ausser er prüft seinen ARP-Cache.

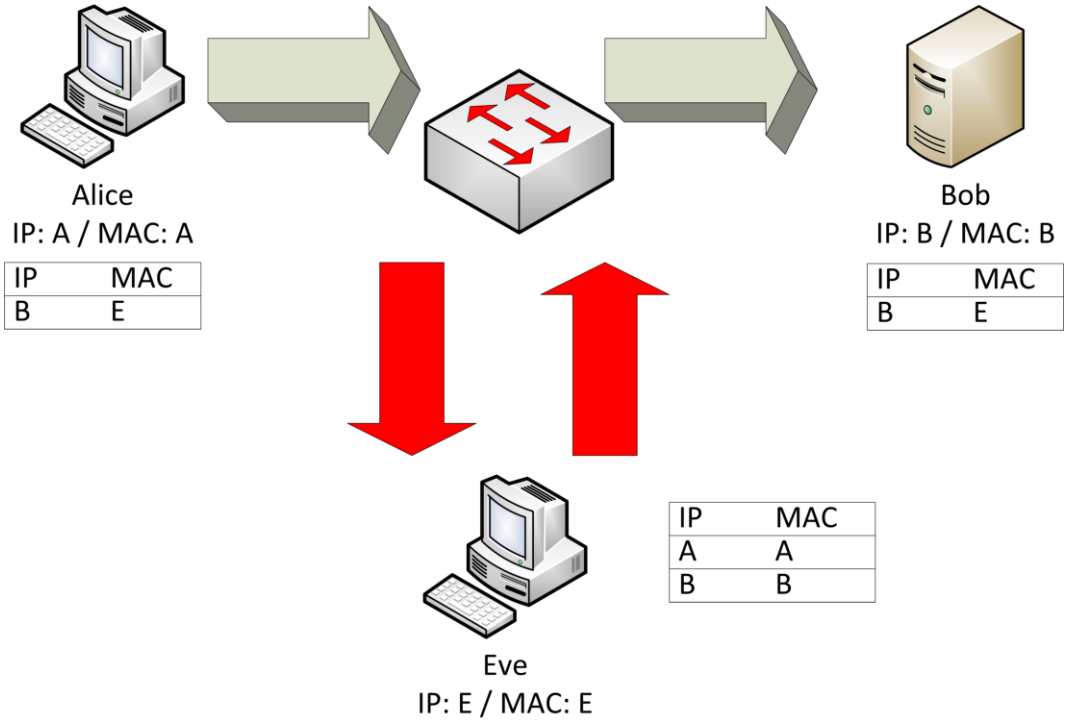


Abbildung 2.3: ARP-Spoofing

2.2.2 Funktionsweise

Ein Angreifer sendet gefälschte ARP-Pakete ins Netzwerk. Dabei hängt er einer IP-Adresse im Netz seine eigene MAC-Adresse an. Dadurch wird der ganze Verkehr an diese MAC-Adresse, sprich auch an die manipulierte IP-Adresse geschickt. Im konkreten Fall, der 2.3, sind «Alice» und «Bob» die beiden Endgerätenutzer, die miteinander kommunizieren. «Eve», der Angreifer, sendet nun manipulierte ARP-Pakete, mit der IP-Adresse von «Alice» und «Bob» und dazugehörend seine MAC Adresse. Damit werden alle Pakete, die an «Alice» und «Bob» geschickt werden, an «Eve» weiter geleitet.

2.3 Proxy

Ein Proxy dient als Schnittstelle zwischen zwei Netzwerkgeräten (siehe Abbildung 2.4). Er nimmt Anfragen entgegen und vermittelt sie weiter. Je nach dem für was ein Proxy benutzt wird, beeinflusst er den Verkehr. Bei einem transparenten Proxy, bekommen die beiden Kommunikationspartner nicht mit, dass ein Vermittlungsgerät dazwischen geschaltet ist. In dieser Arbeit wird kein transparenter Proxy verwendet, das Smartphone weiss also, dass es über einen dritten Kommunikationspartner geleitet wird. [18]

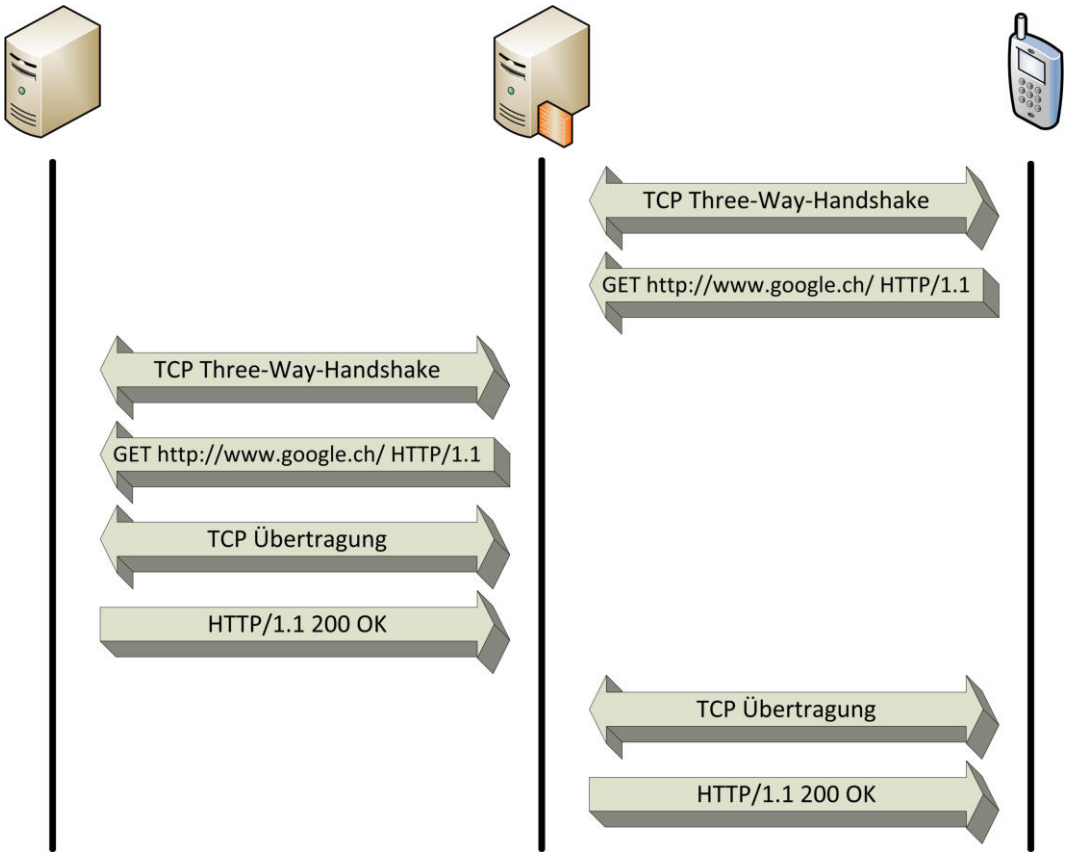


Abbildung 2.4: Proxy Verbindungsaufbau mit «www.google.ch»

2.4 Transport Layer Security (TLS)

Transport Layer Security ist eine Weiterentwicklung von Secure Socket Layer (SSL). Es dient der sicheren Datenübertragung im Internet. Vor allem wird es von HTTPS (Hypertext Transfer Protocol Secure) verwendet. TLS ist auf dem Transport Layer des TCP/IP-Modells (2.5) zu finden. Es verschlüsselt die Daten vom Layer oberhalb, dem Application-Layer, und reicht sie weiter an die dritte Schicht, die Transportschicht, und umgekehrt. [6]

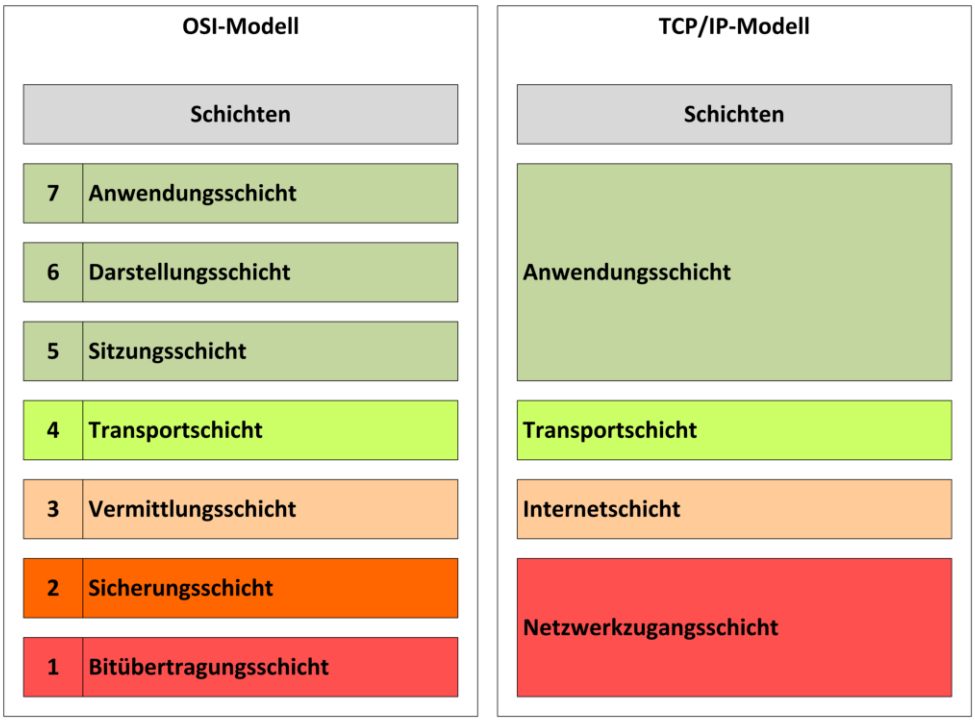


Abbildung 2.5: TCP/IP-Modell [14]

2.4.1 Verbindungsaufbau

Der Verbindungsaufbau kommt meistens vom Client. Er baut über TCP eine Verbindung zum Server auf. Nach dem Three-Way-Handshake authentifiziert sich der Server gegenüber dem Client mit einem Zertifikat, welches der Client auf Echtheit prüft. Die Nachrichten können auf zwei verschiedene Methoden verschlüsselt werden. Entweder wird, wie unten beschrieben, eine Zufallszahl (siehe Abbildung 2.6) generiert oder es wird ein sogenannter Diffie-Hellman-Schlüsseltausch durchgeführt. Dabei schicken die beiden Kommunikationspartner einander jeweils eine Nachricht über den noch unsicheren Kanal. Mit einem Algorithmus wird aus den beiden Nachrichten einen Schlüssel generiert. Die danach folgenden Pakete werden mit diesem Schlüssel verschlüsselt.

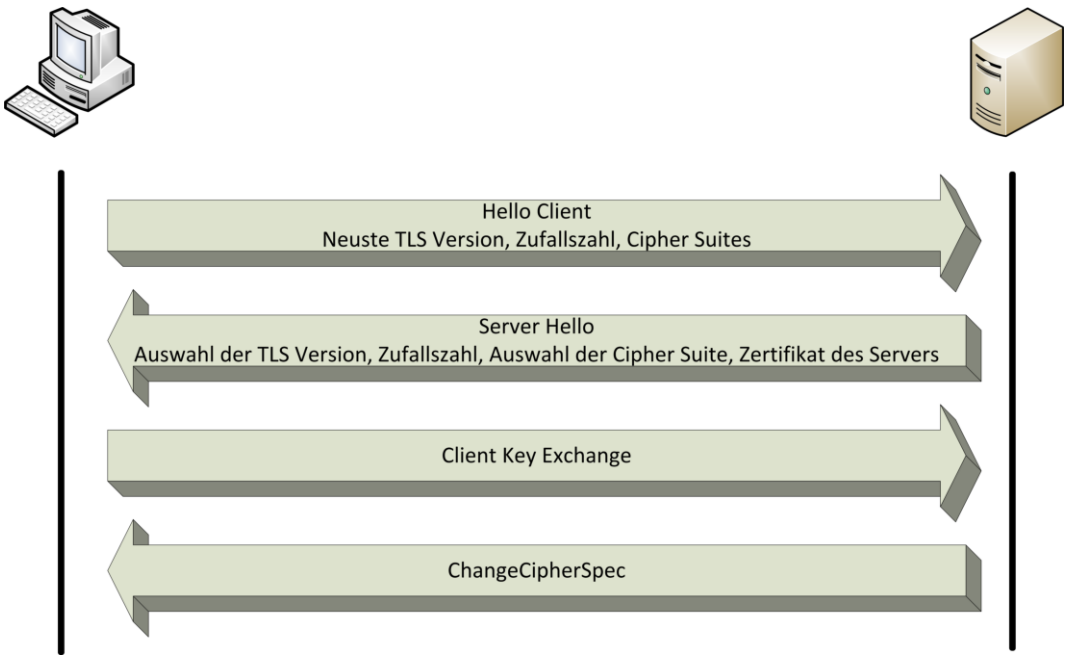


Abbildung 2.6: TLS Handshake

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	10.0.0.6	62.2.156.89	TCP	66	50083 > https [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.02042000	62.2.156.89	10.0.0.6	TCP	66	https > 50083 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1380 SACK_PERM=1 WS=32
3	0.02047600	10.0.0.6	62.2.156.89	TCP	54	50083 > https [ACK] Seq=1 Ack=1 Win=66240 Len=0
4	0.02085800	10.0.0.6	62.2.156.89	TLSv1	223	Client Hello
5	0.04046700	62.2.156.89	10.0.0.6	TCP	60	https > 50083 [ACK] Seq=1 Ack=170 Win=15680 Len=0
6	0.11987400	62.2.156.89	10.0.0.6	TLSv1	1434	Server Hello
7	0.12018000	62.2.156.89	10.0.0.6	TCP	1434	[TCP segment of a reassembled PDU]
8	0.12025300	10.0.0.6	62.2.156.89	TCP	54	50083 > https [ACK] Seq=170 Ack=2761 Win=66240 Len=0
9	0.12587400	62.2.156.89	10.0.0.6	TLSv1	1434	Certificate, Server Key Exchange
10	0.12587500	62.2.156.89	10.0.0.6	TLSv1	60	Server Hello Done
11	0.12589700	10.0.0.6	62.2.156.89	TCP	54	50083 > https [ACK] Seq=170 Ack=4143 Win=66240 Len=0
12	0.13101800	10.0.0.6	62.2.156.89	TLSv1	252	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
13	0.23156200	62.2.156.89	10.0.0.6	TLSv1	336	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
14	0.23215600	10.0.0.6	62.2.156.89	TLSv1	432	Application Data, Application Data
15	0.33525200	62.2.156.89	10.0.0.6	TLSv1	1434	Application Data, Application Data

Abbildung 2.7: Wireshark-Auszug TLS Handshake zu https://www.cnlab.ch

Wie auf der Abbildung 2.7 erkennbar ist, läuft der TLS Handshake wie folgt ab:

1. Der Client schickt ein «Client Hello» Paket. Darin stehen die höchste TLS Version, die er unterstützt, eine zufällig generierte Zahl und eine Auswahl von Chipher-Suites, die der Client unterstützt. Die Chipher-Suites ist eine Sammlung von Algorithmen zum Schlüsselaustausch, Verschlüsseln und Authentifizieren
2. Der Server antwortet mit einem «Server Hello» Paket. Damit bestätigt er die Version von TLS und gibt an welche Chipher-Suite er verwenden will. Zusätzlich schickt auch er eine Zufallszahl, die er generiert hat.
3. Der Server sendet nun im nächsten Paket das «Certificate Message». Dieses enthält sein Zertifikat.

4. Mit einem «Server Hello Done» Paket markiert der Server das Ende seines Handshakes.

5. Als nächstes überprüft der Client das Zertifikat des Servers, ob dieses richtig ist. Danach wiederum generiert der Client aus seiner Zufallszahl und der des Servers eine neue Zufallszahl, die dann benutzt wird um den Verkehr zu verschlüsseln. Diese Zufallszahl sendet er nun dem Server, wobei er das Paket mit dem Public Key des Servers verschlüsselt.

6. Das letzte Pakete, das noch unverschlüsselt übertragen wird, kommt vom Server und ist das «ChangeCipherSpec» Paket. In diesem teilt er dem Client mit, dass von nun an jedes von ihm geschickte Paket authentisiert ist.

2.4.2 Diffie-Hellman-Schlüsselaustausch

Gemäss dem Wikipedia-Eintrag wird der Diffie-Hellman-Schlüsselaustausch wie folgt definiert:

> Der Diffie-Hellman-Schlüsselaustausch oder Diffie-Hellman-Merkle-Schlüsselaustausch ist ein Protokoll aus dem Bereich der Kryptografie. Mit ihm erzeugen zwei Kommunikationspartner einen geheimen Schlüssel, den nur diese beiden kennen. Dieser Schlüssel wird üblicherweise verwendet, um verschlüsselte Nachrichten mittels eines symmetrischen Kryptosystems zu übertragen. [13]

Die Funktionsweise kann wie folgt erklärt werden. «Alice» und «Bob» wollen miteinander kommunizieren, haben jedoch nur ein unsicheres Medium zu Verfügung. Damit eine verschlüsselte Verbindung möglich ist, brauchen Beide einen gemeinsamen Schlüssel.

Hierfür benötigen sie beim Diffie-Hellmann Schlüsselaustausch eine Primzahl p und eine Primitivwurzel g (modulo p). Diese Zahlen sollen so gewählt werden, dass 2 <= g <= p-2 gilt. Diese zwei Zahlen müssen nicht geschützt werden und können somit auch über ein unsicheres Medium übertragen werden.

Zusätzlich sind zwei Zufallszahlen (a und b) nötig. Jeder Kommunikationspartner hat eine Zufallszahl, welche geheim ist. Diese Zahlen sollten deshalb nicht übertragen werden.

Mit diesen vier Zahlen können zwei Berechnungen gemacht werden. «Alice» kann die Berechnung

$$A = g^a mod p \tag{2.1}$$

tätigen und «Bob» die Berechnung

$$B = g^b mod p \tag{2.2}$$

Die Resultate dürfen untereinander ausgetauscht werden.

Sobald «Alice» die Zahl von «Bob» erhalten hat, ist eine weitere Berechnung möglich:

$$K = B^a mod p \tag{2.3}$$

Sinngemäss kann «Bob» folgende Berechnung durchführen:

$$K = A^b mod p \tag{2.4}$$

Die zwei erhaltenen K sind absolut identisch und K kann somit als Schlüssel für die verschlüsselte Verbindung eingesetzt werden.

Die nachstehende Grafik in der Abbildung 2.8 zeigt ein Beispiel eines Schlüsselaustausches.

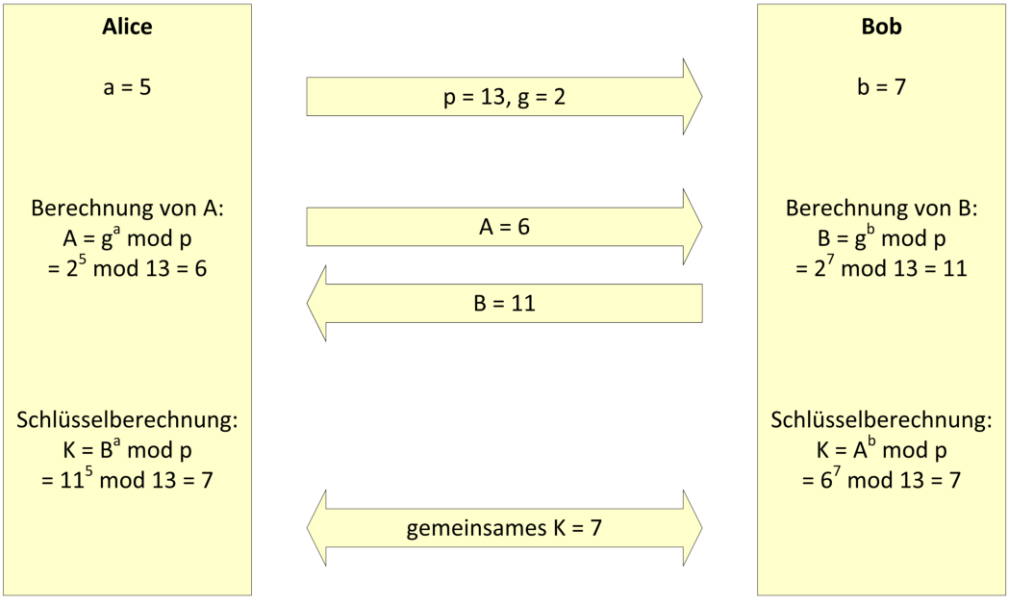


Abbildung 2.8: Ablauf Diffie-Hellman-Schlüsselaustausch

Erklärung zum Beispiel

Der Schlüsselaustausch besteht aus den folgenden Schritten:

1. «Alice» bestimmt die Zahlen p und g. Daraufhin teilt «Alice» diese Zahlen an «Bob» mit.
2. «Alice» und «Bob» bestimmen ihre eigenen Zufallszahlen (a und b).
3. Im Anschluss berechnen sie mit den gegebenen Zahlen A und B.
4. Die entsprechenden Resultate können dem Gegenüber, «Alice» beziehungsweise «Bob», mitgeteilt werden.
5. Aufgrund der erhaltenen Nummer kann der gemeinsame Schlüssel (K) berechnet werden.
6. Der errechnete und gemeinsame Schlüssel K kann dann für die weitere Verschlüsselung verwendet werden.

Über den unverschlüsselten Kanal werden also in diesem Beispiel die Zahlen 2, 13, 6 und 11 übertragen. Der gemeinsame Key (7) wird nicht übertragen, sodass es geheim bleibt.

Zusätzlich ist anzumerken, dass es sich bei diesem Beispiel um einfache Zahlen handelt. In der Praxis werden Zahlen verwendet mit bis zu mehreren hunderten Stellen. Hinzu kommt, dass der Schlüsselaustausch nicht mehr sicher ist, wenn sich ein Angreifer als Man-in-the-Middle dazwischen schalten und Pakete abändern kann.

Dieser Abschnitt zum Diffie-Hellman-Schlüsselaustausch wurde in Anlehnung an den Wikipedia-Artikel geschrieben. Auch das Beispiel wurde von dort übernommen.

2.5 Zertifikat

Ein Public-Key-Zertifikat ist ein elektronisches Dokument, das verifiziert, dass ein Public-Key zu einer Person oder Firma gehört. In der Abbildung 2.9 ist als Beispiel das Zertifikat von cnlab gezeigt. Wenn man die Webseite https://www.cnlab.ch aufruft bekommt man dieses Zertifikat, man kann also annehmen, dass dieses tatsächlich vom cnlab kommt. Ein gefälschtes Zertifikat stellt die Abbildung 2.10 dar. Dieses wurde vom WebScarab erstellt.

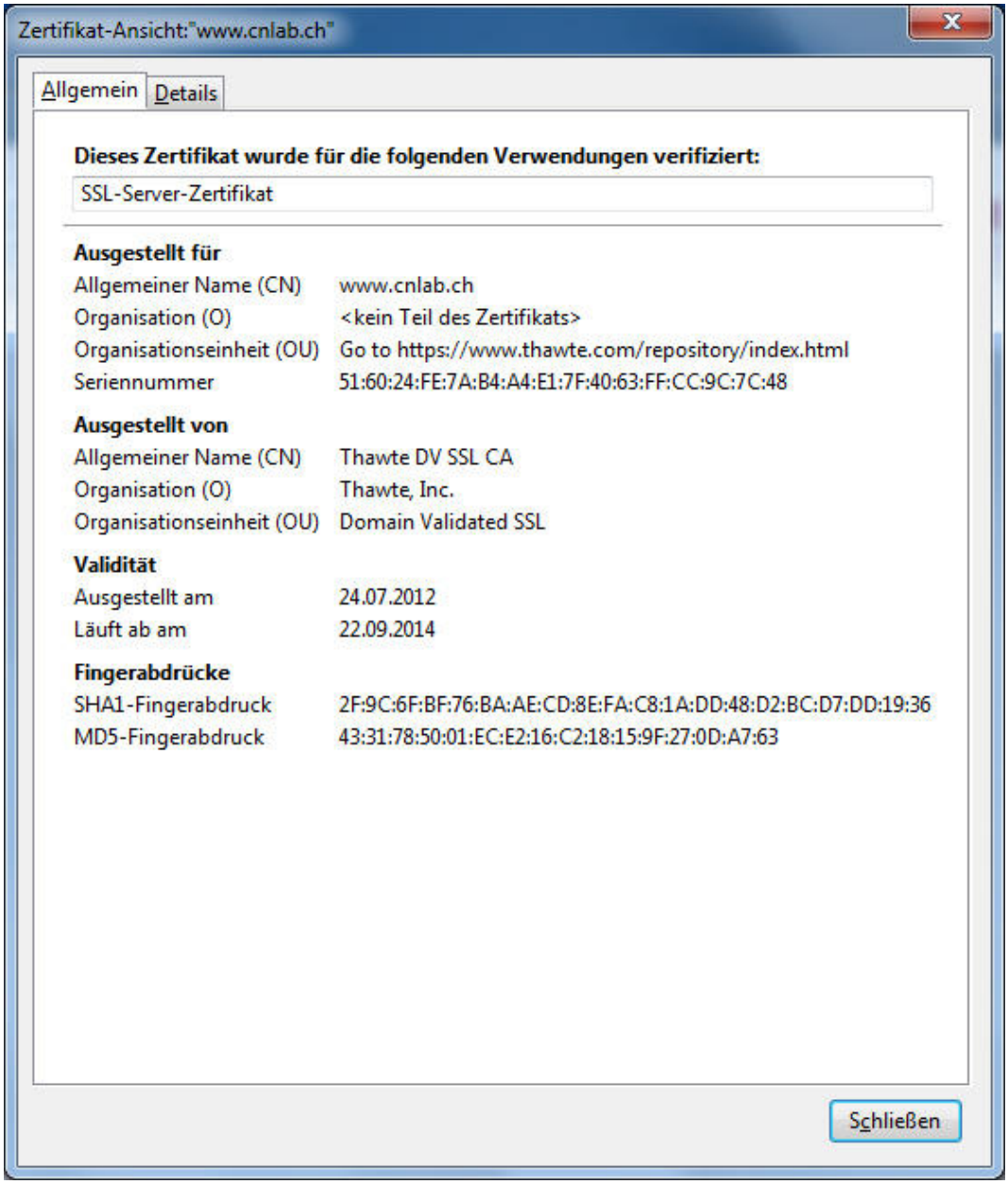


Abbildung 2.9: Zertifikat von https://www.cnlab.ch

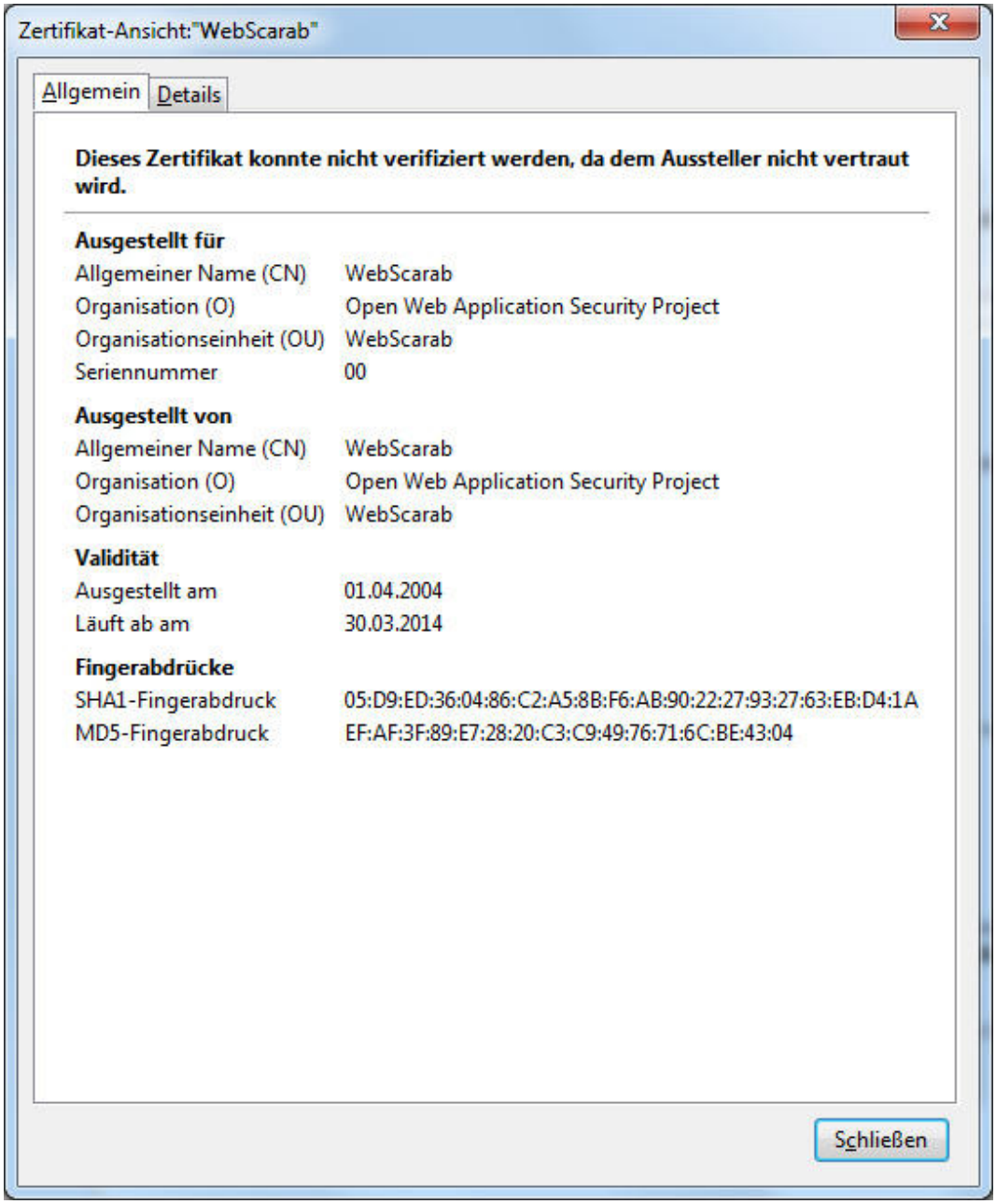


Abbildung 2.10: Gefälschtes Zertifikat von WebScarab

2.6 Beschreibungen der untersuchten Apps

In den nachfolgenden Abschnitten werden die untersuchten Apps aufgelistet. Als Hilfestellung für die Beschreibungen wurde der «App Store» [1] beziehungsweise der «Google Play

Store» [5] verwendet.

Google Latitude

Mit Google Latitude auf dem Mobiltelefon kann geprüft werden, wo sich Freunde aufhalten. Die Standorte von Freunden und Familie sind auf einer Karte zu sehen. Dies ermöglicht es, mit diesen Personen ganz einfach in Kontakt zu bleiben.

	iOS	**Android**
Version	2.2.1	Funktion von Google Maps
Voraussetzung	iPhone, iPod touch und iPad (iOS 4.0)	-

Der Grund für die Analyse dieses Apps ist unter anderem, dass es zu den meist geladenen Apps gehört. Zusätzlich interessiert, wie viele Daten Google wirklich sammelt, das heisst, wie oft ein Standort ohne Wissen des Smartphone Benutzers abgefragt wird.

20 Minuten

Diese App bietet 24 Stunden am Tag die News von der Zeitung 20 Minuten an.

	iOS	**Android**
Version	6.5	2.2.2
Voraussetzung	iPhone, iPod touch und iPad (iOS 4.3)	Android 2.2

Die Wahl fiel auf dieses App, weil es eines von vielen News Apps ist und sehr häufig heruntergeladen wird.

Dropbox

Dropbox bietet einen kostenlosen Service an um Fotos, Dokumente und Videos mit Freunden zu teilen.

	iOS	**Android**
Version	1.5.7	2.2
Voraussetzung	iPhone, iPod touch und iPad (iOS 4.0)	-

Dropbox ist eines der meist geladenen Apps für Datenaustausch. Es ist aber fraglich, ob die Entwickler auch ein Augenmerk auf die Sicherheit gelegt haben.

Facebook

Das App bietet die Möglichkeit die Funktionen der Social Media Plattform bequem unterwegs auf dem Smartphone zu benutzen

	iOS	**Android**
Version	5.3	1.9.11
Voraussetzung	iPhone, iPod touch und iPad (iOS 4.3)	-

Als die beliebteste Social Media Plattform ist es besonders interessant, die App dazu zu analysieren. Wie viel Wert legt Facebook auf die Sicherheit der App?

Bad Piggies

Bad Piggies ist ein App von den Machern von Angry Birds. Es ist ein Spiel in dem es darum geht, abhängig vom gewählten Spiellevel, ein mehr oder weniger schwieriges Fahrzeug zu bauen.

	iOS	**Android**
Version	1.1.0	1.0.0
Voraussetzung	iPhone, iPod touch und iPad (iOS 4.0)	Android 2.2

Um die Wahl der Apps mit einem Game abzurunden, fiel die Wahl auf das aktuell beliebteste Game im AppStore. Wie wird Werbung übertragen und wie oft tätigt diese App eine Verbindungen ins Internet?

Zattoo

Zattoo ist ein Live TV App um über 100 Fernsehsender live sehen zu können.

	iOS	**Android**
Version	1.3.7	1.0.8
Voraussetzung	iPhone, iPod touch und iPad (iOS 4.0)	Android 2.1

Die Frage die sich hier stellt ist, sammelt diese App Daten für die Auswertung von Benutzergeräten? Und wie viel Traffic wird dadurch und durch das Streamen verursacht?

Google Plus

Mit Google Plus hat man die Möglichkeit die Social Platform von Google direkt auf dem Smartphone zu haben.

	iOS	**Android**
Version	3.4	3.2.0.3
Voraussetzung	iPhone, iPod touch und iPad (iOS 5.0)	-

Um für das Facebook App einen Vergleich zu haben, wurde Google Plus gewählt. Inwiefern unterscheiden sich die beiden Apps in der Sicherheitsfrage und welche der beiden Apps verursacht mehr Datenmenge.

Waze

Waze ist ein gemeinschaftsbasiertes Verkehrs- und Navigations-App, mit welchem Verkehrsdaten, Strassenberichte, Unfälle und vieles mehr ausgetauscht werden können.

	iOS	**Android**
Version	3.5.1	3.5.1.4
Voraussetzung	iPhone, iPod touch und iPad (iOS 4.3)	2.0 oder höher

Mit Waze wird GPS ständig im Hintergrund gebraucht. Damit wird die Position vom Smartphone die ganz Zeit mitgeteilt. Die Frage ist nun, was wird da genau an Information ausgetauscht und inwiefern werden persönliche Daten verschickt.

iOf

iOf ist ein App für Angehörige der Schweizer Armee. Dieser App sind unter anderem diverse Abkürzungen, taktische Begriffe, Reglemente, WK-Daten und Lieder zu entnehmen.

	iOS	**Android**
Version	1.3.9	-
Voraussetzung	iPhone, iPod touch und iPad (iOS 4.3)	-

Herr Klomp hat in Auftrag gegeben, dass er dieses App gerne auf die Sicherheit untersucht haben möchte.

Raiffeisen

Raiffeisen bietet für seine Kunden die Möglichkeit, Informationen zu Raiffeisen-Bancomaten und -Banken stehts in der Nähe zu haben. Ebenso bietet Raiffeisen die Möglichkeit, «Mobile Banking» zu nutzen. Durch «Mobile Banking» können sämtliche Funktionen des E-Bankings auch unterwegs genutzt werden.

	iOS	**Android**
Version	1.12	1.0.4
Voraussetzung	iOS 3.1 oder höher	2.1 oder höher

Der Zugriff auf das E-Banking mit einem mobilen Gerät kann als sehr heikel eingestuft werden. Schliesslich sollten die persönlichen Vertragsnummer oder das Kennwort nicht an Dritte gelangen. Ebenfalls sollte eine Zahlung nicht manipuliert werden können durch einen Man-in-the-Middle-Angriff. Aus diesem Grund sollte geprüft werden, wie die Raiffeisen App diese heiklen Bankdaten übermittelt.

Kapitel 3

Versuche

Da es sich bei dieser Studienarbeit um eine Analysearbeit handelt, mussten zuerst einige Versuche gemacht werden. So musste zum Beispiel herausgefunden werden, welcher Proxy eingesetzt werden sollte. Untersucht wurden die Proxy's von Burp und WebScarab. Nähere Informationen zu diesen Versuchen befinden sich im Anhang.

Im nachfolgenden Abschnitt wird zuerst ein Vergleich zwischen den zwei untersuchten Proxy's gemacht. Danach folgen weitere Abschnitte zu weiteren Versuchen mit dem iPad und dem iPhone.

3.1 Direkter Vergleich Burp Proxy und WebScarab

Im diesem Abschnitt werden die Vor- und Nachteile zwischen dem Burp Proxy und WebScarab aufgezeigt sowie der Entscheid getroffen, welches Tool für die Studienarbeit verwendet wird. Dabei werden beide Proxy's kurz erklärt.

Beide Tools bieten die Möglichkeit auf einem Computer einen Proxy zu betreiben. Dieser Proxy löst sowohl unverschlüsselte HTTP-Verbindungen wie auch verschlüsselte HTTPS-Verbindungen auf. Für letzteres muss das entsprechende Zertifikat auf dem Gerät (beispielsweise einem iPhone) installiert werden.

Burp Proxy

Wie sich in einem ersten Versuch zeigte, lässt sich der Burp Proxy leicht bedienen und ist schnell einsatzbereit. Der Proxy selbst wirkt aber im praktischen Einsatz etwas träge, sodass die Anfragen erst mit Verzögerungen beantwortet wurden. Ebenso wird bei jedem Starten von Burp eine neue Seriennummer gesetzt. Diese Nummer kommt bei den Zertifikaten zum Zuge. Standardmässig sendet der Burp Proxy bei verschlüsselten Verbindungen immer ein Zertifikat lautend auf den Namen der aufgerufenen Webseite. Dieses Zertifikat wird jedoch bei jedem Starten von Burp wieder neu generiert und ist somit nicht identisch zu vorhergehenden. Das Zertifikat unterscheidet sich in der Seriennummer. Da der Fingerprint eines Zertifikates auf allen Angaben beruht, ist es dann völlig anders.

Bei der Verwendung von Burp muss jedes mal das «Certification Authority (kurz CA)»-Zertifikat von PortSwigger exportiert werden und auf dem Endgerät (beispielsweise einem

iPhone) installiert werden. PortSwigger wird als Zertifizierungsstelle des Burp Proxy verwendet. Dieser Export und Import kann als deutlicher Nachteil für die in dieser Arbeit geplanten Untersuchungen angesehen werden.

Um die Daten genauer zu analysieren, muss eine zusätzliche Applikation eingesetzt werden, die selbst programmiert werden kann. Um diese weitere Applikation zu nutzen, ist es notwendig die Daten in einem sinnvollen Format abspeichern zu können. Der Burp Proxy erlaubt dies, mittels Export der History-Einträge als XML. Dabei werden diverse Informationen wie HTTP-Methode (GET oder POST), Inhalt, User Agent und weitere Informationen abgespeichert. Die generierte Datei kann so schnell mehrere Megabytes betragen (je nach dem welche Datenmengen heruntergeladen wurden).

Gemäss der Aufgabenstellung wird gewünscht, mehrere Geräte abhören zu können und entsprechend zu analysieren. Dies scheint möglich zu sein, wenn die Tabelle im Burp Proxy ausgewertet wird. Die Einträge beinhalten die Angabe, auf welchem Port des Proxys das Paket übermittelt wurde. So wäre eine Trennung mittels unterschiedlichen Portnummern möglich. Dies ist zwar eher eine Poor-Man-Solution aber dennoch möglich.

Diese Überlegungen bringen jedoch nicht viel. Leider sind die notwendigen Angaben in der Exportdatei nicht erwähnt. In der XML-Datei sind lediglich die Angaben zur HTTP-Anfrage und der entsprechenden Antwort ersichtlich. Soll die Datei analysiert werden, müssten die einzelnen XML-Tags ausgelesen werden. Zusatzinformationen wie die Portnummer des Proxys werden weggelassen. So besteht keine Möglichkeit, unterschiedliche beziehungsweise mehrere Geräte auseinander zu halten.

Für den Einsatz in der Studienarbeit ist der Burp Proxy weniger geeignet. Ebenso ist der Export und das Einlesen der entsprechenden Daten für den Benutzer aufwendig und es entstehen schnell Fehler, da dies manuell gemacht werden muss. Aus diesem Grund wird der WebScarab genauer betrachtet.

WebScarab

WebScarab ist wie der Burp Proxy ein in Java realisiertes Programm. Die Verwendung ist ähnlich einfach, sie bietet jedoch diverse Zusätze, welche bei einem unerfahrenen Benutzer zu Verwirrungen führen könnten.

WebScarab bietet jedoch den entscheidenden Vorteil, dass er den Export der übermittelten Daten zulässt. So können sämtliche Pakete auf der Festplatte des Computers gespeichert werden. Genauer gesagt wird für jeden Request eine Datei erstellt. Die entsprechende Response wird ebenfalls in einer zusätzlichen Datei abgespeichert. So entstehen also pro Request und Response je eine Datei.

Die einzelnen Dateien enthalten sowohl die HTTP-Requests wie auch -Responses im Klartext. Bei den Response-Dateien trifft dies nicht immer ganz zu, da oft auch Bilder oder Ähnliches via HTTP heruntergeladen werden.

Zusätzlich bietet WebScarab die Möglichkeit, die Dateien fortlaufend zu erstellen. Wird also eine Webseite aufgerufen, werden so beispielsweise zehn Dateien erstellt. Wird zu einem späteren Zeitpunkt erneut eine andere Webseite aufgerufen, werden weitere Dateien erstellt. Die Resultate müssen also nicht jeweils erneut exportiert werden, sondern können direkt von der Festplatte beziehungsweise vom Speicherort nachgeladen werden.

Ebenfalls speichert WebScarab beim HTTP-Request die Information, von welcher IP-Adresse die Anfrage kommt. Konkret wird im HTTP-Request der Header «X-Forwarded-For» gesetzt. Dieser beinhaltet bei WebScarab die IP-Adresse des Endgerätes. Dies erlaubt auf einfache Weise die Unterscheidung der Geräte.

Fazit

Auf der nachfolgenden Tabelle (3.1) werden nochmals die wichtigsten Eigenschaften der beiden Programme im Vergleich gezeigt:

	Burp Proxy	**WebScarab**
Portabel	Ja (Java)	Ja (Java)
Proxy Funktionalität	Ja	Ja
mögliche Zertifikatseinstellungen	Self-signed Zertifikat Generiertes CA-signed Zertifikat pro Host Generiertes CA-signed Zertifikat für einen bestimmten Hostnamen Benutzerdefiniertes Zertifikat (PKCS12)	Vordefiniertes Zertifikat Benutzerdefiniertes Zertifikat (PKCS12)
Log der Request / Responses	Ja Folgende Werte werden aufgelistet: ID, Host, Methode, Parameter, Mod, Status, Länge, MIME Typ, Extension, Titel, Kommentar, SSL, IP, Cookies, Zeitpunkt, Port des Proxys	Ja Folgende Werte werden aufgelistet: ID, Zeitpunkt, Methode, Host, Path, Parameter, Status, Origin, Possible Injection, XSS, CRLF, Set-Cookie, Cookie, Kommentar, Skript
Detaillierte Ansicht Request / Response	Ja	Ja
Exportfunktionalität	Ja, aber nur mit eingeschränkten Daten	Ja, Request und Response werden im «Klartext» abgespeichert
Exportformat	Eine XML-Datei	Pro Request und Response je eine Textdatei

Tabelle 3.1: Übersichtstabelle: direkter Vergleich zwischen dem Burp Proxy und WebScarab

Abschliessend lässt sich sagen, dass es für diese Arbeit WebScarab besser geeignet ist. So kann eine Applikation erstellt werden, welche die Dateien, beispielsweise in regelmässigen Abständen, ausliest und die Ergebnisse fortlaufend ergänzt. Fehler durch einen falschen oder unvollständigen Export können so vermieden werden.

3.2 «Grundrauschen» iPad

In einem weiteren Versuchsaufbau wurde das «Grundrauschen» eines iPads beim Werkszustand analysiert. Das «Grundrauschen» bezeichnet dabei den Netzwerkverkehr, welcher ständig oder in regelmässigen Abständen, ohne spezielle Einwirkung des Benutzers, eintritt. Der Grund für diesen Versuch liegt darin ein Muster im Verhalten des Netzwerkverkehrs herauszufinden. Wird beispielsweise jede Stunde bei Apple nachgefragt, ob ein Update verfügbar ist, so würde stündlich Verkehr erzeugt. Bei einer genauen Analyse könnte dieser regelmässige Verkehr bei der Interpretation miteinbezogen werden.

Um einen solchen Versuch durchzuführen, wurde das iPad auf die Werkseinstellungen zurückgesetzt und die ersten Schritte ausgeführt (Sprachwahl, Länderauswahl,...). Anschliessend musste die WLAN-Verbindung eingerichtet und der Proxy angegeben werden. Als Proxy wurde ein Windows 7 Computer mit laufendem WebScarab verwendet. Zusätzlich hat Wireshark sämtlichen Verkehr aufgezeichnet, welcher über den Proxy lief.

In der Tabelle 3.2 sind die wichtigsten Komponenten dieses Versuches aufgelistet.

IP-Adresse / Subnetzmaske	**Bezeichnung**
10.0.0.1/24	WLAN-Router Netgear Wireless Router WNR1000
10.0.0.3/24	iPad 16 GB Version: 5.1.1 (9B206), Modell: MB292FD
10.0.0.6/24	Windows 7 Computer mit installiertem WebScarab

Tabelle 3.2: Versuch «Grundrauschen» iPad: Netzkomponenten

Bis zu diesem Zeitpunkt wurde keine zusätzliche Applikation geöffnet oder installiert. Der Versuch startete am Mittwochabend (10. Oktober 2012) und endete am Donnerstagmorgen (11. Oktober 2012).

Wie die WebScarab Übersicht am Donnerstagmorgen zeigte, hat es keine Einträge gegeben. Das könnte darauf schliessen lassen, dass das iPad auch wirklich nichts gemacht hat. Da dies aber auf den ersten Blick nicht sehr glaubwürdig scheint, wurden die Wireshark Ergebnisse analysiert. Hierfür wurden die Endpoints genauer betrachtet. Wireshark zeigt in ihrer Übersicht sämtliche Adressen auf, welche in einem aufgezeichneten Paket als Sende- oder Empfängeradresse vermerkt waren. Die nachstehende Abbildung 3.1 zeigt diese Übersicht.

Wie oben erwähnt benutzt das iPad die IP-Adresse «10.0.0.3». Schaut man die Übersicht genau an, kann man diese IP-Adresse gar nicht finden. Dies zeigt sich auch beim Setzen des Filters («ip.addr == 10.0.0.3»). Es werden keine Pakete aufgelistet. Dies wiederum bestätigt die Anzeige des Proxys.

Selbstverständlich ist dann die Frage aufgekommen, ob etwas falsch konfiguriert wurde. Beispielsweise, dass der Proxy nicht richtig eingegeben wurde oder er gar nicht gestartet wurde. Diese Annahme wurde dann aber schnell widerlegt, als auf dem iPad der Safari gestartet wurde und die «www.google.ch»-Website problemlos geladen werden konnte. Entsprechende Einträge waren sowohl im WebScarab wie auch Wireshark vorhanden.

