

Masterarbeit

# Weiterentwicklung von Methoden und Werkzeugen zum Pentest b. mob. Applikationen

zur Erlangung des akademisches Grades eines  
Master of Science

angefertigt von  
**Dominik Gunther Florian Schlecht**  
Matrikelnummer: 00032209

## Betreuer

Erstprüfer:	Prof. Hahndel
Zweitprüfer:	Prof. von Koch
Allianz Deutschland AG:	Herr Muncan und Herr Gerhager

Fakultät:	Elektrotechnik und Informatik
Studiengang:	Informatik
Schwerpunkt:	Security & Safety

Abgabedatum:	01. April 2016
--------------	----------------

Ingolstadt, 5. Juli 2016



# Erklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit bis auf die offizielle Betreuung durch die Betreuer selbstständig und ohne fremde Hilfe verfasst habe.

Die verwendeten Quellen sowie die verwendeten Hilfsmittel sind vollständig angegeben. Wörtlich übernommene Textteile und übernommene Bilder und Zeichnungen sind in jedem Einzelfall kenntlich gemacht.

Ingolstadt, 5. Juli 2016



# Inhaltsverzeichnis

<b>Erklärung</b>	<b>i</b>
<b>1 Einleitung</b>	<b>1</b>
<b>2 Arten von Penetrationstests</b>	<b>3</b>
2.1 Web-Application . . . . .	3
2.2 Wireless . . . . .	3
2.3 Social-Engineering . . . . .	3
2.4 Mobile-Applications . . . . .	3
2.5 Blackbox/Whitebox . . . . .	3
<b>3 Prozesse zu Penetrationstests</b>	<b>5</b>
3.1 Vorbereitung . . . . .	5
3.1.1 Aufwandsschätzung . . . . .	5
3.1.2 Rechtliche Aspekte . . . . .	5
BDSG . . . . .	5
NDA . . . . .	5
Haftungsausschluss . . . . .	5
Absicherung gegen §203STGB . . . . .	5
3.1.3 Technische Aspekte . . . . .	5
Infrastruktur . . . . .	5
Tools . . . . .	5
3.2 Durchführung . . . . .	5
3.2.1 Bewertung von Findings . . . . .	5
CVSS . . . . .	5
Alternative Modelle . . . . .	5
3.2.2 Dokumentation . . . . .	5
3.2.3 Reporting . . . . .	5
3.3 Nachbereitung . . . . .	5
3.3.1 Inhalte . . . . .	5
3.3.2 Vorlage . . . . .	5
<b>4 Penetrationstests mobiler Anwendungen</b>	<b>7</b>
4.1 Bestehende Anwendungen . . . . .	7
4.1.1 All-In-One-Framework: MobSF . . . . .	7
4.1.2 Einzelanwendungen . . . . .	7
• . . . . .	8
4.2 Aktuelle Situation und Vergleich . . . . .	8
4.2.1 IOS . . . . .	8
Emulation . . . . .	8

	Debugging . . . . .	8
	Memory Corruption . . . . .	10
	Ungesicherte Verbindungen . . . . .	11
4.2.2	Windows-Phone . . . . .	11
	Emulation vs. Hardware . . . . .	11
	Debugging . . . . .	11
4.2.3	Android . . . . .	11
	Android-Studio und SDK . . . . .	13
	Compatibility Testing Suite . . . . .	13
	Emulation vs. Hardware . . . . .	13
	Debugging . . . . .	13
	Logcat . . . . .	13
4.3	Anforderungen und Abgleich mit MobSF . . . . .	13
4.4	Weiterentwicklung MobSF . . . . .	14
4.4.1	Struktur . . . . .	14
4.4.2	<i>strings</i> bei iOS-Analyse . . . . .	14
4.5	Labora Aufbau . . . . .	16
4.6	Entwicklung der Umgebung . . . . .	16
4.6.1	Aufbau . . . . .	16
4.6.2	Schnittstellen . . . . .	16
4.6.3	RPC-Service . . . . .	16
	Flask . . . . .	16
	Requests . . . . .	17
4.6.4	Technisches Detail 1 . . . . .	17
4.6.5	Technisches Detail 2 . . . . .	17
4.7	Abgleich mit Anforderungen . . . . .	17
<b>5</b>	<b>Anwendung der Umgebung</b>	<b>19</b>
5.1	Pentest Anwendung 1 . . . . .	19
5.2	Pentest Anwendung 2 . . . . .	19
<b>6</b>	<b>Fazit</b>	<b>21</b>

# 1 Einleitung

Smartphones verbreiten sich immer stärker. Dies zeigt auch eine Statistik von Gardner, nach welcher die Absätze von Mobilgeräten die von herkömmlichen Rechnern und Laptops weiter übertreffen werden [GartnerSales], siehe Tabelle ?? . Mit diesem Fortschritt im Bereich der Verkäufe steigt natürlich auch die Nutzung von Smartphones. Diese können genutzt werden um im Internet zu surfen, Medien-Inhalte wiederzugeben oder zu chatten. Dies sind im Bezug auf Datenschutz und Risiko relativ ungefährliche Anwendungen. Jedoch können auch kritischere Handlungen vollzogen werden, wie zum Beispiel Online-Banking oder das Verwalten von Versicherungs-Verträgen. Das dies gemacht wird, ist eine notwendige Konsequenz aus der Natur des Smartphones und deren User. So ist es einfach praktisch, seinen Kontostand schnell von unterwegs einsehen zu können. Jedoch müssen solche Apps dann so gestaltet sein, dass ein Missbrauch nicht oder nur mit unverhältnismäßig hohem Aufwand möglich ist. Dies ist die Aufgabe der Unternehmen, welche solche Apps bereitstellen. Dies stellt viele vor unerwartet hohe Herausforderungen. Mussten bisher nur lokale oder Web-Anwendungen getestet werden, sind es nun Applikationen auf mobilen Geräten mit einem oft wesentlich höheren Anteil an Web-Services und APIs.