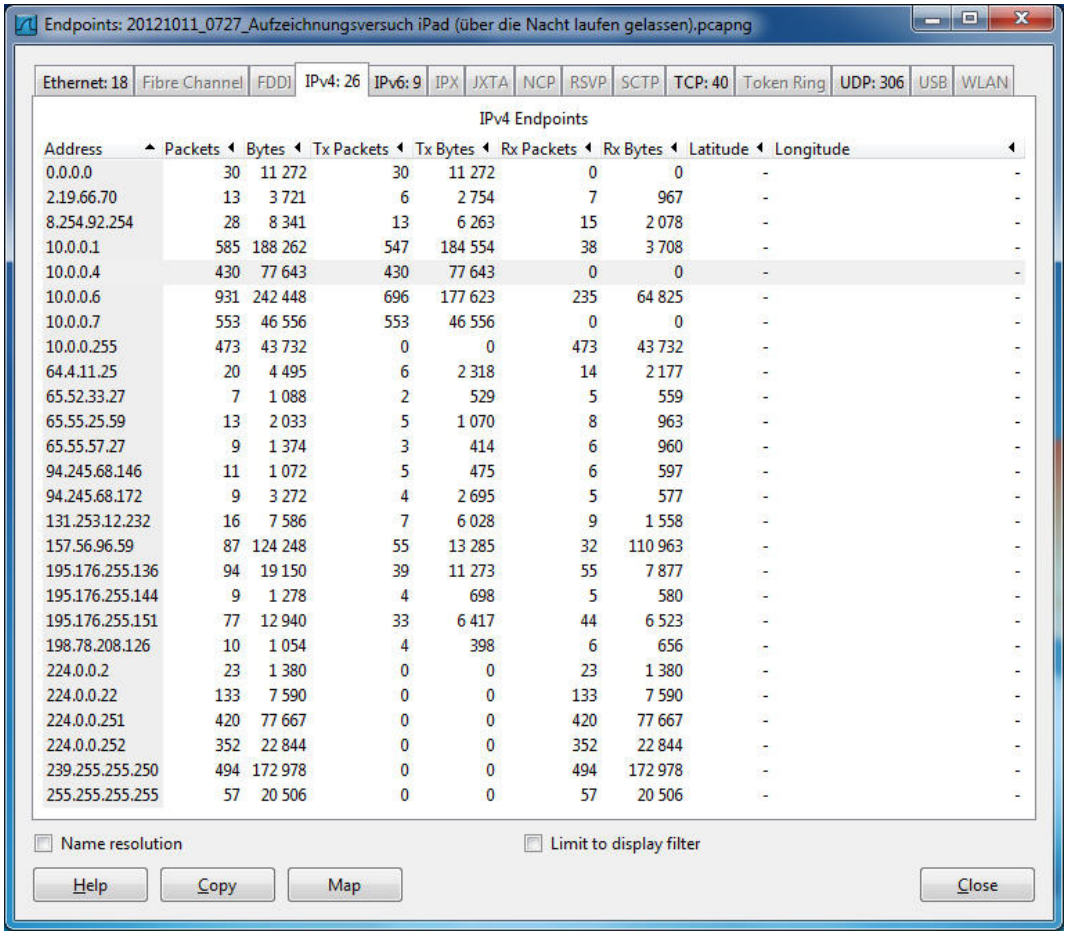


Abbildung 3.1: Übersicht der Endpoints in Wireshark

Fazit

Der erste Test betreffend dem «Grundrauschen» eines zurückgesetzten iPad's zeigte, dass keine Kommunikation zu einem Server des Herstellers gemacht wurde. In konkreten Fall wurden also keine Pakete an einen Server von Apple gesandt. Natürlich wurde bei diesem Versuch nicht geprüft, welche Daten während dem Konfigurationsassistenten genau gemacht wurden. Es kann davon ausgegangen werden, dass sich dies auf ein Minimum von Daten beschränkt.

Ebenfalls ist nicht klar, was passiert, wenn dieser Test über eine längere Zeitdauer durchgeführt würde. Es besteht die Möglichkeit, dass alle 24 Stunden eine Meldung gemacht wird und diese Meldungen beim Testversuch «verpasst» wurden.

Somit sind die gewonnenen Erkenntnisse mit Vorsicht zu geniessen. Sie scheinen jedoch für den aktuellen Versuchsaufbau korrekt zu sein, sodass wir es aktuell so belassen.

3.3 «Grundrauschen» iPhone

Wie bereits beim iPad wurde auch beim iPhone versucht das «Grundrauschen» festzustellen. Dafür wurde das iPhone ebenfalls auf die Werkeinstellungen zurückgesetzt. Anschliessend ist der Proxy gestartet worden. In diesem Fall ein Windows 7 Computer mit laufendem WebScarab und Wireshark.

In der Tabelle 3.3 sind die wichtigsten Komponenten dieses Versuches aufgelistet.

IP-Adresse / Subnetzmaske	Bezeichnung
10.0.0.1/24	WLAN-Router Netgear Wireless Router WNR1000
10.0.0.4/24	iPhone 3G 8 GB Version: 4.2.1 (8C148), Modell: MB504FD
10.0.0.7/24	Windows 7 Computer mit installiertem WebScarab

Tabelle 3.3: Versuch «Grundrauschen» iPhone: Netzkomponenten

Auch auf diesem iPhone wurde bis dahin keine Applikationen installiert. Der Versuch wurde am Donnerstagnachmittag (11. Oktober 2012) gestartet und lief für zwei Stunden.

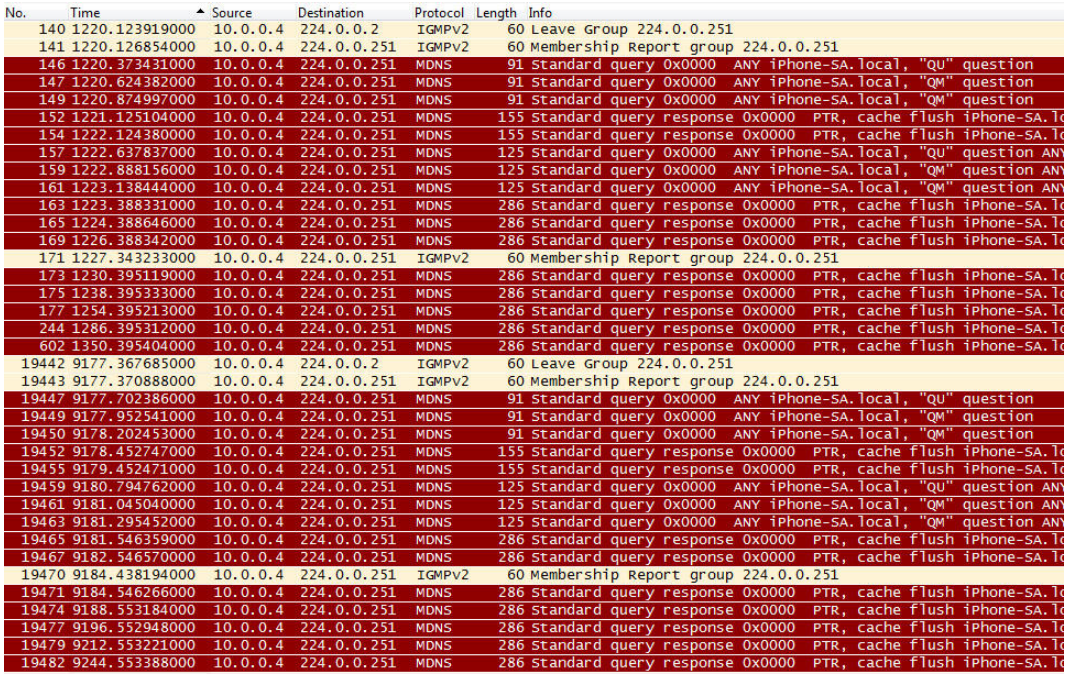
Wie erwartet blieb auch beim iPhone der Proxy leer. Es wurden also keine Daten via HTTP sowie HTTPS übertragen. Da Wireshark deutlich mehr Protokolle erkennt und auflistet, kommt es zu einer unerwarteten Diskrepanz. Im Wireshark sind diverse «Multicast DNS»-Pakete (kurz MDNS) zu sehen, die das iPhone von sich aus versendet hat. Diese stammen vom Bonjour-Dienst des iPhones, welcher dafür zuständig ist, andere Geräte im Netzwerk ausfindig zu machen, um beispielsweise Musik zu teilen. [15] In der Abbildung 3.2 sieht man diese Pakete, die das iPhone an die Multicastadresse 224.0.0.251 schickt.

Fazit

Natürlich kann sich bei der Auswertung der Daten gefragt werden, wieso das iPad nicht auch diese MDNS-Pakete verschickt. Es ist davon auszugehen, dass es bei der Bonjour-Version, die auf dem iPad ist, und mit der installierten iOS Version eine Konflikt gibt.

Der Übertragungsweg dieser MDNS-Pakete kann nur auf zwei Arten abgelaufen sein. Entweder hat das iPhone die Pakete direkt (also ohne Kenntnis des Proxys) dem WLAN-Router zugestellt, welcher die Pakete an alle weitergeleitet hat oder es wurde direkt an den Proxy zugestellt. Letzteres scheint jedoch unwahrscheinlicher, da WebScarab ein Proxy für HTTP- und HTTPS-Verbindungen ist.

Dies würde gleichzeitig bedeuten, dass das iPhone in der Lage ist, anderweitig Verbindungen aufzubauen. Kurz gesagt, der Proxy wird umgangen. Sollte dies der Fall sein, besteht die Möglichkeit, dass dies auch beim iPad der Fall war. Um dies genauer zu prüfen, wird ein weiterer Versuch mit sogenanntem Port-Mirroring durchgeführt.



No.	Time	Source	Destination	Protocol	Length	Info
140	1220.123919000	10.0.0.4	224.0.0.2	IGMPv2	60	Leave Group 224.0.0.251
141	1220.126854000	10.0.0.4	224.0.0.251	IGMPv2	60	Membership Report group 224.0.0.251
146	1220.373431000	10.0.0.4	224.0.0.251	MDNS	91	Standard query 0x0000 ANY iPhone-SA.local, "QU" question
147	1220.624382000	10.0.0.4	224.0.0.251	MDNS	91	Standard query 0x0000 ANY iPhone-SA.local, "QM" question
149	1220.874997000	10.0.0.4	224.0.0.251	MDNS	91	Standard query 0x0000 ANY iPhone-SA.local, "QM" question
152	1221.125104000	10.0.0.4	224.0.0.251	MDNS	155	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
154	1222.124380000	10.0.0.4	224.0.0.251	MDNS	155	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
157	1222.637837000	10.0.0.4	224.0.0.251	MDNS	125	Standard query 0x0000 ANY iPhone-SA.local, "QU" question ANY
159	1222.888156000	10.0.0.4	224.0.0.251	MDNS	125	Standard query 0x0000 ANY iPhone-SA.local, "QM" question ANY
161	1223.138444000	10.0.0.4	224.0.0.251	MDNS	125	Standard query 0x0000 ANY iPhone-SA.local, "QM" question ANY
163	1223.388331000	10.0.0.4	224.0.0.251	MDNS	286	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
165	1224.388646000	10.0.0.4	224.0.0.251	MDNS	286	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
169	1226.388342000	10.0.0.4	224.0.0.251	MDNS	286	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
171	1227.343233000	10.0.0.4	224.0.0.251	IGMPv2	60	Membership Report group 224.0.0.251
173	1230.395119000	10.0.0.4	224.0.0.251	MDNS	286	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
175	1238.395333000	10.0.0.4	224.0.0.251	MDNS	286	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
177	1254.395213000	10.0.0.4	224.0.0.251	MDNS	286	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
244	1286.395312000	10.0.0.4	224.0.0.251	MDNS	286	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
602	1350.395404000	10.0.0.4	224.0.0.251	MDNS	286	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
19442	9177.367685000	10.0.0.4	224.0.0.2	IGMPv2	60	Leave Group 224.0.0.251
19443	9177.370888000	10.0.0.4	224.0.0.251	IGMPv2	60	Membership Report group 224.0.0.251
19447	9177.702386000	10.0.0.4	224.0.0.251	MDNS	91	Standard query 0x0000 ANY iPhone-SA.local, "QU" question
19449	9177.952541000	10.0.0.4	224.0.0.251	MDNS	91	Standard query 0x0000 ANY iPhone-SA.local, "QM" question
19450	9178.202453000	10.0.0.4	224.0.0.251	MDNS	91	Standard query 0x0000 ANY iPhone-SA.local, "QM" question
19452	9178.452747000	10.0.0.4	224.0.0.251	MDNS	155	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
19455	9179.452471000	10.0.0.4	224.0.0.251	MDNS	155	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
19459	9180.794762000	10.0.0.4	224.0.0.251	MDNS	125	Standard query 0x0000 ANY iPhone-SA.local, "QU" question ANY
19461	9181.045040000	10.0.0.4	224.0.0.251	MDNS	125	Standard query 0x0000 ANY iPhone-SA.local, "QM" question ANY
19463	9181.295452000	10.0.0.4	224.0.0.251	MDNS	125	Standard query 0x0000 ANY iPhone-SA.local, "QM" question ANY
19465	9181.546359000	10.0.0.4	224.0.0.251	MDNS	286	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
19467	9182.546570000	10.0.0.4	224.0.0.251	MDNS	286	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
19470	9184.438194000	10.0.0.4	224.0.0.251	IGMPv2	60	Membership Report group 224.0.0.251
19471	9184.546266000	10.0.0.4	224.0.0.251	MDNS	286	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
19474	9188.553184000	10.0.0.4	224.0.0.251	MDNS	286	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
19477	9196.552948000	10.0.0.4	224.0.0.251	MDNS	286	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
19479	9212.553221000	10.0.0.4	224.0.0.251	MDNS	286	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo
19482	9244.553388000	10.0.0.4	224.0.0.251	MDNS	286	Standard query response 0x0000 PTR, cache flush iPhone-SA.lo

Abbildung 3.2: Grundrauschen des iPhones

3.4 Analyse «Port Mirroring»

In den vorhergehenden Versuchen wurde immer ein Proxy beim iPhone oder iPad angegeben. Dabei ist jedoch nicht klar, ob auch wirklich alle Datenverbindungen via Proxy aufgebaut werden oder ob Apps diesen Proxy umgehen. Letzteres würde bedeuten, dass die App die Einstellungen des Telefons ignoriert und eine direkte Verbindung versucht aufzubauen. Mit den bisherigen Versuchsaufbauen wäre dies immer möglich gewesen, da der Internetzugang ohne konkrete Authentisierung möglich ist.

Um eine genauere Analyse machen zu können, reicht ein einfacher Layer 2 Switch, welcher die Konfiguration von «Port Mirroring» erlaubt. Im konkreten Fall wurde dazu ein «HP ProCurve Switch 1700-8» verwendet. Insgesamt waren für diesen Versuchsaufbau folgende Komponenten im Einsatz (Tabelle 3.4):

IP-Adresse / Subnetzmaske	Bezeichnung
10.0.0.1/24	WLAN-Router Netgear Wireless Router WNR1000
10.0.0.6/24	BackTrack5 R3 Computer mit installiertem WebScarab
10.0.0.9/24	iPhone 4 16 GB Version: 5.1.1 (9B206), Modell: MC603FD
192.168.2.10	HP ProCurve Switch 1700-8
192.168.2.100	Windows 7 Computer mit installiertem Wireshark

Tabelle 3.4: Analyse «Port Mirroring»: Netzkomponenten

Der Versuchsaufbau selbst wird mit nachstehender Skizze (Abbildung 3.3) beschrieben:

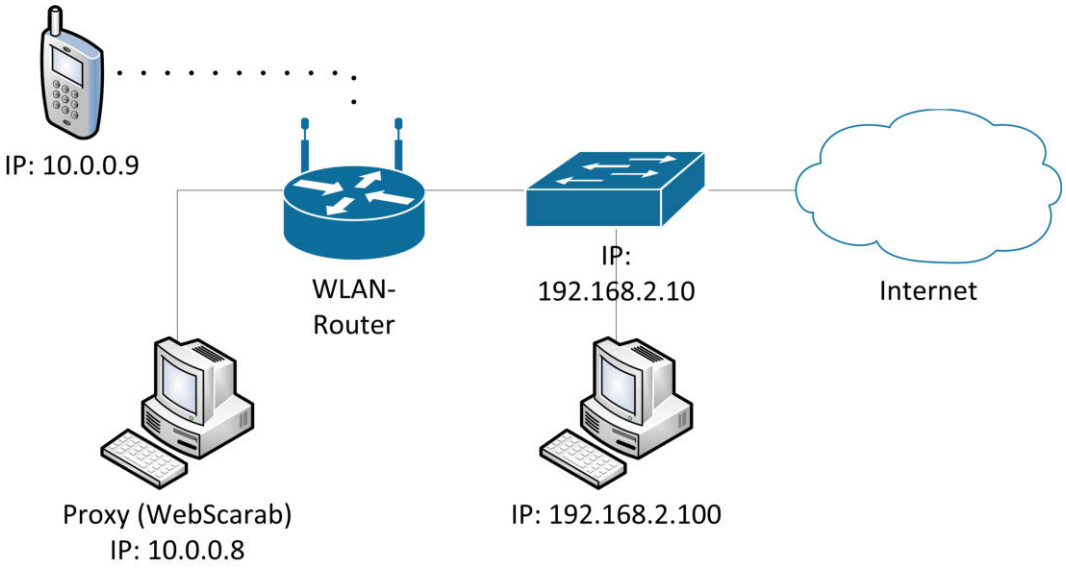


Abbildung 3.3: Analyse «Port Mirroring»: Aufbau Netzkomponenten

Wie auf der Grafik dargestellt, ist am WLAN-Router das iPhone (IP: «10.0.0.9») und der BackTrack5 R3 Computer (IP: «10.0.0.8») angehängt. Letzterer dient in diesem Versuch als Proxy. Das iPhone hat dabei die entsprechenden Angaben in den Einstellungen eingetragen. Das WAN-Interface des WLAN-Router war bei den vorhergehenden Versuchen immer direkt mit dem HSR-Netz verbunden. In diesem Versuch wird jedoch noch ein Switch (Web-Interface IP: «192.168.2.10») dazwischen gesetzt. Der WLAN-Router ist also mit dem Switch verbunden. Dieser wiederum stellt die Verbindung mit dem HSR-Netz sicher.

Um den ausgehenden Traffic des WLAN-Routers vollständig abfangen zu können, ist ein weiterer Computer (IP: «192.168.2.100») notwendig. Hierfür wird ein normaler Windows 7 Computer verwendet. Dieser ist ebenfalls am Switch angehängt.

Damit das Abfangen des Traffics möglich ist, sind einige Konfigurationsschritte notwendig. Es ist von Vorteil, wenn der verwendete Switch zu Beginn auf die Werkseinstellungen zurückgesetzt wird. So kann ein Fehlverhalten durch eine falsche und veraltete Konfiguration minimiert werden. Beim verwendeten «HP ProCurve Switch 1700-8» kann dies mittels verbinden des Port 1 und 2 erreicht werden. Nach diesem Schritt gelten die Standardeinstellungen von HP. Konkret kann das Web-Interface des Switches via «192.168.2.10» erreicht werden. Das Standardpasswort für den Login kann dabei leer gelassen werden (Abbildung 3.4).

Um überhaupt auf den Switch zugreifen zu können, benötigt der verwendete Windows 7 Computer eine IP-Adresse. Diese muss manuell konfiguriert werden, da der Switch standardmässig keine IP-Adressen verteilt. In diesem Versuch wurde dabei die IP-Adresse «192.168.2.100» fix beim Windows 7 Computer eingetragen.

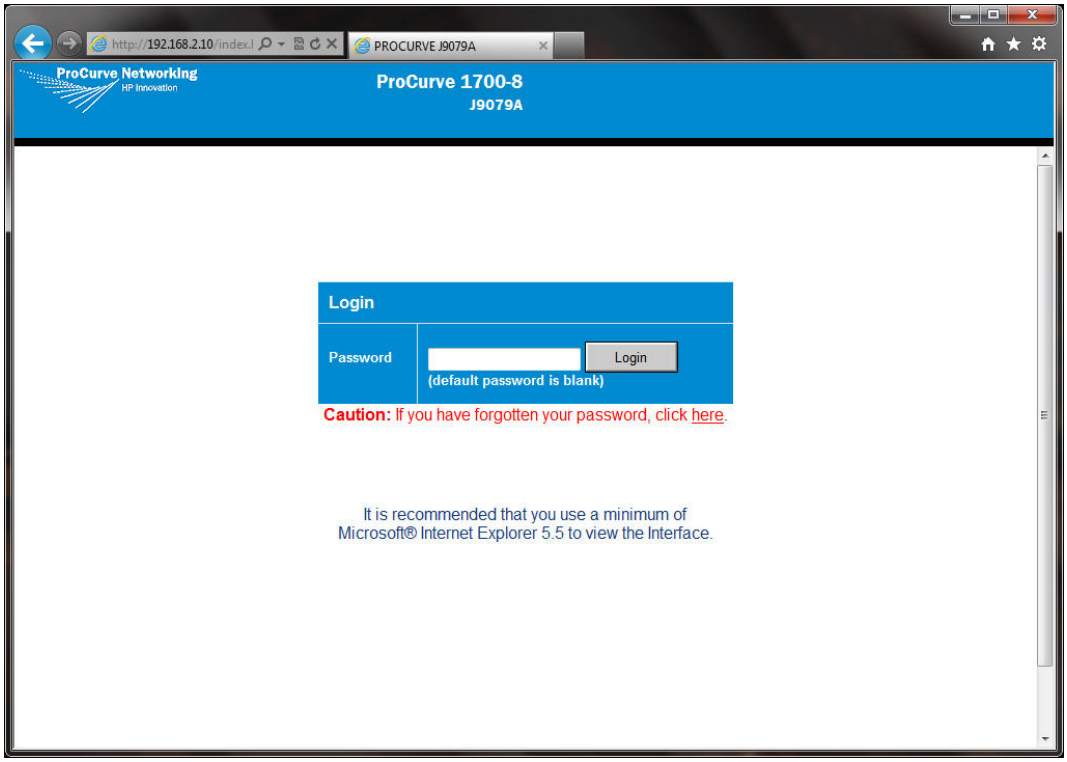


Abbildung 3.4: «HP ProCurve Switch 1700-8»: Web-Interface Login

Der Zugriff auf den Switch beziehungsweise das Web-Interface sollte nun möglich sein. Dabei erscheint nach dem Einlogen eine Übersicht mit den wichtigsten Systeminformationen (siehe Grafik in Abbildung 3.5). Die Übersicht zeigt weiter, welche Ports aktuell belegt sind. Aktuell wird Port 3 als «Up» gekennzeichnet. An diesem Port ist der Windows 7 Computer angehängt, welcher den Traffic mithören soll. Alle anderen Komponenten wurden noch nicht verbunden.

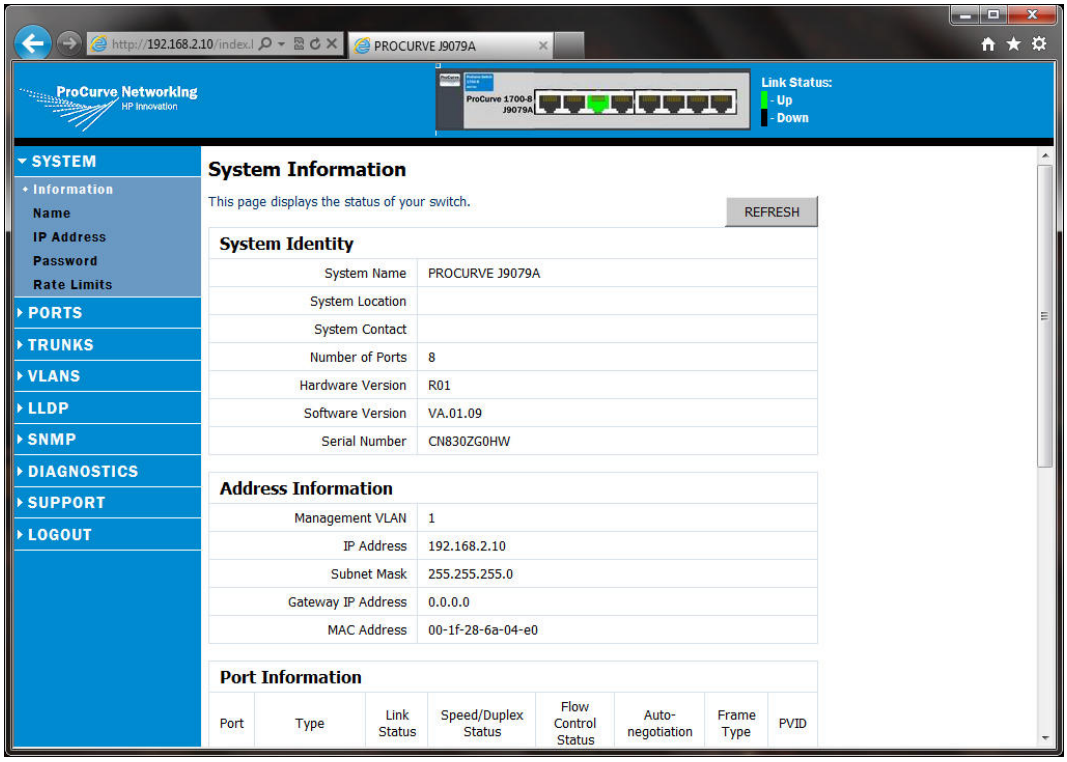


Abbildung 3.5: «HP ProCurve Switch 1700-8»: Systeminformationen

Damit der Traffic auch wirklich weitergeleitet wird, muss dies noch konfiguriert werden. Unter dem Navigationspunkt «Ports» erscheint nach dem Klicken der Link «Port Mirroring». Wird diese Seite angewählt, erscheint die Konfigurationsseite zu Port Mirroring (Abbildung 3.6).

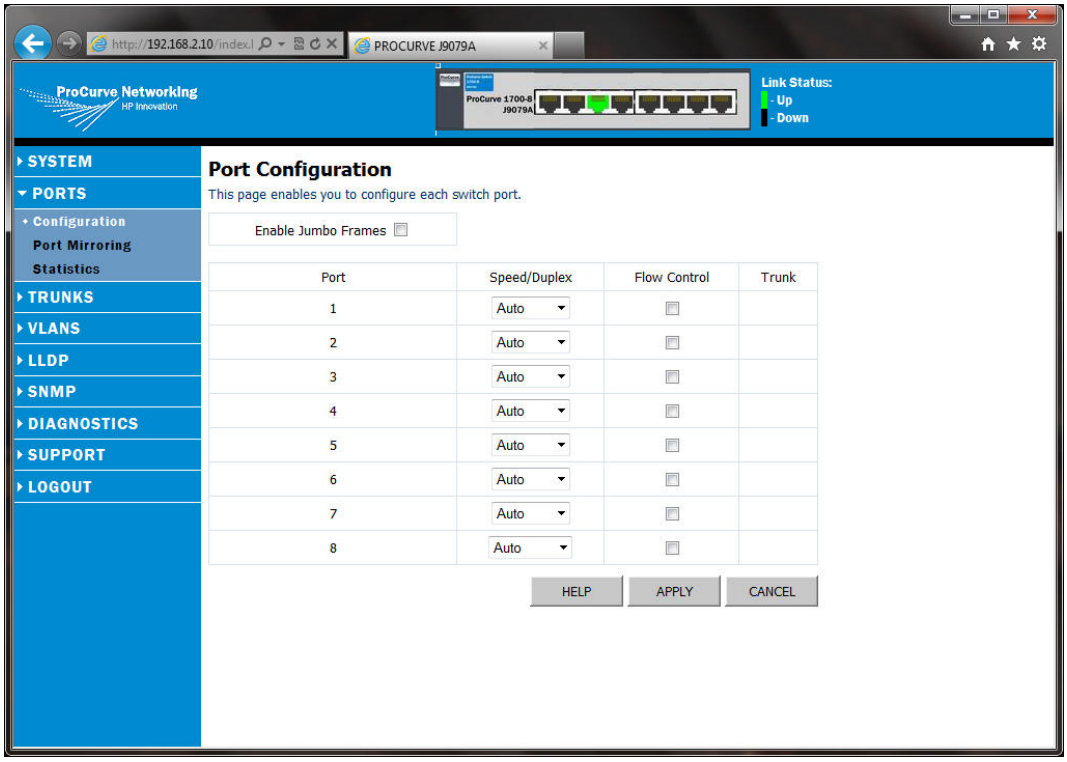


Abbildung 3.6: «HP ProCurve Switch 1700-8»: Ports

Bei diesem Versuch wurden die Ports wie folgt belegt:

- Port 1: WLAN-Router
- Port 2: HSR-Netz («Internet»)
- Port 3: Windows 7 Computer für Port Mirroring

Alle weiteren Ports werden nicht weiter benötigt. Um das Port Mirroring also realisieren zu können, muss der Traffic von Port 1 auf Port 3 gespiegelt werden. Dies ist mit folgender Konfiguration möglich (Abbildung 3.7):

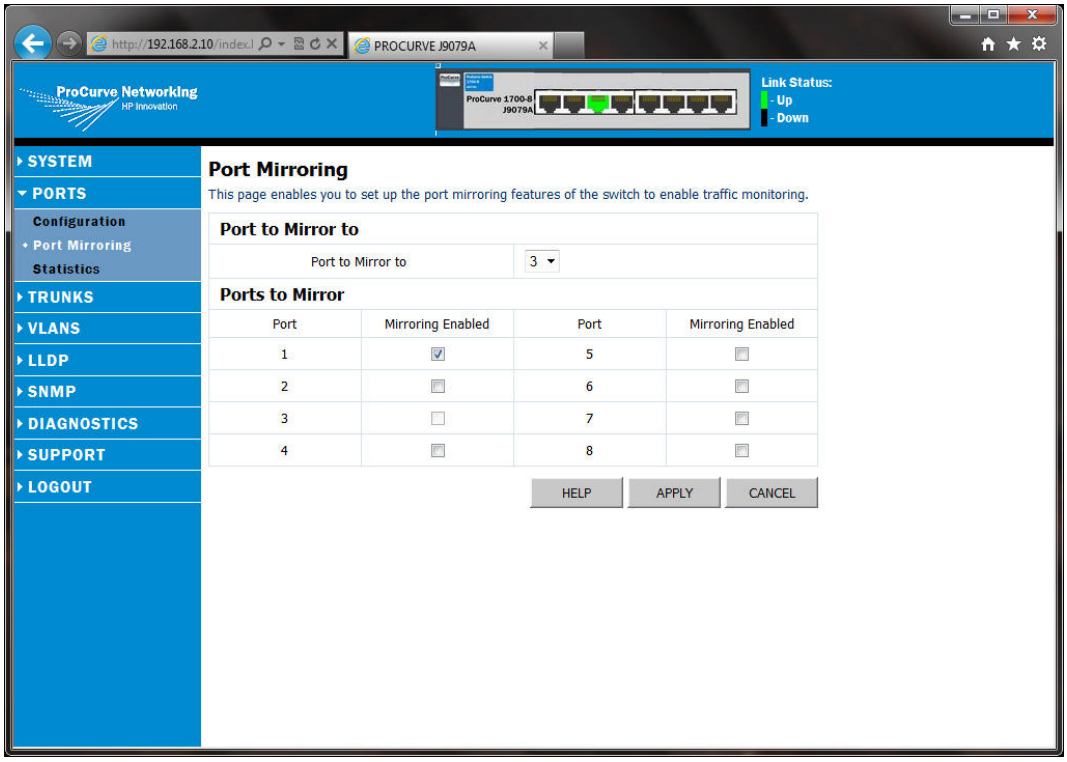


Abbildung 3.7: «HP ProCurve Switch 1700-8»: Port Mirroring

Nach dem Übernehmen der Einstellungen ist das Port Mirroring aktiv und sowohl der WLAN-Router wie auch das HSR-Netz kann nun auch an den Switch angehängt werden. Die Startseite des Web-Interface kennzeichnet daraufhin Port 1 bis 3 mit dem Link Status «up» (Abbildung 3.8).

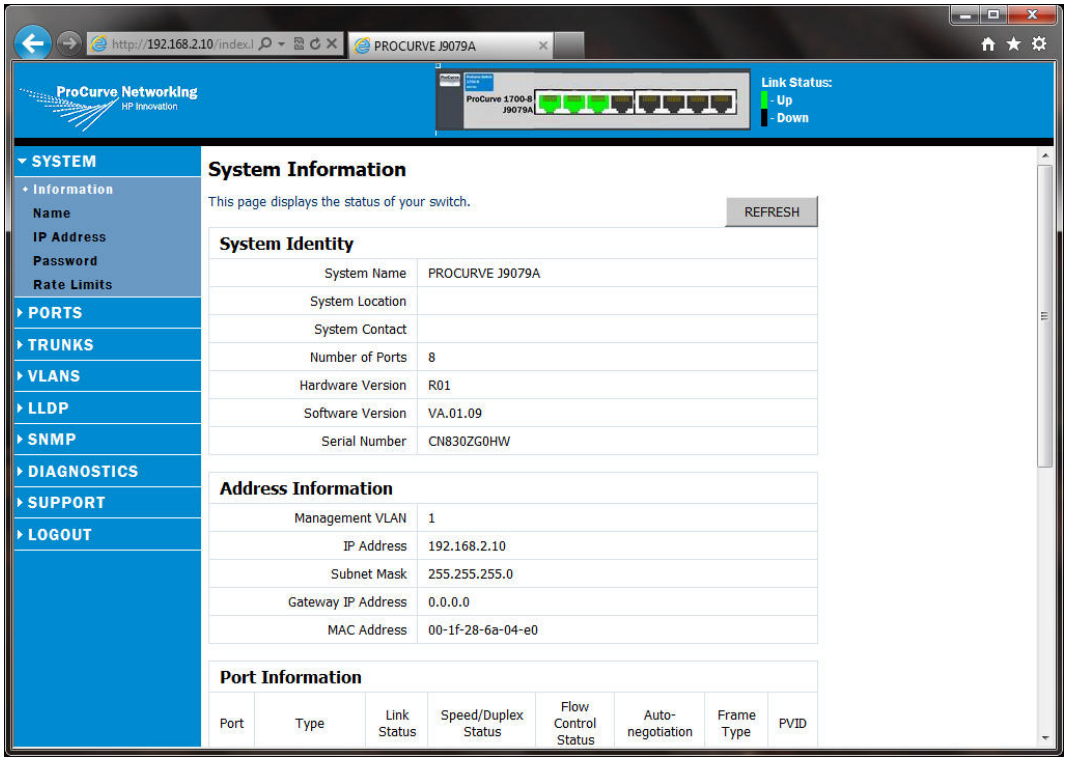


Abbildung 3.8: «HP ProCurve Switch 1700-8»: Systeminformationen

Um mit dem eigentlichen Versuch fortfahren zu können, ist eine App nötig, welche eine Verbindung zu einem Server aufbaut. Dabei sollte das verwendete Protokoll mit Vorteil weder HTTP noch HTTPS sein. Bekanntlich werden diese zwei Protokolle über den Proxy geleitet, sodass die Erkenntnis dieses Versuchs minimal wäre.

Als mögliches App kann der Speedtest von der Firma cnlab AG verwendet werden. Die App führt eine Geschwindkeitsmessung der aktuellen Bandbreite durch. Dabei wird eine TCP-Verbindung zu einem Testserver aufgebaut. Sollte die Annahme stimmen, dass eine App die Proxyangaben nicht berücksichtigt und einfach so eine Verbindung zum Server aufbaut, sollten die TCP-Pakete beim BackTrack5-System nicht sichtbar sein. Gleichzeitig sollte jedoch der zusätzliche Windows 7 Computer, welcher eine Kopie des gesamten Traffics erhält, diese TCP-Pakete sehen können.

Das App von cnlab kann auf dem AppStore gesucht und installiert werden. Nach der Installation wird beim ersten Programmstart nachgefragt, ob der aktuelle Standort verwendet werden darf. Für diesen Versuch ist diese Angabe nicht weiter relevant, sodass dem problemlos zugestimmt werden kann.

Bevor ein Test gestartet wird, muss nochmals geprüft werden, ob Wireshark sowohl auf dem BackTrack5 System läuft (und aufzeichnet) sowie auf dem Windows 7 Computer, welcher am Mirror-Port angeschlossen ist. Ist dies sichergestellt, kann der Test gestartet werden und es wird eine Geschwindigkeitsmessung durchgeführt.

Abbildung 3.9: «cnlab Speed»: Ergebnisanzeige

Nach der Durchführung des Tests empfiehlt es sich die Aufzeichnungen der beiden Wiresharks zu stoppen und die Ergebnisse zu speichern.

Werden die zwei Wireshark-Aufzeichnungen miteinander verglichen, so fällt auf, dass die Aufnahme am Mirror-Port deutlich mehr aufgezeichnet hat. Wie in der App von cnlab vermerkt wurde, wurden während dem Test 30.41 MB Daten transferiert (siehe 3.9). Diese Datenübertragung fand zwischen dem Telefon und dem Referenzserver (hier «hsi.bluewin.ch») statt. Wie eine kurze Abfrage in «nslookup» gezeigt hat, ist der Referenzserver von Bluewin unter der IP-Adresse «195.186.135.224» erreichbar.

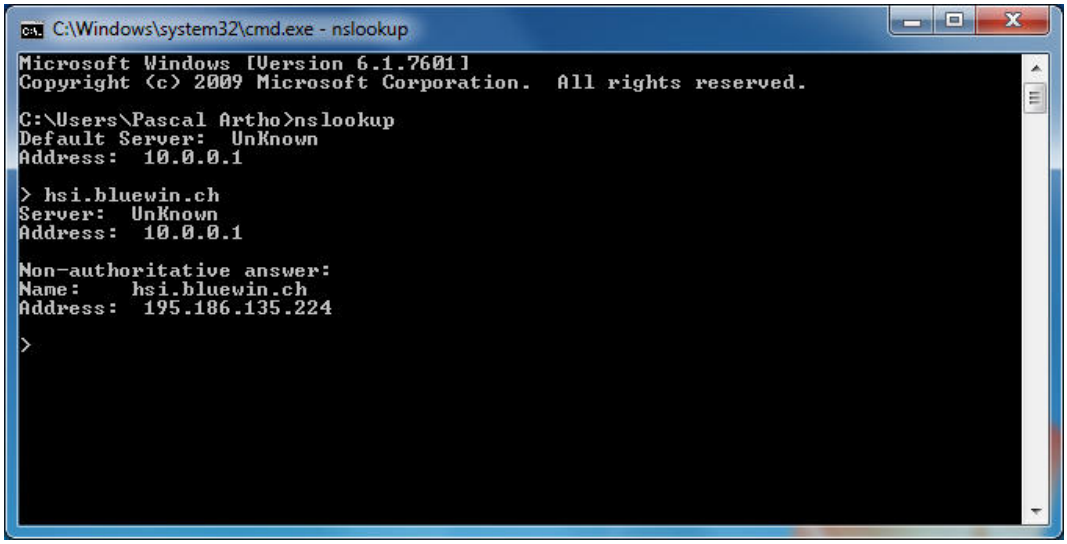


Abbildung 3.10: «nslookup» für «hsi.bluewin.ch»

In den Aufzeichnungen von Wireshark müsste also diese IP-Adresse beziehungsweise der aufgelöste Namen («hsi.bluewin.ch») als Endpoint aufgeführt sein. Im konkreten Versuch wurden folgende «TCP Endpoints» aufgelistet (siehe Abbildung 3.11 und 3.12).

> *Diese Screenshots können sich bei weiteren Versuchen ändern und es müssen nicht sämtliche Adressen aufgelistet werden. Die Einträge sind abhängig von der vorhandenen Netzwerkinfrastruktur, sowie der Aktivität im Netzwerk.*

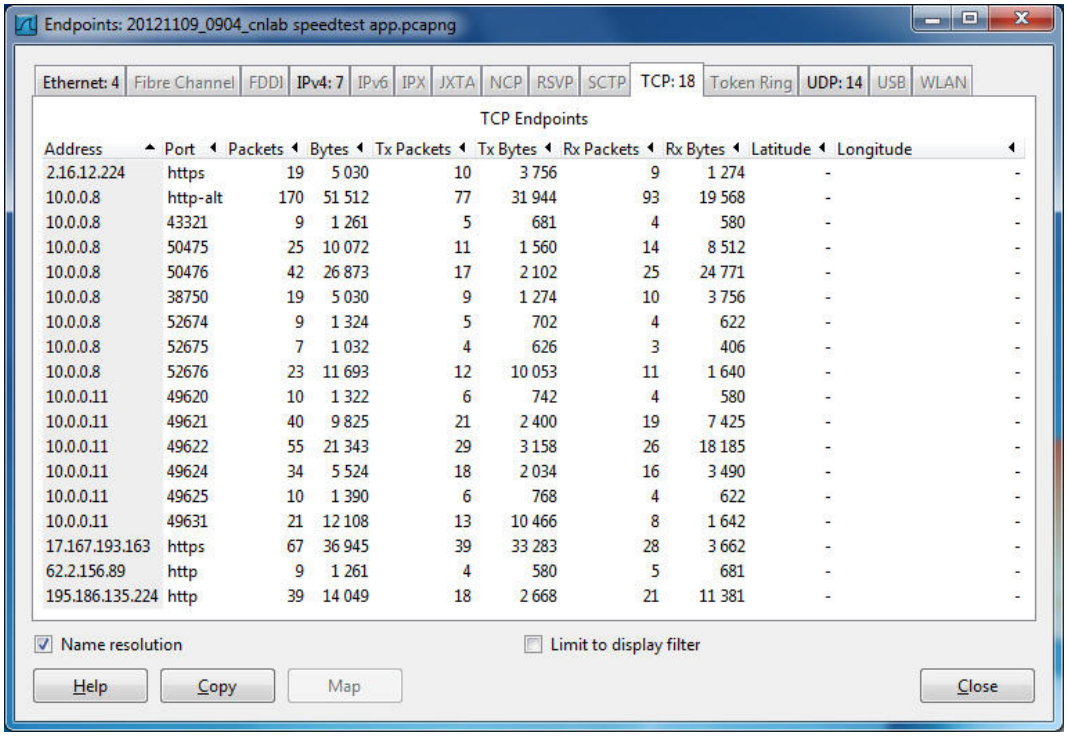


Abbildung 3.11: Wireshark: Übersicht «TCP Endpoints» beim Proxy

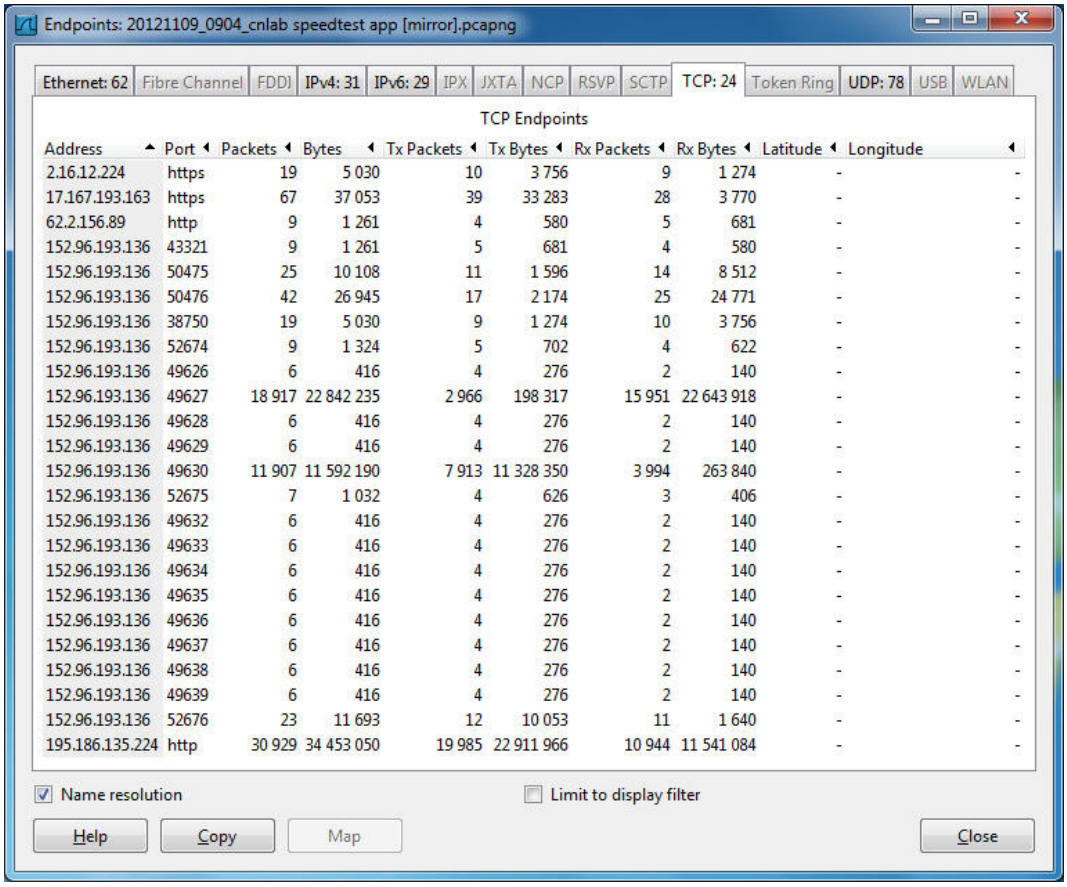


Abbildung 3.12: Wireshark: Übersicht «TCP Endpoints» beim Mirror Port

In beiden Anzeigen war der Eintrag der IP-Adresse «195.186.135.224» vorhanden. Diese werden nun nebeneinander aufgeführt, um einen konkreten Vergleich zu machen:

	Aufzeichnung Proxy	**Aufzeichnung Mirror-Port**
Address	195.186.135.224	195.186.135.224
Port	HTTP	HTTP
Packets	39	30’929
Bytes	14’049	34’453’050
Tx Packets	18	19’985
Tx Bytes	2’668	22’911’966
Rx Packets	21	10’944
Rx Bytes	11’381	11’541’084
Latitude	-	-
Longitude	-	-

Tabelle 3.5: Vergleich «TCP Endpoints»: «195.186.135.224»

Die Tabelle zeigt die Unterschiede ziemlich deutlich. In der Aufzeichnung am Mirror-Port des Switches wurden etwa 34MB Daten transportiert. Bei der Aufzeichnung beim BackTrack-System sind lediglich Daten im Bereich von etwa 14 KB transportiert worden.

Diese Erkenntnis lässt darauf schliessen, dass die App selbstständig und ohne Kenntnis des Proxys eine Verbindung aufbauen kann. Dabei wird die Angabe des Proxys auf dem iPhone schlichtweg ignoriert. Der angegebene Wert in der App von etwas mehr als 30 MB scheint einigermassen korrekt zu sein.

Fazit

Einmal mehr zeigt sich, dass eine genaue und detaillierte Analyse von mobilen Apps nicht ganz einfach ist. Mit den Ergebnissen des Proxy lassen sich lediglich HTTP- sowie HTTPS-Verbindungen analysieren. Um auch die zusätzlichen übermittelten Daten miteinzubeziehen, muss das Port-Mirroring verwendet werden und die Daten von Wireshark ebenfalls miteinbezogen werden.

Als Ziel unserer Applikation wurde festgelegt, mehrere Geräte gleichzeitig zu analysieren. Hierfür ist es notwendig, die Geräte auf Stufe IP unterscheiden zu können. Da im aktuellen Versuchsaufbau ein Layer 2 Switch verwendet wurde, können nur die MAC-Adressen unterschieden werden. Der verwendete WLAN-Router sendet jedoch jedes Paket mit seiner eigenen MAC-Adresse, sodass das schlussendliche Gerät, welches am WLAN-Router angehängt ist, unbekannt bleibt.

Aufgrund dessen muss eine alternative «Port Mirroring»-Variante gefunden werden. In einem weiteren Versuch wurde deshalb ein AirPcap-Stick verwendet, welcher den WLAN-Verkehr mitsniffen kann.

3.5 Analyse «Port Mirroring mittels AirPcap-Stick»

Wie im vorgehenden Versuch festgestellt wurde, kann zwar sämtlicher Verkehr aufgezeichnet werden, jedoch können die verwendeten Geräte nicht unterschieden werden. Aus diesem Grund wird nun versucht, mittels einem AirPcap-Stick den WLAN-Netzwerkverkehr mitzuhören. Der AirPcap-Stick wird hierfür an den Windows 7 eingesteckt. Weiter werden für diesen Versuch zwei Geräte verwendet, welche mit dem WLAN verbunden sind. Konkret waren folgende Komponenten im Einsatz (Tabelle 3.6):

IP-Adresse / Subnetzmaske	Bezeichnung
10.0.0.1/24	WLAN-Router Netgear Wireless Router WNR1000
10.0.0.3/24	iPad 16 GB Version: 5.1.1 (9B206), Modell: MB292FD
10.0.0.6/24	Windows 7 Computer mit installiertem Wireshark (32Bit) sowie AirPcap Control Panel
10.0.0.9/24	iPhone 4 16 GB Version: 5.1.1 (9B206), Modell: MC603FD
-	AirPcap-Stick Modell: AirPcap Classic, Media: 802.11 b/g

Tabelle 3.6: Analyse «Port Mirroring mittels AirPcap-Stick»: Netzkomponenten

Der Versuchsaufbau selbst kann mit nachstehender Skizze (Abbildung 3.13)weiter beschrieben werden:

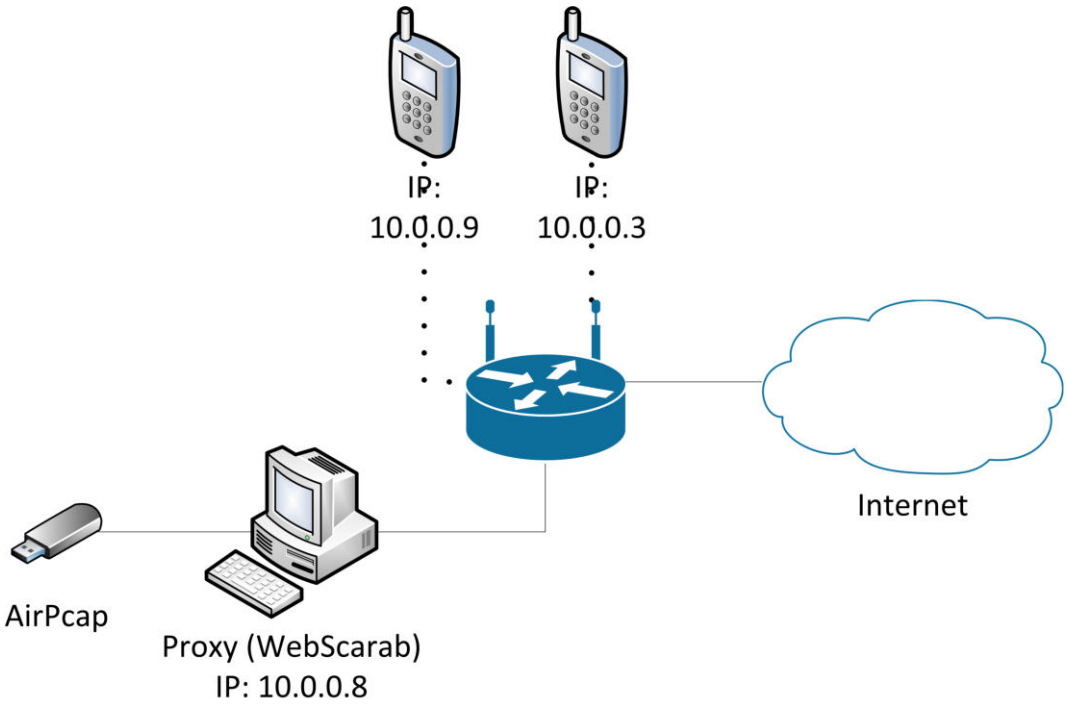


Abbildung 3.13: Analyse «Port Mirroring mittels AirPcap-Stick»: Aufbau Netzkomponenten

Für den Versuchsaufbau ist es dabei zwingend notwendig, dass die 32Bit-Version von Wireshark verwendet wird. Ansonsten kann der AirPcap-Stick nicht verwendet werden für die Aufzeichnung.

Wie weiter auf der Grafik in Abbildung 3.13 erkannt werden kann, ist am WLAN-

Router das iPhone, das iPad und der Windows 7 Computer angeschlossen. Letzterer wird in diesem Versuch benötigt, um den WLAN-Netzwerkverkehr abzuhören. Ein Proxy wurde in diesem Versuch nicht verwendet. Es wird lediglich versucht, TCP-Verbindungen von einzelnen Geräten unterscheiden zu können. Das WAN-Interface des WLAN-Routers ist direkt mit dem HSR-Netz verbunden. Der Layer 2 Switch, welcher im vorhergehenden Versuch verwendet wurde, wurde wieder entfernt.

Um den drahtlosen Netzwerkverkehr des WLAN-Routers abfangen zu können, muss der verwendete AirPcap-Stick konfiguriert werden. Hierfür wird die Software «AirPcap Control Panel» benötigt, welche mit dem AirPcap-Stick mitgeliefert wird. Die Konfiguration muss dabei so getroffen werden, dass der AirPcap-Stick auf dem gleichen Kanal mithört, wie der Access Point die Signale aussendet. In diesem Versuchsaufbau betrifft es den Kanal 13, welcher auf dem AirPcap-Stick konfiguriert wurde.

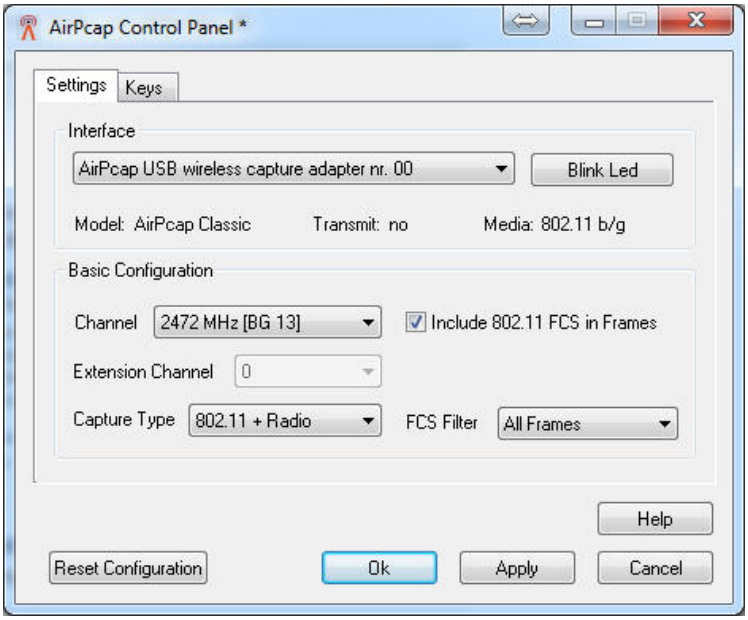


Abbildung 3.14: «AirPcap Konfiguration»

Mit dieser Konfiguration ist der AirPcap-Stick bereits einsatzbereit und kann verwendet werden.

Um mit dem Versuch fortfahren zu können, wird auch hier eine App nötig, welche eine Verbindung zu einem Server aufbaut. Da sich die App von der Firma cnlab AG beim letzten Versuch bewährt hat, wird diese nochmals eingesetzt.

Auf dem iPhone und dem iPad muss also die App installiert und gestartet werden. Bevor jedoch eine Geschwindigkeitsmessung gemacht wird, muss auf dem verwendeten Windows 7 Computer Wireshark gestartet werden. Beim Starten einer Aufzeichnung fällt auf, dass ein neues Interface verfügbar ist. Ein «AirPcap USB wireless capture adapter nr. 00» wird nun zusätzlich angezeigt. Das AirPcap-Stick kann so als Interface ausgewählt und die Aufzeichnung gestartet werden.

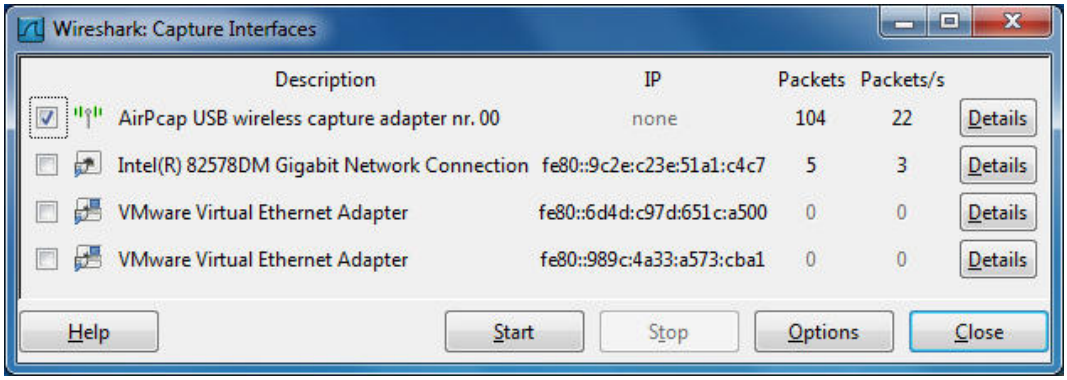


Abbildung 3.15: Wireshark: Capture Interfaces

Auf dem iPhone und dem iPad werden nun nacheinander je ein Speedtest gestartet.

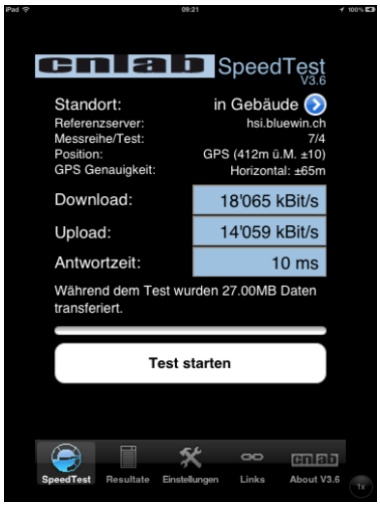


Abbildung 3.16: «cnlab Speed»: Ergebnisanzeige iPhone

Abbildung 3.17: «cnlab Speed»: Ergebnisanzeige iPad

Nach der Durchführung des Tests empfiehlt es sich die Aufzeichnung bei Wireshark zu stoppen und die Ergebnisse zu speichern.

Wie aus dem vorhergehenden Versuch bekannt, ist der Referenzserver «hsi.bluewin.ch» unter der IP-Adresse «195.186.135.224» erreichbar. Bei Wireshark kann also bereits mal nach dieser IP-Adresse gefiltert werden. Dies funktioniert am einfachsten mit dem Filter «ip.addr == 195.186.135.224».

Mit Hilfe dieses Filters kann dann eine bereinigte Statistik «Conversations» angezeigt werden.

Diese Screenshots können sich bei weiteren Versuchen ändern und es müssen

nicht sämtliche Adressen aufgelistet werden. Die Einträge sind abhängig von der vorhandenen Netzwerkinfrastruktur, sowie der Aktivität im Netzwerk.

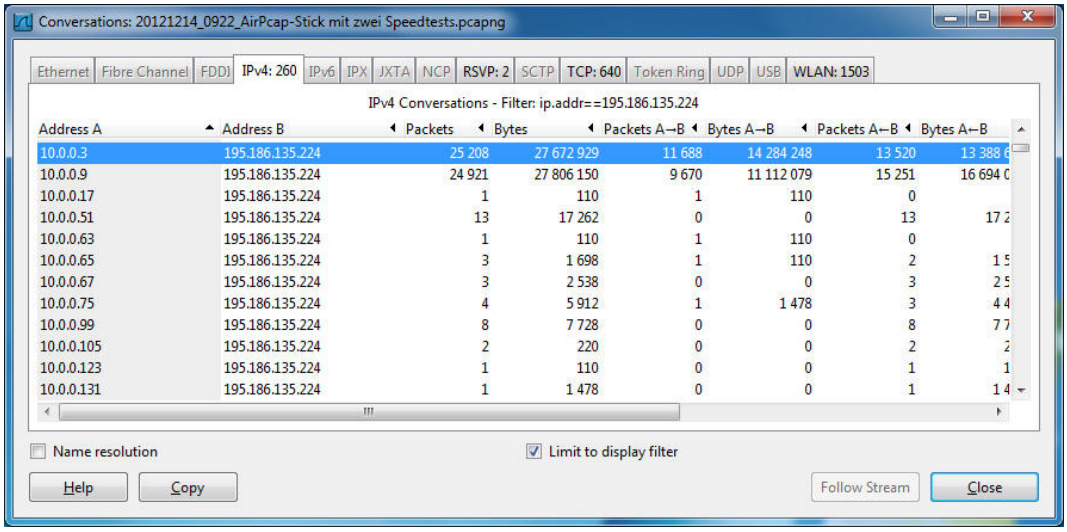


Abbildung 3.18: Wireshark: Übersicht «Conversations»

In dieser Statistik sind dann die zwei Einträge vorhanden, welche auf den Speedtest zurückzuführen sind. Die Geräte können also mit Hilfe eines AirPcap-Sticks unterschieden und die Daten zugleich in einer Wiresharkaufzeichnung gespeichert werden.

Die Übertragungsmengen, welche in der Statistik aufgeführt werden, stimmen etwa mit den Datenmengen, welche von cnlab angegeben werden, überein.

Fazit

Im Verlauf des Versuches hat sich gezeigt, dass sich der AirPcap-Stick als sehr praktikabel erweist. Unterschiedliche Geräte können erkannt und deren Daten mittels Wiresharkaufzeichnung gespeichert werden.

Aufgrund des AirPcap-Sticks werden die Wiresharkaufzeichnungen ein wenig aufgebläht. So werden beispielsweise alle «Beacon-Frames», welche ein WLAN-Router aussendet um sein Netz bekannt zu machen, ebenfalls aufgezeichnet.

Es sollte ein IPv4-Filter für die Aufnahme eingesetzt werden.

Kapitel 4

Softwaredokumentation

Das Kapitel Softwareentwicklung beschäftigt sich mit dem Aufbau des AppAnalysators und der Konzepte dahinter.

Die Applikation ist in drei Teile aufgebaut. Zum einen ist es das Auslesen der Daten aus den WebScarab und Wireshark Dateien und das Schreiben in die Datenbank. Zum anderen ist es das Auslesen der Daten aus der Datenbank und das Darstellen als Objekte. Der letzte Teil ist das UI und die Komponenten dazu. Als Datenbank wurde eine SQLite Datenbank verwendet. Der Aufbau ist im Kapitel «Datenspeicherung» genauer erläutert.

4.1 Domainanalyse

Die für die Applikation relevanten Komponenten sind im nachfolgenden Domain-Model dargestellt:

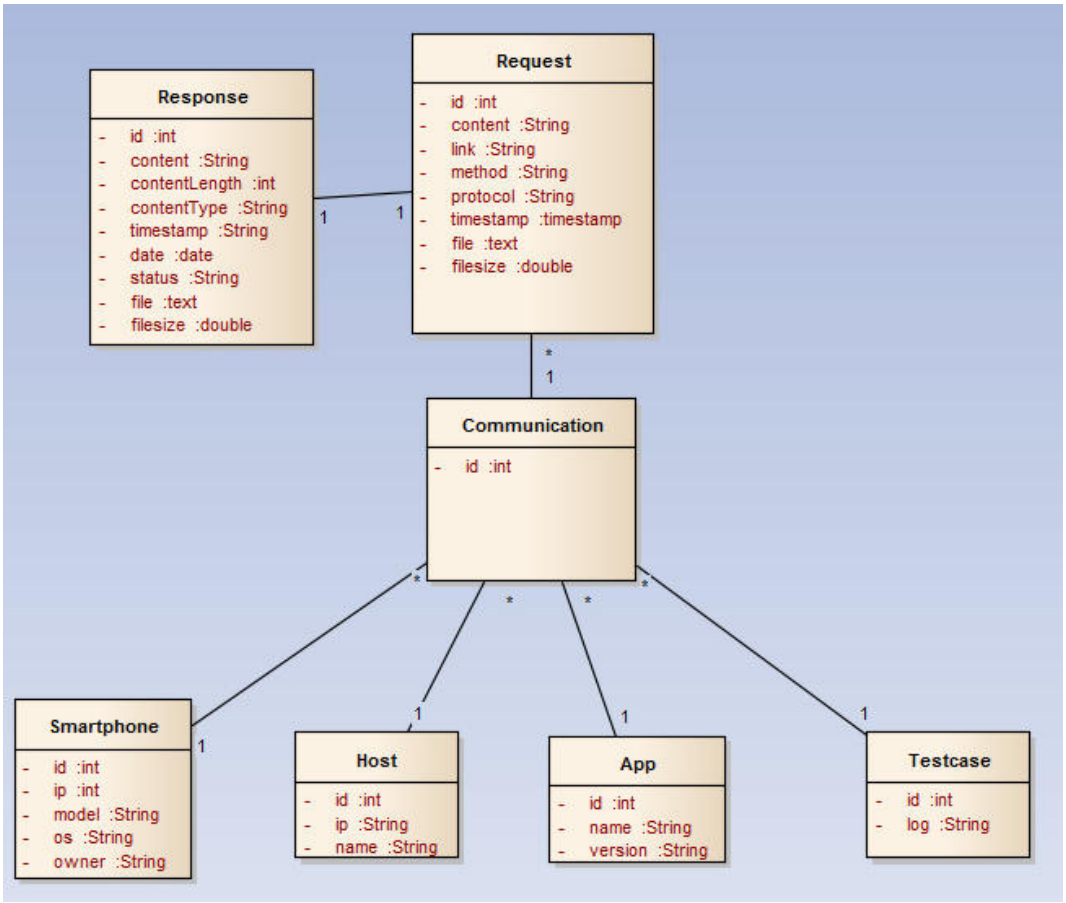


Abbildung 4.1: Domain-Model

In den nachkommenden Abschnitten werden die konkreten Bestandteile des Domain-Modells erklärt.

4.1.1 Request

Der Request abstrahiert einen Request, der von einem Smartphone aus an einen Host gemacht wurde. Es handelt sich also beispielsweise um eine Anfrage an einen Webserver. Ein Request wird jeweils einer Communication und einer Response zugeordnet.

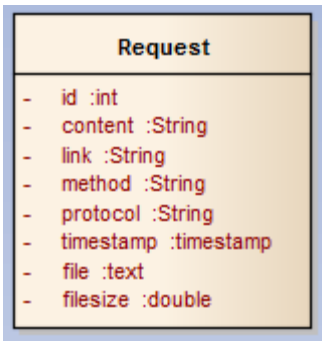


Abbildung 4.2: Request

Attribut	**Beschreibung**
id	Die ID identifiziert einen Request mit einer eindeutigen Nummer.
content	Das Attribut Content repräsentiert den Inhalt des gesendeten Pakets.
link	Der Link entspricht der Adresse, welche durch das Smartphone kontaktiert wurde.
method	Die Methode des benutzten HTTP- und HTTPS-Protokolls.
protocol	Repräsentiert das benutzte Protokoll des Requests.
timestamp	Der Timestamp speichert den Zeitpunkt, an welchem der Request gestartet ist.
file	Die gesamte, von Webscarab erstellte, Datei für einen Request befindet sich in diesem Attribut.
filesize	Entspricht der Grösse (Anzahl Bytes) eines Requests.

4.1.2 Response

Die Response abstrahiert einen Response, der von einem Host zurück an eine Smartphone getätigt wird.

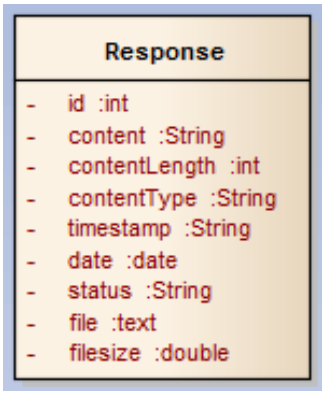


Abbildung 4.3: Response

Attribut	Beschreibung
id	Die ID identifiziert eine Response mit einer eindeutigen Nummer.
content	Im Attribut Content repräsentiert den Inhalt des erhaltenen Pakets.
contentLength	Dieses Attribut gibt die Länge des Contents zurück.
conentType	Der Content Type gibt den Typ des Inhalts zurück. Beispielsweise, dass der Inhalt als XML codiert ist.
timestamp	Der Timestamp speichert den Zeitpunkt, an welchem die Response erhalten wurde.
date	Date entspricht dem Zeitpunkt, an welchem die Response entstand. Es entspricht somit der Zeit des Server, bei welcher dieser die Antwort generiert hat.
status	Dieses Attribut repräsentiert den Response Status eines HTTP beziehungsweise HTTPS Requests.
file	Die gesamte, von Webscarab erstellte, Datei für eine Response befindet sich in diesem Attribut.
filesize	Entspricht der Grösse des Response.

4.1.3 Communication

Die Communication entspricht einer Sammlung von Requests und Resonses von einem bestimmten Smartphone an einen bestimmten Host. Einer Communication wird jeweils ein Host, ein Smartphone, eine App und ein Testcase zugeordnet.

Abbildung 4.4: Communication

Attribut	Beschreibung
id	Die ID identifiziert eine Communication mit einer eindeutigen Nummer.

4.1.4 Smartphone

Ein Smartphone repräsentiert ein Gerät, welches getestet wird.

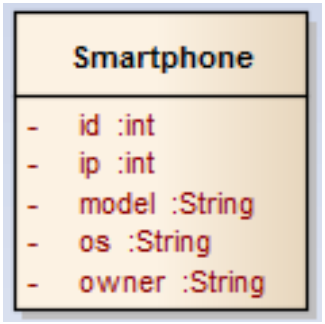


Abbildung 4.5: Smartphone

Attribut	Beschreibung
id	Die ID identifiziert ein Smartphone mit einer eindeutigen Nummer.
ip	Dieses Attribut steht für die IP-Adresse eines Devices.
model	Das Model repräsentiert den Typ eine Smartphones (beispielsweise «iPhone5»).
os	Das OS ist das Betriebssystem, welches auf dem Device läuft.
owner	Hier kann der Besitzer des Devices eingetragen werden.

4.1.5 Host

Ein Host repräsentiert eine Adresse eines Webservers, auf welche eine Verbindung hergestellt wird. In den meisten Fällen handelt es sich dabei um einen Webserver.

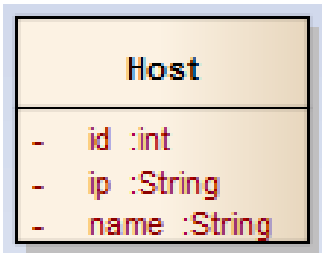


Abbildung 4.6: Host

Attribut	Beschreibung
id	Die ID identifiziert einen Host mit einer eindeutigen Nummer.
ip	Dieses Attribut steht für die IP-Adresse eines Hosts.
name	Das Attribut «name» repräsentiert den Hostnamen / Domainnamen.

4.1.6 App

Ein App repräsentiert eine Applikation, von welcher eine Verbindung getätigt wird.

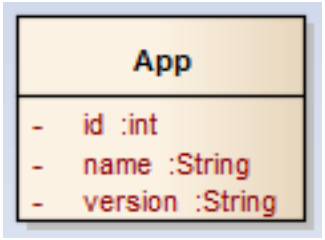


Abbildung 4.7: App

Attribut	Beschreibung
id	Die ID identifiziert ein App eindeutig.
name	Dieses Attribut steht für den Name der App.
version	Dies repräsentiert die Version des Apps.

4.1.7 Testcase

Ein Testcase entspricht einer gestarteten Messung

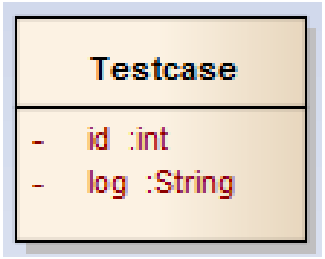


Abbildung 4.8: Testcase

Attribut	Beschreibung
id	Die ID identifiziert ein App mit einer eindeutigen Nummer.
log	In diesem Attribut werden optionale Zusatzinformationen eines Testcase gespeichert.

4.2 Packages

Die Applikation lässt sich in vier verschiedene Schichten unterteilen. UserInterface, Service, Domain und DAO.

Packageübersicht

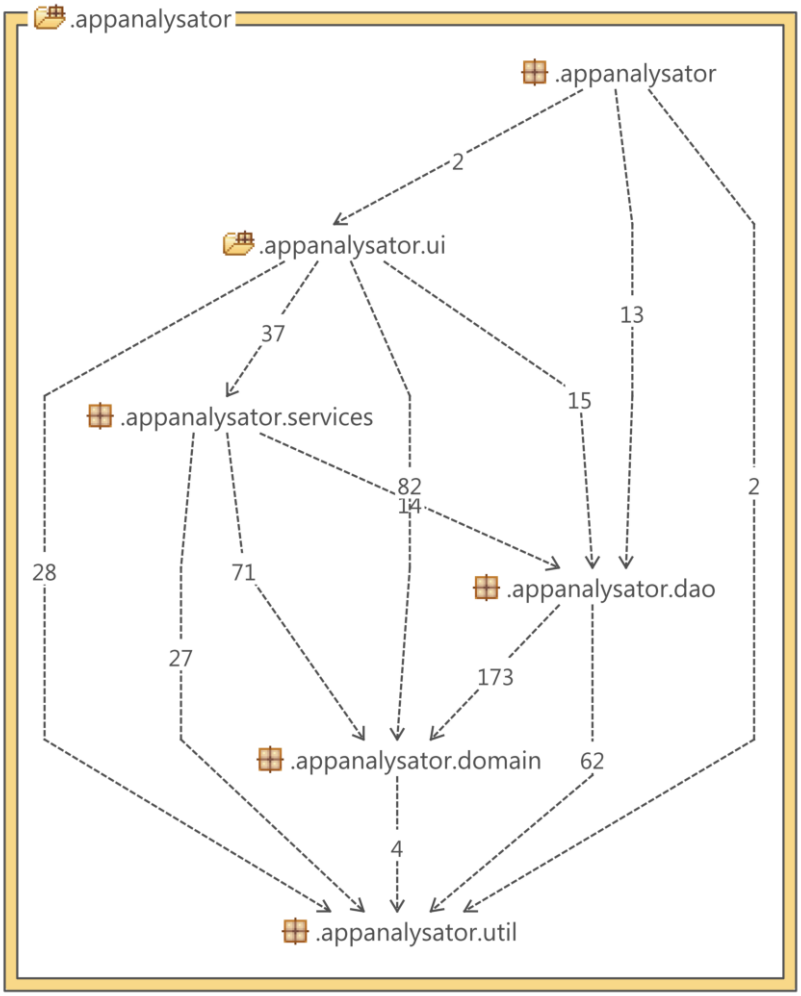


Abbildung 4.9: Packageübersicht

Package	Beschreibung
ch.hsr.appanalysator	In diesem Package ist die Main Methode beinhaltet, sie initialisiert das Logfile, die Datenbankverbindung und das Userinterface.
ch.hsr.appanalysator.ui	Alle für das User Interface relevanten Dateien, wie Frames, TableModels, CellRenderer und weitere für das UserInterface verwendete Klassen sind in diesem Package beinhaltet.
ch.hsr.appanalysator.domain	Die Definitionen der zu verwendenden Klassen werden hier beschrieben.
ch.hsr.appanalysator.dao	Dieses Package beinhaltet die Data Access Objects, welche die Objekte der Datenbank repräsentieren. Weiter werden die Datenbankabfragen und Updates definiert.
ch.hsr.appanalysator.util	Hier sind die Klassen beinhaltet, welche zur Programmunterstützung dienen.
ch.hsr.appanalysator.service	Im diesem Package befindet sich die Logik, um die externen Dateien von Wireshark und WebScarab einzulesen. Die Dateien werden in diesen Klassen ausgelesen und in die Datenbank eingefügt.

Tabelle 4.1: Beschreibung der Packages

4.3 Datenspeicherung

Um die Daten speichern zu können, ist eine Datenbank notwendig. Die verwendete SQLite-Datenbank hat dabei den entscheidenden Vorteil, dass kein Datenbankserver benötigt wird. Die Funktionen von SQlite reichen für die Anforderungen dieser Applikation aus. Da die SQlite Datenbank nur aus einer Datei besteht, welche alle Tabellen und Views beinhaltet, hält sich auch der benötigte Speicherplatz in Grenzen.

Request

Attribut	Beschreibung
id	INTEGER
content	VARCHAR
link	VARCHAR
method	VARCHAR
protocol	VARCHAR
timestamp	VARCHAR
responseId	INTEGER
communicationId	INTEGER
file	NTEXT
filesize	DOUBLE

Response

Attribut	Beschreibung
id	INTEGER
content	VARCHAR
contentLength	INTEGER
contentType	VARCHAR
timestamp	VARCHAR
date	VARCHAR
status	VARCHAR
file	INTEGER
filesize	DOUBLE

Communication

Attribut	Beschreibung
id	INTEGER
appId	INTEGER
hostId	INTEGER
smartphoneId	INTEGER
testcaseId	INTEGER
protocol	VARCHAR

Smartphone

Attribut	Beschreibung
id	INTEGER
ip	VARCHAR
model	VARCHAR
os	VARCHAR
owner	VARCHAR

Host

Attribut	Beschreibung
id	INTEGER
ip	VARCHAR
name	VARCHAR

App

Attribut	Beschreibung
id	INTEGER
name	VARCHAR
version	VARCHAR

Testcase

Attribut	Beschreibung
id	INTEGER
log	VARCHAR

Kapitel 5

Ergebnisse der untersuchten Apps

Im Kapitel mit den Grundlagen wurden einige Apps aufgelistet, welche mit dem im Anhang aufgezeichneten Aufbau und dem AppAnalysator getestet wurden. Die Ergebnisse zu den einzelnen Tests sind im Anhang beigelegt. In den folgenden Abschnitten wird nun auf Auffälligkeiten bei den Ergebnissen eingegangen. Ebenso wird das Ergebnis des «iOf»-App genauer betrachtet. Der Industriepartner, Philipp Klomp, Nomasis AG, hat gewünscht, das Verhalten des App ebenfalls zu untersuchen.

5.1 Allgemeine Auffälligkeiten

Bei den Testversuchen ist aufgefallen, dass viele Apps welche mit persönlichen Daten umgehen mit der sichereren HTTPS-Verbindung arbeiten. Weiter ist nicht aufgefallen, dass im Hintergrund unbewusst persönliche Daten, wie beispielsweise Kamerabilder, hochgeladen wurden.

Vorbildlich hat sich Google mit dem App «Google Plus» gezeigt. «Google Plus» ist, wie beispielsweise Facebook, eine Social Media Plattform. Die App ermöglicht es dabei Statusmeldungen, Bilder, Videos oder ähnliches auf «Google Plus» zu verbreiten.

Wird die App gestartet und meldet sich der Benutzer an, scheint es, als ob während der Anmeldephase ein internes Google-Zertifikat oder eine spezielle Zahl geprüft wird. Bei einem zwischengeschalteten Proxy, wie es beim Test der Fall war, schlägt die Anmeldung fehl. Ein Verbindungsunterbruch zwischen dem Testgerät und dem Google-Server scheint unwahrscheinlich zu sein. Wird der Proxy aus den Einstellungen entfernt, klappt die Anmeldung problemlos. Wird aber danach der Proxy erneut eingegeben und die App gestartet, schlägt die Verbindung wieder fehl.

Diese Erkenntnisse unterstützen die Annahme, dass Google eine Sicherheitsmassnahme in die «Google Plus»-App eingebaut hat. Kann dieses Zertifikat oder die Kennzahl nicht überprüft werden, interpretiert der «Google-Plus»-Server den Verbindungsaufbau als inkorrekt, sodass der Verbindungsaufbau abgebrochen wird. Dies bedeutet wiederum, dass Google bei der App einen Man-in-the-Middle-Angriff verhindert. Der Verwendung der App kann also als «ungefährlich» betrachtet werden, zumindest was Angriffe über einen Man-in-the-Middle betrifft.

Ebenfalls als vorbildlich hat sich die App «Waze» gezeigt. Zusätzlich zur verwendeten

HTTPS-Verbindung schien, beim durchgeführten Test, der Benutzername zusätzlich verschlüsselt zu sein. Anstatt der Emailadresse, wie dies bei anderen Apps gemacht wird, wurde «BA3olgZfQZmVm8BoaE7K5Q» verwendet. Für einen Man-in-the-Middle ist somit nicht gleich ersichtlich, wer sich angemeldet hat. Gleichzeitig ist anzumerken, dass «Waze» auch TCP-Verbindungen aufbaut und als Kommunikationsmittel benutzt. Was bei diesen Verbindungen konkret übertragen wird, ist nicht klar.

Allgemein wurden jedoch während allen Tests sehr viele Informationen zum verwendeten Gerät übertragen. Dabei können Informationen zur Displaygrösse oder zur installierten iOS-Version relativiert und als «nicht heikel» angesehen werden. Bei vielen Apps wurde aber auch oft der Mobilfunknetzbetreiber, der sogenannte Carrier, mitgeliefert. Der Hintergrund beziehungsweise die Notwendigkeit solche spezifische Informationen mitzusenden, kann aus Sicht des Benutzer als sehr fraglich angesehen werden. Aus diesem Grund können nur Annahmen getroffen werden.

Als möglicher Grund kann mit grosser Sicherheit eine statistische Erfassung der Benutzer sein. Diese Statistiken könnten daraufhin auch weiterverkauft beziehungsweise weitergereicht werden. Ebenfalls kann als möglicher Grund die benutzerspezifische Werbung angesehen werden. Werden Informationen zum Carrier übermittelt, können auf der App Werbungen der Konkurrenz oder des korrekten Betreibers aufgeschaltet werden.

Als heikler kann das Versenden der sogenannten UUID angesehen werden. Es handelt sich dabei um den «Universally Unique Identifier» eines Gerätes, sodass sich jedes Gerät eindeutig identifizieren lässt. In den durchgeführten Tests war diese Kennzeichnung bei Facebook und iOf problemlos auffindbar.

Ebenso bedenklich sind die Ergebnisse des Test der «Raiffeisen»-App. Die Absicherung gegen einen Man-in-the-Middle-Angriff scheint dort nicht gegeben zu sein, sodass Informationen wie beispielsweise die E-Banking Vertragsnummer und das persönliche Passwort problemlos ausgelesen werden kann.

5.2 Ergebnis iOf

Wie im Anhang dokumentiert, besitzt die «iOf»-App sehr viele statische Daten, also Daten, welche fix in die Applikation programmiert wurden. Die dynamischen Inhalte wie beispielsweise Vereinsinformationen werden zum benötigten Zeitpunkt geladen. Dabei wird eine HTTPS-Verbindung mit dem Webserver der Entwicklungsfirma (RedDev GmbH) aufgebaut. Diese Übertragungsart kann für öffentlich zugängliche Daten, wie es bei iOf der Fall ist, als ausreichend angesehen werden.

Was berücksichtigt werden muss beziehungsweise auch beachtet werden sollte, dass bei jedem versendeten GET-Request Informationen zum Client-Gerät, zum Modell und zum Betriebssystem gemacht werden. Weiter ist auch die UUID in jedem Request vorhanden.

Die Übermittlung dieser Angaben kann sehr in Frage gestellt werden. Insbesondere die UUID scheint für die korrekte Darstellung und Interpretation der empfangenden Antworten nicht weiter relevant zu sein. Auch hier kann nur angenommen werden, dass es sich um eine statistische Erfassung des Benutzersverhalten handelt.

5.3 Fazit

Wie in den oberen Abschnitten erwähnt, wird bei vielen Apps bereits jetzt eine HTTPS-Verbindung aufgebaut. Dennoch können die Ergebnisse der durchgeführten Tests nicht alle Fragen mit Sicherheit beantworten. Bei Google Latitude wird beispielsweise ein grosser Teil der versendeten Daten zusätzlich verschlüsselt. Es ist deshalb nicht klar ersichtlich, welche Daten genau übermittelt werden. Unter Umständen kann es sein, dass Google Latitude umliegende und auffindbare WLAN-Netze an Google übermittelt. Wird noch ein genauer Standort mittels GPS-Daten übermittelt, weiss Google genau, wo sich welche WLAN-Netze befinden. Dies erlaubt Google, den Benutzer auch mittels der SSID des WLAN-Netzes orten zu können.

Kapitel 6

Schlussfolgerungen

Die in der Aufgabenstellung gewünschten Funktionen sind im AppAnalysator vorhanden. Mit der entwickelten Applikation besteht die Möglichkeit, verschiedene Apps und Smartphones gleichzeitig untersucht werden können. Dies wurde an einigen Tests im Anhang C genauer aufgezeigt.

Der AppAnalysator importiert dabei Daten aus dem Proxy (WebScarab) sowie den Wireshark-Messungen. Die gesammelten Daten werden strukturiert in einer SQLite-Datenbank abgespeichert. Zu einem späteren Zeitpunkt kann natürlich auch eine andere Datenbank in Betracht gezogen werden. Die Datenbank ermöglicht es, die Daten mit SQL-Abfragen auszulesen.

Bei den drei Darstellungsmöglichkeiten wurde versucht, möglichst informative Ansichten zu generieren. Die Ansichten wurden in Zusammenarbeit mit dem Betreuer und dem Industriepartner entwickelt und optimiert. In der «Timeline» werden beispielsweise lediglich die Zeitpunkte aufgezeigt, welche auch wirklich benötigt wurden. Damit soll die Übersichtlichkeit für den Benutzer bestehen bleiben. Die Übersichtlichkeit umzusetzen war jedoch alles andere als einfach.

Ebenfalls kam es zu einer Kompromisslösung bezüglich der Datenmengen. Die Aufzeichnungen durch Wireshark werden begrenzt. Konkret werden nur die ersten 200 Bytes eines Paketes abgespeichert. Diese 200 Bytes beinhalten die wichtigsten Informationen zum Sender und Empfänger, welche für die weitere Analyse durch den AppAnalysator benötigt werden. Informationen zum Inhalt der Übertragung wird jedoch in den meisten Fällen abgeschnitten und ein Speicherplatzproblem zu verhindern.

Bei WebScarab wurde eine solche Einschränkung jedoch nicht getätigt. Sämtliche Informationen des Requests und der Response werden in die Datenbank mit eingefügt. Dies hat jedoch zur Folge, dass die Applikation nicht geeignet ist, um beispielsweise mehrere grössere YouTube-Videos anzusehen. YouTube Videos werden in einem HTTP Response an den Empfänger gesandt, sodass der Proxy die ganze Filmdatei mitspeichert. Dies erhöht zum einen den Speicherbedarf der Festplatte durch die WebScarab Dateien. Zum anderen entstehen aber auch zusätzliche Anforderungen an die SQLite-Datenbank. Schlussendlich muss die Datenbank diese Daten ebenfalls verwalten können.

Mit der aktuellen Version des AppAnalysators können lediglich iOS-Geräte analysiert werden. Die Android Plattform ermöglicht es nicht, einen globalen Proxy für das ganze

Smartphone einzutragen. Dadurch kann nicht mit Sicherheit gesagt werden, dass der komplette Verkehr über den Proxy geht. Als Weiterführung der Arbeit wäre es denkbar, eine Lösung für dieses Problem zu finden.

Allgemein hat sich gezeigt, dass die Aufgabestellung nicht ganz trivial ist, wie sie auf den ersten Blick erscheinen mag. Es gab dabei einige Fragen, welche zu Beginn der Arbeit nicht genau beantwortet werden konnten. Als kleines Beispiel: Es war lange nicht klar, wie eine App am einfachsten identifiziert werden kann. In einem Versuch zeigte sich dann, dass der Name des Apps im «User-Agent» des HTTP-Request-Header eingetragen wird. Aber auch hier zeigte sich schnell, dass es sogar innerhalb einer App zu Unterscheidungen kommen kann. Diese Unterscheidungen müssen jedoch in Kauf genommen werden, da das Verhalten des Benutzers nicht bestimmt ist.

Nichts desto trotz, konnte die Aufgabenstellung mit den oben erwähnten Einschränkungen erfüllt werden.

Kapitel 7

Persönliche Berichte

Persönlicher Bericht Pascal Roman Artho

Nach 14 Wochen blicke ich nun zurück auf die Studienarbeit. Bei der Bewerbung zur Studienarbeit im Sommer 2012 war mir noch nicht klar, was alles in dieser Arbeit gemacht werden muss. Dies änderte sich jedoch nachdem die ersten Sitzungen mit dem Betreuer, Herrn Prof. Dr. Peter Heinzmann, und später mit dem Industriepartner, Herrn Philipp Klomp, Nomasis AG, stattgefunden haben.

Schnell konnten die Wünsche und Ansprüche des Betreuers und dem Industriepartner herausgehört werden. Wie diese umgesetzt werden können, war damals noch nicht bekannt. So kam es, dass einige Versuche durchgeführt werden mussten, um mit der Materie vertraut zu werden. Dabei mussten auch die Grundlagen gelernt beziehungsweise in Erinnerung gerufen werden. So war mir beispielsweise der genaue Ablauf eines TLS-Verbindungsaufbau nicht bekannt.

Bei den Versuchen kam es ebenfalls immer wieder zu einzelnen Rückschlägen. Plötzlich konnte die verschlüsselte Verbindung nicht mehr aufgebrochen werden und «niemand» wusste warum. Inzwischen habe ich einiges dazugelernt und weiss jetzt, was das Problem sein könnte, wenn der Proxy nicht funktioniert. Dieses Wissen aufzubauen nahm jedoch viel Zeit in Anspruch, da ein Versuch immer vorbereitet werden musste. Ein iPad oder iPhone musste bei jedem Versuch auf die Werkseinstellungen zurückgesetzt werden, um wieder von «null» beginnen zu können.

Ich bin jedoch der Meinung, dass ich vieles aus der Studienarbeit lernen konnte. Es kam beispielsweise immer wieder dazu, dass unterschiedliche Ansätze getestet wurden, wie zum Beispiel der Burp Proxy und der WebScarab. Auf den ersten Blick schien der Burp Proxy «viel einfacher und besser» zu sein als der WebScarab. Zu einem späteren Zeitpunkt stellten Rebekka und ich jedoch schnell fest, dass WebScarab mehr Möglichkeiten bietet. Insbesondere beim Abspeichern von Request und Responses kann der Burp Proxy nicht «mithal-

ten». Ähnliches galt für den Versuch mit dem Port Mirroring. Die Lösung mit dem Layer 2 Switch schien auf den ersten Blick die beste Lösung zu sein. Es zeigte sich jedoch schnell, dass dort nicht alle benötigten Informationen vorhanden sind und der AirPcap-Stick mehr Informationen liefert.

Da meine Kenntnisse im Programmieren von Applikationen eingeschränkt sind, war es für mich eine sehr spannende und abwechslungsreiche Arbeit auch in Bezug auf das Programmieren. Das Einlesen der Daten aus den WebScarab Dateien sowie den Aufzeichungen von Wireshark war nicht so einfach, wie ich zuerst vielleicht mal gedacht hatte. Immer wieder kam es zu Zusätzen, um spezielle Fälle abfangen zu können. Es sollten beispielsweise nur Dateien gelesen werden, welche zuvor nicht schon gelesen wurden. Dass heisst, dass eine Möglichkeit gefunden werden musste, die Dateien zu überspringen und nur die «neuen» Dateien einlesen zu lassen.

Die gewonnenen Erkenntnisse in Bezug auf das Thema der Studienarbeit haben mich ein wenig zum Nachdenken gebracht. Persönlich finde ich es sehr erschreckend, dass diverse Informationen an irgendwelche Webserver gesendet werden. Obwohl ich ebenfalls ein Smartphone habe (mit dem Android Betriebssystem), kann ich nicht genau sagen, welche Daten genau versandt werden. Wird davon ausgegangen, dass ähnlich viele Daten versandt werden wie bei iOS-Geräten, ist dies eine sehr erschreckende Bilanz.

Fazit

Während der Studienarbeit konnte ich meine Kenntnisse stark erweitern. Sei es in Bezug auf die Projektplanung, wie viel Zeit für eine Tätigkeit benötigt wird, oder aber auch in Bezug auf das Teamwork, eine Arbeit mit einer Mitstudentin durchzuführen. Die Arbeit mit dem Betreuer, Herrn Prof. Dr. Peter Heinzmann, und dem Industriepartner, Herrn Philipp Klomp, Nomasis AG, war sehr angenehm und produktiv.

So gesehen blicke ich auf eine sehr spannende und interessante Studienarbeit zurück.

Pascal Roman Artho, Dezember 2012

Persönlicher Bericht Rebekka Zahler

Die Studienarbeit habe ich als eine sehr interessante und lehrreiche Arbeit erlebt. Ich konnte das bisher an der HSR gelernte sehr gut umsetzen, sei es fachlich bezogen oder auch die Planung einer Projekts.

Zu Beginn empfand ich es als schwierig, mich in das Projekt zu vertiefen, weil sehr lange nicht genau klar war, was das Ergebnis der Arbeit sein soll. Wir waren ziemlich frei, was sich nicht immer positiv auswirkte. Zusätzlich hätten wir viel früher mit der Softwareentwicklung beginnen sollen. Wir hatten uns nicht

genau geeinigt, was alles realisiert werden soll. Daher wussten wir nie genau, wie viel Zeit uns für die einzelnen Teile noch bleibt, damit wir am Schluss genügend Zeit in die Dokumentation investieren konnten. Ich denke im Allgemeinen ist eine Forschungsarbeit ohne genaues Ziel immer schwieriger zu realisieren. Aber ich empfand es als gute Erfahrung.

Da Pascal Artho und ich bisher während dem Studium kein grösseres Projekt miteinander durchgeführt haben, gingen wir das Risiko ein, das wir vielleicht nicht gut miteinander arbeiten können. Allerdings wurde ich vom Gegenteil überzeugt. Es war sehr angenehm mit meinem Partner zu arbeiten, und ich freue mich sehr auf eine gemeinsame Bachelorarbeit.

Fazit

Schlussendlich bin ich sehr zufrieden mit unserer Arbeit. Aus den anfänglichen Schwierigkeiten, weil uns das Wissen aus dem Modul «Informationssicherheit 1» fehlte, haben wir viel gelernt. Trotz einigen weiteren Rückschlägen während der Arbeit, haben wir das Beste aus den jeweiligen Situationen gemacht. Ich hoffe das wir auch für die Bachelorarbeit so gut zusammenarbeiten werden.