Device Type	2015	2016	2017	2018
Traditional PCs (Desk-Based and Notebook)	244	228	223	216
Ultramobiles (Premium)	45	57	73	90
PC Market	289	284	296	306
Ultramobiles (Basic and Utility)	195	188	188	194
Computing Devices Market	484	473	485	500
Mobile Phones	1,917	1,943	1,983	2,022
Total Devices Market	2,401	2,416	2,468	2,521

Abbildung 1.1: Worldwide Devices Shipments by Device Type, 2015-2018 (Millions of Units)[GartnerSales]

Die Allianz Deutschland AG ist ein solches Unternehmen. Bisher waren viele Wege zum Kunden analog, also per Brief. Es gab nicht viele elektronische Schnittstellen. Vor 2 Jahren wurde eine Digitalisierungs-Strategie gegenüber beschlossen und die Web-Anwendung "Meine-Allianz" entwickelt. In dieser können Versicherungsnehmer nicht nur Verträge anschauen, sondern auch Änderungen an diesen vornehmen. Sollte diese Anwendung eine schwerwiegende Schwachstelle aufweisen, hätten dies für die Allianz nicht nur Schäden in Bezug auf die Reputation, sondern eventuell zusätzliche rechtliche Folgen, da Krankendaten unter §203 StGB fallen. Um dies zu verhindern, sind Prozesse in der Anwendungsentwicklung notwendig, welche die Sicherheit einer Anwendung sicherstellen.

Diese Prozesse umfassen Komponenten wie Source-Scanning, Penetrations-Test sowie eine regelmäßig wiederkehrende Prüfung von Anwendungen, selbst nach dem Release.

In dieser Arbeit werden als Hinleitung allgemeine Prozesse und Fakten rund um die Applikations-Sicherheit erläutert. Im weiterführenden Teil wird ein bereits bestehendes Open-Source-Tool zum automatischen Testen mobiler Anwendungen um Features wie TODO erweitert.



## **2 Arten von Penetrationstests**

**2.1 Web-Application**

**2.2 Wireless**

**2.3 Social-Engineering**

**2.4 Mobile-Applications**

**2.5 Blackbox/Whitebox**



## **3 Prozesse zu Penetrationstests**

### **3.1 Vorbereitung**

#### **3.1.1 Aufwandsschätzung**

#### **3.1.2 Rechtliche Aspekte**

BDSG

NDA

Haftungsausschluss

Absicherung gegen §203STGB

#### **3.1.3 Technische Aspekte**

Infrastruktur

Tools

### **3.2 Durchführung**

#### **3.2.1 Bewertung von Findings**

CVSS

Alternative Modelle

#### **3.2.2 Dokumentation**

#### **3.2.3 Reporting**

### **3.3 Nachbereitung**

#### **3.3.1 Inhalte**

#### **3.3.2 Vorlage**



## 4 Penetrationstests mobiler Anwendungen

### 4.1 Bestehende Anwendungen

Trotz der relativ neuen Thematik der Mobilen Applikationen gibt es schon einige Programme und Applikationen, die bei der Identifizierung von Schwachstellen helfen können. Im Folgenden sind diese unterteilt in *All-In-One-Framework* und Einzelanwendungen. Die Namen sind hierbei sprechend: Sogenannte *All-In-One-Frameworks* bündeln mehrere kleine Anwendungen und automatisieren den Ablauf oder vereinfachen die Bedienung.

#### 4.1.1 All-In-One-Framework: MobSF

*MobSF* ist das einzige, derzeit öffentlich Verbreitete All-In-One-Framework zur Analyse von Mobilen Applikationen. Es ist eine Plattform zur statischen Analyse von Android und iOS-Apps sowie zur dynamischen Analyse von Android Apps. Es bündelt viele kleinere Anwendungen, welche unter 4.1.2 aufgeführt sind, in einer einfachen Weboberfläche. Es ist Open-Source, in *Python* geschrieben und steht in *GIT* frei zur Verfügung.<sup>1</sup> Die aktuelle Version ist *0.9.2 beta*.

Es unterstützt die statische Analyse von Apps in den Formaten *APK* und *IPA* sowie aus einfach komprimierten Archiven (*zip*). Zusätzlich beinhaltet *MobSF* einen eingebauten API Fuzzer und ist in der Lage, API-spezifische Schwachstellen wie XXE, SSRF oder Path Traversal zu erkennen (TODO Auflisten).

#### 4.1.2 Einzelanwendungen

Das All-In-One-Framework MobSF greift im Hintergrund oft auf eigenständige Tools zurück. Da es für Penetration-Test oft hilfreich ist, diese ohne ein umgebendes Framework nutzen zu können, sind im Folgenden die wichtigsten Tools kurz aufgeführt.

Für Android-Apps:

**jd-core:** *jd-core* ist eine Java Decompiler für Java 5 und spätere Versionen. Er steht unter <http://jd.benow.ca/> zur Verfügung und kann zum Beispiel über das auf der selben Seite zur Verfügung gestellte JD-GUI genutzt werden.