Rebekka Zahler, Dezember 2012

Glossary

AirPcap USB Stick zum Wireless Sniffen. 44

Apps Applikation für Smartphone. 1

ARP Address Resolution Protocol. 15

CA Certificate Authority. 27

DAO Data Access Objects. 55

DNS Domain Name System. 13

GET Methode von HTTP. 28

GPS Global Positioning System. 99

HSR Hochschule für Technik Rapperswil. 34

HTTP Hypertext Transfer Protocol. 1

HTTPS Hypertext Transfer Protocol Secure. 1

iOS Standard-Betriebssystem von Apple. 1

IP Internet Protocol. 6

JAR Java Archive. 81

JDK Java Development Kit. 125

MAC-Adresse Media-Access-Control-Adresse. 14

MDNS Multicast Domain Name System. 32

OS Operating System. 53

OSI Open Systems Interconnection. 14

PKCS12 Containerformat für Zertifikate. 29

POST Methode von HTTP. 28

SSID Service Set Identifier. 60

SSL Secure Sockets Layer. 16

SVN Apache Subversion. 125

TCP Transmission Control Protocol. 1

TLS Transport Layer Security. 16

UDP User Datagram Protocol. 1

URL Uniform Resource Locator. 5

USB Universal Serial Bus. 1, 46

UUID Universally Unique Identifier. 59

WAN Wide Area Network. 34

WLAN Wireless Local Area Network. 1

XML Extensible Markup Language. 28

Literaturverzeichnis

[1] Apple. App Store-Downloads bei iTunes.
https://itunes.apple.com/de/genre/ios/id36?mt=8, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[2] www.zdnet.de Björn Greif. 700.000 Android-Apps: Google zieht mit Apple gleich.
http://www.zdnet.de/88129581/google-zieht-mit-700-000-android-apps-mit-apple-gleich, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[3] cnlab AG. WebScarab, zuletzt geändert am 10. Oktober 2012.

[4] c't. Gut App-geschaut, Netzwerkverkehr von Smartphones kontrollieren,
c't 2012, Heft 7.

[5] Google. Android Apps auf Google Play.
https://play.google.com/store?hl=de, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[6] Prof. Dr. Peter Heinzmann. Informationssicherheit 1 Folien.

[7] Thomas Leichtenstern. Arp-Grundlagen und Spoofing (Gastbeitrag tecCHANNEL).
http://www.microsoft.com/germany/technet/datenbank/articles/600593.mspx, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[8] Raiffeisenbank Schweiz. Sicherheitshinweise Mobile Banking.
www.raiffeisen.ch/mb-sicherheit, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[9] Raiffeisenbank Schweiz. Sicherer Einstieg durch dreistufiges Login.
http://www.raiffeisen.ch/web/sicherer+einstieg+durch+dreistufiges+login, zuletzt besucht am 20. Dezember 2012, 17:00 Uhr.

[10] PortSwigger Web Security. Burp Proxy.
http://www.portswigger.net/burp/proxy.html, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[11] www.zdnet.de Stefan Beiersmann. Zahl der Smartphone-Nutzer übersteigt erstmals die Milliardengrenze.
http://www.zdnet.de/88127635/zahl-der-smartphone-nutzer-uebersteigt-erstmals-milliardengrenze, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[12] Wikipedia. Address Resolution Protocol.
http://en.wikipedia.org/wiki/Address_Resolution_Protocol, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[13] Wikipedia. Diffie-Hellman-Schlüsselaustausch.
http://de.wikipedia.org/wiki/Diffie-Hellman-Schlüsselaustausch, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[14] Wikipedia. Internetprotokollfamilie.
http://de.wikipedia.org/wiki/Internetprotokollfamilie, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[15] Wikipedia. Multicast dns.
http://en.wikipedia.org/wiki/Multicast_DNS, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[16] Wikipedia. OSI Model.
http://en.wikipedia.org/wiki/OSI_model, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[17] Wikipedia. OWASP WebScarab Project.
http://de.wikipedia.org/wiki/BackTrack, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[18] Wikipedia. Proxy server.
http://en.wikipedia.org/wiki/Proxy_server, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[19] Wireshark. Man Page mergecap.
http://www.wireshark.org/docs/man-pages/mergecap.html, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[20] Wireshark. Man Page tshark.
http://www.wireshark.org/docs/man-pages/tshark.html, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

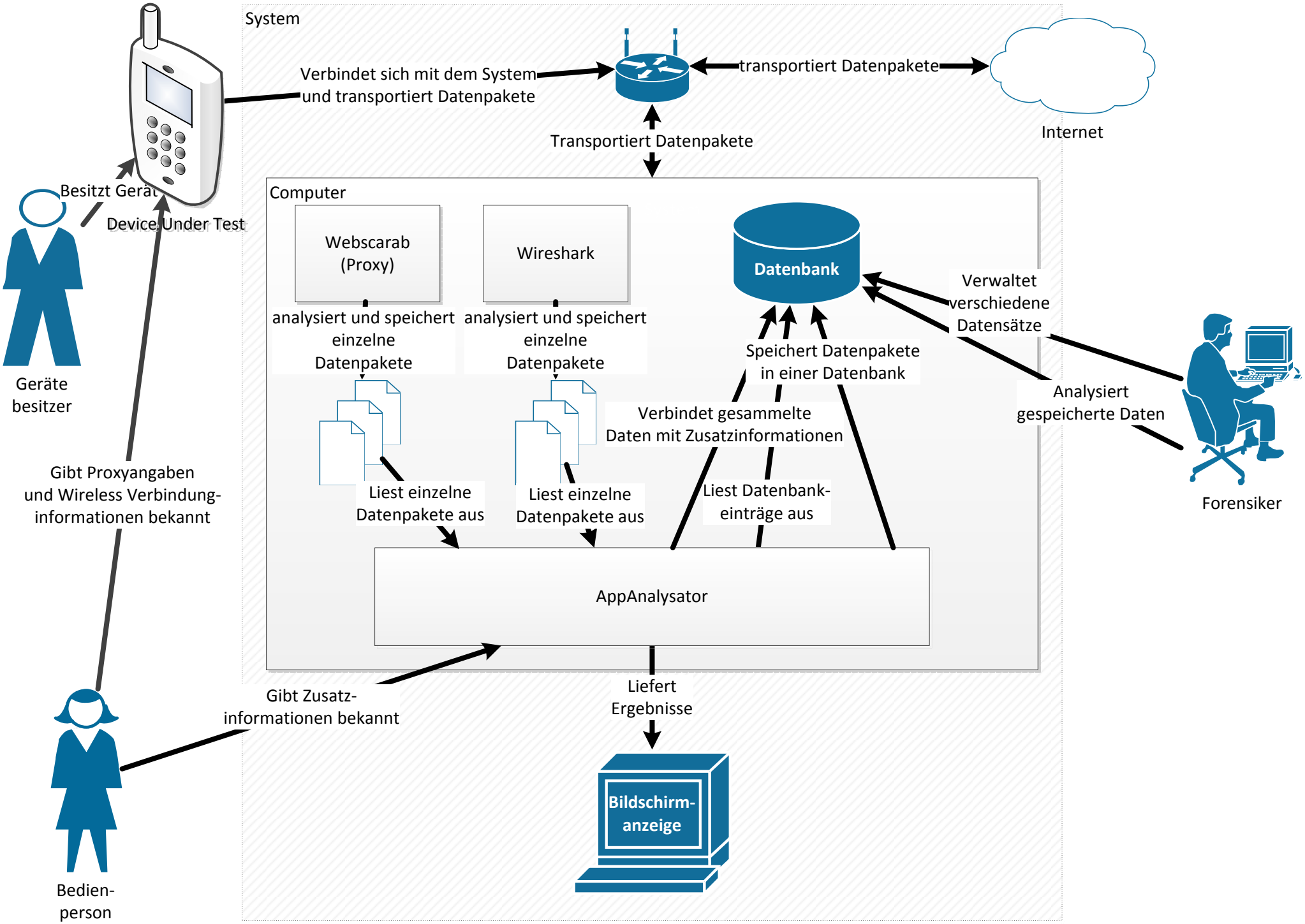
[21] www.golem.de. Webseite erlaubt Whatsapp-Nachrichten im Namen anderer.
http://www.golem.de/news/sicherheitsluecke-webseite-erlaubt-whatsapp-nachrichten-im-namen-anderer-1209-94741.html, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[22] www.owasp.org. OWASP WebScarab Project.
https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

[23] www.ubuntuusers.de. iptables2.
http://wiki.ubuntuusers.de/iptables2, zuletzt besucht am 19. Dezember 2012, 16:00 Uhr.

Anhang A

Systemgrenzen


System
Verbindet sich mit dem System
und transportiert Datenpakete
transportiert Datenpakete
Internet
Transportiert Datenpakete
Besitzt Gerät
Device Under Test
Geräte
besitzer
Computer
Webscarab
(Proxy)
Wireshark
Datenbank
analysiert und speichert
einzelne
Datenpakete
analysiert und speichert
einzelne
Datenpakete
Verwaltet
verschiedene
Datensätze
Speichert Datenpakete
in einer Datenbank
Analysiert
gespeicherte Daten
Forensiker
Verbindet gesammelte
Daten mit Zusatzinformationen
Liest einzelne
Datenpakete aus
Liest einzelne
Datenpakete aus
Liest Datenbank-
einträge aus
Gibt Proxyangaben
und Wireless Verbindung-
informationen bekannt
AppAnalysator
Gibt Zusatz-
informationen bekannt
Liefert
Ergebnisse
Bildschirm-
anzeige
Bedien-
person

Anhang B

Versuche

In diesem Kapitel wird über die gesammelten Ergebnisse der ersten Einsätze des Proxy berichtet. Als Hilfestellung wurde der Artikel aus dem c't 2012, Heft 7 [4], sowie eine Anleitung [3] von Herrn Prof. Dr. Peter Heinzmann verwendet.

B.1 Burp Proxy

Als erstes wird das Programm «Burp Proxy» genauer betrachtet. Dieses eignet sich insbesondere für iOS-Geräte und Windows-Systeme. Gemäss der Website von Burp Proxy [10] wird die Applikation wie folgt erklärt:

> Burp Proxy is an intercepting proxy server for security testing of web applications. It operates as a man-in-the-middle between your browser and the target application [...].

Es ist also eine Applikation, welche als Man-in-the-Middle fungiert und durch die der gesamte Traffic analysiert werden kann, indem ein Proxy angegeben wurde. Am einfachsten lässt sich eine solche Analyse an einem Windows-Computer oder iOS-Gerät zeigen, da bei diesen Geräten die Proxyangaben auf dem gesamten System gelten.

Als Test wird der Burp Proxy heruntergeladen und auf einem Windows 7 Computer ausgeführt. Damit der Test funktioniert, muss sich dieser Computer im gleichen Netz befinden wie das Zielobjekt (in diesem Fall ein iPad). Einfachheitshalber und um den Betrieb des HSR-Netzes nicht zu stören werden diese zwei Geräte über einen WLAN-Router verbunden. Dieser ist mit dem HSR-Netz verbunden.

Das Netz besteht für diesen Versuchsaufbau aus den in Tabelle B.1 aufgelisteten Komponenten.

IP-Adresse / Subnetzmaske	Bezeichnung
10.0.0.1/24	WLAN-Router Netgear Wireless Router WNR1000
10.0.0.3/24	iPad 16 GB Version: 5.1.1 (9B206), Modell: MB292FD
10.0.0.6/24	Windows 7 Computer mit installiertem Burp Proxy

Tabelle B.1: Burp-Versuch: Netzkomponenten

Um diesen Proxy nun auch verwenden zu können, sind ein paar Konfigurationen nötig. Im Burp muss beim Reiter «Proxy» der intercept-Modus abgeschaltet werden. Wenn dieser Modus abgeschaltet ist (also mittels Klick auf «intercept is on»), ist es nicht nötig jede einzelne HTTP-Anfrage zu bestätigen. Da die Standardkonfiguration von Burp nur Verbindungen vom lokalen Loopback-Interface des Rechners entgegen nimmt, muss dies ebenfalls angepasst werden. Dies kann durch das Entfernen eines Häkchens angepasst werden.

Abbildung B.1: «intercept is off» wurde aktiviert

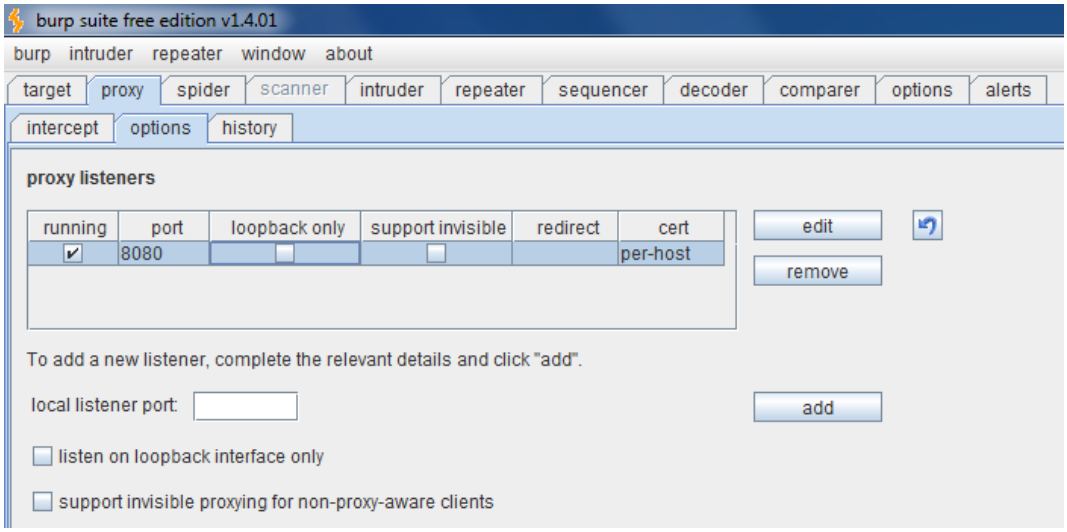


Abbildung B.2: Das Häkchen von «listen on loopback interface only» wurde entfernt

Weitere Konfigurationen am Burp Proxy sind nicht mehr notwendig. Die Ansicht kann also auf den Reiter «Target» gewechselt werden. Aktuell sollte diese Anzeige noch leer sein und keine Einträge haben. Schliesslich wurde der Proxy ja noch niemandem bekannt gegeben.

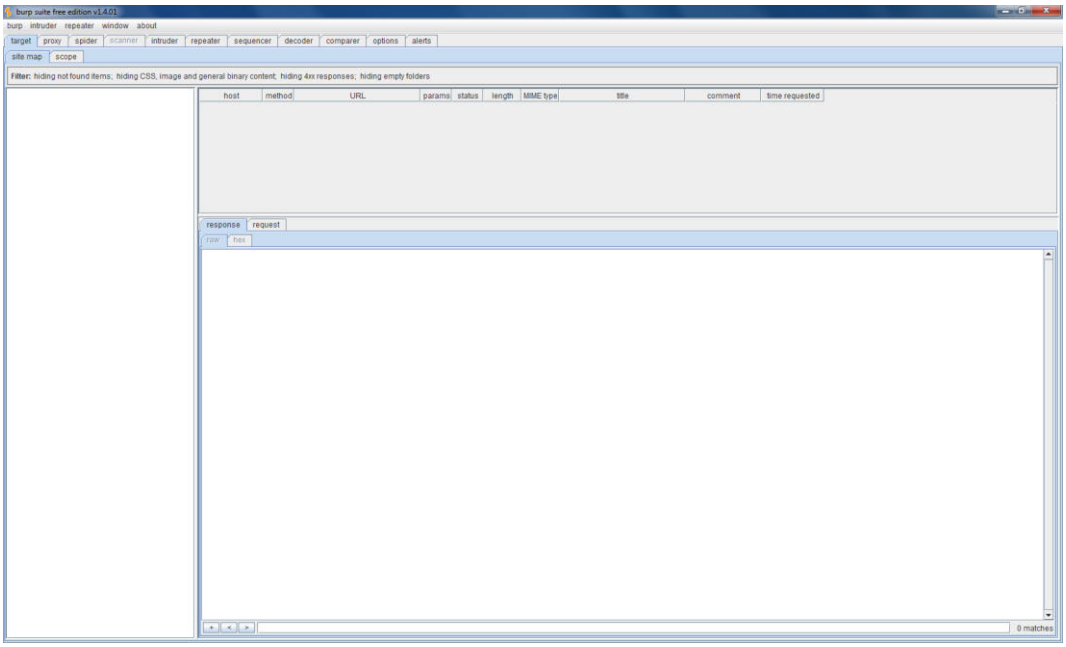


Abbildung B.3: Aktuell wurden noch keine Pakete über den Burp Proxy geleitet

Dies kann geändert werden indem auf dem iPad die Angaben zum Proxy ergänzt werden. Diese Konfiguration kann bei der entsprechenden WLAN-Verbindung vorgenommen werden. In diesem Fall betrifft es die SSID «rzahler». Für die Angabe des Proxy's wird die IP-Adresse sowie der Port benötigt, auf dem Burp läuft. Diese Angaben sind mittels dem Befehl «ipconfig -all» beziehungsweise aus der Burp-Applikation ablesbar.

Abbildung B.4: Übersicht der verfügbaren WLAN-Netzwerke auf dem iPad

Abbildung B.5: Eintragen der Angaben des Burp Proxy

Wird jetzt auf dem iPad eine Webseite wie beispielsweise «www.google.ch» aufgerufen, so läuft sämtlicher Verkehr über den Proxy, also über einen Man-in-the-Middle. Einfache HTTP-Anfragen sind somit im Klartext sichtbar:

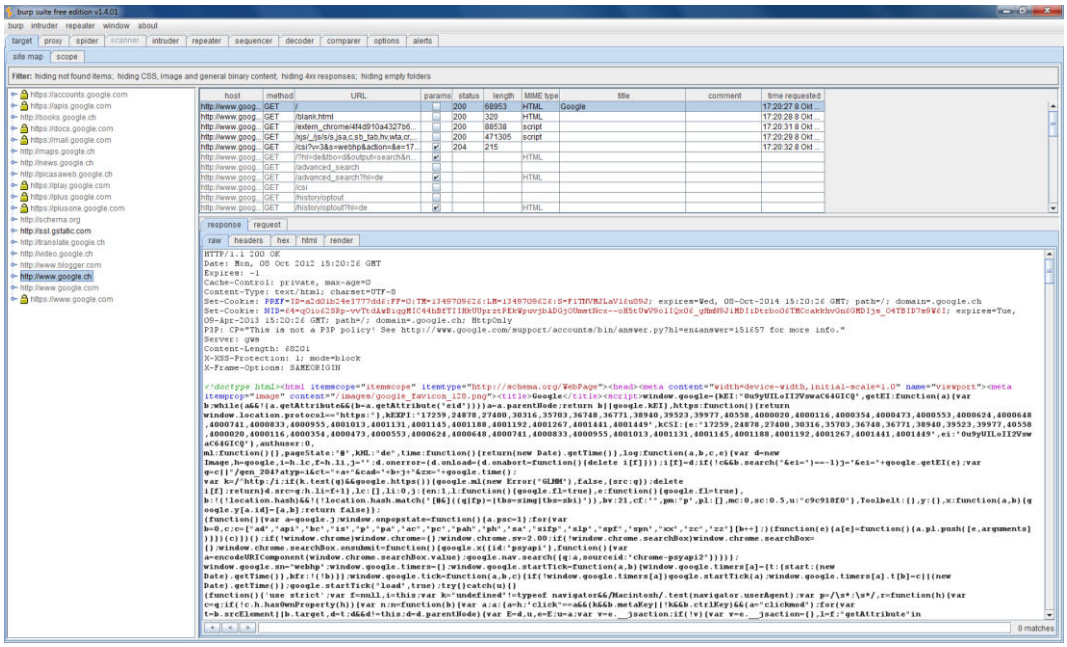


Abbildung B.6: generierte Einträge im Burp Proxy durch die Anfrage auf «http://www.google.ch»

Diese Erkenntnis, dass alles im Klartext sichtbar ist, ist recht beunruhigend. Doch zum Glück gibt es noch die verschlüsselten HTTPS-Verbindungen. Wie sich jedoch schnell zeigt, können auch diese mit Burp umgangen werden. Am einfachsten lässt sich dies mit dem Webzugang auf den Mailserver der Hochschule für Technik Rapperswil, kurz HSR, zeigen.

Auf dem iPad wird nun «https://webmail.hsr.ch» aufgerufen. Dabei erscheint eine Sicherheitswarnung betreffend dem Zertifikat.

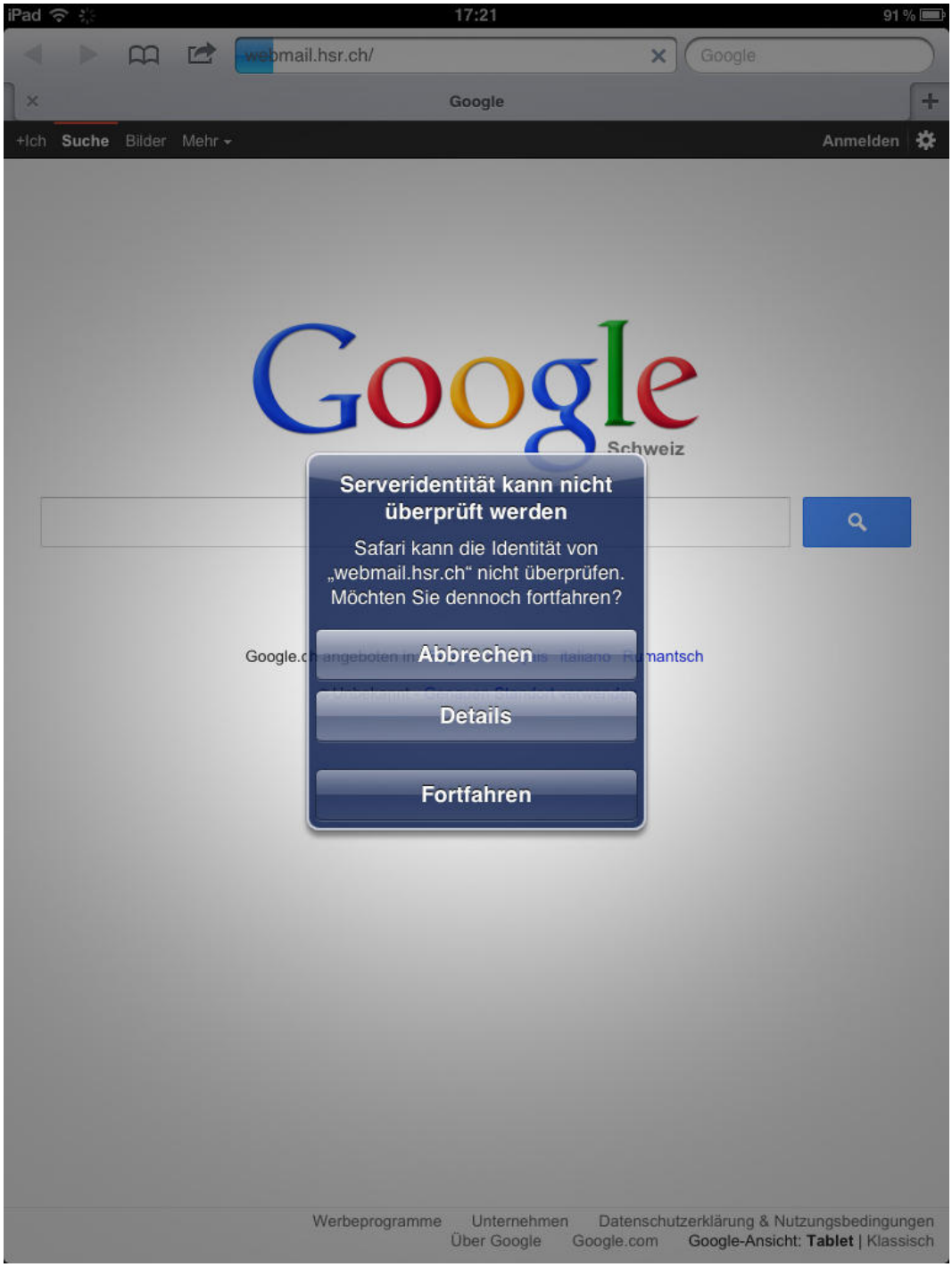


Abbildung B.7: Das iPad warnt den Benutzer betreffend der Serveridentität der «https://webmail.hsr.ch»-Webseite

Wie sich beim persönlichen Test schnell gezeigt hat, klickt der Benutzer des iPads ganz automatisch auf «Fortfahren». Schliesslich muss das Zertifikat angenommen werden. Wird dieses nicht angenommen, kommt man nicht weiter, und in der Regel geht man ja nicht davon aus, dass dieses Zertifikat gefälscht ist. Wenn das Zertifikat jedoch genauer betrachtet wird, zeigt sich schnell, dass dieses Zertifikat nicht vom HSR Mailserver kommt.

Abbildung B.8: Kurzform des gefälschten Zertifikats

Abbildung B.9: Detaillierte Angaben zum gefälschten Zertifikat

Wie oben erwähnt, wurde jedoch bereits auf «Fortfahren» geklickt, sodass der gesamte Verkehr des iPads unter der Obhut des Burp-Proxy ist. Es wurden zwei unterschiedliche verschlüsselte Verbindungen aufgebaut. Zum einen besteht eine verschlüsselte Verbindung

zwischen dem iPad und dem Burp-Proxy, welche mit dem gefälschten Zertifikat aufgebaut wurde. Zum anderen besteht eine verschlüsselte Verbindung zwischen dem Burp-Proxy und dem Mailserver der Hochschule. Diese Verbindung wurde mit dem offiziellen HSR-Zertifikat verschlüsselt. Der Proxy, also der Man-in-the-Middle, sieht somit sämtliche Daten im Klartext (obwohl die Übermittlung der Daten eigentlich verschlüsselt ist). (siehe Abbildung 2.1)

Um dies zu verdeutlichen, wird beim obigen Beispiel weitergefahren und es kann eine Anmeldung mit dem Benutzername «hmuster» und dem Kennwort «hansli» simuliert werden.

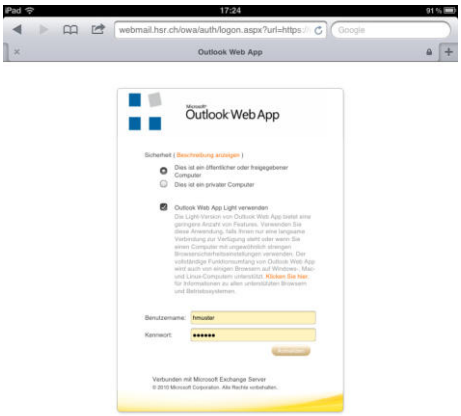


Abbildung B.10: Anzeige der gewöhnlichen Loginmaske von «https://webmail.hsr.ch»

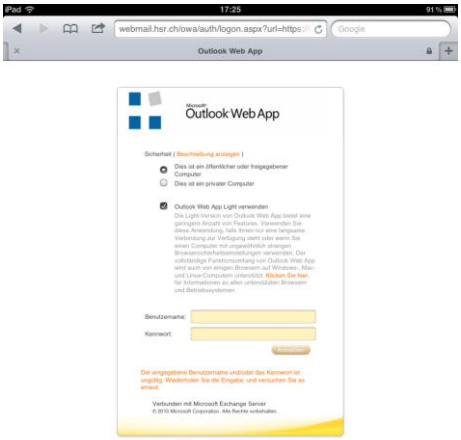


Abbildung B.11: Anmeldung auf «https://webmail.hsr.ch» fehlgeschlagen

Selbstverständlich sind diese Angaben frei erfunden, sodass eine Anmeldung auf dem Mailserver nicht möglich ist. Ein Blick auf den Burp-Proxy zeigt jedoch einen POST-Befehl, welcher die Anmeldedaten im Klartext beinhaltet.

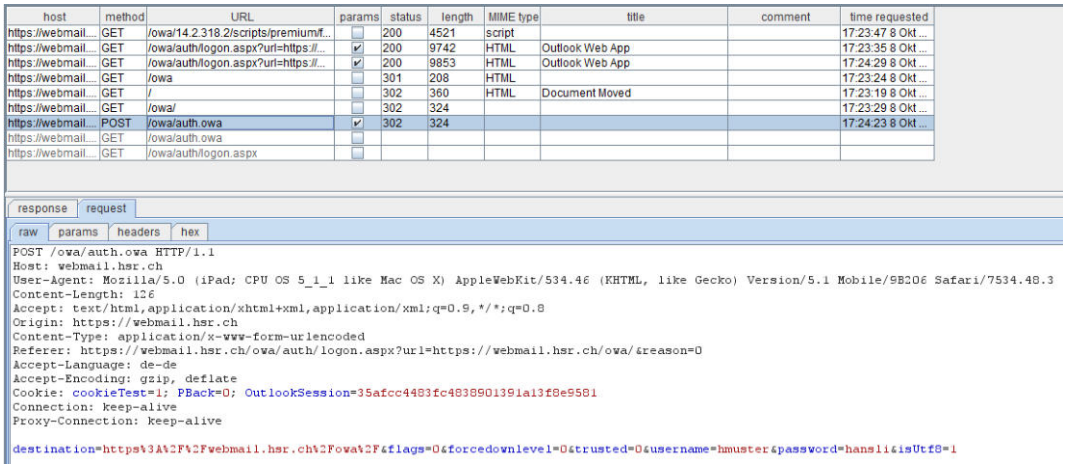


Abbildung B.12: «POST»-Befehl von «https://webmail.hsr.ch» im Burp Proxy. Der Benutzername sowie das Passwort ist im Klartext ersichtlich.

Fazit

Die Erkenntnisse, welche mit diesem Burp Proxy gefunden werden konnten, sind sehr beängstigend. Wie es scheint, sind sämtliche Daten im Klartext zu lesen, selbst bei eigentlich

verschlüsselten Verbindungen.

Natürlich kann man dies beschönigen, indem begründet wird, dass der Proxy beim Endgerät konfiguriert werden muss. Unter gewissen Umständen kann dies jedoch schneller eintreten als überhaupt gedacht. Als mögliches Beispiel wäre ein Restaurant oder Café denkbar, welches seinen Kunden WLAN anbietet. Um dies zu nutzen, müsse der Kunde jedoch ein Proxy eintragen. Einen Grund weshalb ein solcher Proxy gebraucht wird, kann eigentlich frei erfunden werden (beispielsweise «damit die Bandbreite fair aufgeteilt wird»). Benötigt ein Kunde den WLAN-Zugang, so tätigt dieser den Eintrag und der Verkehr kann nun analysiert und ausspioniert werden.

Selbstverständlich soll hier keinem Unternehmen vorgeworfen werden, dass dies auch wirklich so gemacht wird. Es wäre jedoch eine denkbare Möglichkeit, wie der Proxy beim Endgerät eingetragen wird.

Sobald der Proxy beim Endgerät einmal eingetragen ist, kann der Man-in-the-Middle sämtlichen unverschlüsselten Datenverkehr analysieren. Verschlüsselte Daten können nicht mit hundertprozentiger Sicherheit analysiert werden, da der Benutzer das Zertifikat akzeptieren muss. Es kann aber davon ausgegangen werden, dass dies automatisch akzeptiert wird, schliesslich geht man in der Regel davon aus, dass das Zertifikat korrekt ist. Wird das Zertifikat nicht akzeptiert, schlägt die Verbindung fehl und der Benutzer kann sich die Seite nicht anzeigen lassen. Dies ist dann oftmals nicht im Sinne des Benutzers, sodass er das Zertifikat dann schlussendlich akzeptiert.

B.2 WebScarab

Als Zweites wird als Alternative der WebScarab Proxy betrachtet, welcher wie der Burp Proxy HTTP- und HTTPS-Protokolle analysieren lässt. Gemäss der «OWASP WebScarab Project [22]»-Webseite wird WebScarab wie folgt erklärt:

> WebScarab is a framework for analysing applications that communicate using the HTTP and HTTPS protocols. [...] In its most common usage, WebScarab operates as an intercepting proxy, allowing the operator to review and modify requests created by the browser before they are sent to the server, and to review and modify responses returned from the server before they are received by the browser. [...]

Die Applikation selbst ist, wie der Burp Proxy, in Java geschrieben und somit Plattform unabhängig. Im zweiten Testversuch wird versucht, einen transparenten Proxy aufzubauen. Ein transparenter Proxy bezeichnet einen Proxy, welcher dem Endgerät nicht bekannt gegeben wird. Das Endgerät soll also nichts vom Proxy wissen und normal kommunizieren können. Aus diesem Grund wird die Linux Version «BackTrack» als Betriebssystem verwendet. Dieses System ist speziell zur Überprüfung der Sicherheit innerhalb eines Netzwerkes gedacht. Entsprechende Tools sind dabei schon vorinstalliert und können so direkt genutzt werden. [17]

BackTrack kann am einfachsten als virtuelle Maschine genutzt werden. Das entsprechende ZIP-Archiv kann beispielsweise direkt vom Mirror-Server [1] von Switch herunterge-

[1] http://mirror.switch.ch/ftp/mirror/backtrack/BT5R3-GNOME-VM-32.7z

laden werden.

Nach dem Herunterladen und Entpacken kann die virtuelle Maschine mittels VMware Player gestartet werden. Vor dem Starten sollte darauf geachtet werden, dass der Netzwerkadapter sich im «Bridge-Modus» befindet. So erhält die virtuelle Maschine vom DHCP-Server eine eigene IP-Adresse. Für das Starten des Betriebssystems ist der Benutzername «root» und das Passwort «toor» notwendig. Um die graphische Oberfläche zu starten, muss der Befehl «startx» ausgeführt werden.

Um auf die eigentliche Anwendung, WebScarab, zurückzukommen, wird nun die neuste Version von WebScarab heruntergeladen. Am einfachsten funktioniert dies über die «SourceForge.net»-Webseite [2], wo sich das JAR-Archiv «webscarab-selfcontained-[numbers].jar» befindet.

Nach dem Herunterladen der JAR-Datei kann dieses mit dem Befehl «java -jar ./webscarab-selfcontained-[numbers].jar» ausgeführt werden. Wenn die JAR-Datei ausgeführt wird, kann WebScarab im gleichen Umfang wie Burp genutzt werden. Auch hier sind einige Anpassungen noch notwendig, welche später erläutert werden. Da WebScarab in diesem Versuch für Android-Geräte genutzt werden soll, sollte die Applikation noch nicht gestartet werden.

Zuerst ist ein sogenanntes ARP-Spoofing notwendig. Dies ist notwendig, da es beim Android-Betriebssystem nicht so einfach ist, einen systemweiten Proxy zu setzen. Mit dem ARP-Spoofing wird dem Endgerät (beispielsweise einem Android-Gerät) angegeben, dass der BackTrack-Computer der Gateway sei. Gegenüber dem Gateway / WLAN-Router gibt sich der BackTrack-Computer als Endgerät aus. So schickt das Endgerät sämtliche Pakete an den BackTrack-Computer. Dieser kann den Verkehr analysieren und weiterleiten. Der Gateway geht davon aus, dass der BackTrack-Computer das Endgerät sei. So können auch die Antworten auf dem BackTrack-System analysiert und entsprechend weitergeleitet werden.

Um den Testversuch etwas zu vereinfachen, wird anstatt einem Android-Gerät ein Windows 7 Computer als Endgerät eingesetzt. Für diesen Versuch werden folgende Komponenten genutzt:

IP-Adresse / Subnetzmaske	**Bezeichnung**
10.0.0.1/24	WLAN-Router Netgear Wireless Router WNR1000
10.0.0.6/24	Windows 7 Computer
10.0.0.8/24	BackTrack (Version 5R3) System

Tabelle B.2: WebScarab-Versuch: Netzkomponenten

Das ARP-Spoofing kann mit folgenden Befehlen gestartet werden:

- «arpspoof -t 10.0.0.6 10.0.0.1»
- «arpspoof -t 10.0.0.1 10.0.0.6»

Der erste Befehl schickt an das Zielsystem mit der IP-Adresse 10.0.0.6 gefälschte ARP-Pakete, die es glauben machen, dass der WLAN-Router mit der IP-Adresse 10.0.0.1 unter

[2] http://sourceforge.net/projects/owasp/files/WebScarab/

der MAC-Adresse BackTrack-Rechners erreichbar ist. Der zweite Befehl macht dasselbe für den WLAN-Router. Dabei wird ihm mitgeteilt, dass sich das Zielsystem (10.0.0.6) an der MAC-Adresse des BackTrack-Systems befindet.

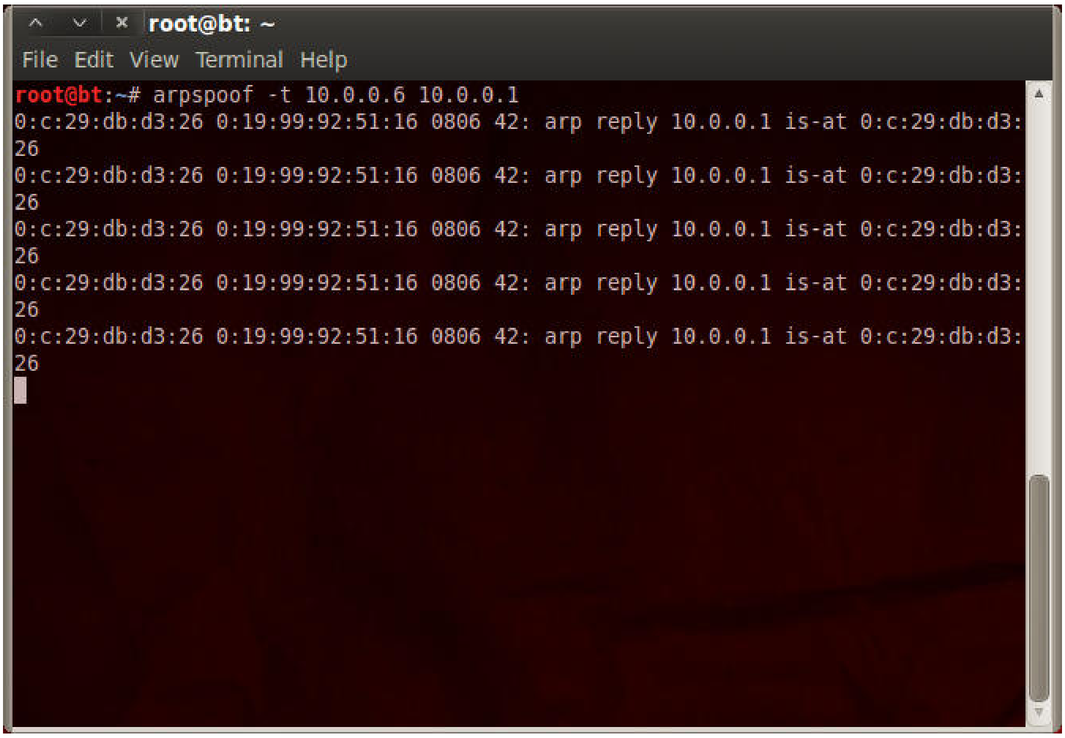


Abbildung B.13: «arpspoof -t 10.0.0.6 10.0.0.1» läuft

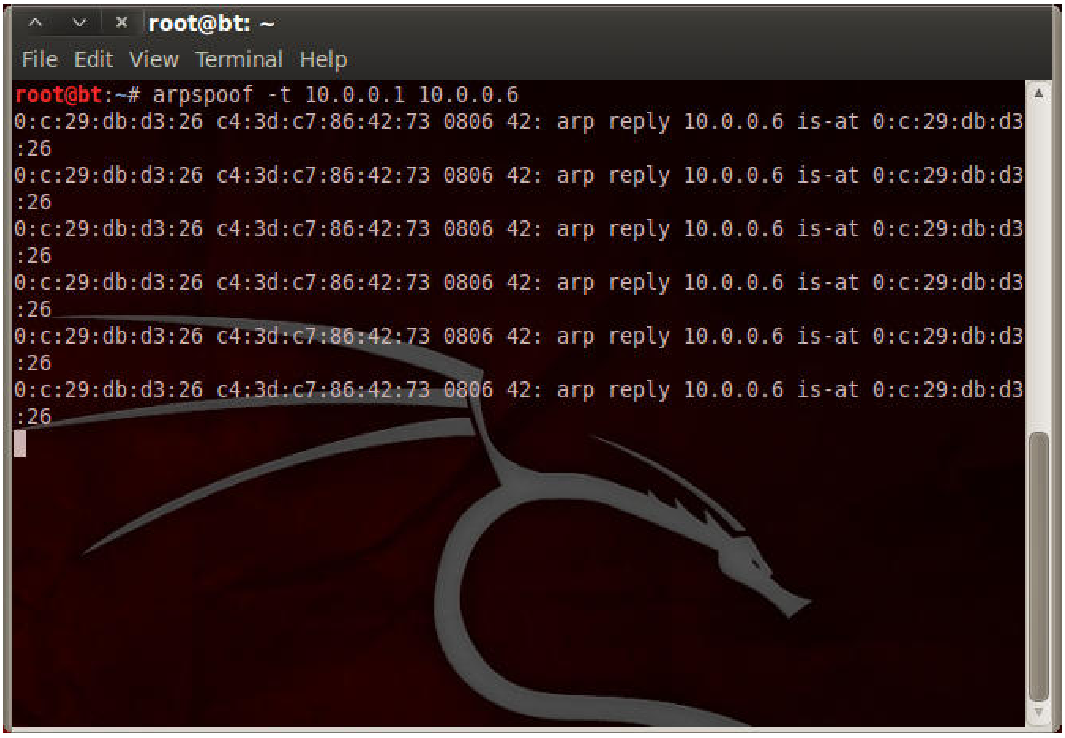


Abbildung B.14: «arpspoof -t 10.0.0.1 10.0.0.6» läuft

Der Zielrechner hat im Anschluss keinen Zugang mehr ins Internet. Sämtliche HTTP-Anfragen und Ping-Versuche enden mit Fehlern.

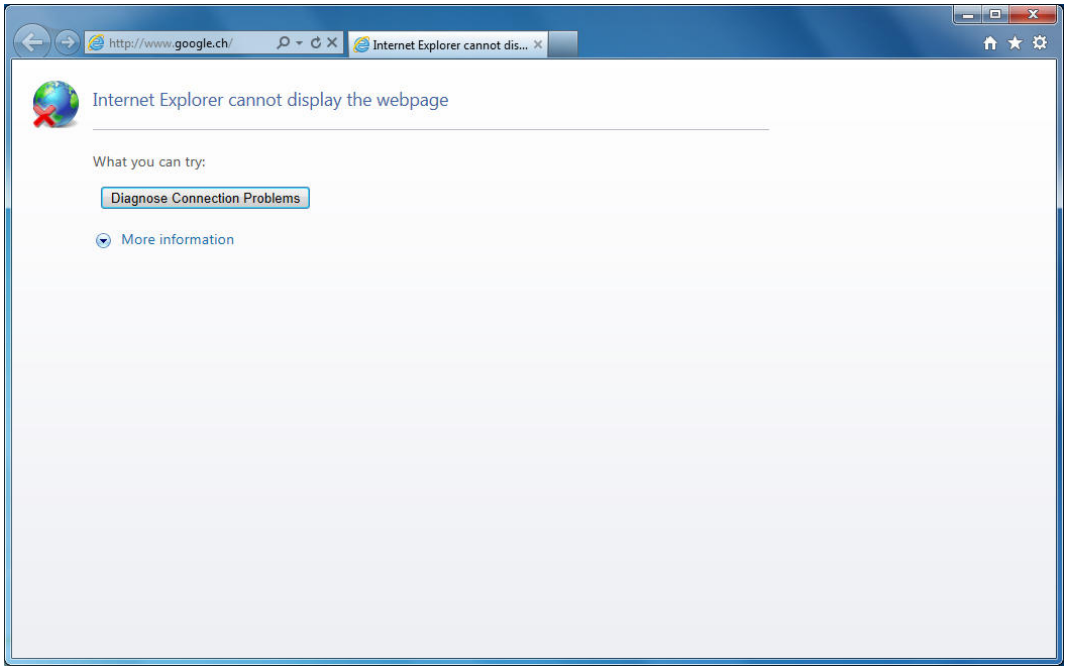


Abbildung B.15: «http://www.google.ch» konnte nicht gefunden werden

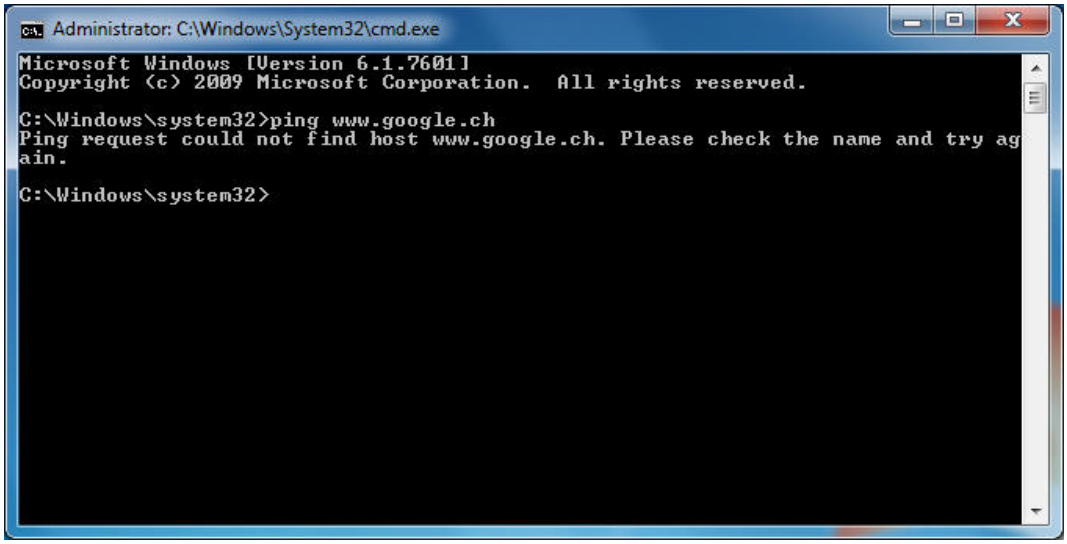


Abbildung B.16: Host konnte nicht gefunden werden für «ping www.google.ch»

Um dies zu korrigieren, muss der BackTrack-Computer als «Router» konfiguriert werden. Mit dem Befehl «echo 1 > /proc/sys/net/ipv4/ip_forward» wird dies erreicht. Das BackTrack-System leitet dabei sämtliche IP-Pakete weiter, sodass Ping-Versuche und HTTP-

Anfragen wieder möglich sind.

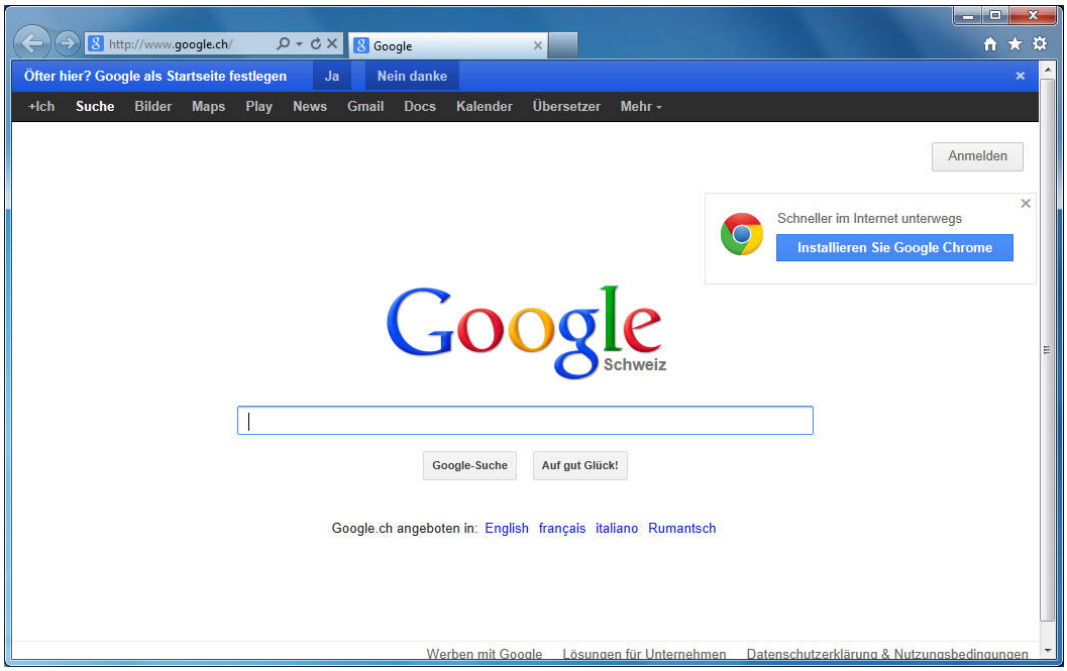


Abbildung B.17: «http://www.google.ch» kann wieder angezeigt werden

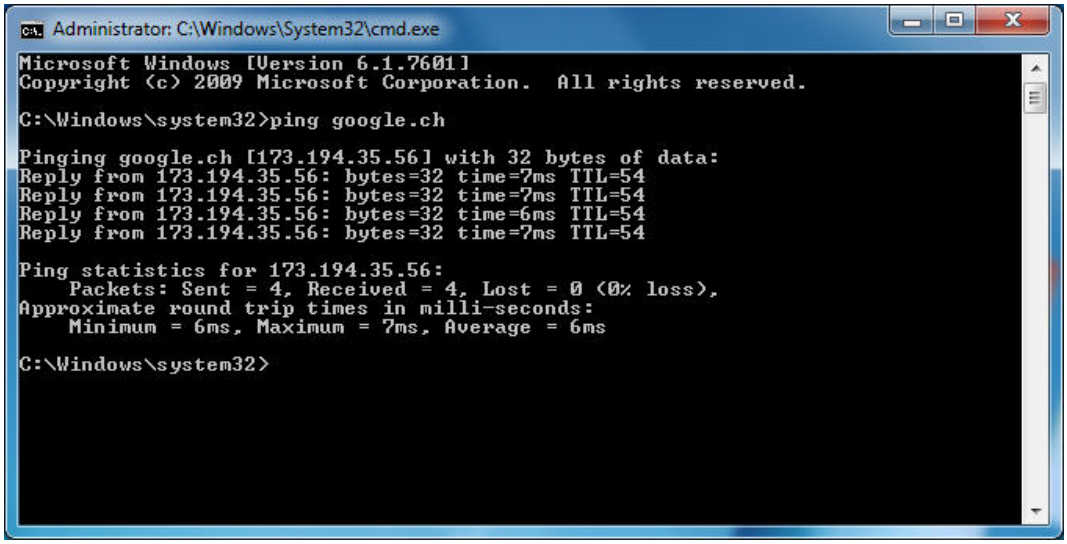


Abbildung B.18: Ping Anfrage auf «www.google.ch» wird beantwortet

Um WebScarab zu testen, wird dieser gestartet. Beim ersten Start ist dabei noch eine Änderung notwendig, um sämtliche Funktionalitäten des WebScarab zu nutzen. Dies kann mittels «Tools -> Use full-featured interface» angepasst werden. Dadurch werden sämtliche Funktionen von WebScarab aktiviert. Um die Änderung zu übernehmen ist ein Neustart des WebScarab notwendig.

Im Anschluss kann im Reiter «Proxy», beim Register «Listener», der vorhandene und vordefinierte Proxy angeschaut werden. Wie erkennt werden kann, lautet die Adresse «127.0.0.1» (was dem Localhost entspricht) und hört auf den Port «8008». Die Adresse muss auf «*» geändert werden.

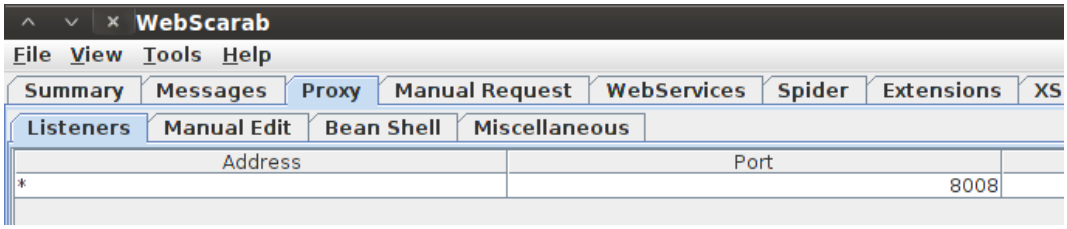


Abbildung B.19: WebScarab Proxy-Konfiguration

Da jedoch dieser Proxy auf dem Zielsystem nicht angegeben werden soll, müssen die «iptables» des BackTrack-Systems angepasst werden. Aus diesem Grund sind folgende Befehle notwendig:

- «iptables -t nat -A PREROUTING -i eth0 -p tcp –dport 80 -j REDIRECT –to-port 8008»
- «iptables -t nat -A PREROUTING -i eth0 -p tcp –dport 443 -j REDIRECT –to-port 8008»

Befehl (/ Zusatz)	**Beschreibung**
iptables	Mit iptables kann man Regeln definieren die den Netzwerkzugriff für bestimmte IP Adressen erlauben oder verwehren. Definiert man die richtigen Regeln, kann man sein System vor unbefugten Zugriffen schützen und diese schon am Netzwerkinterface abweisen.
-t nat	-t Tabelle Diese Filterregel gilt für die Tabelle «Tabelle».
-A PREROUTING	-A Chain Regel wird an die Kette «Chain» angehängt.
-i eth0	-i Netzwerkschnittstelle Das Paket wird nur geprüft, wenn es über die definierte Netzwerkschnittstelle eingegangen ist.
-p tcp	-p Protokoll Das Paket wird nur geprüft, wenn es gemäss «IP-Protokoll» ist (z.B. TCP, UDP, ICMP).
–dport 80	–dport Port-Nr oder -destination-port Port-Nr Das Paket wird nur geprüft, wenn es an die definierte Port-Nummer gesendet wird. Muss zwingend in Verbindung mit -p benutzt werden!
-j REDIRECT	-j Aktion Legt fest, welche Aktion auf das Paket angewendet werden soll, wenn alle Prüfkriterien erfüllt wurden.
–to-port 8008	–to-port Legt fest, an welchen Port diese Pakete weitergeleitet werden

Tabelle B.3: «iptables»-Befehlserklärung [23]

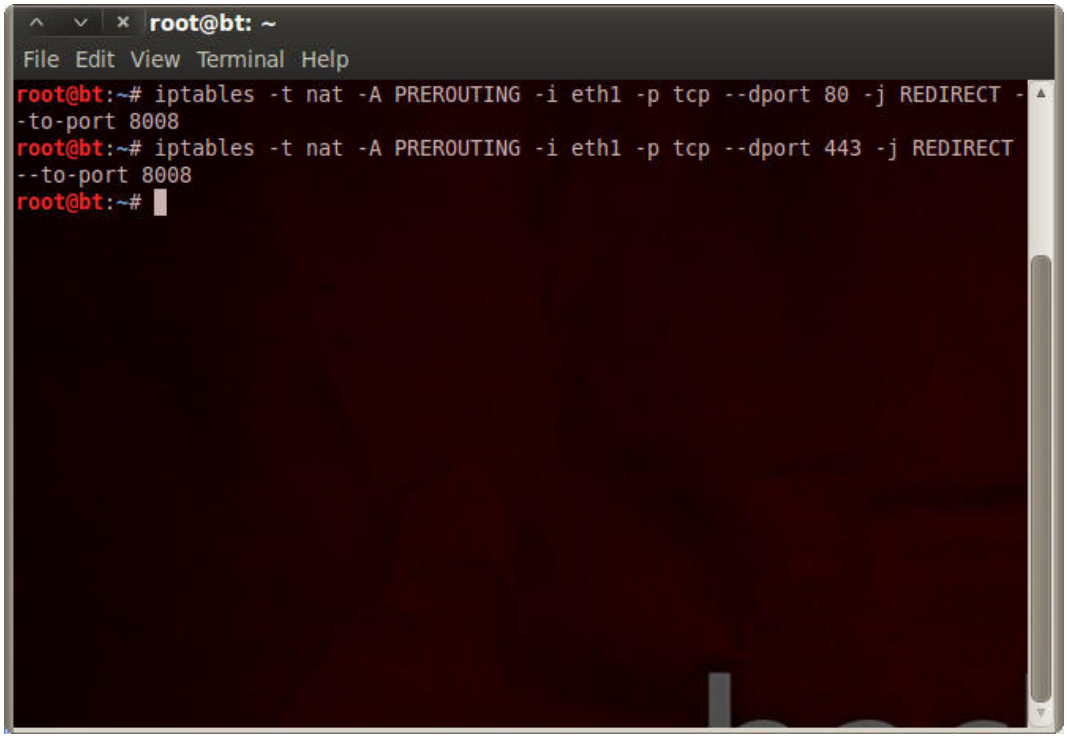


Abbildung B.20: «iptables»-Einträge wurden gesetzt

Nach dieser Eingabe scheitert jedoch die HTTP-Anfrage auf «http://www.google.ch». Ping Versuche sind weiterhin möglich.

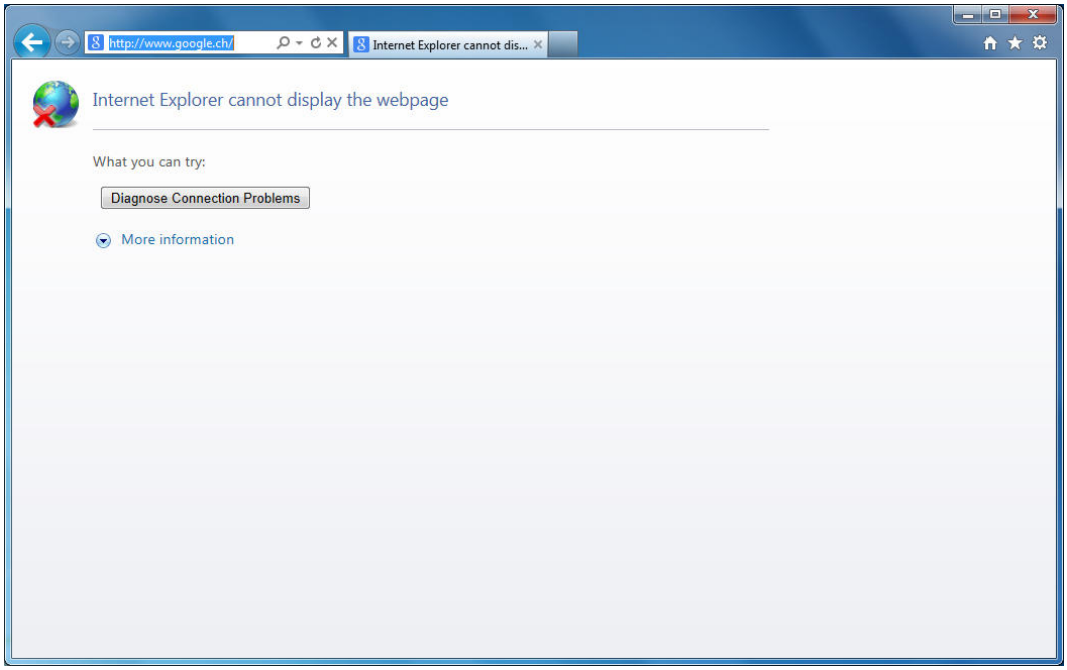


Abbildung B.21: «http://www.google.ch» kann nicht mehr gefunden werden

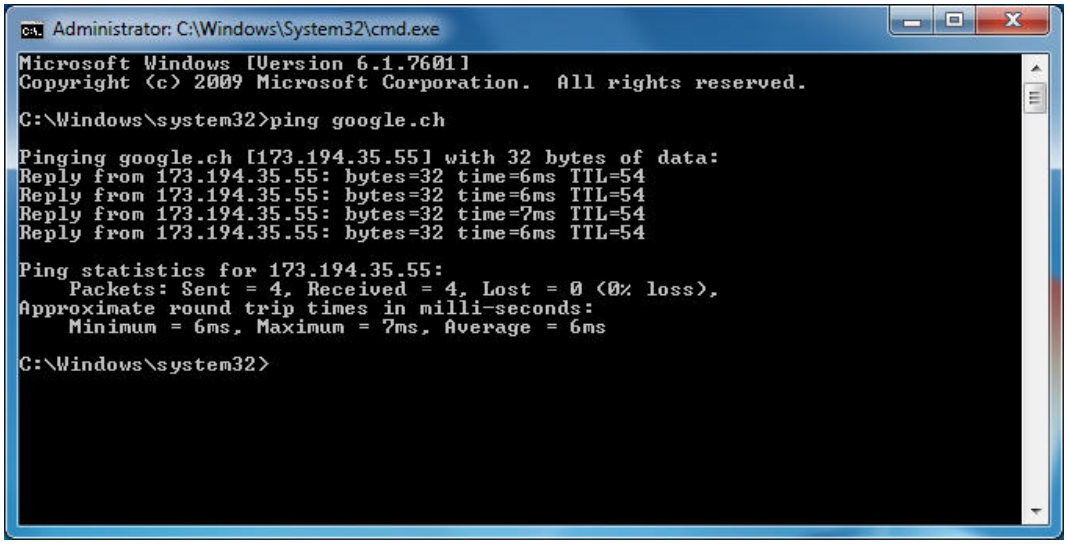


Abbildung B.22: Ping Anfrage auf «www.google.ch» wird weiterhin beantwortet

Nach Rücksprache mit Herrn Prof. Dr. Peter Heinzmann stellte sich heraus, dass das Benutzen der offiziellen Version des WebScarab-Proxy nur unter bestimmten Bedingungen funktioniert. Am Besten funktioniert es mittels Angabe des Proxys (analog zum Burp

Proxy). HTTP-Anfragen können so genau analysiert werden. Bei den verschlüsselten Verbindung steht man jedoch mit der offiziellen WebScarab-Version weiterhin an.

Wird eine HTTPS-Verbindung aufgebaut, liefert der WebScarab ein eigenes (gefälschtes) Zertifikat. Der Name des Zertifikats stimmt dabei nicht mit dem Namen der Webseite überein. WebScarab liefert dabei standardmässig immer das gleiche Zertifikat. Der Benutzer wird dann beispielsweise bei der Verwendung von Mozilla Firefox gewarnt: «Dieser Verbindung wird nicht vertraut». Schaut man sich die weiteren Informationen zum Zertifikat an, sieht man, dass als Zertifikat-Status angegeben wird, dass es sich um eine falsche Webseite sowie um eine unbekannte Identität handelt. Zum Vergleich: Der Burp Proxy erstellt das Zertifikat «on-the-fly». Dadurch wird das Zertifikat lediglich als «Unbekannte Identität» angesehen.

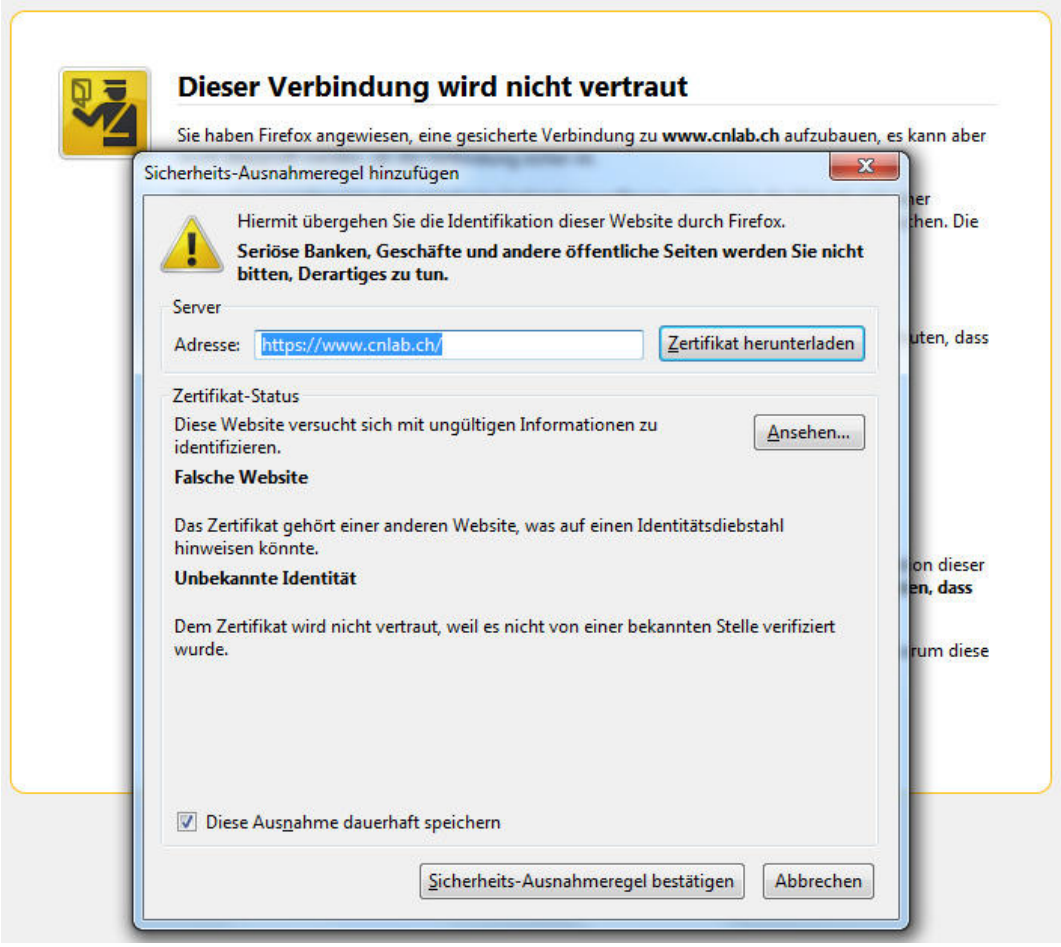


Abbildung B.23: Zertifikatswarnung im Mozilla Firefox bei Verwendung des WebScarab

Für jede verschlüsselte Webseiten müsste also ein Zertifikat vorbereitet werden. Dieser Aufwand ist ziemlich hoch, insbesondere, da der Betreiber des WebScarab-Proxy's nicht weiss, welche Webseite durch das Endgerät aufgerufen wird.

Fazit

Wie im zweiten Versuch festgestellt werden konnte, ist die Realisation eines transparenten Proxys alles andere als einfach. Die Benutzung der offiziellen Version von WebScarab eignet sich daher nicht für eine Studienarbeit. Sind jedoch die Webseiten bekannt und die Anzahl überschaubar, scheint es denkbar, die entsprechenden Zertifikate zu generieren und einzubinden.

Wird jedoch eine angepasste Version von WebScarab verwendet, welche von cnlab ausgearbeitet wurde, kann WebScarab in einem ähnlichen Umfang wie der Burp Proxy verwendet werden. Dabei kann ein eigenes Zertifikat erstellt und in WebScarab beziehungsweise beim Endgerät eingebunden werden. Verschlüsselte Verbindungen werden daraufhin ohne weitere Nachfrage aufgebrochen und im Klartext dargestellt.

Anhang C

Test

Die im Grundlagenkapitel erwähnten Apps wurden getestet und in diesem Kapitel dokumentiert.

Bei den Tests waren folgende Komponenten im Einsatz:

IP-Adresse / Subnetzmaske	Bezeichnung
10.0.0.1/24	WLAN-Router Netgear Wireless Router WNR1000
10.0.0.7/24	Windows 7 Computer mit installiertem WebScarab
10.0.0.9/24	iPhone 4 16 GB Version: 5.1.1 (9B206), Modell: MC603FD
10.0.0.12/24	iPad 3 32 GB Version 6.0.1 (10A523), Modell: MD367FD

Tabelle C.1: Test: Netzkomponenten

Die Entscheidung, ob das iPhone oder das iPad verwendet wird, ist nach eigener Einschätzung gefällt worden. Es wurde dabei das Gerät ausgewählt, welches am geeignetsten schien. So wurde beispielsweise für die «20 Minuten Online»-App das iPad gewählt, um die Nachrichten aus aller Welt in angenehmen Grösse lesen zu können.

Die Tests der Apps wurden wie folgt auf die Geräte verteilt (Tabelle C.2):

iPhone 4	iPad 3
• 20 Minuten Online • Facebook • Google Latitude • Google Plus • Zattoo • Raiffeisen	• Bad Piggies • Dropbox • iOf • Waze

Tabelle C.2: Test: Zuordnung der Apps auf die Geräte

Das verwendete iPhone beziehungsweise das iPad wurde jeweils vor dem Test auf die Werkseinstellungen zurückgesetzt. Daraufhin wurde das zu testende App sowie das CA-Zertifikat installiert.

Auf die Analyse der Android Apps wurde verzichtet, da eine genaue Analyse mit dem vorhandenen Setup nicht möglich ist.

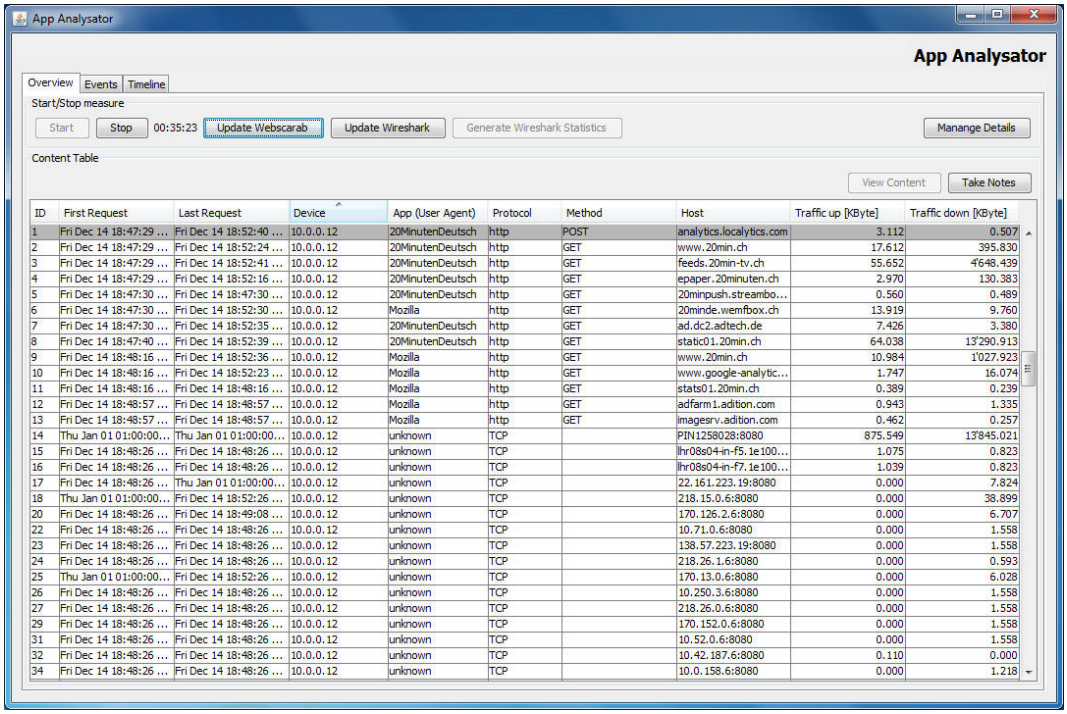
C.1 Test «20 Minuten Online»

Nach dem Starten der «20 Minuten Online»-App wurden einzelne Nachrichten abgerufen, um ein typisches Benutzerverhalten zu simulieren. Dabei wurden sowohl Nachrichten mit Bildern wie auch Werbungen geladen und auf dem Display dargestellt.

Abbildung C.1: Screenshot «20 Minuten Online»: Übersicht

Abbildung C.2: Screenshot «20 Minuten Online»: Themenbereich Wissen

Ergebnis



ID	First Request	Last Request	Device	App (User Agent)	Protocol	Method	Host	Traffic up [KByte]	Traffic down [KByte]
1	Fri Dec 14 18:47:29 ...	Fri Dec 14 18:52:40 ...	10.0.0.12	20MinutenDeutsch	http	POST	analytics.localytics.com	3.112	0.507
2	Fri Dec 14 18:47:29 ...	Fri Dec 14 18:52:24 ...	10.0.0.12	20MinutenDeutsch	http	GET	www.20min.ch	17.612	395.830
3	Fri Dec 14 18:47:29 ...	Fri Dec 14 18:52:41 ...	10.0.0.12	20MinutenDeutsch	http	GET	feeds.20min-tv.ch	55.652	4'648.439
4	Fri Dec 14 18:47:29 ...	Fri Dec 14 18:52:16 ...	10.0.0.12	20MinutenDeutsch	http	GET	epaper.20minuten.ch	2.970	130.383
5	Fri Dec 14 18:47:30 ...	Fri Dec 14 18:47:30 ...	10.0.0.12	20MinutenDeutsch	http	GET	20minpush.streambo...	0.560	0.489
6	Fri Dec 14 18:47:30 ...	Fri Dec 14 18:52:30 ...	10.0.0.12	Mozilla	http	GET	20minde.wemfbox.ch	13.919	9.760
7	Fri Dec 14 18:47:30 ...	Fri Dec 14 18:52:35 ...	10.0.0.12	20MinutenDeutsch	http	GET	ad.dc2.adtech.de	7.426	3.380
8	Fri Dec 14 18:47:40 ...	Fri Dec 14 18:52:39 ...	10.0.0.12	20MinutenDeutsch	http	GET	static01.20min.ch	64.038	13'290.913
9	Fri Dec 14 18:48:16 ...	Fri Dec 14 18:52:36 ...	10.0.0.12	Mozilla	http	GET	www.20min.ch	10.984	1'027.923
10	Fri Dec 14 18:48:16 ...	Fri Dec 14 18:52:23 ...	10.0.0.12	Mozilla	http	GET	www.google-analytic...	1.747	16.074
11	Fri Dec 14 18:48:16 ...	Fri Dec 14 18:48:16 ...	10.0.0.12	Mozilla	http	GET	stats01.20min.ch	0.389	0.239
12	Fri Dec 14 18:48:57 ...	Fri Dec 14 18:48:57 ...	10.0.0.12	Mozilla	http	GET	adfarm1.adition.com	0.943	1.335
13	Fri Dec 14 18:48:57 ...	Fri Dec 14 18:48:57 ...	10.0.0.12	Mozilla	http	GET	imagesrv.adition.com	0.462	0.257
14	Thu Jan 01 01:00:00...	Thu Jan 01 01:00:00...	10.0.0.12	unknown	TCP		PIN1258028:8080	875.549	13'845.021
15	Fri Dec 14 18:48:26 ...	Fri Dec 14 18:48:26 ...	10.0.0.12	unknown	TCP		lhr08s04-in-f5.1e100...	1.075	0.823
16	Fri Dec 14 18:48:26 ...	Fri Dec 14 18:48:26 ...	10.0.0.12	unknown	TCP		lhr08s04-in-f7.1e100...	1.039	0.823
17	Fri Dec 14 18:48:26 ...	Thu Jan 01 01:00:00...	10.0.0.12	unknown	TCP		22.161.223.19:8080	0.000	7.824
18	Thu Jan 01 01:00:00...	Fri Dec 14 18:52:26 ...	10.0.0.12	unknown	TCP		218.15.0.6:8080	0.000	38.899
20	Fri Dec 14 18:48:26 ...	Fri Dec 14 18:49:08 ...	10.0.0.12	unknown	TCP		170.126.2.6:8080	0.000	6.707
22	Fri Dec 14 18:48:26 ...	Fri Dec 14 18:48:26 ...	10.0.0.12	unknown	TCP		10.71.0.6:8080	0.000	1.558
23	Fri Dec 14 18:48:26 ...	Fri Dec 14 18:48:26 ...	10.0.0.12	unknown	TCP		138.57.223.19:8080	0.000	1.558
24	Fri Dec 14 18:48:26 ...	Fri Dec 14 18:48:26 ...	10.0.0.12	unknown	TCP		218.26.1.6:8080	0.000	0.593
25	Thu Jan 01 01:00:00...	Fri Dec 14 18:52:26 ...	10.0.0.12	unknown	TCP		170.13.0.6:8080	0.000	6.028
26	Fri Dec 14 18:48:26 ...	Fri Dec 14 18:48:26 ...	10.0.0.12	unknown	TCP		10.250.3.6:8080	0.000	1.558
27	Fri Dec 14 18:48:26 ...	Fri Dec 14 18:48:26 ...	10.0.0.12	unknown	TCP		218.26.0.6:8080	0.000	1.558
29	Fri Dec 14 18:48:26 ...	Fri Dec 14 18:48:26 ...	10.0.0.12	unknown	TCP		170.152.0.6:8080	0.000	1.558
31	Fri Dec 14 18:48:26 ...	Fri Dec 14 18:48:26 ...	10.0.0.12	unknown	TCP		10.52.0.6:8080	0.000	1.558
32	Fri Dec 14 18:48:26 ...	Fri Dec 14 18:48:26 ...	10.0.0.12	unknown	TCP		10.42.187.6:8080	0.110	0.000
34	Fri Dec 14 18:48:26 ...	Fri Dec 14 18:48:26 ...	10.0.0.12	unknown	TCP		10.0.158.6:8080	0.000	1.218

Abbildung C.3: AppAnalysator-«Overview»: «20 Minuten Online»

Gemäss dem AppAnalysator wurden die Nachrichten über HTTP-Verbindungen abgerufen. Nennenswerte Verbindungen gibt es nicht, lediglich eine HTTP-Verbindungen sendet einen POST-Request. Diese wurde für die Statistik aufgebaut.

Auffallend ist jedoch, dass ziemlich viele Daten übertragen wurden. In fünf Minuten wurden etwas mehr als 14 MB von «20min.ch» heruntergeladen. Die Datenmengen sind auf die Nachrichten und deren Inhalte wie Bilder zurück zu führen. Aufgrund der Displaygrösse werden dadurch qualitativ hochstehende Bilder geladen und auf dem iPad angezeigt.

C.2 Test «Bad Piggies»

Um das Benutzerverhalten bei «Bad Piggies» zu simulieren, wurden einzelne Spielrunden gemacht.

Abbildung C.4: Screenshot «Bad Piggies»: Spielstart

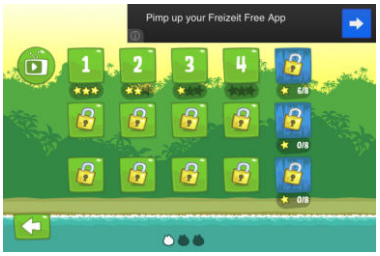
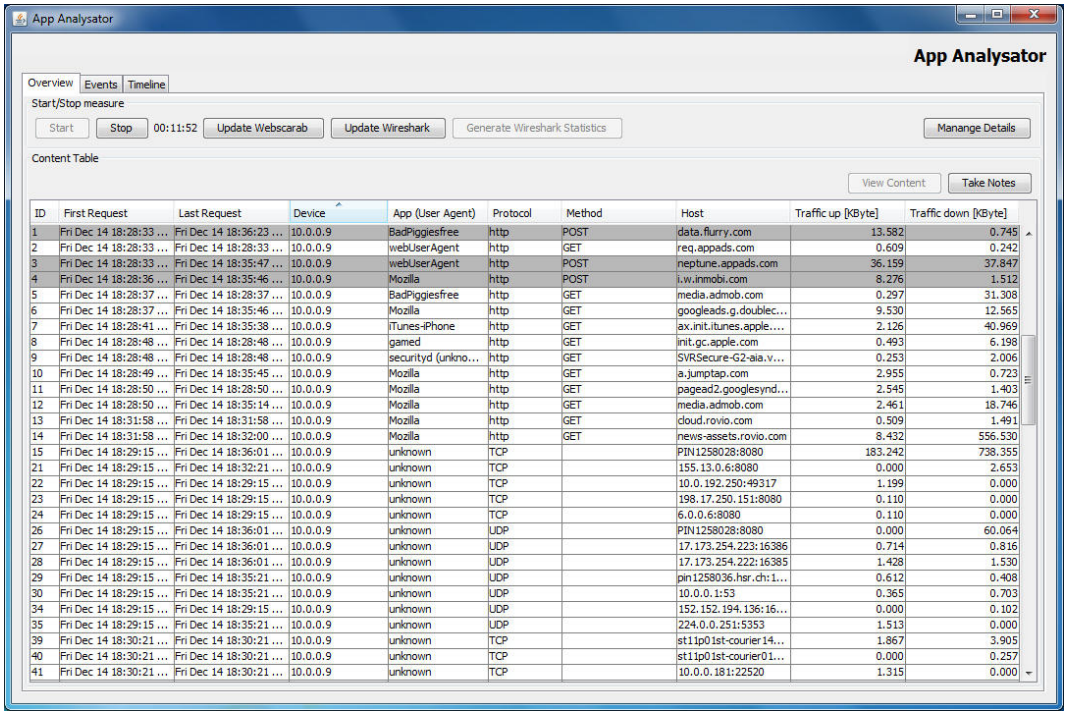


Abbildung C.5: Screenshot «Bad Piggies»: Level Übersicht

Ergebnis



ID	First Request	Last Request	Device	App (User Agent)	Protocol	Method	Host	Traffic up [KByte]	Traffic down [KByte]
1	Fri Dec 14 18:28:33 ...	Fri Dec 14 18:36:23 ...	10.0.0.9	BadPiggiesfree	http	POST	data.flurry.com	13.582	0.745
2	Fri Dec 14 18:28:33 ...	Fri Dec 14 18:28:33 ...	10.0.0.9	webUserAgent	http	GET	req.appads.com	0.609	0.242
3	Fri Dec 14 18:28:33 ...	Fri Dec 14 18:35:47 ...	10.0.0.9	webUserAgent	http	POST	neptune.appads.com	36.159	37.847
4	Fri Dec 14 18:28:36 ...	Fri Dec 14 18:35:46 ...	10.0.0.9	Mozilla	http	POST	i.w.inmobi.com	8.276	1.512
5	Fri Dec 14 18:28:37 ...	Fri Dec 14 18:28:37 ...	10.0.0.9	BadPiggiesfree	http	GET	media.admob.com	0.297	31.308
6	Fri Dec 14 18:28:37 ...	Fri Dec 14 18:35:46 ...	10.0.0.9	Mozilla	http	GET	googleads.g.doublec...	9.530	12.565
7	Fri Dec 14 18:28:41 ...	Fri Dec 14 18:35:38 ...	10.0.0.9	iTunes-iPhone	http	GET	ax.init.itunes.apple....	2.126	40.969
8	Fri Dec 14 18:28:48 ...	Fri Dec 14 18:28:48 ...	10.0.0.9	gamed	http	GET	init.gc.apple.com	0.493	6.198
9	Fri Dec 14 18:28:48 ...	Fri Dec 14 18:28:48 ...	10.0.0.9	securityd (unkno...	http	GET	SVRSecure-G2-aia.v...	0.253	2.006
10	Fri Dec 14 18:28:49 ...	Fri Dec 14 18:35:45 ...	10.0.0.9	Mozilla	http	GET	a.jumptap.com	2.955	0.723
11	Fri Dec 14 18:28:50 ...	Fri Dec 14 18:28:50 ...	10.0.0.9	Mozilla	http	GET	pagead2.googlesynd...	2.545	1.403
12	Fri Dec 14 18:28:50 ...	Fri Dec 14 18:35:14 ...	10.0.0.9	Mozilla	http	GET	media.admob.com	2.461	18.746
13	Fri Dec 14 18:31:58 ...	Fri Dec 14 18:31:58 ...	10.0.0.9	Mozilla	http	GET	cloud.rovio.com	0.509	1.491
14	Fri Dec 14 18:31:58 ...	Fri Dec 14 18:32:00 ...	10.0.0.9	Mozilla	http	GET	news-assets.rovio.com	8.432	556.530
15	Fri Dec 14 18:29:15 ...	Fri Dec 14 18:36:01 ...	10.0.0.9	unknown	TCP		PIN1258028:8080	183.242	738.355
21	Fri Dec 14 18:29:15 ...	Fri Dec 14 18:32:21 ...	10.0.0.9	unknown	TCP		155.13.0.6:8080	0.000	2.653
22	Fri Dec 14 18:29:15 ...	Fri Dec 14 18:29:15 ...	10.0.0.9	unknown	TCP		10.0.192.250:49317	1.199	0.000
23	Fri Dec 14 18:29:15 ...	Fri Dec 14 18:29:15 ...	10.0.0.9	unknown	TCP		198.17.250.151:8080	0.110	0.000
24	Fri Dec 14 18:29:15 ...	Fri Dec 14 18:29:15 ...	10.0.0.9	unknown	TCP		6.0.0.6:8080	0.110	0.000
26	Fri Dec 14 18:29:15 ...	Fri Dec 14 18:36:01 ...	10.0.0.9	unknown	UDP		PIN1258028:8080	0.000	60.064
27	Fri Dec 14 18:29:15 ...	Fri Dec 14 18:36:01 ...	10.0.0.9	unknown	UDP		17.173.254.223:16386	0.714	0.816
28	Fri Dec 14 18:29:15 ...	Fri Dec 14 18:36:01 ...	10.0.0.9	unknown	UDP		17.173.254.222:16385	1.428	1.530
29	Fri Dec 14 18:29:15 ...	Fri Dec 14 18:35:21 ...	10.0.0.9	unknown	UDP		pin1258036.hsr.ch:1...	0.612	0.408
30	Fri Dec 14 18:29:15 ...	Fri Dec 14 18:35:21 ...	10.0.0.9	unknown	UDP		10.0.0.1:53	0.365	0.703
34	Fri Dec 14 18:29:15 ...	Fri Dec 14 18:29:15 ...	10.0.0.9	unknown	UDP		152.152.194.136:16...	0.000	0.102
35	Fri Dec 14 18:29:15 ...	Fri Dec 14 18:35:21 ...	10.0.0.9	unknown	UDP		224.0.0.251:5353	1.513	0.000
39	Fri Dec 14 18:30:21 ...	Fri Dec 14 18:30:21 ...	10.0.0.9	unknown	TCP		st11p01st-courier14...	1.867	3.905
40	Fri Dec 14 18:30:21 ...	Fri Dec 14 18:30:21 ...	10.0.0.9	unknown	TCP		st11p01st-courier01...	0.000	0.257
41	Fri Dec 14 18:30:21 ...	Fri Dec 14 18:30:21 ...	10.0.0.9	unknown	TCP		10.0.0.181:22520	1.315	0.000

Abbildung C.6: AppAnalysator-«Overview»: «Bad Piggies»

Beim Starten der App und während den einzelnen Spielen kam es regelmässig zu HTTP-Requests an «neptune.appads.com». Dabei kam es zu POST-Requests, wo diverse Handyinformationen mitgeschickt werden. Unter anderem wurde da auch Informationen zum Mobiltelefonanbieter (im englischen «Carrier») mitgeschickt.

Es scheint als werden diese Informationen mitgeschickt, um möglichst benutzerspezifische Werbungen anzeigen zu können. In wie fern die Benutzer dies wünschen, ist sehr fraglich. Weiter ist davon auszugehen, dass der Benutzer die Übermittlung der Daten in der Regel gar nicht mitbekommt.

C.3 Test «Dropbox»

Dropbox kann als Datenablage genutzt werden. Hierfür musste ein Dropbox-Account erstellt werden. Nach dem Erstellen wird gleich vorgeschlagen den Kamera-Upload zu aktivieren. Dropbox untersucht dabei den Speicher des Smartphones auf vorhandene Kamerabilder. Sobald Dropbox Bilder findet, werden diese nach und nach auf die Dropbox hochgeladen.

Um den Test der App zu dokumentieren, wurden einige Screenshots angefertigt. Diese wurden von Dropbox automatisch als Kamerabild interpretiert, sodass die Screenshots gleich hochgeladen wurden.

Das Öffnen der einzelnen Dropbox-Ordner ist problemlos möglich.

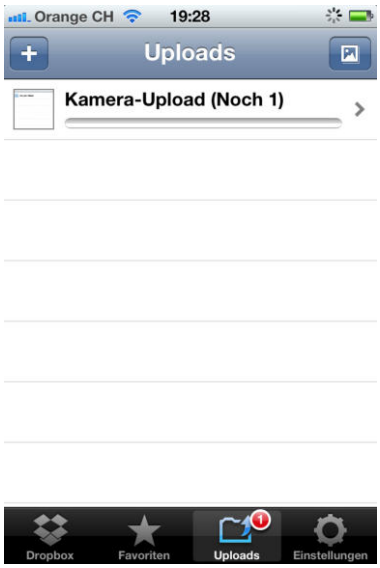


Abbildung C.7: Screenshot «Dropbox»: Uploads

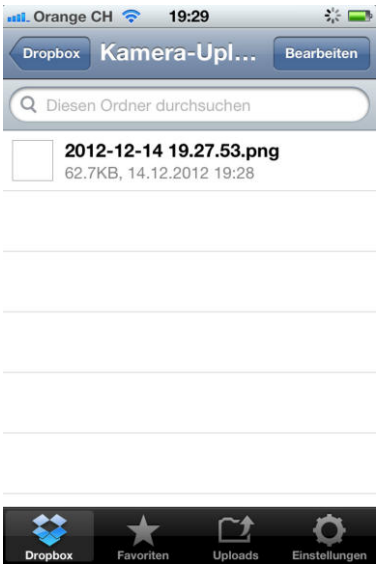
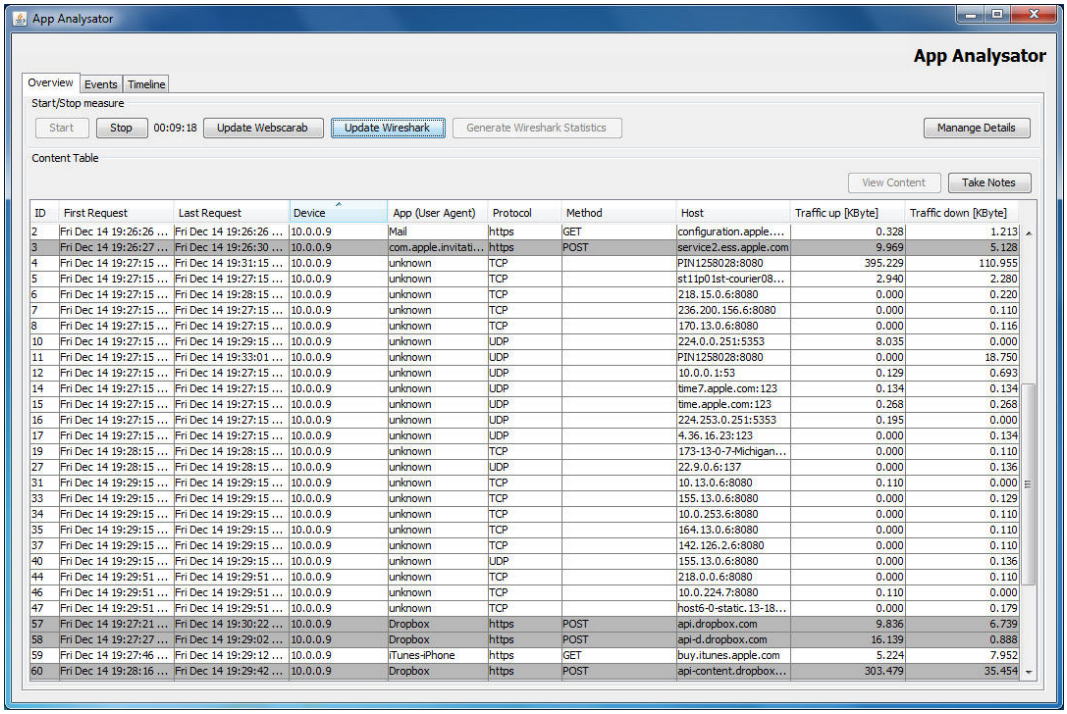


Abbildung C.8: Screenshot «Dropbox»: Kamera-Upload

Ergebnis



ID	First Request	Last Request	Device	App (User Agent)	Protocol	Method	Host	Traffic up [KByte]	Traffic down [KByte]
2	Fri Dec 14 19:26:26 ...	Fri Dec 14 19:26:26 ...	10.0.0.9	Mail	https	GET	configuration.apple....	0.328	1.213
3	Fri Dec 14 19:26:27 ...	Fri Dec 14 19:26:30 ...	10.0.0.9	com.apple.invitati...	https	POST	service2.ess.apple.com	9.969	5.128
4	Fri Dec 14 19:27:15 ...	Fri Dec 14 19:31:15 ...	10.0.0.9	unknown	TCP		PIN1258028:8080	395.229	110.955
5	Fri Dec 14 19:27:15 ...	Fri Dec 14 19:27:15 ...	10.0.0.9	unknown	TCP		st11p01st-courier08...	2.940	2.280
6	Fri Dec 14 19:27:15 ...	Fri Dec 14 19:28:15 ...	10.0.0.9	unknown	TCP		218.15.0.6:8080	0.000	0.220
7	Fri Dec 14 19:27:15 ...	Fri Dec 14 19:27:15 ...	10.0.0.9	unknown	TCP		236.200.156.6:8080	0.000	0.110
8	Fri Dec 14 19:27:15 ...	Fri Dec 14 19:27:15 ...	10.0.0.9	unknown	TCP		170.13.0.6:8080	0.000	0.116
10	Fri Dec 14 19:27:15 ...	Fri Dec 14 19:29:15 ...	10.0.0.9	unknown	UDP		224.0.0.251:5353	8.035	0.000
11	Fri Dec 14 19:27:15 ...	Fri Dec 14 19:33:01 ...	10.0.0.9	unknown	UDP		PIN1258028:8080	0.000	18.750
12	Fri Dec 14 19:27:15 ...	Fri Dec 14 19:27:15 ...	10.0.0.9	unknown	UDP		10.0.0.1:53	0.129	0.693
14	Fri Dec 14 19:27:15 ...	Fri Dec 14 19:27:15 ...	10.0.0.9	unknown	UDP		time7.apple.com:123	0.134	0.134
15	Fri Dec 14 19:27:15 ...	Fri Dec 14 19:27:15 ...	10.0.0.9	unknown	UDP		time.apple.com:123	0.268	0.268
16	Fri Dec 14 19:27:15 ...	Fri Dec 14 19:27:15 ...	10.0.0.9	unknown	UDP		224.253.0.251:5353	0.195	0.000
17	Fri Dec 14 19:27:15 ...	Fri Dec 14 19:27:15 ...	10.0.0.9	unknown	UDP		4.36.16.23:123	0.000	0.134
19	Fri Dec 14 19:28:15 ...	Fri Dec 14 19:28:15 ...	10.0.0.9	unknown	TCP		173-13-0-7-Michigan...	0.000	0.110
27	Fri Dec 14 19:28:15 ...	Fri Dec 14 19:28:15 ...	10.0.0.9	unknown	UDP		22.9.0.6:137	0.000	0.136
31	Fri Dec 14 19:29:15 ...	Fri Dec 14 19:29:15 ...	10.0.0.9	unknown	TCP		10.13.0.6:8080	0.110	0.000
33	Fri Dec 14 19:29:15 ...	Fri Dec 14 19:29:15 ...	10.0.0.9	unknown	TCP		155.13.0.6:8080	0.000	0.129
34	Fri Dec 14 19:29:15 ...	Fri Dec 14 19:29:15 ...	10.0.0.9	unknown	TCP		10.0.253.6:8080	0.000	0.110
35	Fri Dec 14 19:29:15 ...	Fri Dec 14 19:29:15 ...	10.0.0.9	unknown	TCP		164.13.0.6:8080	0.000	0.110
37	Fri Dec 14 19:29:15 ...	Fri Dec 14 19:29:15 ...	10.0.0.9	unknown	TCP		142.126.2.6:8080	0.000	0.110
40	Fri Dec 14 19:29:15 ...	Fri Dec 14 19:29:15 ...	10.0.0.9	unknown	UDP		155.13.0.6:8080	0.000	0.136
44	Fri Dec 14 19:29:51 ...	Fri Dec 14 19:29:51 ...	10.0.0.9	unknown	TCP		218.0.0.6:8080	0.000	0.110
46	Fri Dec 14 19:29:51 ...	Fri Dec 14 19:29:51 ...	10.0.0.9	unknown	TCP		10.0.224.7:8080	0.110	0.000
47	Fri Dec 14 19:29:51 ...	Fri Dec 14 19:29:51 ...	10.0.0.9	unknown	TCP		host6-0-static.13-18...	0.000	0.179
57	Fri Dec 14 19:27:21 ...	Fri Dec 14 19:30:22 ...	10.0.0.9	Dropbox	https	POST	api.dropbox.com	9.836	6.739
58	Fri Dec 14 19:27:27 ...	Fri Dec 14 19:29:02 ...	10.0.0.9	Dropbox	https	POST	api-d.dropbox.com	16.139	0.888
59	Fri Dec 14 19:27:46 ...	Fri Dec 14 19:29:12 ...	10.0.0.9	iTunes-iPhone	https	GET	buy.itunes.apple.com	5.224	7.952
60	Fri Dec 14 19:28:16 ...	Fri Dec 14 19:29:42 ...	10.0.0.9	Dropbox	https	POST	api-content.dropbox...	303.479	35.454

Abbildung C.9: AppAnalysator-«Overview»: «Dropbox»

Dropbox verwendet als Datenübertragungsmittel HTTPS-Verbindungen. Dabei werden beispielsweise bei der Registrierung die Emailadresse und das Passwort nicht zusätzlich verschlüsselt.