**Dex2Jar (d2j):** <https://github.com/pxb1988/dex2jar>

**Dex2Smali:** <https://github.com/JesusFreke/smali>

**Jar2Java:** *asasd*

**enjarify:** <https://github.com/google/enjarify>

---

<sup>1</sup><https://github.com/ajinabraham/Mobile-Security-Framework-MobSF>

**procyon** : <https://bitbucket.org/mstrobels/procyon/overview>

Für iOS-Apps:

**otool:** -L -hv -Iv  
          [-L] Test

•

## 4.2 Aktuelle Situation und Vergleich

### 4.2.1 IOS

#### Emulation

Die Emulation von iOS-Geräten ist derzeit mit der Verwendung von XCode möglich. XCode wiederum ist nur unter Mac OSX erhältlich. Da Mac OSX laut EULA nur auf „Apple-branded computers“ verwendet werden darf [AppleEULA], ist die Simulation von iOS-Geräten nur unter Apple-Hardware möglich. Nach der Installation über den in Mac OSX enthaltenen App-Store kann ein virtualisierter iPhone über die Schritte XCode, Open Developer Tools, Simulator gestartet werden.

#### Debugging

Als Debugger unter Mac OS X hat sich LLDB etabliert und stellt das Pendant zu GDB unter Linux dar. LLDB ist kostenlos verfügbar, Open-Source und steht unter der <https://opensource.org/licenses/UoI-NCSA.php> Lizenz, welche die Vervielfältigung und Veränderung des Quellcodes unter Hinweis auf LLVM erlaubt.

LLDB sollte auf jedem Mac OS X System mit XCode automatisch installiert sein und lässt sich im Terminal über das Kommando

```
1 lldb
```

aufrufen. Eine Gegenüberstellung von GDB-Kommandos zu LLDB steht unter <http://lldb.llvm.org/lldb-gdb.html> zur Verfügung.

Kompiliert man eine Applikation in XCode, wird diese in einem emulierten iPhone gestartet und direkt ein Fenster LLDB hergestellt. Die ausgeführte Applikation ist in LLDB automatisch ausgewählt.

Ein Ziel dieser Arbeit ist jedoch das Automatisieren von Analysen, weshalb das Ausführen der grafischen Oberfläche nicht optimal ist.

Leider ist nicht erkennbar, wie LLDB und das emulierte iPhone eine Verbindung herstellen. Eine Auflistung der offenen Sockets auf dem System legt jedoch nahe, dass auf dem iPhone das Programm *debugserver* gestartet wird, welches Remote-Debugging mit LLDB erlaubt. Es bleibt herauszufinden, wie die Debugging-Session auf dem simulierten iPhone ohne XCode hergestellt werden kann.

Nach dem Artikel [https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/gdb\\_to\\_lldb\\_transition\\_guide/document/lldb-terminal-workflow-tutorial.html](https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/gdb_to_lldb_transition_guide/document/lldb-terminal-workflow-tutorial.html) von Apple, ist es