Weiter konnte nichts auffälliges herausgefunden werden. Dropbox macht selbst keine selbstständigen Anfragen auf Änderungen. Beim Durchgehen durch die einzelnen Ordner werden die Übersichten jeweils nachgeladen.

C.4 Test «Facebook»

Um das Facebook App nutzen zu können, ist es notwendig einen Facebook Account besitzen. Die Registrierung funktioniert mittels Aufruf der mobilen Facebook-Seite im Webbrowser.

Nach der Registrierung und der Bestätigung können die Login-Daten im Facebook-App genutzt werden. Mit der App können die Facebook typischen Funktionen wie beispielsweise Statusmeldungen benutzt werden.

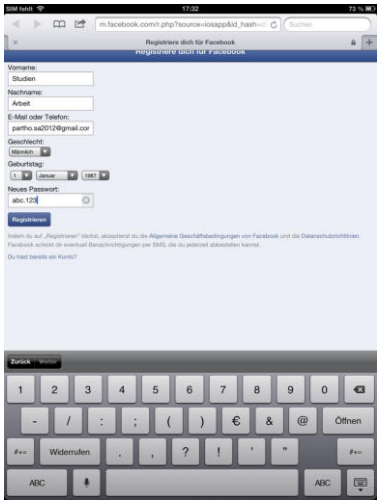


Abbildung C.10: Screenshot «Facebook»: Registrierung im Webbrowser

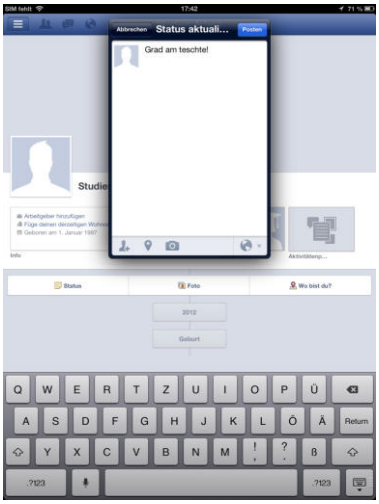
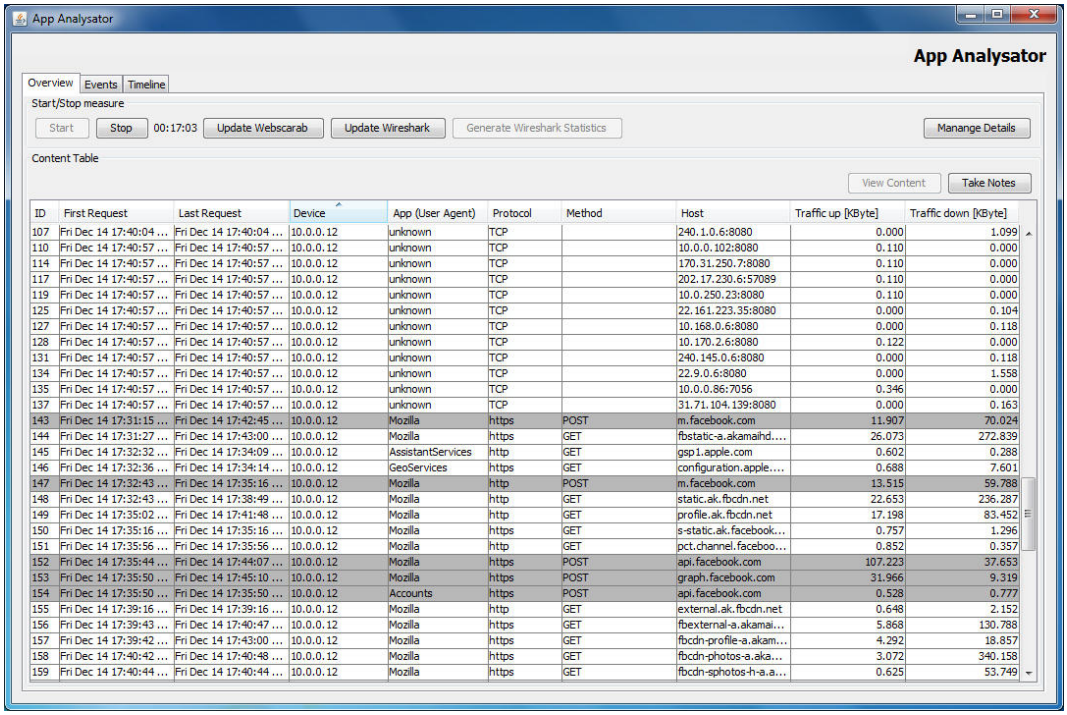


Abbildung C.11: Screenshot «Facebook»: Statusmeldung posten

Ergebnis



ID	First Request	Last Request	Device	App (User Agent)	Protocol	Method	Host	Traffic up [KByte]	Traffic down [KByte]
107	Fri Dec 14 17:40:04 ...	Fri Dec 14 17:40:04 ...	10.0.0.12	unknown	TCP		240.1.0.6:8080	0.000	1.099
110	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		10.0.0.102:8080	0.110	0.000
114	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		170.31.250.7:8080	0.110	0.000
117	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		202.17.230.6:57089	0.110	0.000
119	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		10.0.250.23:8080	0.110	0.000
125	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		22.161.223.35:8080	0.000	0.104
127	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		10.168.0.6:8080	0.000	0.118
128	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		10.170.2.6:8080	0.122	0.000
131	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		240.145.0.6:8080	0.000	0.118
134	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		22.9.0.6:8080	0.000	1.558
135	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		10.0.0.86:7056	0.346	0.000
137	Fri Dec 14 17:40:57 ...	Fri Dec 14 17:40:57 ...	10.0.0.12	unknown	TCP		31.71.104.139:8080	0.000	0.163
143	Fri Dec 14 17:31:15 ...	Fri Dec 14 17:42:45 ...	10.0.0.12	Mozilla	https	POST	m.facebook.com	11.907	70.024
144	Fri Dec 14 17:31:27 ...	Fri Dec 14 17:43:00 ...	10.0.0.12	Mozilla	https	GET	fbstatic-a.akamaihd....	26.073	272.839
145	Fri Dec 14 17:32:32 ...	Fri Dec 14 17:34:09 ...	10.0.0.12	AssistantServices	http	GET	gsp1.apple.com	0.602	0.288
146	Fri Dec 14 17:32:36 ...	Fri Dec 14 17:34:14 ...	10.0.0.12	GeoServices	https	GET	configuration.apple....	0.688	7.601
147	Fri Dec 14 17:32:43 ...	Fri Dec 14 17:35:16 ...	10.0.0.12	Mozilla	http	POST	m.facebook.com	13.515	59.788
148	Fri Dec 14 17:32:43 ...	Fri Dec 14 17:38:49 ...	10.0.0.12	Mozilla	http	GET	static.ak.fbcdn.net	22.653	236.287
149	Fri Dec 14 17:35:02 ...	Fri Dec 14 17:41:48 ...	10.0.0.12	Mozilla	http	GET	profile.ak.fbcdn.net	17.198	83.452
150	Fri Dec 14 17:35:16 ...	Fri Dec 14 17:35:16 ...	10.0.0.12	Mozilla	https	GET	s-static.ak.facebook...	0.757	1.296
151	Fri Dec 14 17:35:56 ...	Fri Dec 14 17:35:56 ...	10.0.0.12	Mozilla	http	GET	pct.channel.faceboo...	0.852	0.357
152	Fri Dec 14 17:35:44 ...	Fri Dec 14 17:44:07 ...	10.0.0.12	Mozilla	https	POST	api.facebook.com	107.223	37.653
153	Fri Dec 14 17:35:50 ...	Fri Dec 14 17:45:10 ...	10.0.0.12	Mozilla	https	POST	graph.facebook.com	31.966	9.319
154	Fri Dec 14 17:35:50 ...	Fri Dec 14 17:35:50 ...	10.0.0.12	Accounts	https	POST	api.facebook.com	0.528	0.777
155	Fri Dec 14 17:39:16 ...	Fri Dec 14 17:39:16 ...	10.0.0.12	Mozilla	http	GET	external.ak.fbcdn.net	0.648	2.152
156	Fri Dec 14 17:39:43 ...	Fri Dec 14 17:40:47 ...	10.0.0.12	Mozilla	https	GET	fbexternal-a.akamai...	5.868	130.788
157	Fri Dec 14 17:39:42 ...	Fri Dec 14 17:43:00 ...	10.0.0.12	Mozilla	https	GET	fbcdn-profile-a.akam...	4.292	18.857
158	Fri Dec 14 17:40:42 ...	Fri Dec 14 17:40:48 ...	10.0.0.12	Mozilla	https	GET	fbcdn-photos-a.aka...	3.072	340.158
159	Fri Dec 14 17:40:44 ...	Fri Dec 14 17:40:44 ...	10.0.0.12	Mozilla	https	GET	fbcdn-sphotos-h-a.a...	0.625	53.749

Abbildung C.12: AppAnalysator-«Overview»: «Facebook»

Bei der Registrierung des Facebook-Accounts wurde zwar die sichere Variante mit HTTPS verwendet, jedoch wurden die Anmeldedaten im «Klartext» übertragen. Die verwendete Emailadresse und das Passwort kann also problemlos herausgelesen werden.

Weiter werden Informationen zum Gerät übermittelt. So wird beispielsweise die UUID («Universally Unique Identifier») des Gerätes übermittelt. Es werden aber auch Angaben zum iOS-Betriebssystem, zum Gerätetyp und dem Verbindungstyp übermittelt.

C.5 Test «Google Latitude»

Google Latitude verwendet zur Authentisierung einen Google Account. Es erlaubt Personen zu lokalisieren und den Standort mit Freunden zu teilen.

So können Freunde ihren Standort teilen und sehen, wo sich die anderen gerade befinden. Anhand der GPS-Informationen kann der aktuelle Standort problemlos auf wenige Meter herausgefunden werden.

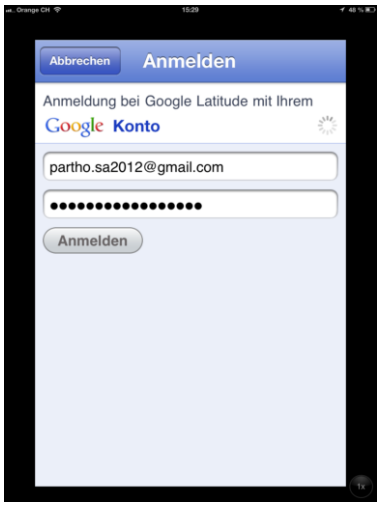


Abbildung C.13: Screenshot «Google Latitude»: Anmeldung

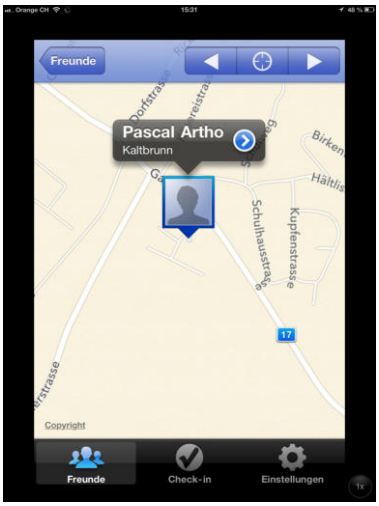
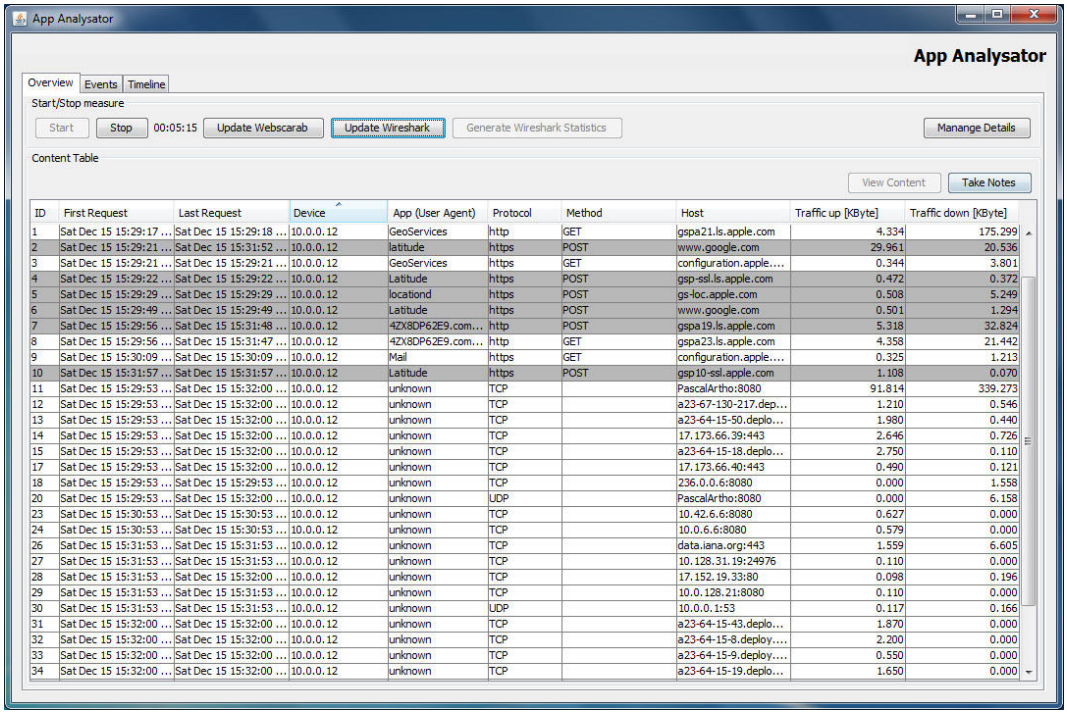


Abbildung C.14: Screenshot «Google Latitude»: Standort

Ergebnis



ID	First Request	Last Request	Device	App (User Agent)	Protocol	Method	Host	Traffic up [KByte]	Traffic down [KByte]
1	Sat Dec 15 15:29:17 ...	Sat Dec 15 15:29:18 ...	10.0.0.12	GeoServices	http	GET	gspa21.ls.apple.com	4.334	175.299
2	Sat Dec 15 15:29:21 ...	Sat Dec 15 15:31:52 ...	10.0.0.12	latitude	https	POST	www.google.com	29.961	20.536
3	Sat Dec 15 15:29:21 ...	Sat Dec 15 15:29:21 ...	10.0.0.12	GeoServices	https	GET	configuration.apple....	0.344	3.801
4	Sat Dec 15 15:29:22 ...	Sat Dec 15 15:29:22 ...	10.0.0.12	Latitude	https	POST	gsp-ssl.ls.apple.com	0.472	0.372
5	Sat Dec 15 15:29:29 ...	Sat Dec 15 15:29:29 ...	10.0.0.12	locationd	https	POST	gs-loc.apple.com	0.508	5.249
6	Sat Dec 15 15:29:49 ...	Sat Dec 15 15:29:49 ...	10.0.0.12	Latitude	https	POST	www.google.com	0.501	1.294
7	Sat Dec 15 15:29:56 ...	Sat Dec 15 15:31:48 ...	10.0.0.12	4ZX8DP62E9.com...	http	POST	gspa19.ls.apple.com	5.318	32.824
8	Sat Dec 15 15:29:56 ...	Sat Dec 15 15:31:47 ...	10.0.0.12	4ZX8DP62E9.com...	http	GET	gspa23.ls.apple.com	4.358	21.442
9	Sat Dec 15 15:30:09 ...	Sat Dec 15 15:30:09 ...	10.0.0.12	Mail	https	GET	configuration.apple....	0.325	1.213
10	Sat Dec 15 15:31:57 ...	Sat Dec 15 15:31:57 ...	10.0.0.12	Latitude	https	POST	gsp10-ssl.apple.com	1.108	0.070
11	Sat Dec 15 15:29:53 ...	Sat Dec 15 15:32:00 ...	10.0.0.12	unknown	TCP		PascalArtho:8080	91.814	339.273
12	Sat Dec 15 15:29:53 ...	Sat Dec 15 15:32:00 ...	10.0.0.12	unknown	TCP		a23-67-130-217.dep...	1.210	0.546
13	Sat Dec 15 15:29:53 ...	Sat Dec 15 15:32:00 ...	10.0.0.12	unknown	TCP		a23-64-15-50.deplo...	1.980	0.440
14	Sat Dec 15 15:29:53 ...	Sat Dec 15 15:32:00 ...	10.0.0.12	unknown	TCP		17.173.66.39:443	2.646	0.726
15	Sat Dec 15 15:29:53 ...	Sat Dec 15 15:32:00 ...	10.0.0.12	unknown	TCP		a23-64-15-18.deplo...	2.750	0.110
17	Sat Dec 15 15:29:53 ...	Sat Dec 15 15:32:00 ...	10.0.0.12	unknown	TCP		17.173.66.40:443	0.490	0.121
18	Sat Dec 15 15:29:53 ...	Sat Dec 15 15:29:53 ...	10.0.0.12	unknown	TCP		236.0.0.6:8080	0.000	1.558
20	Sat Dec 15 15:29:53 ...	Sat Dec 15 15:32:00 ...	10.0.0.12	unknown	UDP		PascalArtho:8080	0.000	6.158
23	Sat Dec 15 15:30:53 ...	Sat Dec 15 15:30:53 ...	10.0.0.12	unknown	TCP		10.42.6.6:8080	0.627	0.000
24	Sat Dec 15 15:30:53 ...	Sat Dec 15 15:30:53 ...	10.0.0.12	unknown	TCP		10.0.6.6:8080	0.579	0.000
26	Sat Dec 15 15:31:53 ...	Sat Dec 15 15:31:53 ...	10.0.0.12	unknown	TCP		data.iana.org:443	1.559	6.605
27	Sat Dec 15 15:31:53 ...	Sat Dec 15 15:31:53 ...	10.0.0.12	unknown	TCP		10.128.31.19:24976	0.110	0.000
28	Sat Dec 15 15:31:53 ...	Sat Dec 15 15:32:00 ...	10.0.0.12	unknown	TCP		17.152.19.33:80	0.098	0.196
29	Sat Dec 15 15:31:53 ...	Sat Dec 15 15:31:53 ...	10.0.0.12	unknown	TCP		10.0.128.21:8080	0.110	0.000
30	Sat Dec 15 15:31:53 ...	Sat Dec 15 15:31:53 ...	10.0.0.12	unknown	UDP		10.0.0.1:53	0.117	0.166
31	Sat Dec 15 15:32:00 ...	Sat Dec 15 15:32:00 ...	10.0.0.12	unknown	TCP		a23-64-15-43.deplo...	1.870	0.000
32	Sat Dec 15 15:32:00 ...	Sat Dec 15 15:32:00 ...	10.0.0.12	unknown	TCP		a23-64-15-8.deploy....	2.200	0.000
33	Sat Dec 15 15:32:00 ...	Sat Dec 15 15:32:00 ...	10.0.0.12	unknown	TCP		a23-64-15-9.deploy....	0.550	0.000
34	Sat Dec 15 15:32:00 ...	Sat Dec 15 15:32:00 ...	10.0.0.12	unknown	TCP		a23-64-15-19.deplo...	1.650	0.000

Abbildung C.15: AppAnalysator-«Overview»: «Google Latitude»

Bei der Anmeldung mit dem Google Account wird eine HTTPS-Verbindung aufgebaut. Die Verbindungen werden durch den Proxy problemlos aufgebrochen, sodass die Emailadresse und das Passwort im «Klartext» ersichtlich ist.

Zusätzlich werden weitere Daten übertragen. Diese sind jedoch nicht weiter erkennbar, da sie verschlüsselt sind. Dies geschieht beispielsweise auch beim expliziten Suchen des Standorts. Was jedoch genau übermittelt wird, ist nicht erkennbar.

C.6 Test «Google Plus»

Das App für Google Plus kann mit einem Google Account verwendet werden. Google Plus bietet einen ähnlichen Funktionsumfang wie Facebook.

Die App kann jedoch nicht weiter getestet werden, da eine Fehlermeldung erscheint: «Die Verbindung wurde unterbrochen. Bitte melden Sie sich erneut in Google+ an»

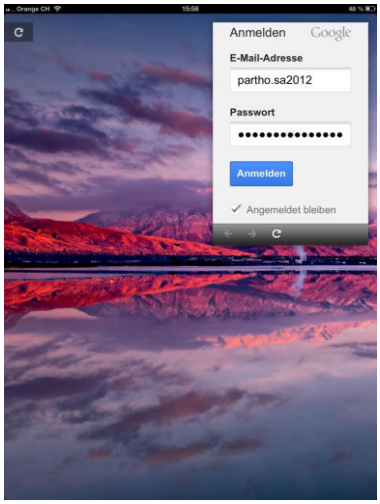


Abbildung C.16: Screenshot «Google Plus»: Anmeldung

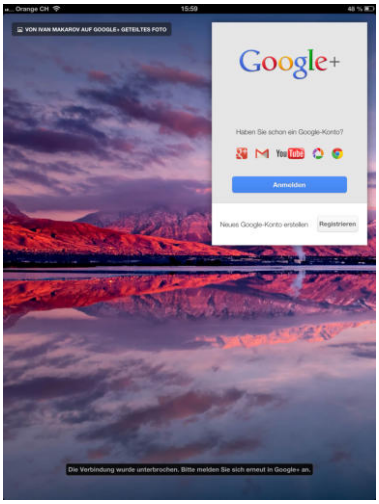


Abbildung C.17: Screenshot «Google Plus»: Fehlermeldung Verbindungsunterbruch

Ergebnis

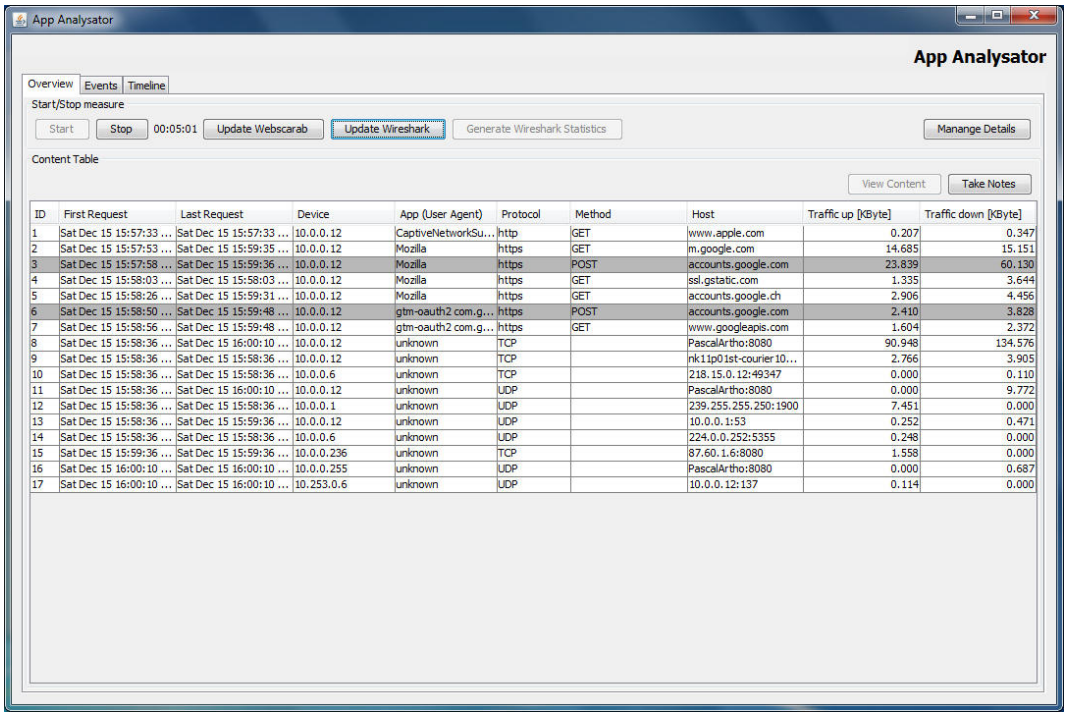


Abbildung C.18: AppAnalysator-«Overview»: «Google Plus»

Eine Verbindung mit dem Google Plus Service ist nicht möglich. Es scheint als ob mit dem vorhandenen Testaufbau keine Verbindung zulässt.

Es gibt Grund zur Annahme, dass Google im Google Plus App ein Zertifikat oder ein spezielle Zahl hinterlegt hat. Wird nun ein Proxy verwendet, welcher sich dazwischen schaltet, kann dieses Zertifikat oder die Zahl vom App nicht übertragen werden. So kann keine Verbindung aufgebaut werden und die Anmeldung schlägt fehl.

Dass etwas in der App hinterlegt ist, wird dadurch unterstützt, da die Anmeldung ohne Proxyangabe problemlos funktioniert.

Was aufgefallen ist bei der Anmeldung, dass die Google Account Informationen überprüft werden konnten. Wird ein falsches Passwort eingegeben, schlägt die Verbindung schon vorher fehl.

Allgemein verwendet Google Plus HTTPS für die Übertragung der Daten. Das Passwort und die Emailadresse ist aber auch hier im «Klartext» ablesbar, da der Proxy dazwischen geschaltet werden konnte. Ein Man-in-the-Middle kann die Zugangsdaten nutzen, sodass die Sicherheitsmassnahme von Google nicht viel bringt.

Sonstige Daten können nicht weiter herausgelesen werden. Es werden weitere Daten verschlüsselt übertragen.

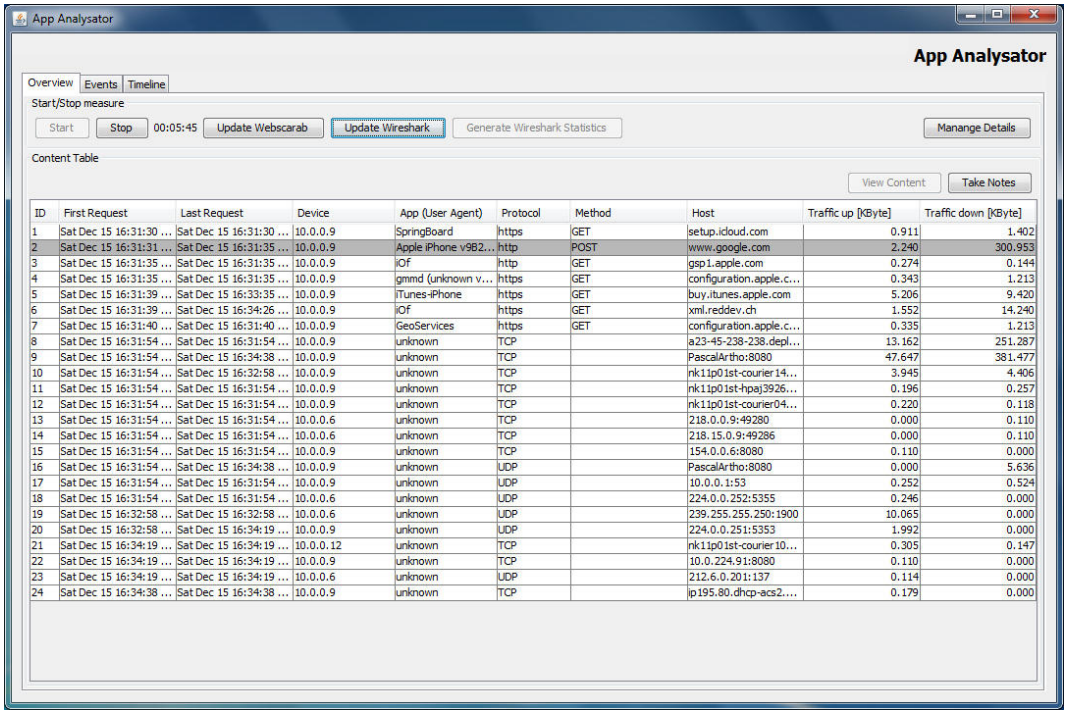
C.7 Test «iOf»

Die iOf App stellt diverse Informationen zum schweizerischen Militär zu Verfügung. So können beispielsweise «statische Informationen» wie die militärischen Grade und Abzeichen sowie die Bekleidung angesehen werden. Es besteht jedoch auch die Möglichkeit, «dynamische Informationen» wie beispielsweise WK-Daten abzurufen.

Abbildung C.19: Screenshot «iOf»: Startbildschirm

Abbildung C.20: Screenshot «iOf»: Schweiz

Ergebnis



ID	First Request	Last Request	Device	App (User Agent)	Protocol	Method	Host	Traffic up [KByte]	Traffic down [KByte]
1	Sat Dec 15 16:31:30 ...	Sat Dec 15 16:31:30 ...	10.0.0.9	SpringBoard	https	GET	setup.icloud.com	0.911	1.402
2	Sat Dec 15 16:31:31 ...	Sat Dec 15 16:31:35 ...	10.0.0.9	Apple iPhone v9B2...	http	POST	www.google.com	2.240	300.953
3	Sat Dec 15 16:31:35 ...	Sat Dec 15 16:31:35 ...	10.0.0.9	iOf	http	GET	gsp1.apple.com	0.274	0.144
4	Sat Dec 15 16:31:35 ...	Sat Dec 15 16:31:35 ...	10.0.0.9	gmmd (unknown v...	https	GET	configuration.apple.c...	0.343	1.213
5	Sat Dec 15 16:31:39 ...	Sat Dec 15 16:33:35 ...	10.0.0.9	iTunes-iPhone	https	GET	buy.itunes.apple.com	5.206	9.420
6	Sat Dec 15 16:31:39 ...	Sat Dec 15 16:34:26 ...	10.0.0.9	iOf	https	GET	xml.reddev.ch	1.552	14.240
7	Sat Dec 15 16:31:40 ...	Sat Dec 15 16:31:40 ...	10.0.0.9	GeoServices	https	GET	configuration.apple.c...	0.335	1.213
8	Sat Dec 15 16:31:54 ...	Sat Dec 15 16:31:54 ...	10.0.0.9	unknown	TCP		a23-45-238-238.depl...	13.162	251.287
9	Sat Dec 15 16:31:54 ...	Sat Dec 15 16:34:38 ...	10.0.0.9	unknown	TCP		PascalArtho:8080	47.647	381.477
10	Sat Dec 15 16:31:54 ...	Sat Dec 15 16:32:58 ...	10.0.0.9	unknown	TCP		nk11p01st-courier14...	3.945	4.406
11	Sat Dec 15 16:31:54 ...	Sat Dec 15 16:31:54 ...	10.0.0.9	unknown	TCP		nk11p01st-hpaj3926...	0.196	0.257
12	Sat Dec 15 16:31:54 ...	Sat Dec 15 16:31:54 ...	10.0.0.9	unknown	TCP		nk11p01st-courier04...	0.220	0.118
13	Sat Dec 15 16:31:54 ...	Sat Dec 15 16:31:54 ...	10.0.0.6	unknown	TCP		218.0.0.9:49280	0.000	0.110
14	Sat Dec 15 16:31:54 ...	Sat Dec 15 16:31:54 ...	10.0.0.6	unknown	TCP		218.15.0.9:49286	0.000	0.110
15	Sat Dec 15 16:31:54 ...	Sat Dec 15 16:31:54 ...	10.0.0.9	unknown	TCP		154.0.0.6:8080	0.110	0.000
16	Sat Dec 15 16:31:54 ...	Sat Dec 15 16:34:38 ...	10.0.0.9	unknown	UDP		PascalArtho:8080	0.000	5.636
17	Sat Dec 15 16:31:54 ...	Sat Dec 15 16:31:54 ...	10.0.0.9	unknown	UDP		10.0.0.1:53	0.252	0.524
18	Sat Dec 15 16:31:54 ...	Sat Dec 15 16:31:54 ...	10.0.0.6	unknown	UDP		224.0.0.252:5355	0.246	0.000
19	Sat Dec 15 16:32:58 ...	Sat Dec 15 16:32:58 ...	10.0.0.6	unknown	UDP		239.255.255.250:1900	10.065	0.000
20	Sat Dec 15 16:32:58 ...	Sat Dec 15 16:34:19 ...	10.0.0.9	unknown	UDP		224.0.0.251:5353	1.992	0.000
21	Sat Dec 15 16:34:19 ...	Sat Dec 15 16:34:19 ...	10.0.0.12	unknown	TCP		nk11p01st-courier10...	0.305	0.147
22	Sat Dec 15 16:34:19 ...	Sat Dec 15 16:34:19 ...	10.0.0.9	unknown	TCP		10.0.224.91:8080	0.110	0.000
23	Sat Dec 15 16:34:19 ...	Sat Dec 15 16:34:19 ...	10.0.0.6	unknown	UDP		212.6.0.201:137	0.114	0.000
24	Sat Dec 15 16:34:38 ...	Sat Dec 15 16:34:38 ...	10.0.0.9	unknown	TCP		ip195.80.dhcp-acs2....	0.179	0.000

Abbildung C.21: AppAnalysator-«Overview»: «iOf»

Viele Daten sind statisch programmiert und benötigen keine Internetverbindung. Die meisten Informationen, welche einem Webserver geladen werden, werden über HTTPS geladen. Es wird jeweils ein GET-Request übermittelt, welcher eine XML-Datei als Antwort erhält. In diesem GET-Request werden Informationen zum Client-Gerät, zum Modell und zum Betriebssystem gemacht. Weiter ist auch die UUID in diesem Request vorhanden.

Anzumerken ist, dass das Abrufen der WK-Daten über die unverschlüsselte HTTP-Verbindung getätigt wird. Dies ist wohl darauf zurück zu führen, dass die Daten für WK's von der Webseite des Bundes geladen wird. Die App wurde jedoch nicht durch den Bund entwickelt, sondern durch die Firma «RedDev GmbH».

C.8 Test «Raiffeisen»-App

> Tätigen Sie Ihre Zahlungen oder Börsenaufträge wann und wo Sie wollen. Mit Mobile Banking stehen Ihnen sämtliche Funktionen des E-Bankings auch unterwegs zur Verfügung. [8]

So lautet der Werbespruch für die «Raiffeisen»-App auf der Anleitung für die genannte App. Weiter sei der Zugriff auf das E-Banking von Raiffeisen mit der App sehr einfach. Die App muss lediglich im Store heruntergeladen werden und es kann los gehen.

Im AppStore von Apple ist die App schnell gefunden und wie alle anderen Apps schnell installiert. Gleich beim ersten Start wird nach der Freigabe des aktuellen Standorts gefragt. Wofür dieser Standort gebraucht wird, ist auf den ersten Blick nicht klar. Erst während dem Benutzen der App fällt auf, dass der Standort wohl benötigt wird, um den nächstgelegenen Bancomat anzuzeigen.

Die App bietet dem Benutzer also Informationen zu Bancomaten in der näheren Umgebung sowie allgemeine Informationen im Bereich Finanzen und Notfallsnummern. Zusätzlich besteht die Möglichkeit, auf den E-Banking Account des persönlichen Raiffeisenkontos zu zugreifen.

Um sich beim E-Banking anzumelden, muss die Vertragsnummer und das persönliche Passwort eingegeben werden. Die Benutzung der Mobile Services muss hierbei jedoch einmalig in der Web-Applikation freigegeben werden.

In der Standardversion des E-Banking verweist Raiffeisen dabei auf Sicherheitshinweise bezüglich Mobile Banking. Raiffeisen warnt dabei unter anderem: «Beachten Sie, falls Mobile Banking im öffentlichen Raum nutzen, dass Unberechtigte durch Beobachten Ihre Bankinformationen einsehen können.»

Werden die Sicherheitsinformationen akzeptiert, gelten diese als Verstanden und der Benutzer willigt ein, das Mobiltelefon zu schützen. Danach steht dem Mobile Banking nichts mehr im Wege.

Die nachfolgenden Bildern wurden mit einem schwarzen Balken anonymisiert, da es sich um private und persönliche Daten handelt.

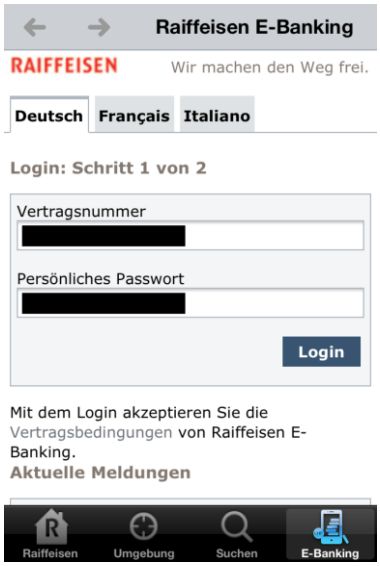


Abbildung C.22: Screenshot «Raiffeisen»: E-Banking Anmeldeformular

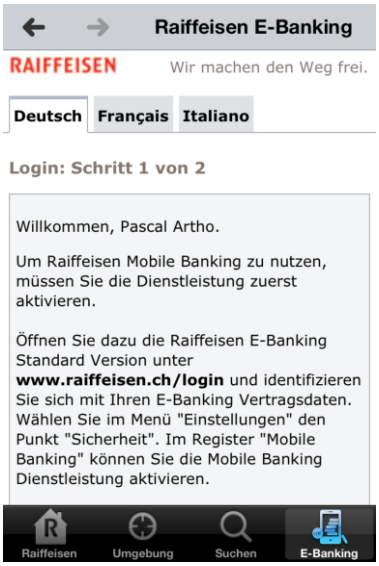


Abbildung C.23: Screenshot «Raiffeisen»: E-Banking Begrüssung

Fazit

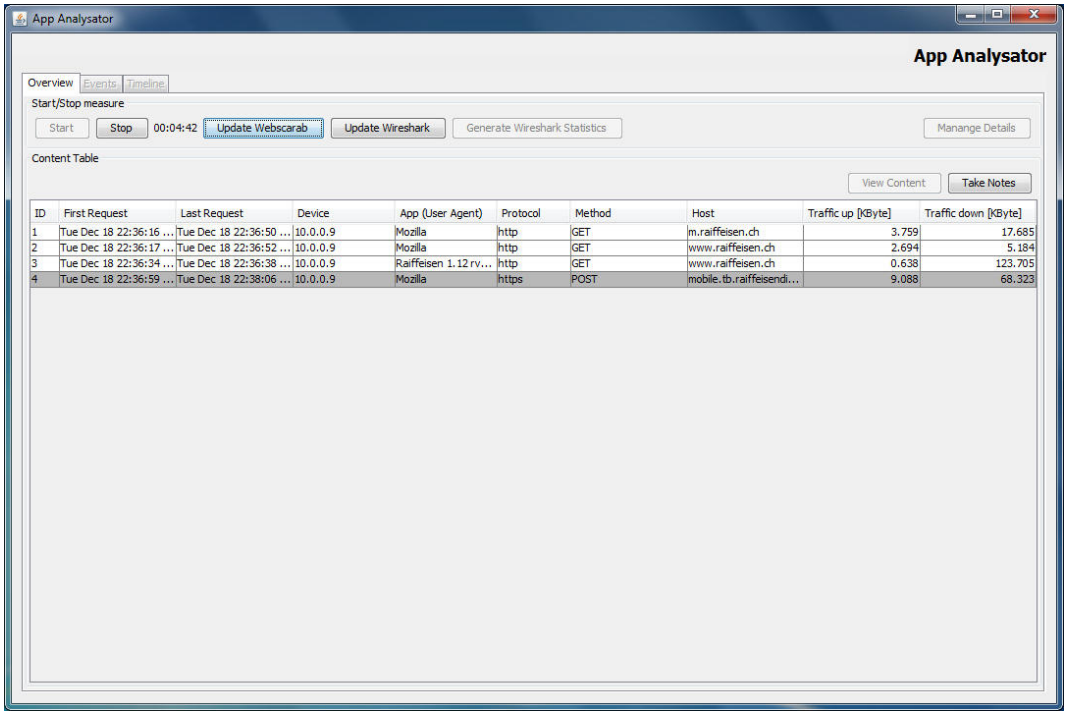


Abbildung C.24: AppAnalysator-«Overview»: «Raiffeisen»

Die Ergebnisse bezüglich der «Raiffeisen»-App sind sehr erschreckend. Zwar verwendet die «Raiffeisen»-App für die Übertragung der persönlichen Kontodaten eine verschlüsselte HTTPS-Verbindung. In einem einfachen Test konnte jedoch ein Proxy dazwischen geschalten werden. Das selbst erstelle Zertifikat wurde durch das «Raiffeisen»-App problemlos akzeptiert. Die Daten konnten also problemlos über den Proxy gesandt werden, sodass ein Man-in-the-Middle zu den Zugangsdaten kommen kann.

Zwar warnt die Raiffeisenbank davor, dass in einem öffentlichen Netz unberechtigte Zugriff haben könnten. Bei einer Bank sollte jedoch davon ausgegangen werden, dass viel Wert auf Sicherheit gelegt wird. Wie sich mit dem vorhandenen Testaufbau gezeigt hat, ist die «Raiffeisen»-App aber alles andere als gegen eine Man-in-the-Middle-Attacke geschützt.

Im Test wurden die nachfolgenden Daten in einer verschlüsselten HTTPS-Verbindung übermittelt: «CID=&lang=de&CID_NUM=XXXXX&CID_CLEAR=XXXX&CID_MOB=XXXXX-XXXX&PWD=XXXXXXXXXX&date=18.12.2012+22%3A37%3A04&os=iPhone&osPlatform=Macintosh+OS+X»
(Aus Datenschutzgründen wurden die korrekten Daten durch «X» ersetzt)

Die E-Banking-Vertragsnummer, die sogenannte «CID_MOB», wird im Klartext übermittelt. Ebenso auch das persönliche Passwort, welches im POST-Befehl bei «PWD» mitgegeben wird.

Wie es scheint bleibt die Raiffeisenbank ihrem Slogan, «Wir machen den Weg frei», treu. Diese Mal richtet sich dieser Slogan aber leider eher gegen den Kunden und für einen potentiellen Angreifer. Es ist jedoch davon auszugehen, dass die Raiffeisenbank das Risiko eines Schadenfalls abgeschätzt hat und zum Schluss gekommen ist, dass der notwendige Aufwand sind nicht lohnt. Ebenso schützt Raiffeisenbank die Kunden durch ein dreistufiges Login. Dieser Identifizierungsmechanismus verwendet eine individuelle E-Banking Vertragsnummer, ein persönliches Passwort und einen Passwortzusatz, der jeweils nur während eines einzigen Logins gültig ist. [9]

C.9 Test «Waze»

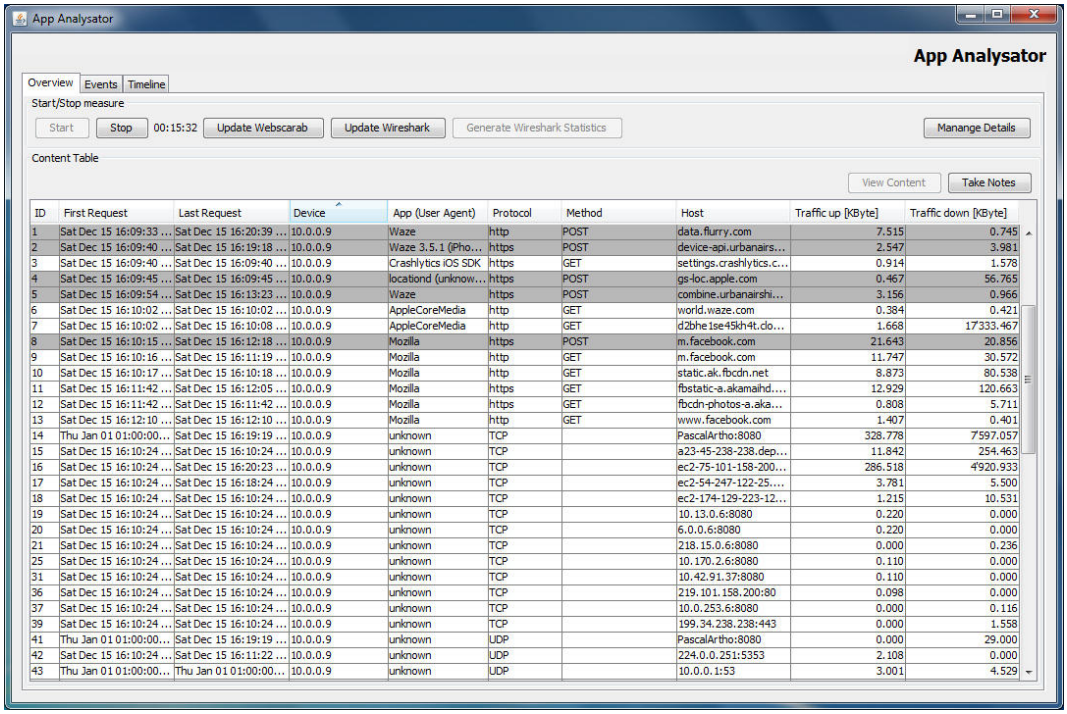
Waze bietet die Möglichkeit den Benutzeraccount mit einem Facebook Account zu verbinden. Hierfür ist eine Anmeldung bei Facebook nötig und eine kurze Bestätigung, dass die Facebook-Daten mit Waze geteilt werden dürfen.

Ist die Anmeldung abgeschlossen, wird der aktuelle Standort abgerufen. Mit dem aktuellen Standort kann Waze ausfindig machen, welche Meldungen in der näheren Umgebung gemacht wurden. Bei diesen Meldungen handelt es sich um Verkehrsinformationen wie beispielsweise Staumeldungen. Diese Meldungen werden jeweils von anderen Benutzer abgesetzt.

Abbildung C.25: Screenshot «Waze»: Übersicht Meldungen

Abbildung C.26: Screenshot «Waze»: Kartenausschnitt

Ergebnis



ID	First Request	Last Request	Device	App (User Agent)	Protocol	Method	Host	Traffic up [KByte]	Traffic down [KByte]
1	Sat Dec 15 16:09:33 ...	Sat Dec 15 16:20:39 ...	10.0.0.9	Waze	http	POST	data.flurry.com	7.515	0.745
2	Sat Dec 15 16:09:40 ...	Sat Dec 15 16:19:18 ...	10.0.0.9	Waze 3.5.1 (iPho...	https	POST	device-api.urbanairs...	2.547	3.981
3	Sat Dec 15 16:09:40 ...	Sat Dec 15 16:09:40 ...	10.0.0.9	Crashlytics iOS SDK	https	GET	settings.crashlytics.c...	0.914	1.578
4	Sat Dec 15 16:09:45 ...	Sat Dec 15 16:09:45 ...	10.0.0.9	locationd (unknow...	https	POST	gs-loc.apple.com	0.467	56.765
5	Sat Dec 15 16:09:54 ...	Sat Dec 15 16:13:23 ...	10.0.0.9	Waze	https	POST	combine.urbanairshi...	3.156	0.966
6	Sat Dec 15 16:10:02 ...	Sat Dec 15 16:10:02 ...	10.0.0.9	AppleCoreMedia	http	GET	world.waze.com	0.384	0.421
7	Sat Dec 15 16:10:02 ...	Sat Dec 15 16:10:08 ...	10.0.0.9	AppleCoreMedia	http	GET	d2bhe1se45kh4t.clo...	1.668	17'333.467
8	Sat Dec 15 16:10:15 ...	Sat Dec 15 16:12:18 ...	10.0.0.9	Mozilla	https	POST	m.facebook.com	21.643	20.856
9	Sat Dec 15 16:10:16 ...	Sat Dec 15 16:11:19 ...	10.0.0.9	Mozilla	http	GET	m.facebook.com	11.747	30.572
10	Sat Dec 15 16:10:17 ...	Sat Dec 15 16:10:18 ...	10.0.0.9	Mozilla	http	GET	static.ak.fbcdn.net	8.873	80.538
11	Sat Dec 15 16:11:42 ...	Sat Dec 15 16:12:05 ...	10.0.0.9	Mozilla	https	GET	fbstatic-a.akamaihd....	12.929	120.663
12	Sat Dec 15 16:11:42 ...	Sat Dec 15 16:11:42 ...	10.0.0.9	Mozilla	https	GET	fbcdn-photos-a.aka...	0.808	5.711
13	Sat Dec 15 16:12:10 ...	Sat Dec 15 16:12:10 ...	10.0.0.9	Mozilla	http	GET	www.facebook.com	1.407	0.401
14	Thu Jan 01 01:00:00...	Sat Dec 15 16:19:19 ...	10.0.0.9	unknown	TCP		PascalArtho:8080	328.778	7'597.057
15	Sat Dec 15 16:10:24 ...	Sat Dec 15 16:10:24 ...	10.0.0.9	unknown	TCP		a23-45-238-238.dep...	11.842	254.463
16	Sat Dec 15 16:10:24 ...	Sat Dec 15 16:20:23 ...	10.0.0.9	unknown	TCP		ec2-75-101-158-200...	286.518	4'920.933
17	Sat Dec 15 16:10:24 ...	Sat Dec 15 16:18:24 ...	10.0.0.9	unknown	TCP		ec2-54-247-122-25....	3.781	5.500
18	Sat Dec 15 16:10:24 ...	Sat Dec 15 16:10:24 ...	10.0.0.9	unknown	TCP		ec2-174-129-223-12...	1.215	10.531
19	Sat Dec 15 16:10:24 ...	Sat Dec 15 16:10:24 ...	10.0.0.9	unknown	TCP		10.13.0.6:8080	0.220	0.000
20	Sat Dec 15 16:10:24 ...	Sat Dec 15 16:10:24 ...	10.0.0.9	unknown	TCP		6.0.0.6:8080	0.220	0.000
21	Sat Dec 15 16:10:24 ...	Sat Dec 15 16:10:24 ...	10.0.0.9	unknown	TCP		218.15.0.6:8080	0.000	0.236
25	Sat Dec 15 16:10:24 ...	Sat Dec 15 16:10:24 ...	10.0.0.9	unknown	TCP		10.170.2.6:8080	0.110	0.000
31	Sat Dec 15 16:10:24 ...	Sat Dec 15 16:10:24 ...	10.0.0.9	unknown	TCP		10.42.91.37:8080	0.110	0.000
36	Sat Dec 15 16:10:24 ...	Sat Dec 15 16:10:24 ...	10.0.0.9	unknown	TCP		219.101.158.200:80	0.098	0.000
37	Sat Dec 15 16:10:24 ...	Sat Dec 15 16:10:24 ...	10.0.0.9	unknown	TCP		10.0.253.6:8080	0.000	0.116
39	Sat Dec 15 16:10:24 ...	Sat Dec 15 16:10:24 ...	10.0.0.9	unknown	TCP		199.34.238.238:443	0.000	1.558
41	Thu Jan 01 01:00:00...	Sat Dec 15 16:19:19 ...	10.0.0.9	unknown	UDP		PascalArtho:8080	0.000	29.000
42	Sat Dec 15 16:10:24 ...	Sat Dec 15 16:11:22 ...	10.0.0.9	unknown	UDP		224.0.0.251:5353	2.108	0.000
43	Thu Jan 01 01:00:00...	Thu Jan 01 01:00:00...	10.0.0.9	unknown	UDP		10.0.0.1:53	3.001	4.529

Abbildung C.27: AppAnalysator-«Overview»: «Waze»

Gleich beim Öffnen der App wurde ein unverschlüsselter POST-Befehl an «data.flurry.com» abgesetzt. Die Inhalte dieses Pakets können jedoch als vertretbar angesehen werden. Es werden «lediglich» Informationen zur Bildschirmauflösung und zum verwendeten Betriebssystem übermittelt. Diese scheinen wohl nötig zu sein, um beispielsweise benutzerspezifische Werbungen anzuzeigen.

Nach der Registrierung mittels einem Facebook-Account hat sich Waze angemeldet. Auffällig ist, dass als «userid» nicht die Emailadresse verwendet wurde. Es sieht danach aus, als wird ein verschlüsselter oder zufälliger Benutzername verwendet. In diesem Test war es «BA3olgZfQZmVm8BoaE7K5Q».

Zusätzlich wird zum Benutzername, wie auch schon bei anderen Apps, auch Informationen zum Carrier und zum Verbindungstyp mitgeschickt.

Während dem Benutzen der App wurden nur wenige Daten über HTTP beziehungsweise HTTPS übertragen. Es ist davon auszugehen, dass Meldungen wie beispielsweise die Staumeldungen über TCP übertragen werden, was eine weitere Analyse schwieriger macht.

C.10 Test «Zattoo»

Zattoo erlaubt es beispielsweise auf einem iPad ein Fernsehprogramm live anzuschauen. Damit dies möglich ist, ist ein Benutzeraccount bei Zattoo notwendig. Die Registrierung ist jedoch einfach und schnell erledigt. Nach der Registrierung werden alle verfügbaren Sender aufgelistet. Die Sendernamen werden mit einem aktuellen Vorschaubild ergänzt.

Sobald ein Sender ausgewählt ist, wird nach einer kurzen Werbeeinblendung das Fernsehprogramm abgespielt. Der Wechsel der Sender ist ebenfalls sehr einfach und schnell möglich.

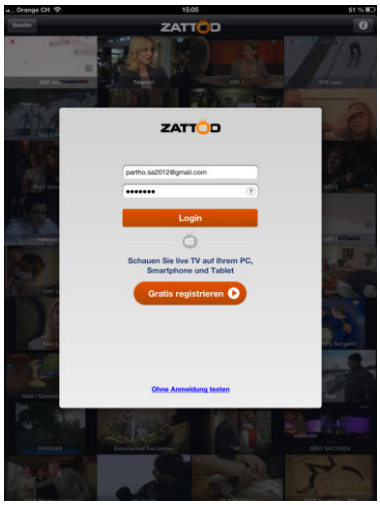


Abbildung C.28: Screenshot «Zattoo»: Benutzeranmeldung

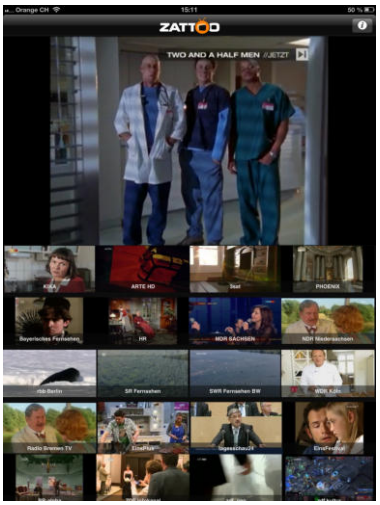
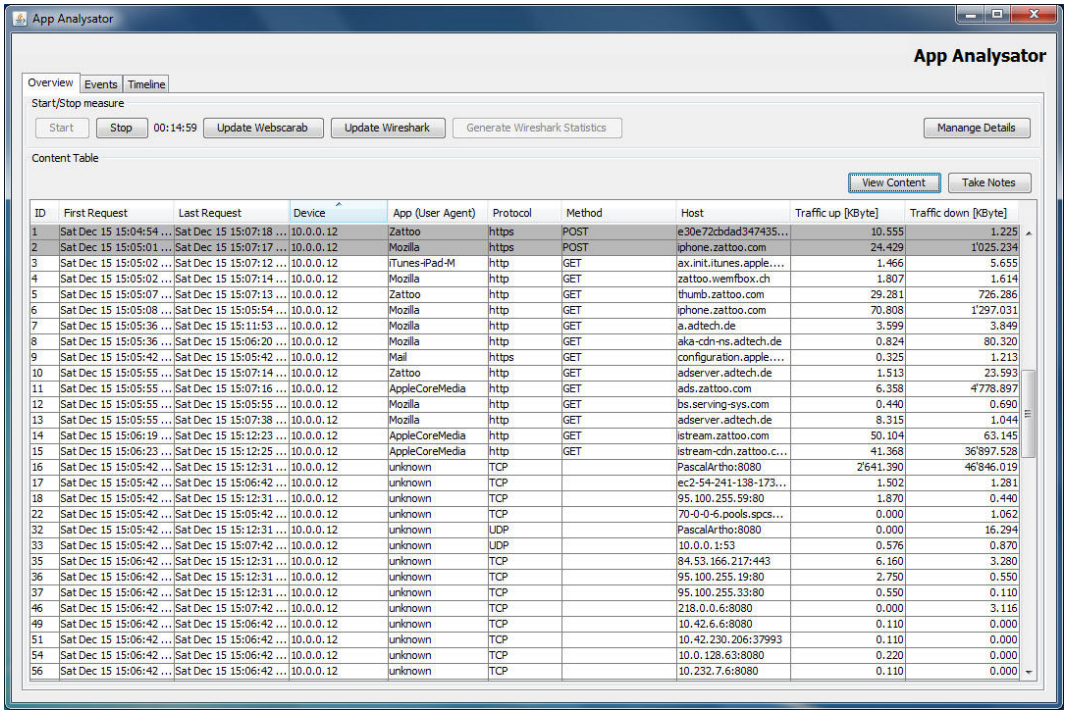


Abbildung C.29: Screenshot «Zattoo»: Wiedergabe Pro7

Ergebnis



ID	First Request	Last Request	Device	App (User Agent)	Protocol	Method	Host	Traffic up [KByte]	Traffic down [KByte]
1	Sat Dec 15 15:04:54 ...	Sat Dec 15 15:07:18 ...	10.0.0.12	Zattoo	https	POST	e30e72cbdad347435...	10.555	1.225
2	Sat Dec 15 15:05:01 ...	Sat Dec 15 15:07:17 ...	10.0.0.12	Mozilla	https	POST	iphone.zattoo.com	24.429	1'025.234
3	Sat Dec 15 15:05:02 ...	Sat Dec 15 15:07:12 ...	10.0.0.12	iTunes-iPad-M	http	GET	ax.init.itunes.apple....	1.466	5.655
4	Sat Dec 15 15:05:02 ...	Sat Dec 15 15:07:14 ...	10.0.0.12	Mozilla	http	GET	zattoo.wemfbox.ch	1.807	1.614
5	Sat Dec 15 15:05:07 ...	Sat Dec 15 15:07:13 ...	10.0.0.12	Zattoo	http	GET	thumb.zattoo.com	29.281	726.286
6	Sat Dec 15 15:05:08 ...	Sat Dec 15 15:05:54 ...	10.0.0.12	Mozilla	http	GET	iphone.zattoo.com	70.808	1'297.031
7	Sat Dec 15 15:05:36 ...	Sat Dec 15 15:11:53 ...	10.0.0.12	Mozilla	http	GET	a.adtech.de	3.599	3.849
8	Sat Dec 15 15:05:36 ...	Sat Dec 15 15:06:20 ...	10.0.0.12	Mozilla	http	GET	aka-cdn-ns.adtech.de	0.824	80.320
9	Sat Dec 15 15:05:42 ...	Sat Dec 15 15:05:42 ...	10.0.0.12	Mail	https	GET	configuration.apple....	0.325	1.213
10	Sat Dec 15 15:05:55 ...	Sat Dec 15 15:07:14 ...	10.0.0.12	Zattoo	http	GET	adserver.adtech.de	1.513	23.593
11	Sat Dec 15 15:05:55 ...	Sat Dec 15 15:07:16 ...	10.0.0.12	AppleCoreMedia	http	GET	ads.zattoo.com	6.358	4'778.897
12	Sat Dec 15 15:05:55 ...	Sat Dec 15 15:05:55 ...	10.0.0.12	Mozilla	http	GET	bs.serving-sys.com	0.440	0.690
13	Sat Dec 15 15:05:55 ...	Sat Dec 15 15:07:38 ...	10.0.0.12	Mozilla	http	GET	adserver.adtech.de	8.315	1.044
14	Sat Dec 15 15:06:19 ...	Sat Dec 15 15:12:23 ...	10.0.0.12	AppleCoreMedia	http	GET	istream.zattoo.com	50.104	63.145
15	Sat Dec 15 15:06:23 ...	Sat Dec 15 15:12:25 ...	10.0.0.12	AppleCoreMedia	http	GET	istream-cdn.zattoo.c...	41.368	36'897.528
16	Sat Dec 15 15:05:42 ...	Sat Dec 15 15:12:31 ...	10.0.0.12	unknown	TCP		PascalArtho:8080	2'641.390	46'846.019
17	Sat Dec 15 15:05:42 ...	Sat Dec 15 15:06:42 ...	10.0.0.12	unknown	TCP		ec2-54-241-138-173...	1.502	1.281
18	Sat Dec 15 15:05:42 ...	Sat Dec 15 15:12:31 ...	10.0.0.12	unknown	TCP		95.100.255.59:80	1.870	0.440
22	Sat Dec 15 15:05:42 ...	Sat Dec 15 15:05:42 ...	10.0.0.12	unknown	TCP		70-0-0-6.pools.spcs...	0.000	1.062
32	Sat Dec 15 15:05:42 ...	Sat Dec 15 15:12:31 ...	10.0.0.12	unknown	UDP		PascalArtho:8080	0.000	16.294
33	Sat Dec 15 15:05:42 ...	Sat Dec 15 15:07:42 ...	10.0.0.12	unknown	UDP		10.0.0.1:53	0.576	0.870
35	Sat Dec 15 15:06:42 ...	Sat Dec 15 15:12:31 ...	10.0.0.12	unknown	TCP		84.53.166.217:443	6.160	3.280
36	Sat Dec 15 15:06:42 ...	Sat Dec 15 15:12:31 ...	10.0.0.12	unknown	TCP		95.100.255.19:80	2.750	0.550
37	Sat Dec 15 15:06:42 ...	Sat Dec 15 15:12:31 ...	10.0.0.12	unknown	TCP		95.100.255.33:80	0.550	0.110
46	Sat Dec 15 15:06:42 ...	Sat Dec 15 15:07:42 ...	10.0.0.12	unknown	TCP		218.0.0.6:8080	0.000	3.116
49	Sat Dec 15 15:06:42 ...	Sat Dec 15 15:06:42 ...	10.0.0.12	unknown	TCP		10.42.6.6:8080	0.110	0.000
51	Sat Dec 15 15:06:42 ...	Sat Dec 15 15:06:42 ...	10.0.0.12	unknown	TCP		10.42.230.206:37993	0.110	0.000
54	Sat Dec 15 15:06:42 ...	Sat Dec 15 15:06:42 ...	10.0.0.12	unknown	TCP		10.0.128.63:8080	0.220	0.000
56	Sat Dec 15 15:06:42 ...	Sat Dec 15 15:06:42 ...	10.0.0.12	unknown	TCP		10.232.7.6:8080	0.110	0.000

Abbildung C.30: AppAnalysator-«Overview»: «Zattoo»

Bei der Registrierung des Zattoo Accounts wird eine verschlüsselte Verbindung zu «iphone.zattoo.com» aufgebaut. Das Passwort und die Emailadresse werden dabei im «Klartext» übertragen. Ebenfalls mitgesendet wird auch der Carrier.

Auffallend ist dass die Kacheln der TV-Sender regelmässig geladen werden. Die Bilder werden jede Minute aktualisiert. Allein durch diese Aktualisierung wird pro Minute rund 250 KB übertragen.

Bei der Auswahl des TV-Senders wird ein POST-Befehl abgesetzt. Darin enthalten ist die Angabe zur Qualität, zum Verbindungstyp und der gewählte Sender. Die Übertragung des Videomaterials wird über HTTP geladen. Somit werden immer ein paar Sekunden des Videos vorgeladen und daraufhin abgespielt.

Konkret wird etwa alle fünf Sekunden ein Request abgesetzt, welcher als Antwort eine kurze Videosequenz erhält. Die Datenmenge dieser Sequenz liegt bei etwa 500 KB. Wird nun eine Minute ein TV Programm angeschaut, entstehen Datenmengen von etwa 6 MB. Je nach Qualität des Videomaterial kann die Datenmenge natürlich abweichen.

Anhang D

Mockups

Nachfolgenden ist die Entwicklung der Mockups im Laufe der Studienarbeit dokumentiert.

Zu Beginn der Arbeit fiel die Entscheidung, eine Tabelle mit den aufgezeichneten Pakete zu erstellen. Siehe Abbildung D.1

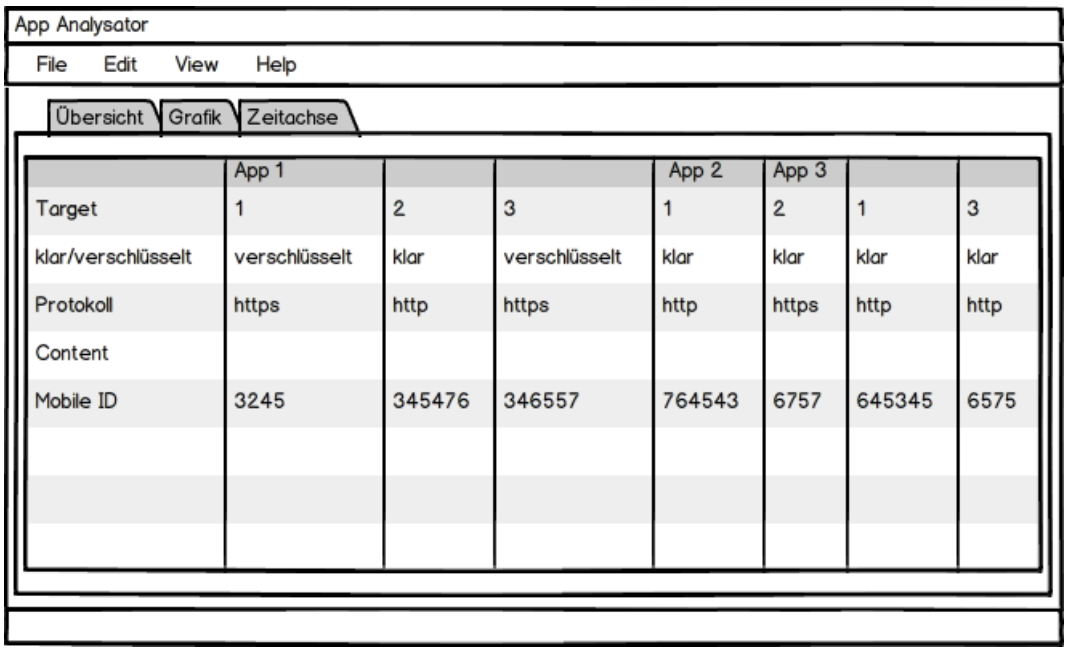


Abbildung D.1: Tab Overview

Der Industriepartner hat die geplante Ansicht mit der Map D.2 nicht als notwendig empfunden, und aus Zeitgründen haben wir sie zum Schuss ganz weggelassen.

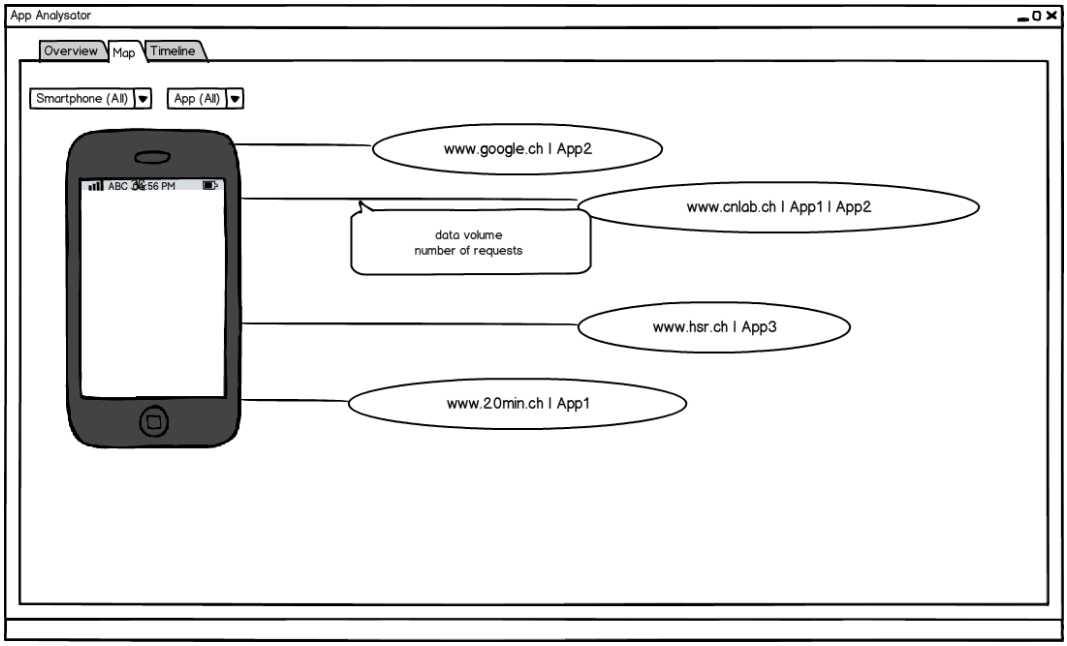


Abbildung D.2: Map

Die beiden weiteren Ansichten Events und die Timeline, waren wie in Abbildung D.3 zusehen ist, in einem Tab geplant. Allerdings haben wir das während dem Programmieren aus Platzgründen auf zwei Tabs ausgelagert.

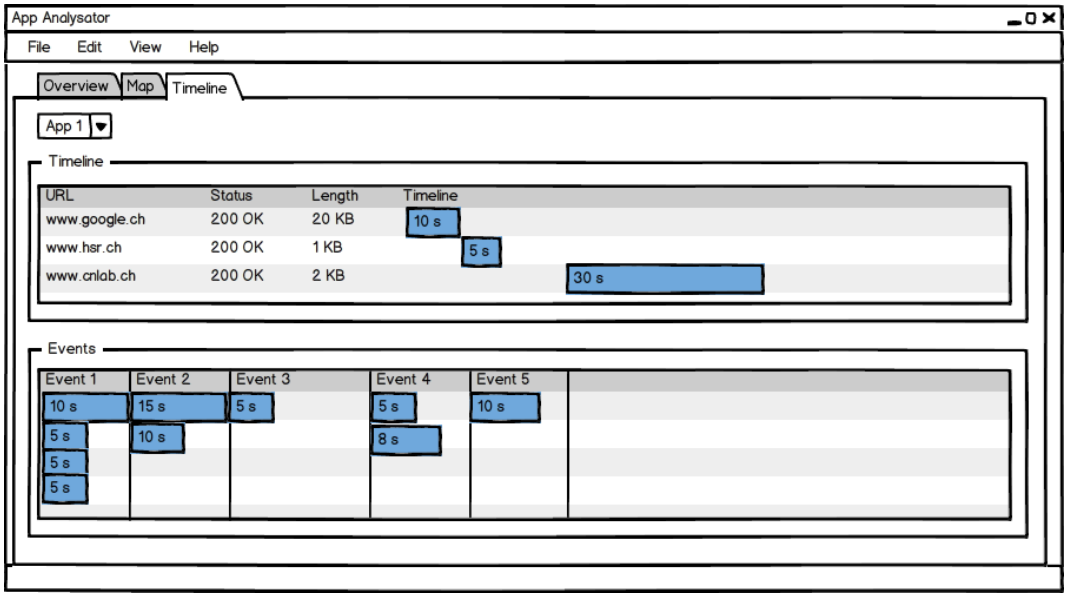


Abbildung D.3: Timeline/Events

Nach einigen Tests und Entwicklungsschritten, fiel die Entscheidung während einer Sitzung auf folgende Ansichten:

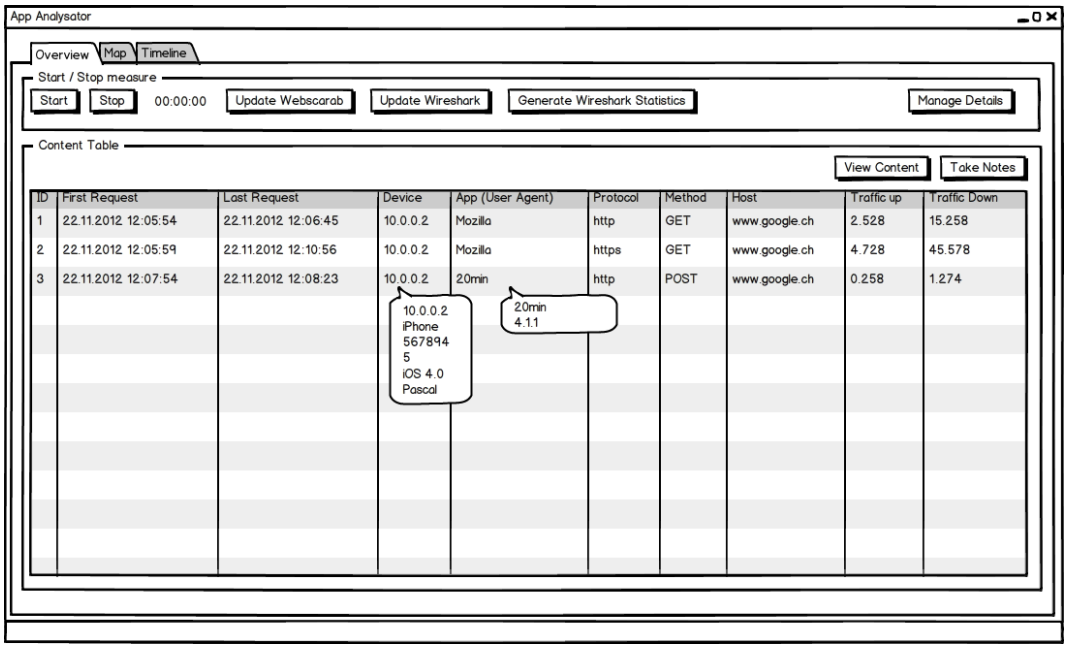


Abbildung D.4: Tab Overview

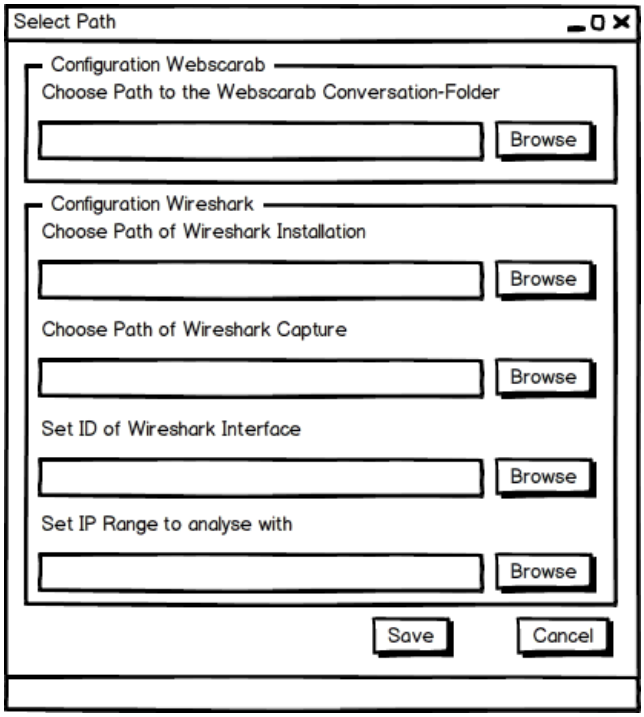


Abbildung D.5: Configure Path

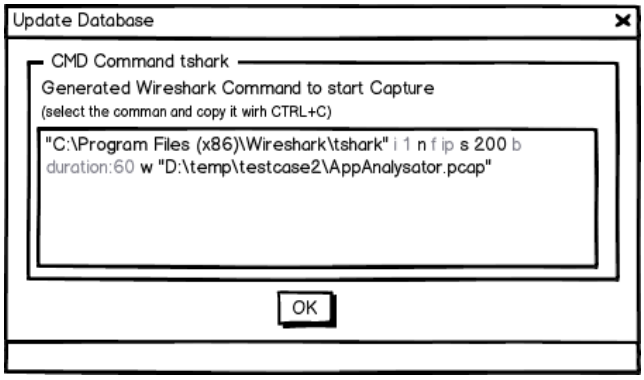


Abbildung D.6: CMD Command tshark

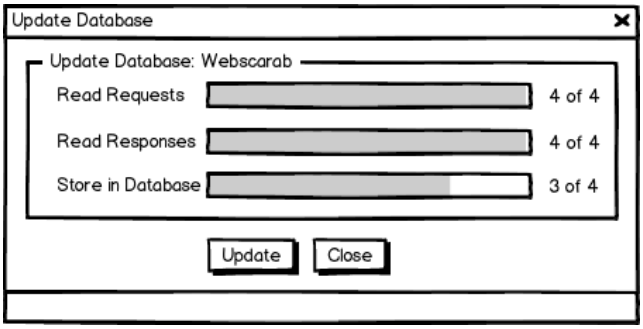


Abbildung D.7: Update Webscarab

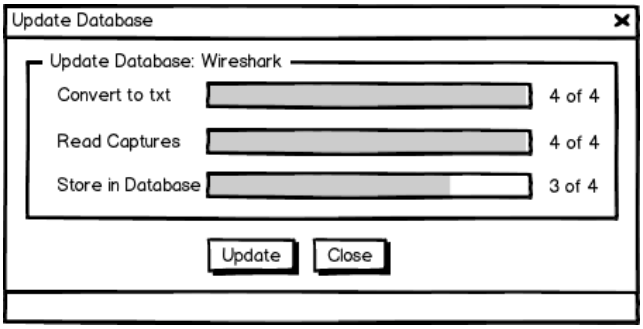


Abbildung D.8: Update Wireshark

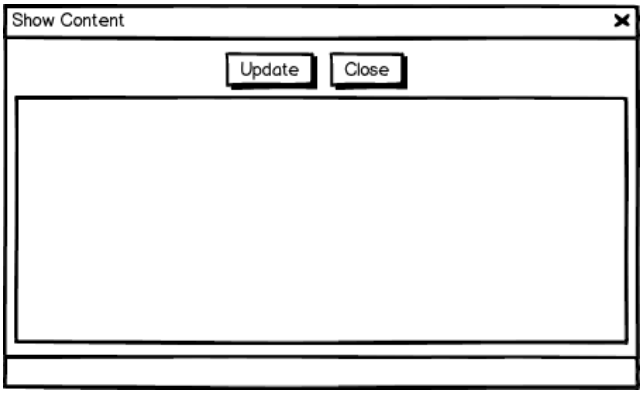


Abbildung D.9: Show Content

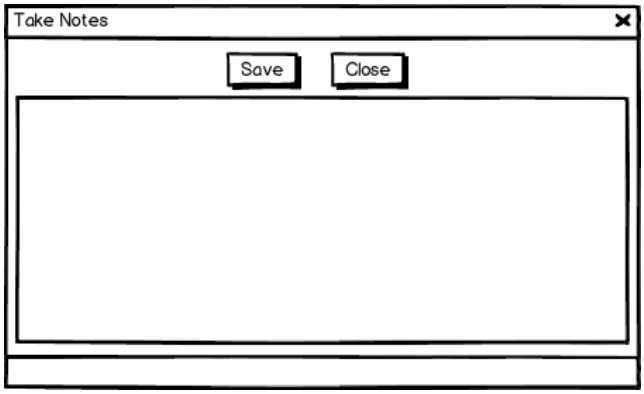


Abbildung D.10: Take Notes

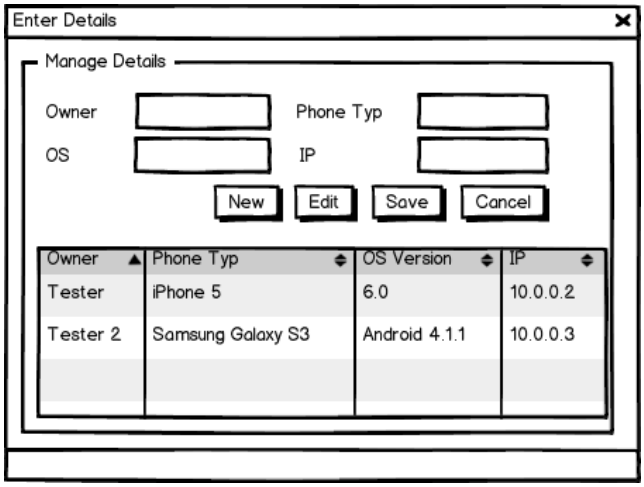


Abbildung D.11: Manage Devices

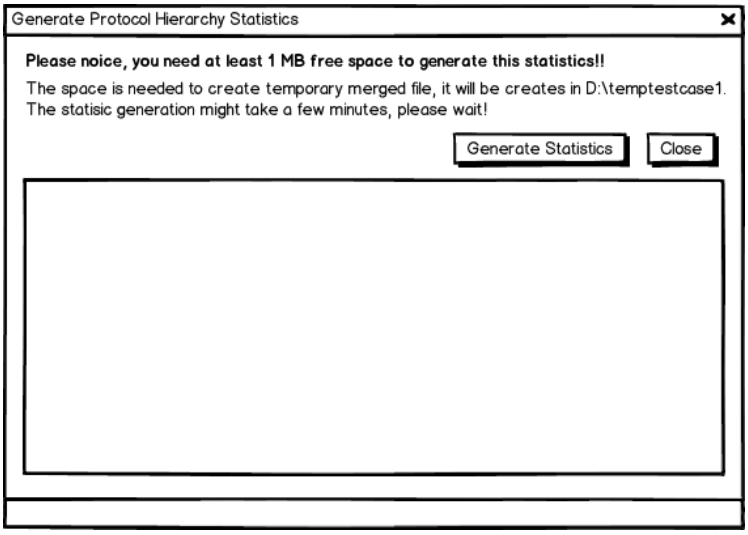


Abbildung D.12: Generate Statisitc

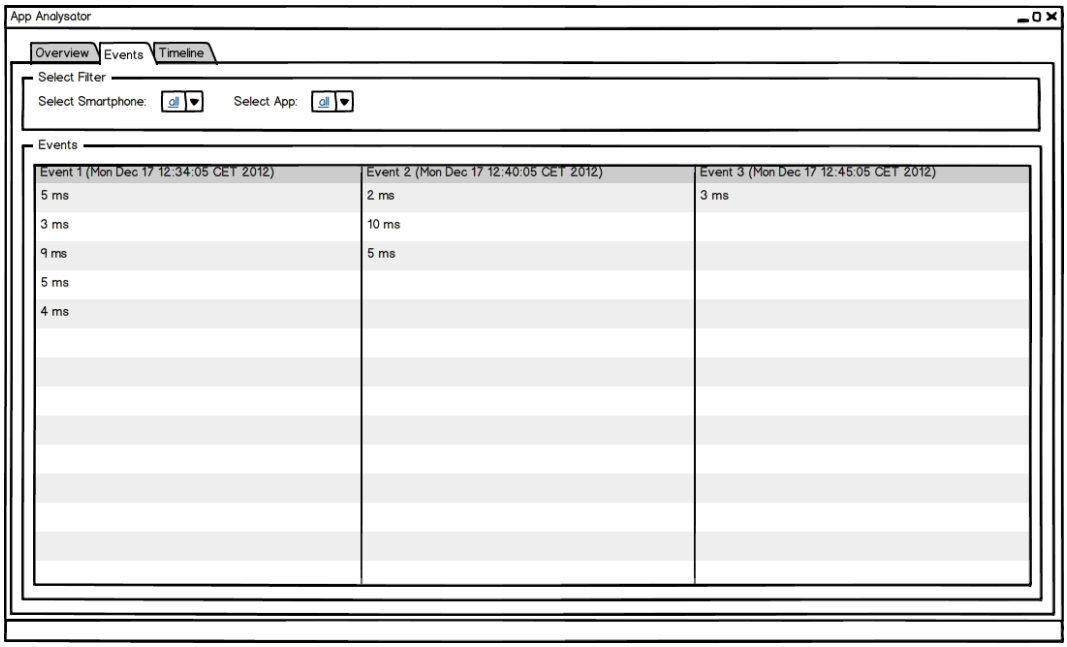


Abbildung D.13: Events

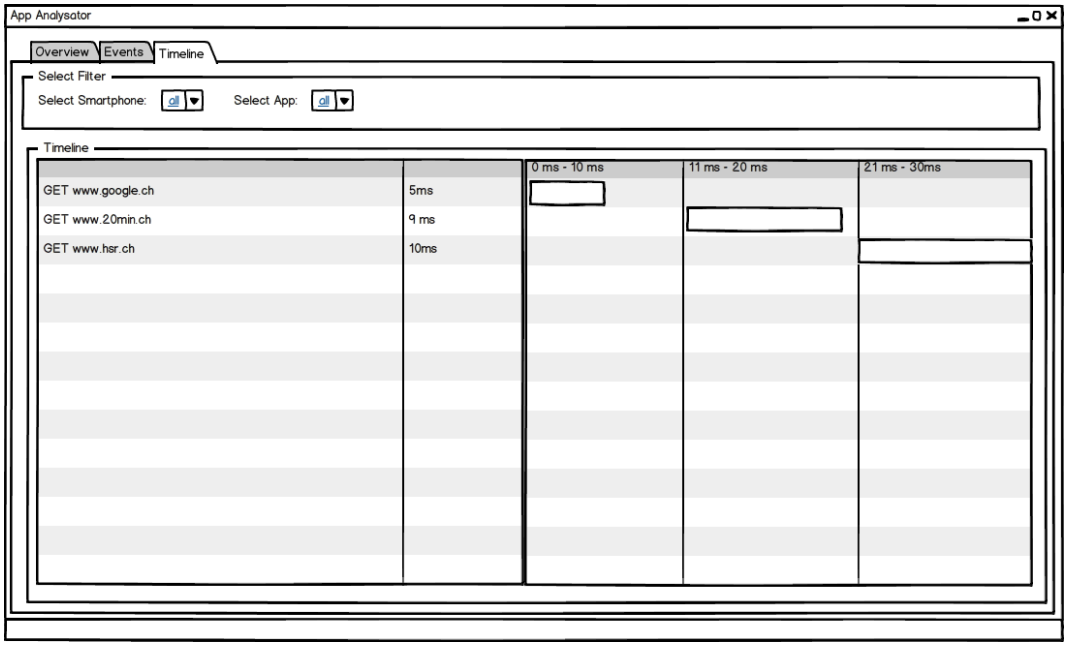


Abbildung D.14: Timeline

Anhang E

Details zur Lösungsfindung

Bei der Entwicklung des AppAnalysator war es notwendig, die Daten von WebScarab und Wireshark zu exportieren wie auch zu importieren. Dieses Kapitel zeigt auf, wie bei WebScarab und Wireshark vorgegangen wurde.

E.1 WebScarab

WebScarab wird als Proxy verwendet und bietet die Möglichkeit, HTTP- und HTTPS-Requests und Responses als Textdatei abzuspeichern. Mit wenigen Klicks kann der Speicherort für diese Daten gewählt werden. Daraufhin werden alle Requests und Responses automatisch und fortlaufend in diesem Ordner abgespeichert. Die Ordnerstruktur, welche durch WebScarab erstellt wird, hat folgenden Aufbau:

```
/
├── conversations/
│   ├── 1-request
│   ├── 1-response
│   ├── 2-request
│   ├── 2-response
│   ├── [...]
│   ├── 10-request
│   └── 10-response
├── fragments/
│   └── [...]
├── conversationlog
├── cookies
└── urlinfo
```

Für die Analyse der WebScarab-Daten ist der «conversations»-Ordner entscheiden. Dieser muss bei der Ausführung des AppAnalysator fix definiert werden. Java erlaubt es den Ordnerinhalt auszulesen und die Dateien durchzugehen.
Bei einem statischen Inhalt eines Ordners wäre das Durchgehen der Dateien nicht weiter problematisch. Da aber unter Umständen immer wieder neue Daten hinzukommen können, muss darauf geachtet werden, dass die Dateiliste lediglich einmal gelesen wird. Die Datei-

liste muss sowohl für das Einlesen der Requests wie auch für das Einlesen der Response verwendet werden. Davon ausgeschlossen müssen die Dateien sein, welche noch benutzt werden (weil beispielsweise der Proxy noch Dateien hineinschreibt) oder welche dessen Zeitpunkt ausserhalb der Messung liegt. Hinzu kommt aber auch, dass bereits gelesene Dateien nicht mehr eingelesen werden.
Nach und nach kamen deshalb zum «einfachen» Einlesen von Dateien immer mehr Bedingungen hinzu, die beachtet werden mussten. Sind diese Bedingungen einmal definiert, kann mit der Dateiliste weitergearbeitet werden. Das nächste Hindernis ist jedoch nicht weit entfernt.
Wie in der Ordnerstruktur angedeutet werden die Requests und Response mit einer Nummer zu Beginn des Namens abgespeichert. Da nach dem ersten Request nicht der zweite folgt, sondern der zehnte, musste die chronologische Reihenfolge mit einer zusätzlichen Methode programmiert werden.
Nachdem aber auch dieses Hindernis überstanden ist, konnten die Dateien eingelesen werden. Dies gestaltete sich als «einfach», da zeilenweise durch die Datei gelesen werden konnte. Selbstverständlich gab es auch hier die eine oder andere Ergänzung, da nicht in jedem Request die selben Headerdaten mitgeliefert werden.
Die gesammelten Daten werden als Java-Objekte zwischengespeichert, um daraufhin in die Datenbank gespeichert zu werden. Der Hintergrund für diesen Zwischenschritt war, dass die Daten zuerst aufbereitet werden sollten. Sind alle Objekte vorbereitet, können diese «in einem Schritt» in die Datenbank übertragen werden. Dabei müssen die Wörter «in einem Schritt» in Anführungszeichen angesehen werden. Selbstverständlich handelt es sich dabei um mehrere SQL-Abfragen und -Einfügeoperationen.
Dieser letzte Schritt war ebenfalls anspruchsvoll, da die Logik des Einfügens und der SQL-Statements vorbereitet werden musste. Schliesslich sollte beispielsweise ein Host nur einmal in der Datenbank eingetragen werden. Wird derselbe Host zu einem späteren Zeitpunkt nochmals bei einem Request aufgerufen, so sollte der gleiche verwendet werden.

E.2 Wireshark

Die Daten aus Wireshark in den AppAnalysator einzulesen war bedeutend aufwendiger. Zum einen musste eine Möglichkeit gefunden werden, eine Wiresharkaufzeichnung auf einfachste Weise zu Starten. Zugleich musste die Aufzeichnung direkt als Datei abgespeichert und in sinnvolle Teildateien aufgeteilt werden können.
Nach einer Internetrecherche kam schnell hervor, dass Wireshark diese Möglichkeit bietet. Konkret ermöglicht Wireshark mit Hilfe des Kommandozeilenbefehls «tshark.exe» eine Aufzeichnung zu starten. Die Anleitung auf «http://www.wireshark.org/docs/man-pages/tshark.html» ist dabei sehr hilfreich.
Mit Zusatzoptionen können weitere Einstellungen zur Aufzeichnungsart vorgenommen werden. Konkret wurden folgenden Optionen gewählt:

Befehl	Beschreibung
-i 1	-i <capture interface> Kennzeichnet das Interface, an welchem der Verkehr aufgezeichnet werden soll. Mittels «tshark -D» können die Interfaces aufgelistet und die Nummer des Interface abgelesen werden.
-n	-n verhindert die Namensauflösung von Hostname sowie TCP und UDP Ports.
-f ip	-f <capture filter> Setzt einen Aufzeichnungsfilter. Konkret werden durch den AppAnalysator lediglich IP-Pakete aufgezeichnet.
-s 200	-s <capture snaplen> Setzt die Aufzeichnungslänge eines Paketes. Ohne dieses Option wird standardmässig bis zu 65535 Bytes pro Paket aufgenommen. Um Speicher zu sparen, reichen 200 Bytes für die wichtigsten Informationen zu Quelle und Empfänge des Paketes aus.
-b duration:60	-b <capture ring buffer option> Die Option ermöglicht es die Aufzeichnung in mehrere Dateien zu unterteilen. Als mögliches Unterteilungskriterium kann eine Zeitdauer (hier 60 Sekunden) verwendet werden. So wird jede Minute eine neue Aufzeichnungsdatei erstellt.