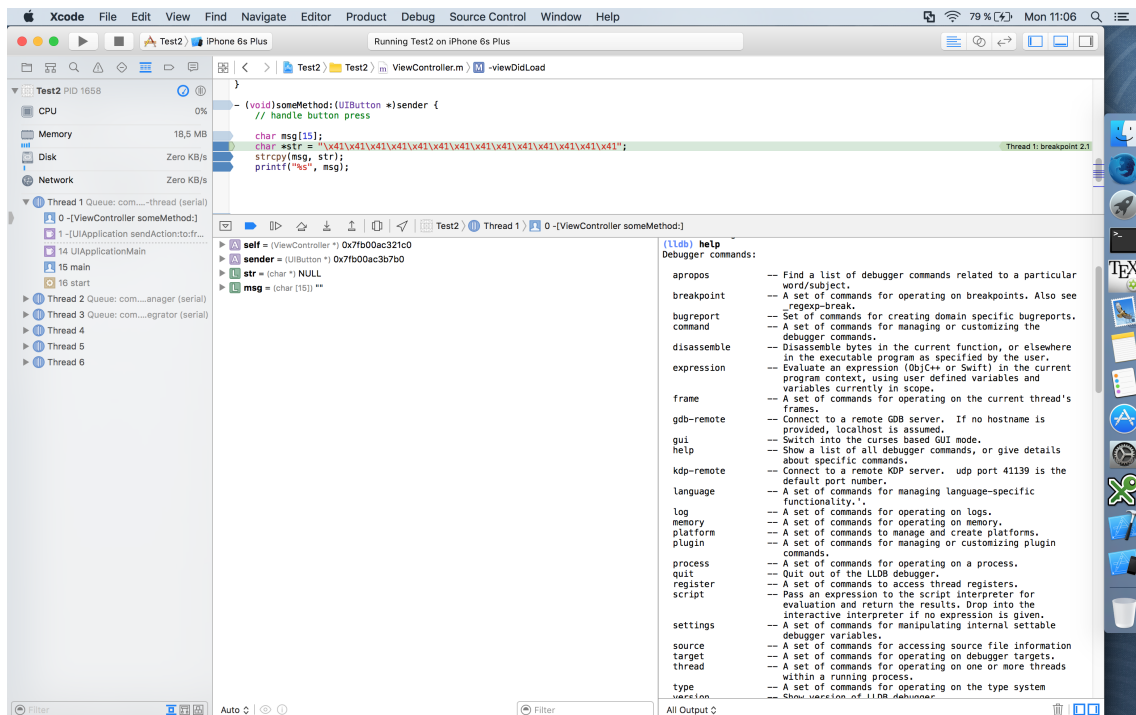


Abbildung 4.1: LLDB in XCode

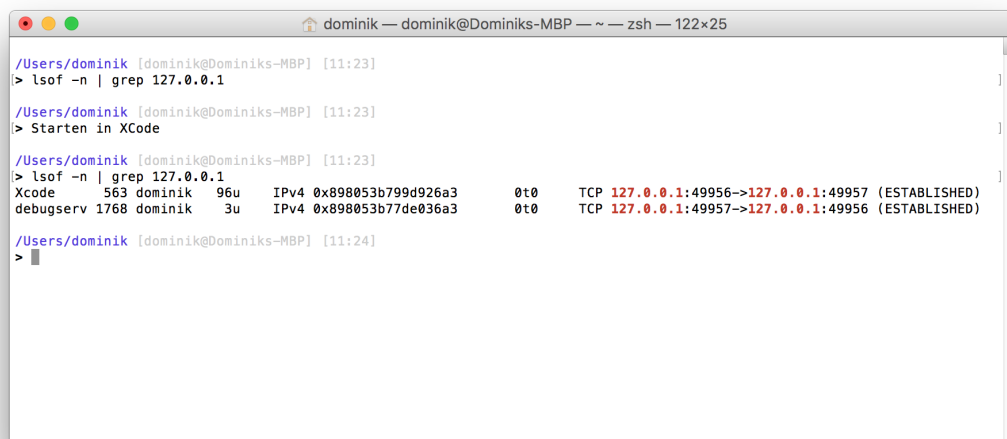


Abbildung 4.2: Vergleich der offenen Pipes vor und nach der Ausführung der Applikation in XCode

möglich, mit LLDB eine App auch als „Standalone Debugger“, also ohne XCode, zu verwenden. Dies ist in Abbildung 4.3 aufgezeigt.



```
dominik — lldb — lldb — lldb Library/Developer/Xcode/DerivedData/Test2-dzkepdcdrczeblbvysnxcbrwmp/Build/Products/Debug-iphonesimulator/Test2.app [12:06]
> lldb Library/Developer/Xcode/DerivedData/Test2-dzkepdcdrczeblbvysnxcbrwmp/Build/Products/Debug-iphonesimulator/Test2.app
(lldb) target create "Library/Developer/Xcode/DerivedData/Test2-dzkepdcdrczeblbvysnxcbrwmp/Build/Products/Debug-iphonesimulator/Test2.app"
Current executable set to 'Library/Developer/Xcode/DerivedData/Test2-dzkepdcdrczeblbvysnxcbrwmp/Build/Products/Debug-iphonesimulator/Test2.app' (x86_64).
(lldb) run --stop-at-entry
Process 2750 launched: '/Users/dominik/Library/Developer/Xcode/DerivedData/Test2-dzkepdcdrczeblbvysnxcbrwmp/Build/Products/Debug-iphonesimulator/Test2.app/Test2' (x86_64)
2016-06-27 12:06:59.691 Test2[2750:72230] *** Assertion failure in Boolean BKSDisplayServicesStart(), /BuildRoot/Library/Caches/com.apple.xbs/Sources/BackBoardServicesFramework_Sim/backboarddemon-2.34/BackBoardServices/BKSDisplayServices.m:38
2016-06-27 12:06:59.696 Test2[2750:72230] *** Terminating app due to uncaught exception 'NSInternalInconsistencyException', reason: 'backboardd isn't running, result: 268435459 isAlive: 0'
*** First throw call stack:
(
  0  CoreFoundation          0x0000000101104d85 __exceptionPreprocess + 165
  1  libobjc.A.dylib         0x0000000100b78deb objc_exception_throw + 48
  2  CoreFoundation          0x0000000101104bea +[NSError raise:format:arguments:] + 106
  3  Foundation              0x00000001007c2e1e -[NSAssertionHandler handleFailureInFunction:file:lineNumber:description:] + 169
  4  BackBoardServices       0x00000001045aeceb BKSDisplayServicesStart + 279
  5  UIKit                   0x00000001014bafcf _UIApplicationMainPreparations + 164
  6  UIKit                   0x00000001014baeda UIApplicationMain + 124
  7  Test2                   0x0000000100691a8f main + 111
  8  libdyld.dylib           0x00000001038d492d start + 1
  9  ???                     0x0000000000000002 0x0 + 2
)
libc++abi.dylib: terminating with uncaught exception of type NSException
Process 2750 stopped
* thread #1: tid = 0x11a26, 0x0000000103c18f06 libsystem_kernel.dylib`__pthread_kill + 10, queue = 'com.apple.main-thread', stop reason = signal SIGABRT
  frame #0: 0x0000000103c18f06 libsystem_kernel.dylib`__pthread_kill + 10
libsystem_kernel.dylib`__pthread_kill:
-> 0x103c18f06 <+10>: jae    0x103c18f10    ; <+20>
    0x103c18f08 <+12>: movq   %rax, %rdi
    0x103c18f0b <+15>: jmp    0x103c137cd    ; cerror_nocancel
    0x103c18f10 <+20>: retq
(lldb)
```

Abbildung 4.3: LLDB als Standalone Debugger

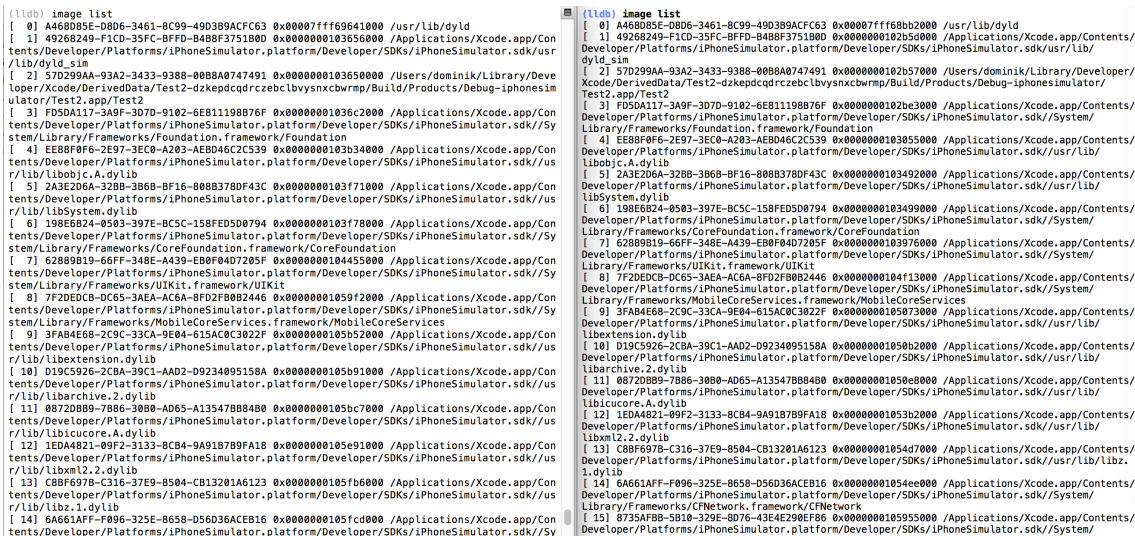
Um zu verifizieren, dass die App auf einem simulierten iPhone ausgeführt wird, können entweder die geöffneten Prozesse (siehe Abbildung ??) oder die geladenen Bibliotheken der Programme (siehe Abbildung 4.4) verglichen werden.

Bei Methoden zeigen, dass die App auf dem simulierten iPhone gestartet wird. Bei den Prozesse ist zu beobachten, dass vor Start von LLDB nur der Hintergrund-Service ausgeführt wurde. Nach dem Start von LLDB dagegen, läuft der gesamte Simulator.

Auch der Vergleich der geladenen Libraries legt nahe, dass die LLDB Standalone und XCode in der gleichen Umgebung ausgeführt werden. Die Adressen im RAM variieren aufgrund von ASLR zwar, aber es werden die Bibliotheken verwendet (am Pfad zu erkennen).

## Memory Corruption Bad Functions





```

(lldb) image list
[ 0] A4680B5E-D806-3461-8C99-490389ACFC63 0x00007fff69641000 /usr/lib/dyld
[ 1] 49268249-F1CD-35FC-8FFD-9408F3751800 0x0000000102b50000 /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator.sdk/usr/lib/dyld_sim
[ 2] 570299AA-93A2-3433-9388-0088A0747491 0x0000000103650000 /Users/dominik/Library/Developer/Xcode/DerivedData/Test2-dzkepdqdcrczcbclbvysnxcwmp/Build/Products/Debug-iphonesimulator/Test2.app/Test2
[ 3] F05DA117-3A9F-307D-9182-6EB1198B76F 0x00000001036c2000 /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator.sdk/System/Library/Frameworks/Foundation.framework/Foundation
[ 4] EE8BF0F6-2E97-3EC0-A203-AEBD46C2C539 0x0000000103b34000 /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator.sdk/usr/lib/libobjc.A.dylib
[ 5] 2A3E2D6A-32B8-3B6B-BF16-80883780F43C 0x0000000103f71000 /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator.sdk/usr/lib/libSystem.dylib
[ 6] 198E6B24-0503-397E-BC5C-158FED5D0794 0x0000000103f78000 /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator.sdk/System/Library/Frameworks/CoreFoundation.framework/CoreFoundation
[ 7] 628B9919-66FF-340E-A439-EB0F94D7205F 0x0000000104450000 /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator.sdk/System/Library/Frameworks/UIKit.framework/UIKit
[ 8] 7F2DEDCB-DC65-3AEA-AC6A-8FD2F80B2446 0x00000001059f2000 /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator.sdk/System/Library/Frameworks/MobileCoreServices.framework/MobileCoreServices
[ 9] 3FAB4E68-2C9C-33CA-9E04-615AC8C3022F 0x0000000105b52000 /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator.sdk/usr/lib/libtextension.dylib
[10] D19C5926-2CBA-39C1-AA02-09234095158A 0x0000000105b91000 /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator.sdk/usr/lib/libarchive.2.dylib
[11] 8072D0B9-7806-380B-AD65-A13547B8B400 0x0000000105bc7000 /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator.sdk/usr/lib/libcure.A.dylib
[12] 1EDA4821-09F2-3133-8C84-9A9187B9FA18 0x0000000105e91000 /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator.sdk/usr/lib/libxml2.2.dylib
[13] C8BF697B-C316-37E9-8504-CB13201A6123 0x0000000105fb6000 /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator.sdk/usr/lib/libx12.2.dylib
[14] 6A661AFF-F096-325E-8658-D56D36ACEB16 0x00000001054ee000 /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator.sdk/System/Library/Frameworks/CFNetwork.framework/CFNetwork
[15] 975AF9B8-5810-322E-8076-43E4E290E786 0x0000000105955000 /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator.sdk/System/
  