Tabelle E.1: «tshark»-Optionen für die Wiresharkaufzeichnungen [20], sinngemässe Übersetzung

Ein möglicher Befehl mit «tshark» kann sein: «"C:\Program Files (x86)\Wireshark\tshark" -i 1 -n -f ip -s 200 -b duration:60 -w D:\temp\testcase1\AppAnalysator.pcap». Dieser Befehl erstellt im Ordner «D: \temp \testcase1 \» eine Datei mit dem Namen: «AppAnalysator_00001_20121218192615.pcap». In den Dateinamen eingefügt ist eine fortlaufende Kennzahl (hier «00001») sowie ein Zeitstempel (hier «20121218192615»).
Die Wiresharkaufzeichnungen können also in einem ähnlichen Sinne wie bei WebScarab angesprochen werden. Der Ordnerinhalt kann aufgelistet werden und auf die einzelnen Dateien zugegriffen werden. Die Wiresharkaufzeichnungsdatei mit der Dateiendung «.pcap» bringt dem AppAnalysator nicht viel. Es ist eine Art Übersicht notwendig, um an summierte Angaben der Aufzeichnung zu kommen. Eine geeignete Statistik konnte in Wireshark schnell gefunden werden. Die sogenannte «Conversations»-Statistik.
Wie sich nach einer weiteren kurzen Recherche zeigte, kann auch diese Statistik mit einem Kommandozeilenbefehl generiert werden. Weitere Informationen hierzu konnte auf der obengenannten Wireshark-Webseite gefunden werden. Am Schluss konnten folgende zwei Befehle zusammengestellt werden:

- «"C:\Program Files (x86)\Wireshark\tshark" -q -n -z conv,tcp,ip.addr==10.0.0.0/24 -r D:\temp\testcase1\AppAnalysator_00001_20121218192615.pcap > D:\temp\testcase1\AppAnalysator_00001_20121218192615_tcp.txt»

- «“C:\Program Files (x86)\Wireshark\tshark” -q -n -z conv,udp,ip.addr==10.0.0.0/24 -r D:\temp\testcase1\AppAnalysator_00001_20121218192615.pcap > D:\temp\testcase1\AppAnalysator_00001_20121218192615_udp.txt»

Auf der Tabelle E.2 sind die zusätzlich verwendeten Optionen für die Statistikgenerierung aufgelistet.

Befehl	**Beschreibung**
-q	-q Verhindert die fortlaufende Ausgabe der aufgezeichneten Pakete
-z conv,tcp, ip.addr== 10.0.0.0/24	-z conv,type[,filter] Generiert eine Tabelle aller «Conversations». Dabei wird ein Filter «ip.addr==10.0.0.0/24» gesetzt. Dieser filtert die Statistik auf eine Netzadresse.
-r [...]	-r <infile> Gibt an von welcher Aufzeichnungsdatei die Daten gelesen werden sollen
> [...]	> <outfile> Leitet / Schreibt die Ausgabe beziehungsweise die Statistik in eine Datei.

Tabelle E.2: «tshark»-Optionen für die Statistikgenerierung [20], sinngemässe Übersetzung

Die zwei Befehle unterscheiden sich dabei lediglich an zwei Stellen. Dabei wird unterschieden zwischen TCP und UDP. In Java konnten diese Befehle nun zusammengestellt werden und als Prozess gestartet werden. Pro Aufzeichnungsdatei wurden so zwei Textdateien erstellt mit den entsprechenden Statistiken.
Ähnlich wie auch bei WebScarab konnten diese daraufhin zeilenweise ausgelesen und als Objekte gespeichert werden. Die vorbereiteten Objekte konnten daraufhin in die Datenbank geschrieben werden.
Generell dauert das Einlesen der Aufzeichnungen von Wireshark länger. Zum einen kann es zu grösseren Aufzeichnungsdateien kommen, welche dann eine längere Zeit in Anspruch nehmen, um die Statistik generieren zu lassen. Zum anderen können trotz Einschränkung der Netzadresse viele Verbindungen aufgezeichnet werden. Da Wireshark jeweils die IP-Adresse der Quelle und des Empfängers auflistet, muss zusätzlich der Host ausfindig gemacht werden. Hierfür wird eine Namensauflösung versucht. Kann der Name nicht aufgelöst werden, wartet Java bis ein gewisses Timeout abgelaufen ist. Erst dann wird im Code weitergefahren.
Unter Umständen könnte eine kürzere Zeitdauer des Timeout’s helfen, den Vorgang zu beschleunigen. Es kann jedoch nicht festgelegt werden, was eine sinnvolle Timeoutzeit ist. Ist die Zeit zu kurz, geht das Programm unter Umständen zu früh weiter und kein Name wird innerhalb des selbstgesetzten Timeout’s aufgelöst.
Um die Angelegenheit zu beschleunigen wird bei jedem Programmstart eine Liste von bereits angefragten Hostnamen erstellt. Ist eine IP-Adresse bereits aufgelöst, so wird direkt dieser Hostname verwendet. Bei gleichbleibenden Hostnamen beschleunigt diese Liste die

Abarbeitung sehr. Um eine falsche Hostnamenauflösung zu verhindern, wird diese Liste beim Beenden des AppAnalysators gelöscht.

Protocol Hierarchy Statistics

Vom Betreuer der Studienarbeit, Herrn Prof. Dr. Peter Heinzmann, wurde der Wunsch geäussert, dass die sogenannte «Protocol Hierarchy Statistics» interessant wäre. Die Statistik listet die aufgezeichneten Pakete auf und gibt die Anzahl sowie die Datenmenge in Bytes.
Eine solche Statistik kann für jeweils eine Datei der Aufzeichnung von Wireshark erstellt werden. Da jedoch mehrere Aufzeichnungsdateien erstellt werden (pro 60 Sekunden eine Datei), ist es zuerst notwendig, die Dateien temporär zu einer «grossen» Datei zusammenzuführen.
Mit Hilfe dieser Datei kann dann eine Statistik generiert werden und in eine Textdatei gespeichert werden.
Konkret sind also folgende Befehle notwendig:

- «“C:\Program Files (x86)\Wireshark\mergecap.exe” -w D:\temp\testcase1\out.pcap D:\temp\testcase1\AppAnalysator_00001_20121218192615.pcap D:\temp\testcase1\AppAnalysator_00002_20121218192715.pcap [...] »
- «“C:\Program Files (x86)\Wireshark\tshark.exe” -q -z io,phs -r D:\temp\testcase1\out.pcap > D:\temp\testcase1\out.txt»

Auf den nachfolgenden Tabellen sind die «mergecap»-Optionen (siehe Tabelle E.3) und die zusätzlichen «tshark»-Optionen (siehe Tabelle E.4) beschrieben.

Befehl	**Beschreibung**
-w [...] [...]	w <outfile> [<infile> ...] Setzt den Namen für die zusammengesetzte Wiresharkaufzeichnungsdatei und gibt eine Liste von Input-Dateien an.

Tabelle E.3: «mergecap»-Optionen für die «Protocol Hierarchy Statistics» [19], sinngemässe Übersetzung

Befehl	**Beschreibung**
-z io,phs	-z io,phs[,filter] Generiert die eine Tabelle aller «Protocol Hierarchy Statistics» mit der Anzahl Pakete sowie der Anzahl Bytes.

Tabelle E.4: «tshark»-Optionen für die «Protocol Hierarchy Statistics» [20], sinngemässe Übersetzung

Die zwei Befehle werden automatisch durch den AppAnalysator gestartet, sodass die von «tshark» generierte Textdatei eingelesen und in einem einfachen Textfeld im AppAnalysator ausgegeben werden.

Anhang F

Projektmanagement

Meilensteine

Meilenstein 1: Demonstration Testaufbau - 05.10.2012
In der dritten Projektwoche, werden die bisher gesammelten Erkenntnisse dem Industriepartner präsentiert. Und das weitere Vorgehen besprochen.

Meilenstein 2: Abgabe des Einleitungskapitel zur Korrektur an den Betreuer - 12.10.2012
Das Einleitungskapitel wird in der vierten Projektwoche abgegeben dem Betreuer zu Einsicht abgegeben.

Meilenstein 3: manuelle Analyse abgeschlossen - 12.10.2012
Um nicht zu viel Zeit in der Analyse zu verbringen, und die Programmierarbeit zu vernachlässigen, wir die Analyse am ebenfalls in der vierten Woche abgeschlossen.

Meilenstein 4: Aufgabenstellung gutgeheissen - 19.10.2012
Da zu Beginn der Arbeit nicht feststeht, welche Ziel verfolgt wird, sollte bis spätestens der fünften Woche eine Aufgabenstellung definiert werden.

Meilenstein 5: Spezifikation des Demonstrators - 02.11.2012
Die genaue Vorstellung der zu entwickelnde Applikation stand zu Beginn der Arbeit noch nicht fest, sollte aber bis spätestens der siebten Woche feststehen.

Meilenstein 6: Demonstrator Release 1 - 23.11.2012
Ein erstes Release des Demonstrators wird in der zehnten Projektwoche fertiggestellt.

Meilenstein 7: Demonstrator 2 (fertig) - 07.12.2012
Das zweite Release der Applikation wird auch das Ende sein und ist auf die 12. Woche festgelegt worden.

Meilenstein 8: Abgaben - 21.12.2012
Der letzte Meilenstein ist die Abgabe der Studienarbeit.

Zeitplan

Für die Studienarbeit werden jedem Student acht ECTS angerechnet, dies bedeutet einen Aufwand von 240 Stunden pro Person. In der Abbildung F.1 und F.2 sind die geleisteten Stunden angegeben.

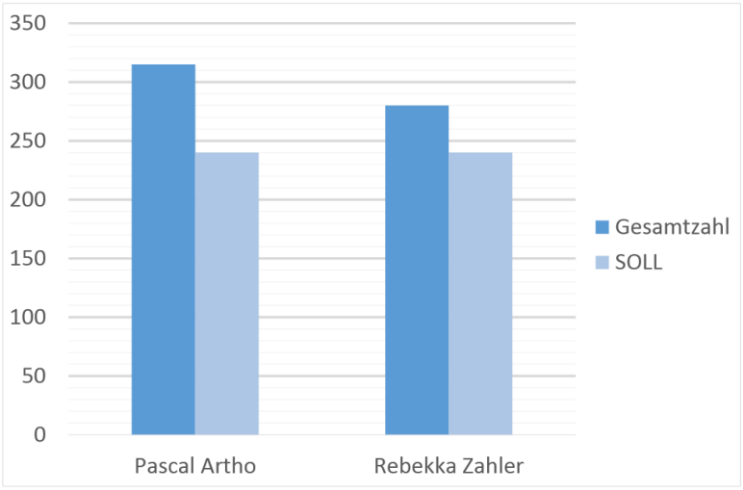


Abbildung F.1: Stundenverteilung Ist/Soll pro Person

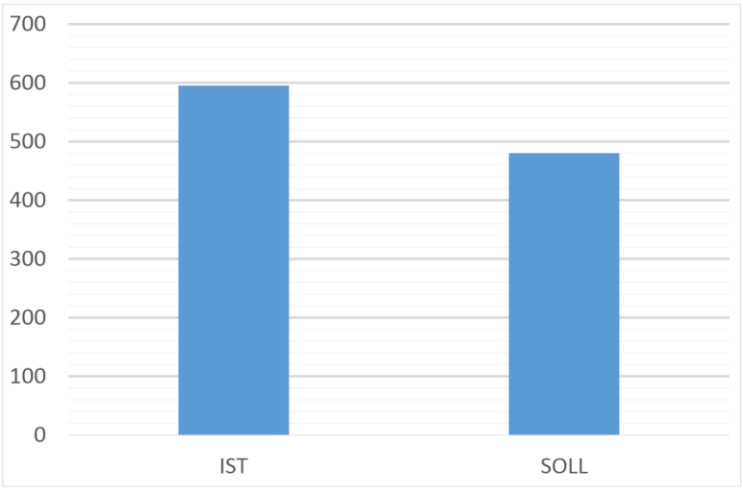


Abbildung F.2: Stundenverteilung Ist/Soll Gesamt

Anhang G

Verwendete Tools

Entwicklung

Für die Entwicklung wurde die Java JDK in der Version 1.7.09 verwendet. Der Javacode wurde im Eclipse Juno geschrieben. Während dem Studium wurde nie eine andere Entwicklungsumgebung benutzt, deshalb sind wir sehr vertraut mit Eclipse.

Dokumentation

Für die Dokumentation der Studienarbeit haben wir Latex benutzt. Um die Dateien zu bearbeiten, verwendeten wir das MikTex 2.9. Der Grund für die Wahl von Latex liegt darin, dass sich diese Lösung gut für Gruppenarbeiten eignet.
Für die Sitzungsprotokoll wurde jeweils Microsoft Word 2012 verwendet, da sich dies zum editieren für alle am besten geeignet hat.

Grafik

Die Grafischen Elemente, wie die Mockups wurden mit Balsamiq Mockups in der Version 2.2.3, das DomainModel mit Enterprise Architekt 9 und die weiteren Grafiken mit Visio von Microsoft Office 2010 erstellt.
In den Modulen Software Engineering eins und Software Engineering zwei Projekt wurden sowohl Balsamiq Mockup als auch Enterprise Architekt eingeführt und benutzt. Weshalb die Wahl auch bei dieser Arbeit auf diese Applikationen fiel.

Analyse

Zusätzlich wurde für die Analyse der Burp Proxy 1.5, WebScarab und Wireshark 1.8.4 benutzt.

Versionsverwaltung und Projektmanagement

Der von der HSR zu Verfügung gestellte virtuelle Server wurde für die Arbeit benutzt. Darauf wurde SVN für die Versionsverwaltung und Redmine für das Projektmanangement installiert.

Anhang H

Anleitung

Diese Anleitung zeigt die Schritt-für-Schritt Konfiguration für eine Analyse mit dem App-Analysator auf. Am Ende dieses Kapitels befindet sich ein Unterkapitel «Zusatzinformationen», in welchem allgemeine Hinweise zur Benutzung notiert sind.

H.1 Versuchsaufbau

Um den Versuch durchführen zu können sind folgende Komponenten und Applikationen nötig:

- Hardware
  - Windows 7 (64Bit) Computer
  - WLAN Router (Netgear Wireless Router WNR1000)
  - AirPcap-Stick (Modell: AirPcap Classic, Media: 802.11 b/g)
  - Testgerät (iPhone 4 16 GB, Version: 5.1.1 (9B206), Modell: MC603FD)
- Software
  - Java Runtime Environment (JRE) (Version 1.7.09)
    (Bezugsquelle: http://www.java.com/de/download/)
  - Wireshark (Version 1.8.4 32Bit)
    (Bezugsquelle: http://www.wireshark.org/download.html)
  - AirPcap Control Panel (wird mit dem AirPcap-Stick mitgeliefert)
  - WebScarab (im Lieferumfang enthalten)
  - «Unlimited Strength Java(TM) Cryptography Extension Policy» (im Lieferumfang enthalten)
  - AppAnalysator-v1.1 (im Lieferumfang enthalten)

Anmerkungen: *Bitte beachten Sie, dass bei der Wireshark Software die 32Bit-Version benötigt wird. Ansonsten kann der AirPcap-Stick nicht verwendet werden für die Wiresharkaufzeichnung.*

Die aufgezählten Hardware-Komponenten können nun für den Versuchsaufbau gemäss nachfolgendem Plan aufgebaut werden. Dabei ist zu beachten, dass der Versuch lediglich funktioniert, wenn sich die Geräte im gleichen Netz / Subnetz befinden. Im aufgezeichneten Plan ist «10.0.0.0/24» die verwendete Netzadresse.

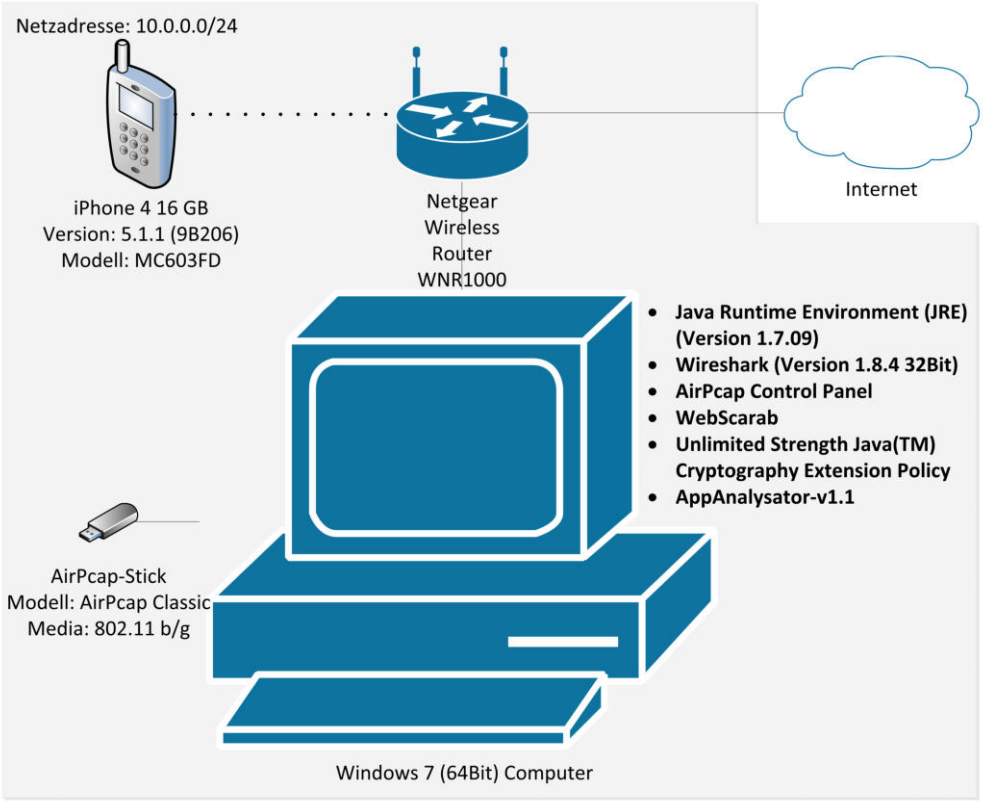


Abbildung H.1: «Versuchsaufbauplan»

H.2 Anforderungen

Der Versuch wurde mit einem Windows 7 (64Bit) Computer durchgeführt. Wir empfehlen ebenfalls einen Windows 7 Computer zu verwenden. Als ersten Schritt gilt es die benötigte Software zu installieren beziehungsweise auf den Computer zu kopieren. Wie bereits erwähnt ist es dabei zwingend notwendig, dass die 32Bit Version von Wireshark verwendt wird. Der weiter benötigte AirPcap-Stick zeichnet den Netzwerkverkehr im WLAN auf und kann den Datenverkehr nur Aufzeichnen, wenn die 32Bit Version von Wireshark verwendet wird. Die Treiber und das Konfigurationsprogramm werden mit dem AirPcap-Stick mitgeliefert.
Nach der erfolgreichen Installation dieser Applikationen müssen zusätzlich im Java Programme-

Ordner «C:\Program Files\Java\jre7\lib\security» die beiden Security Dateien («local_policy.jar» sowie «US_export_policy.jar») ersetzt beziehungsweise ergänzt werden. Diese sind im Ordner «UnlimitedJCEPolicy» beigelegt. Die Dateien selbst sind notwendig, um die verschlüsselten SSL-Verbindungen mit WebScarab aufzubauen.
Vor dem Start des AppAnalysators müssen noch ein paar Konfigurationen getätigt werden, die nun Schritt für Schritt erklärt werden.

H.3 Konfiguration AirPcap

Der AirPcap-Stick wird verwendet um den WLAN-Netzverkehr abzuhören. Die Konfiguration muss dabei so getroffen werden, dass der AirPcap-Stick auf dem gleichen Kanal mithört, wie der Access Point die Signale aussendet. In der Abbildung H.2 sieht man wie die Einstellung auf den Kanal 13 konfiguriert wurde.

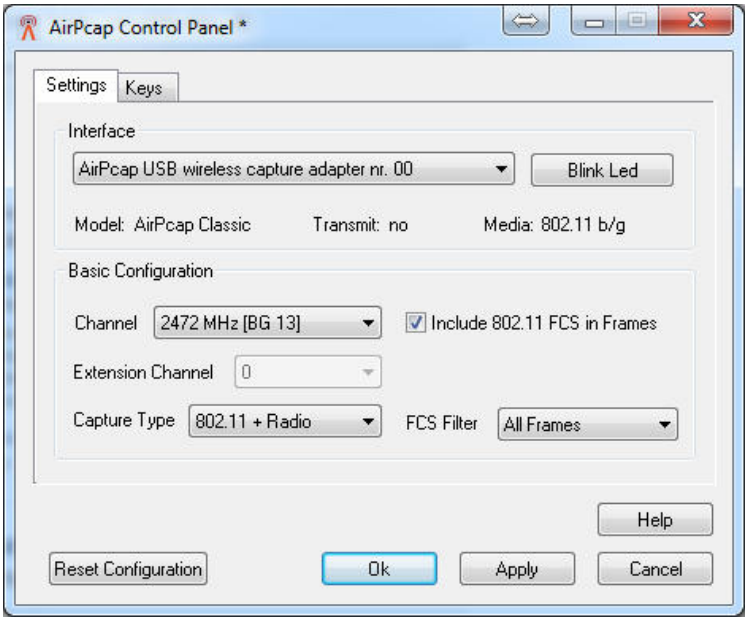


Abbildung H.2: «AirPcap Konfiguration»

Mit dieser Konfiguration ist der AirPcap-Stick bereits einsatzbereit und kann später verwendet werden.

H.4 Konfiguration WebScarab

Als nächstes gilt es WebScarab zu konfigurieren. WebScarab ist ein Proxy, welcher in diesem Versuchsaufbau die HTTP sowie HTTPS Verbindungen aufzeichnet. Die verschlüsselten HTTPS Verbindungen werden dabei entschlüsselt und im Klartext dargestellt.
Beim ersten Start von WebScarab ist eine Änderung notwendig, um sämtliche Funktionalitäten des WebScarab zu benutzen. Dies kann mittels «Tools -> Use full-featured interface» angepasst werden. Um die Änderung zu übernehmen, ist ein Neustart des WebScarab notwendig.
Nach einem weiteren Start des WebScarab muss im Reiter Proxy der Listener folgendermassen eingestellt werden.

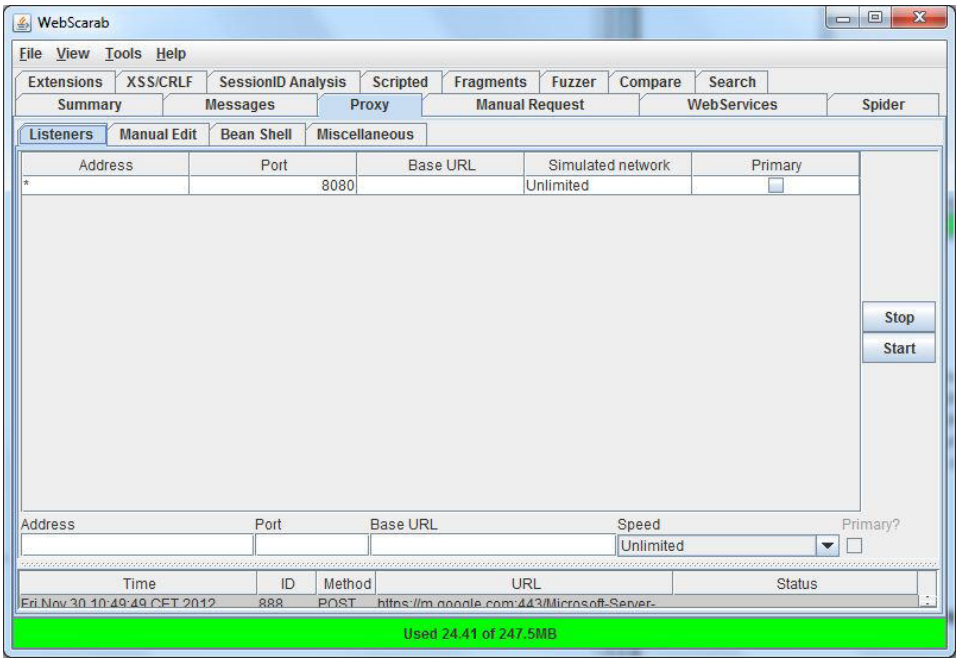


Abbildung H.3: «WebScarab Konfiguration»

Die Spalten «Address» und «Port» geben an, auf welche Adressen beziehungsweise welchen Port der Computer, auf dem der WebScarab läuft, hören soll. Weiter muss ein Speicherort gewählt werden. Dieser ist notwendig um die Requests und Responses, welche der WebScarab aufzeichnet, abzuspeichern. Dafür wählt man unter «File > Save» den gewünschte Ordner aus. Der Speicherort selbst ist später wichtig für die Konfiguration des AppAnalysator.
Damit ist der WebScarab fertig konfiguriert und kann so im Hintergrund laufen gelassen werden.

H.5 Konfiguration Smartphone

Zum Schluss muss das Smartphone für den Versuchsaufbau vorbereitet werden. Damit der verwendete Proxy WebScarab, die verschlüsselten Verbindungen entschlüsseln kann, ist ein gemeinsames Zertifikat notwendig. Dieses muss sowohl im WebScarab verwendet werden, wie auch auf dem Smartphone. Im WebScarab ist dieses Zertifikat bereits konfiguriert, sodass nur noch auf dem Smartphone das Zertifikat installiert werden muss.
Für den Testversuch wurde das Zertifikat in einem öffentlichen Dropbox Ordner abgelegt und ist unter «http://goo.gl/Jvi1M» zugänglich.
Die Installation des Zertifikats ist gänzlich einfach. Auf dem Smartphone kann der obengenannte Link im Browser aufgerufen werden. Bei bestehender Internetverbindung erscheinen automatisch die Einstellungen. Es wird darauf hingewiesen, dass ein Profil installiert werden kann. Dabei wird das geladene Root-Zertifikat als «nicht vertrauenswürdig» betrachtet. Da es sich um ein persönlich generiertes Zertifikat handelt, muss dies jedoch nicht weiter beachtet werden. Das Profil kann somit installiert werden. Nach der Installation wird das Profil als «vertrauenswürdig» eingestuft.

Abbildung H.4: «iPhone»: Shortlink im Safari eingeben

Abbildung H.5: «iPhone»: Shortlink / Zertifikat wird geladen

Abbildung H.6: «iPhone»: Profil installieren

Abbildung H.7: «iPhone»: Zertifikat «AppAnalysator»

Abbildung H.8: «iPhone»: Zertifikatsdetails

Abbildung H.9: «iPhone»: Warnung nicht geprüftes Zertifikat

Abbildung H.10: «iPhone»: Profil installiert, «AppAnalysator» ist vertrauenswürdig

Durch das Akzeptieren des Zertifikats auf dem Smartphone, kann WebScarab die verschlüsselten Verbindungen entschlüsseln.
Anschliessend muss unter den Einstellungen der WLAN Verbindung der Proxy eingetragen werden. Der Port entspricht dem Port, welcher im WebScarab konfiguriert wurde.

Abbildung H.11: «iPhone»: WLAN-Verbindung «rzahler» konfigurieren

Abbildung H.12: «iPhone»: WebScarab Proxy eintragen

H.6 Anleitung AppAnalysator

H.6.1 Vorbereitung

Der AppAnalysator besteht aus zwei Dateien: «AppAnalysator.jar» und «AppAnalysator.db». Letztere befindet sich im Datenbankordner «db». Es ist dabei wichtig, dass die «AppAnalysator.jar»-Datei sowie der Datenbankordner im gleichen Ordner befinden.
Die Applikation kann mittels Doppelklick gestartet werden. Dabei erscheint folgende Oberfläche:

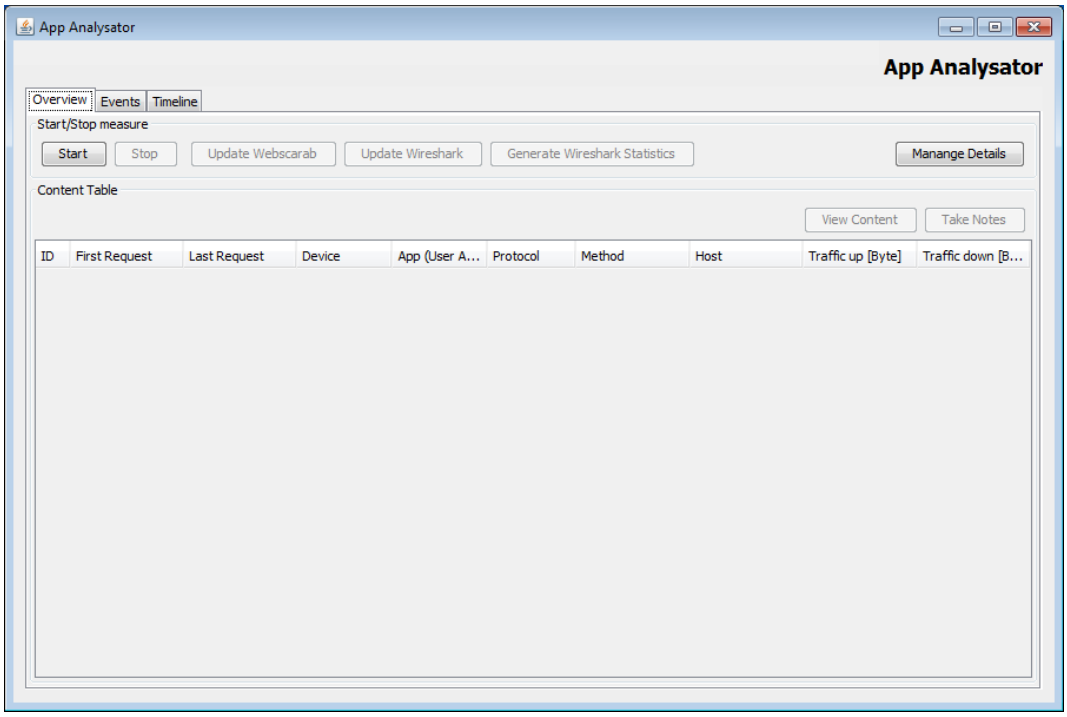


Abbildung H.13: «AppAnalysator nach dem Start»

H.6.2 Konfiguration

Nach dem Start der Applikation kann mit einer Analyse begonnen werden. Hierfür muss der «Start»-Button angeklickt werden. Es erscheint ein neues Fenster in dem einige Konfigurationen vorgenommen werden müssen:

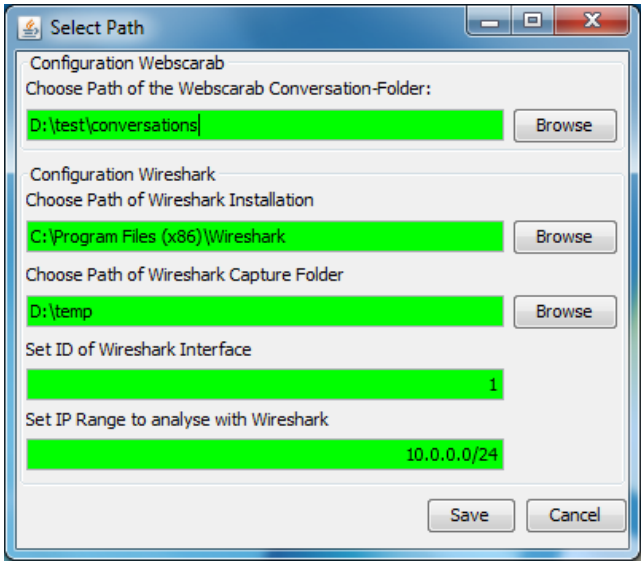


Abbildung H.14: «Select Path Frame»

1. Zuerst muss der Pfad für den WebScarab eingetragen werden. Hier muss der von vorhin angegebene Speicherort angegeben und der Unterordner «Conversations» gewählt werden.

2. Im nächsten Feld ist der Installationsordner von Wireshark gewünscht. Der Standardinstallationspfad ist dabei schon vorgegeben und sollte bei Windows 7 Computer korrekt sein.

3. Als nächstes wird der Pfad angegeben, wo die Aufzeichnungen von Wireshark gespeichert werden sollen. Wählen Sie dabei einen Ordner ohne Sonder- und / oder Leerzeichen. Beispiel für einen möglichen gültigen Pfad: «C:\temp\»

4. Weiter muss das Interface angegeben werden, auf welchem der Wireshark sniffen soll. Um herauszufinden auf welchem Interface sich der AirPcap befindet, kann in der Kommandozeile folgender Befehl verwendet werden: «“C:\Program Files (x86)\Wireshark\tshark” -D». Dieser listet alle Interfaces auf.

5. Als letzte Eingabe muss der IP-Bereich angegeben werden. In der Abbildung H.14 wurde die Netzadresse «10.0.0.0/24» als IP-Bereich angegeben.

Sind all diese Eingaben gemacht und gültig, können die Eingaben mit Klick auf «Save» gespeichert werden. Mit diesem Klick wird automatisch ein Testcase erstellt und eine Stoppuhr gestartet.
Daraufhin erscheint ein weiteres Fenster, welches die getätigten Eingaben zu einem tshark-Befehl zusammengefügt hat. Für die hier vorgeschlagenen Werte lautet der Befehl: «“C:\Program Files (x86)\Wireshark\tshark” -i 1 -n -f ip -s 200 -b duration:60 -w D:\temp\testcase1\AppAnalysator.pcap». Dieser muss in der in der Eingabeaufforderung («cmd.exe») ausgeführt werden, um die Aufzeichnung mit Wireshark zu starten. Am einfachsten geht

dies mittels kopieren des Befehls und dem Einfügen in der Eingabeaufforderung. Die Aufzeichnung selbst kann mit Eingabe von «CTRL+C» in die Eingabeaufforderung gestoppt werden.

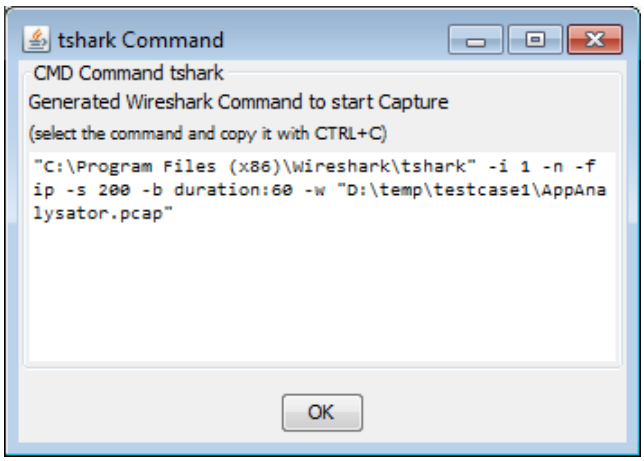


Abbildung H.15: «generierter tshark Befehl»

Bei den Wiresharkaufzeichnungen gilt es anzumerken, dass nur IPv4-Pakete aufgezeichnet werden. Davon werden TCP und UDP-Pakete weiter betrachtet und in die Analyse miteinbezogen.

H.6.3 Durchführung

Nach dem Speichern der Eingaben und dem Starten der Wireshark läuft nun ein Testcase. Über die Button «Update WebScarab» sowie «Update Wireshark» wird die Tabelle mit neuen Paketen gefüllt.

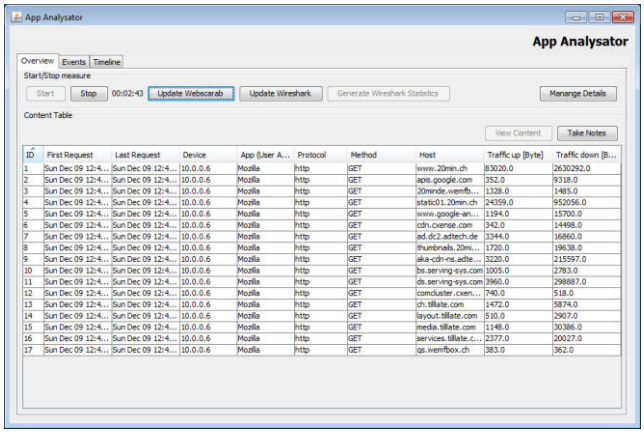


Abbildung H.16: «Overview Frame»

Wenn man zu den jeweiligen Verbindungen den Inhalt ansehen will, klickt man diesen in

der Tabelle an und wählt den Button «View Content». Diese Anzeige ist nur bei HTTP- und HTTPS-Verbindungen möglich. Bei TCP- sowie UDP-Verbindungen ist die Anzeige nicht möglich.

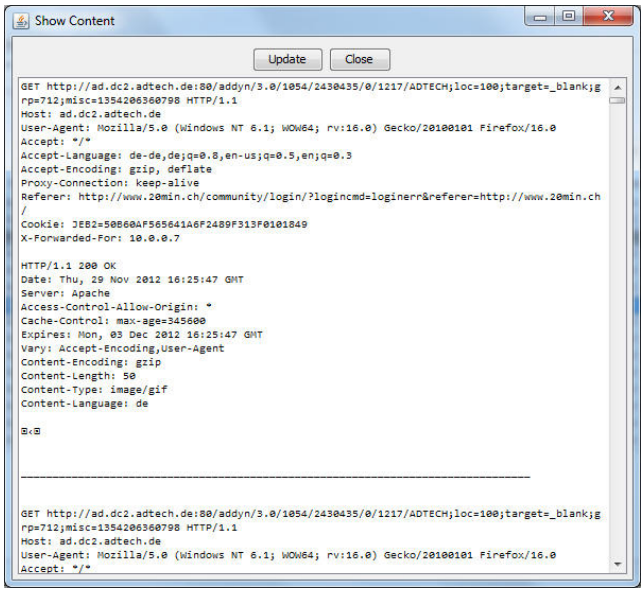


Abbildung H.17: «Content Frame»

Falls Notizen erfasst werden wollen, kann dies über «Take Notes» gemacht werden.

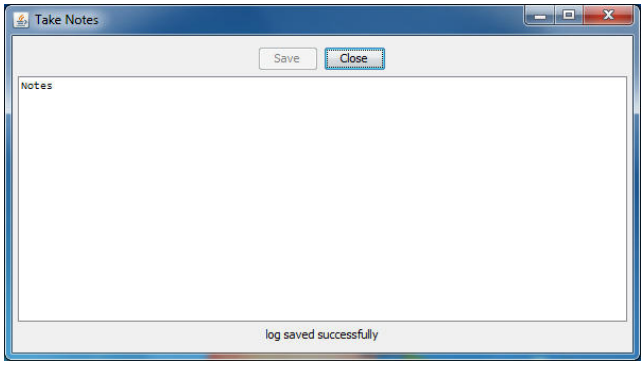


Abbildung H.18: «Notes Frame»

Da der AppAnalysator sich auf die Analyse von TCP- und UDP-Verbindungen beschränkt, kann nicht mit Sicherheit ausgeschlossen werden, dass nicht weitere Pakete übertragen werden. Um diese Unsicherheit etwas zu minimieren, besteht die Möglichkeit, die sogenannte «Protocoll Hierarchy Statistics» anzuzeigen.
In dieser Statistik werden die aufgezeichneten Protokolle aufgelistet und die jeweils übertragenen Pakete und Bytes. Damit diese Statistik erstellt werden kann, werden die einzelnen Aufzeichnungen zu einer «grossen» Datei zusammengefügt.

Je nach Menge der Dateien ist somit die gleiche Menge Speicherplatz temporär notwendig. Die Ausführungszeit ist ebenfalls abhängig von der Menge der Dateien. Aus diesem Grund muss die Generierung der Statistik explizit durch einen Klick auf «Generate Statistics» gestartet werden.
Zum Schluss wird die Statistik in einem Textfeld angezeigt.

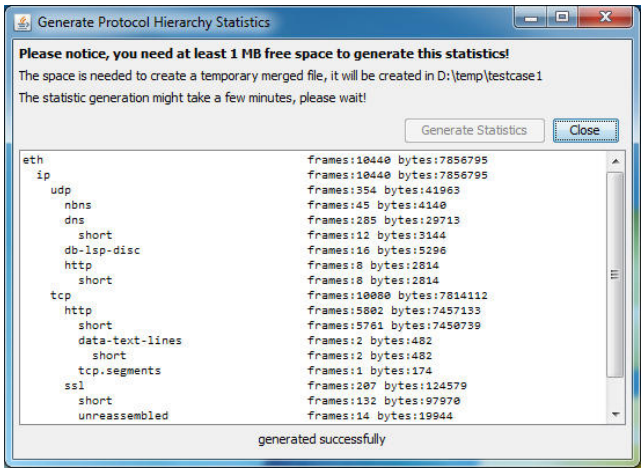


Abbildung H.19: «Generate Protocol Hierarchy Statistics Frame»

H.7 Zusatzinformationen

H.7.1 Allgemeine Hinweise

Bitte beachten Sie, dass in der aktuellen Version einige Einschränkungen bestehen:

1. Bei den Pfadangaben dürfen keine Leerschläge oder sonstige Sonderzeichen in den Pfadnamen vorhanden sein.

2. Bei Klick auf «Update Wireshark» werden die Dateien von den Wireshark Aufzeichnungen ausgelesen. Dabei wird jeweils versucht, den Hostnamen einer IP-Adresse aufzulösen. Es besteht die Möglichkeit, dass dieser Vorgang eine längere Zeitdauer in Anspruch nimmt.
   Bereits abgefragte IP-Adressen und deren Hostnamen werden temporär in einer Liste zwischengespeichert. Die Liste besteht während der Laufzeit des AppAnalysators. Beim nächsten Start des AppAnalysators sind diese jedoch wieder zurückgesetzt und somit leer, um falsche Namensauflösungen zu einem späteren Zeitpunkt zu vermeinden. Grundsätzlich wird jedoch der Vorgang der Namensauflösung beschleunigt, sofern immer etwa die gleichen IP-Adressen aufgerufen werden.

3. Während einer Testdurchführung kann jeweils nur eine Abfrage gleichzeitig auf die SQL-Datenbank gemacht werden. Wird sowohl ein Wireshark-Update wie auch ein WebScarab-Update gemacht, so muss mindestens ein Update warten mit der Speicherung der Daten in die Datenbank.
   Ebenfalls benötigt das Aktualisieren der Tabellen (nach einem Update), ebenfalls Zugriff auf die Datenbank. Sind nun zwei Update-Aufträge gleichzeitig beschäftigt, kann es so auch zu Verzögerungen mit der Aktualisierung kommen.
   Grundsätzlich wird deshalb nicht empfohlen, mehrere Update-Aufträge gleichzeitig durchzuführen.

4. Bei der Ausführung des AppAnalysators wird jeweils eine LOG-Datei erstellt. Darin werden diverse Informationen niedergeschrieben, um mögliche Softwarefehler nachvollziehen zu können. Ohne diese Datei kann unter Umständen ein gefundener Softwarefehler nicht rekonstruiert werden.

5. Ist auf einem Computer zusätzlich zur «Java Runtime Environment» (kurz JRE) auch noch das «Java Development Kit» (kurz JDK) installiert, wird empfohlen, die Policy-Dateien im folgenden Ordner zusätzlich einzufügen:
   «C:\Program Files\Java\jdk1.7.0_09\jre\lib\security»

H.7.2 Anmerkungen zum Speicherplatz

Aufgrund der Wiresharkaufzeichnungen und dem laufenden WebScarab kann es unter gewissen Umständen zu einem erhöhten Speicherbedarf auf der Festplatte kommen. Überprüfen Sie deshalb regelmässig den verfügbaren Speicher Ihrer Festplatte. Sollte der Speicherplatz erschöpft sein, kann die Applikation abstürzen.
Beispiel: Wird die aktuelle Ubuntu CD (Version: 12.10, Desktop i386) mit gut 750 MB von «ftp://mirror.switch.ch» heruntergeladen, erzeugt Wireshark Aufzeichnungsdateien mit

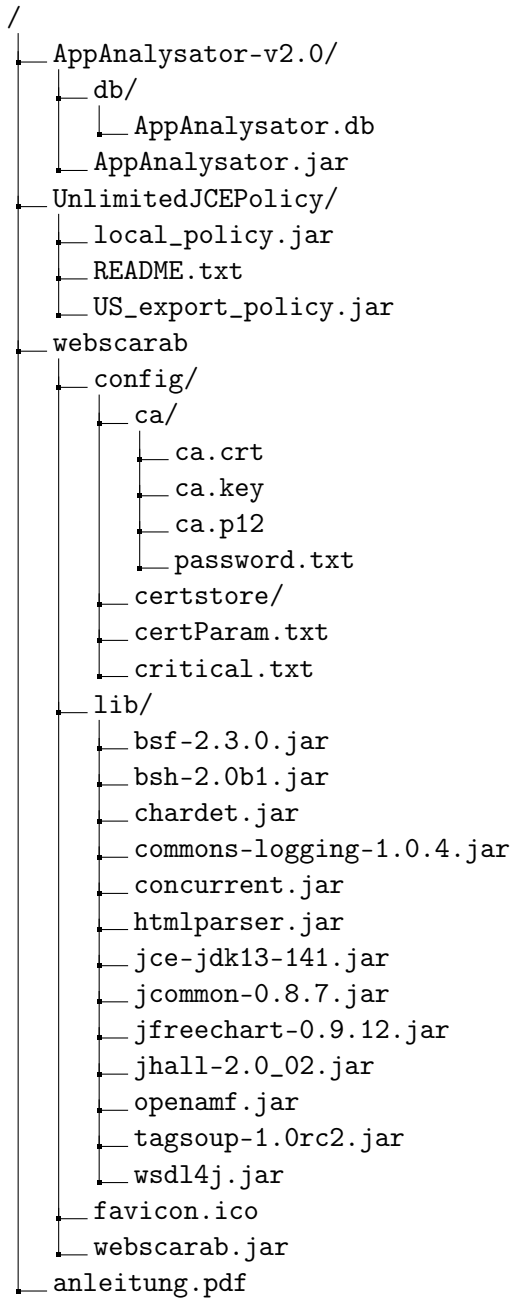
etwas mehr als 140 MB. Wie zu erkennen ist, speichert Wireshark nicht das ganze Paket. Es werden jeweils nur die ersten 200 Bytes eines Paketes gespeichert. Über alles gesehen kann der Speicherbedarf dennoch ziemlich hoch sein.
Ähnliches gilt auch bei WebScarab. Wie oben erwähnt, wurde ein Speicherort für die Requests und Responses festgelegt. Sämtliche Requests und Responses werden also ebenfalls auf der Festplatte abgelegt. Auf Grund dessen wird nicht empfohlen, bewusst grössere Dateien herunterzuladen.
Beispiel: Als grössere Datei kann beispielsweise ein YouTube-Video in höherer Qualität («720p» oder «1080p» sowie längere Videos angesehen werden. Das YouTube-Video wird dabei über eine HTTP-Verbindung verlangt und auch so von YouTube zurückgeschickt. So gesehen wird der gesamte YouTube-Video durch WebScarab in einer Response-Datei gespeichert. Auf der Festplatte mag dies weniger problematisch sein, jedoch wird durch das Einlesen in den AppAnalysator die gesamte Response-Datei in der Datenbank gespeichert. Bei einer einfachen Datenbank, wie der verwendeten SQLite Datenbank, kann dies schnell zu Performance Problemen führen beziehungsweise zu längeren Abfragezeiten.

H.7.3 Lieferumfang

Der Lieferumfang besteht aus den folgenden drei Ordnern:

- Ordner «WebScarab»
  Beinhaltet den WebScarab Proxy mit einem selbst erstellten Zertifikat
- Ordner «UnlimitedJCEPolicy»
  Beinhaltet die «Unlimited Strength Java(TM) Cryptography Extension Policy» von Java
- Ordner «AppAnalysator-v2.0»
  Beinhaltet die AppAnalysator Applikation sowie die Datenbank

Der detaillierte Lieferumfang (mit sämtlichen Dateien) wird mit dem nachfolgenden Dateibaum aufgezeigt:



Anhang I

Inhaltsverzeichnis CD-ROM

Auf der CD-ROM befinden sich folgende Inhalte:

- 01Dokumente
- 02Code
- 03Sitzungsprotokolle
- 04Korrespondenz
- 05Software