```

Abbildung 4.4: LLDB als Standalone Debugger

## Ungesicherte Verbindungen

[https://developer.apple.com/library/ios/documentation/General/Reference/InfoPlistKeyReference/Articles/CocoaKeys.html#//apple\\_ref/doc/other/propertylistkey/NSAppTransportSecurity](https://developer.apple.com/library/ios/documentation/General/Reference/InfoPlistKeyReference/Articles/CocoaKeys.html#//apple_ref/doc/other/propertylistkey/NSAppTransportSecurity)

```

1 NSURL *url = [NSURL URLWithString:@"http://api.ipify.org"];
2 NSData *data = [NSData dataWithContentsOfURL:url];
3 NSString *ret = [[NSString alloc] initWithData:data encoding
    :NSUTF8StringEncoding];
4 NSLog(@"ret=%@", ret);
  
```

```

1 2016-06-28 08:42:56.518 Test2[4789:140270] App Transport
    Security has blocked a cleartext HTTP (http://) resource load
    since it is insecure. Temporary exceptions can be configured
    via your app's Info.plist file.
  
```

## 4.2.2 Windows-Phone

### Emulation vs. Hardware

VS

### Debugging

VS

## 4.2.3 Android

Android ist ein Ursprünglich 2003 von der Android, Inc. entwickeltes mobiles Betriebssystem. 2005 wurde es durch Google übernommen und wird seit dem weiterentwickelt. 2015 liegt es bei einem Marktanteil von TODO %. Aufgrund der Quelloffenheit des Systems wird von

```
1 /Users/dominik [dominik@Dominiks-MacBook-Pro] [12:22]
2 > ps aux | grep Simulator
3 dominik          567    0.0  0.0  2546312   3344   ??  U
   8:51PM    0:00.21 /Applications/Xcode.app/Contents/Developer/
   Library/PrivateFrameworks/CoreSimulator.framework/Versions/A/
   XPCServices/com.apple.CoreSimulator.CoreSimulatorService.xpc/
   Contents/MacOS/com.apple.CoreSimulator.CoreSimulatorService
4 dominik          3330    0.0  0.0  2434840    776 s006  R+
   12:22PM    0:00.00 grep --color=auto --exclude-dir=.bzd --
   exclude-dir=CVS --exclude-dir=.git --exclude-dir=.hg --
   exclude-dir=.svn Simulator
5
6 [Ausführung der App mit LLDB]
7
8 /Users/dominik [dominik@Dominiks-MacBook-Pro] [12:22]
9 > ps aux | grep Simulator
10 dominik          3361  82.6  0.6  2937152 100892   ??  Ss
   12:22PM    0:08.23 /Applications/Xcode.app/Contents/Developer/
   Platforms/iPhoneSimulator.platform/Developer/SDKs/
   iPhoneSimulator.sdk/System/Library/CoreServices/SpringBoard.
   app/SpringBoard
11 dominik          3371  38.4  0.0  2525776   2896   ??  Rs
   12:22PM    0:05.15 /Applications/Xcode.app/Contents/Developer/
   Platforms/iPhoneSimulator.platform/Developer/SDKs/
   iPhoneSimulator.sdk/usr/sbin/notifyd
12 dominik          3344   7.8  0.0  2546992   4560   ??  S
   12:22PM    0:00.97 launchd_sim /Users/dominik/Library/
   Developer/CoreSimulator/Devices/F379F302-76DE-43BA-A6A9-27
   F85C97ED6C/data/var/run/launchd_bootstrap.plist
13 dominik          3389   0.0  0.1  2600072  10244   ??  Ss
   12:22PM    0:00.05 /Applications/Xcode.app/Contents/Developer/
   Platforms/iPhoneSimulator.platform/Developer/SDKs/
   iPhoneSimulator.sdk/usr/libexec/nanoregistrylaunchd
14 dominik          3388   0.0  0.3  2790820  44792   ??  Us
   12:22PM    0:00.24 /Applications/Xcode.app/Contents/Developer/
   Platforms/iPhoneSimulator.platform/Developer/SDKs/
   iPhoneSimulator.sdk/System/Library/PrivateFrameworks/
   ManagedConfiguration.framework/Support/profiled
15 dominik          3387   0.0  0.1  2584432  12680   ??  Ss
   12:22PM    0:00.07 /Applications/Xcode.app/Contents/Developer/
   Platforms/iPhoneSimulator.platform/Developer/SDKs/
   iPhoneSimulator.sdk/usr/libexec/networkd
16 dominik          3386   0.0  0.1  2614364  15684   ??  Ss
   12:22PM    0:00.11 /Applications/Xcode.app/Contents/Developer/
   Platforms/iPhoneSimulator.platform/Developer/SDKs/
   iPhoneSimulator.sdk/System/Library/Frameworks/Accounts.
   framework/accountsd
```

Listing 4.1: Gestartete Prozesse nach LLDB Aufruf

vielen Herstellern auf verschiedensten Plattformen genutzt. Jedoch bringt die weitführende Fragmentierung des Betriebssystems auch Nachteile mit sich. So sind in 2015 nur TODO % der Android-Devices auf einer aktuellen Version.[Drake2014]

### Android-Studio und SDK

Das Android-Studio ist eine umfassende IDE. Sie ermöglicht unter anderem das schnelle Entwickeln und Testen von Apps, sowie die Emulation von beliebigen Android-Versionen. Außerdem ist Android Studio kostenlos, Open-Source und für Linux, Mac und Windows erhältlich. Die aktuelle Version kann unter <http://developer.android.com/sdk/index.html> heruntergeladen werden. Die Installation unter Linux ist vergleichsweise einfach, da nur ein Archiv über das Kommando

```
1 unzip android-studio-ide-143.2739321-linux.zip
```

entpackt werden muss. Für alle anderen Betriebssysteme werden entsprechende Installationsroutinen zur Verfügung gestellt. Anschließend kann die IDE über die Datei „bin/studio.sh“ gestartet werden. Neben dem Android-Studio gibt es noch das Android SDK, welches über die gleiche URL heruntergeladen werden kann. Es enthält wichtige Kommandozeilen-Tools wie *adb*, *fastboot* oder *logcat*, auf welche im weiteren Verlauf noch detailliert eingegangen wird.

### Compatibility Testing Suite

[Drake2014] Seite 18

### Emulation vs. Hardware

Android SDK

### Debugging

Android Debug Bridge[androidDebugBridge]

### Logcat

Android Debug Bridge[androidDebugBridge]

## 4.3 Anforderungen und Abgleich mit MobSF

- Automatisierung
- Blackbox/Whitebox
- Reporting
- False-Positive-Rate

## 4.4 Weiterentwicklung MobSF

Ein Kernelement dieser Arbeit ist die Weiterentwicklung des Mobile Security Frameworks *MobSF*. Die Funktionen des Frameworks sind bereits unter TODO aufgezeigt. Im Folgenden sind die Änderungen dargestellt, welche an dem Framework vorgenommen und veröffentlicht wurden.

### 4.4.1 Syntax

Bei der Neu- oder Reimplementierung wurde auf die Verwendung von offiziellen Style-Guides geachtet. Insbesondere wurde der PEP 8 Standard <sup>2</sup> für Python verwendet, welcher die Lesbarkeit und Wartbarkeit von Python-Code verbessern soll.

### 4.4.2 Struktur

Die Struktur von MobSF war bisher relativ flach. Auf der ersten Ebene findet man die Übergeordneten Module wie den ApiTester, StaticAnalyzer, DynamicAnalyzer sowie den statischen Content, Templates und Kern-Module von MobSF. Dies ist in der Abbildung ?? verdeutlicht. Jedoch hatte die Struktur in den Modulen oft keine saubere Trennung der Aufgaben. So waren im *StaticAnalyzer*-Modul sowohl iOS wie auch Android-Analyse in der *views.py* zusammengefasst. Um hier eine klarere Trennung zu schaffen, wurde die *views.py* aufgliedert in drei Dateien:

**shared\_func.py:** Die *shared\_func.py* enthält alle Funktionen, welche sowohl für iOS als auch Android gebraucht werden. Beispiele sind die Erstellung von Hashes oder das Generieren von PDFs.

**ios.py:** Die Datei *ios.py* enthält alle iOS spezifischen Funktionen zur statischen Analyse.

**android.py:** Die Datei *android.py* enthält alle Android spezifischen Funktionen zur statischen Analyse.

Dies ist im Detail in der Abbildung 4.6 dargestellt.

### 4.4.3 strings bei iOS-Analyse

Zuerst wurde das MobSF um die Fähigkeit erweitert, eine iOS-Applikation mit dem *strings*-Programm zu untersuchen. *strings* durchsucht, insofern keine zusätzlichen Parameter übergeben werden, eine binäre Datei nach 4 aufeinander folgende ASCII-Elemente.

Dies hilft oft bei einer ersten Einschätzung der Anwendung, da oft eine grundlegende Funktionsweise und der Zweck der Software abgeleitet werden kann. Ebenso können eventuell unbeabsichtigt im Programm vergessene Strings in einer App aufgedeckt werden.

## 4.5 Laboraufbau

---

<sup>2</sup><https://www.python.org/dev/peps/pep-0008/>

```
Mobile-Security-Framework-MobSF/  
├── .git/  
├── APITester/  
├── downloads/  
├── DynamicAnalyzer/  
├── LICENSES/  
├── logs/  
├── MalwareAnalyzer/  
├── MobSF/  
├── static/  
├── StaticAnalyzer/  
├── templates/  
└── uploads/
```

Abbildung 4.5: Struktur MobSF auf der ersten Ebene

## 4.6 Entwicklung der Umgebung

### 4.6.1 Aufbau

### 4.6.2 Schnittstellen

### 4.6.3 RPC-Service

Um eine Kommunikation zwischen den verschiedenen Virtuellen Maschinen zu ermöglichen, wurde ein minimaler RPC-Server in Python 3.5 entwickelt. Verwendet wurden hierzu die Bibliotheken *Flask* und *Requests*.

#### Flask

Flask ist ein schneller, minimaler Webserver. Mit nur sehr wenig Code es möglich, eine Schnittstelle bereit zu stellen. Ein Code mit zwei akzeptierenden Funktionen, basierend auf der Schnellstart-Anleitung<sup>3</sup>, ist in Abbildung ?? dargestellt.

Es wird eine minimale Anwendung erstellt, welche auf dem Pfad `/` lokal das print-Statement ausführt und den Text 'Hello World!' zurück gibt. Wird die Anwendung unter dem Pfad `/second_command/` angesprochen, wird ein anderes print-Statement ausgeführt und ein anderer Wert zurück gegeben. Auf diese Weise können schnell API-Funktionen auf verschiedene Pfade gelegt und angesprochen werden.

#### Requests

Requests ist eine Python-Bibliothek, welche einfache Anfragen (sogenannte Requests, daher der Name) an Web-Server ermöglicht. So ist es über ein kurzes Code-Snippet, dargestellt in Abbildung ??, möglich die unter ?? aufgezeigt Schnittstelle anzusprechen.

Wird zuerst der Code ?? und anschließend der Code ?? ausgeführt, wird auf Server-Seite folgende Ausgabe erzeugt:

<sup>3</sup><http://flask.pocoo.org/docs/0.10/quickstart/>

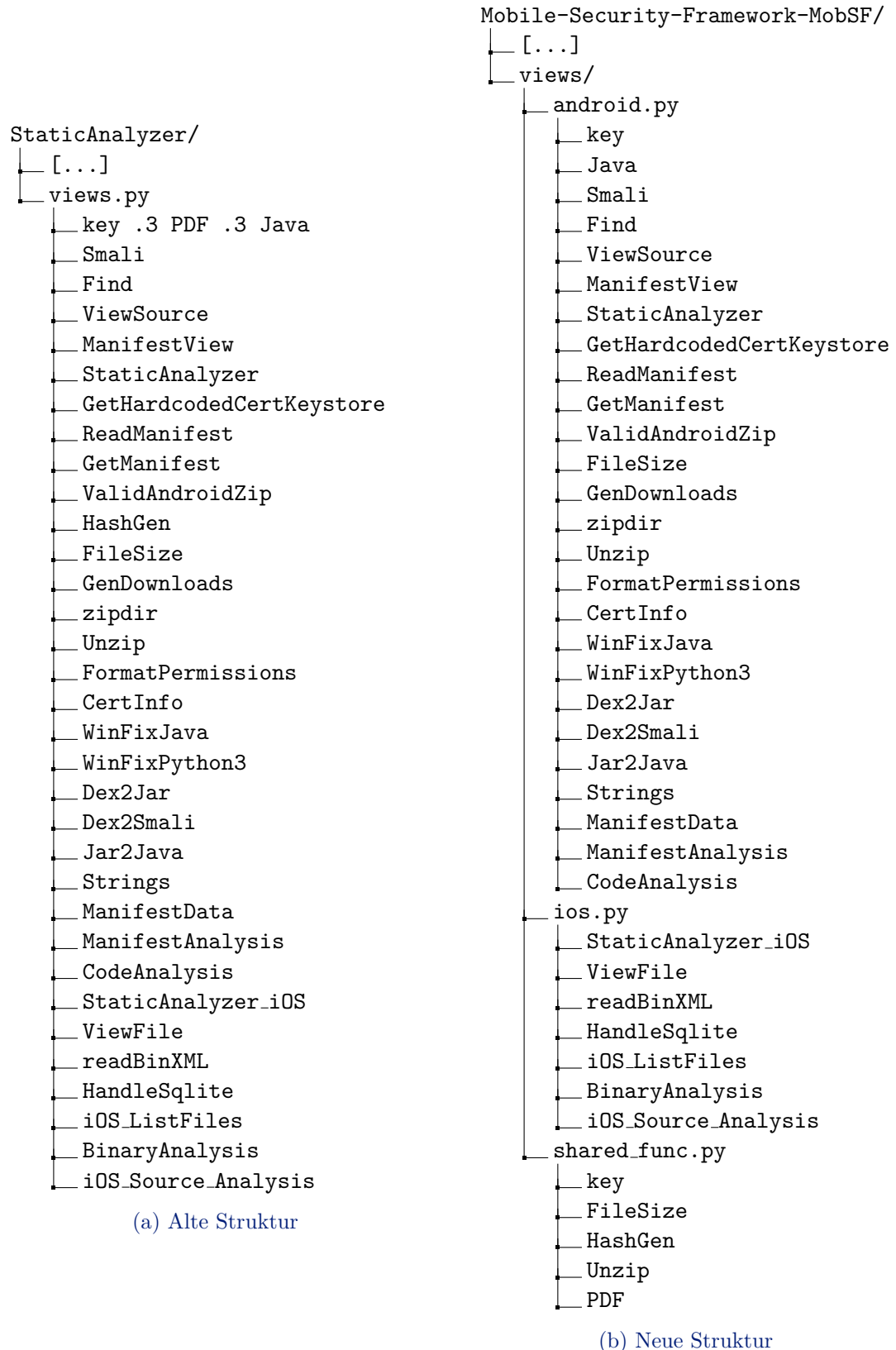


Abbildung 4.6: Vergleich der Struktur von StaticAnalyzer

```
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route('/')
5 def hello_world():
6     print("Execute command!")
7     return 'Hello World!'
8
9 @app.route('/second_command/')
10 def not_hello_world():
11     print("Execute second command!")
12     return 'Goodbye World!'
13
14 if __name__ == '__main__':
15     app.run()
```

Abbildung 4.7: rpc\_client.py

```
1 import requests
2
3 r = requests.get('http://localhost:5000')
4 print(r.text)
5
6 r = requests.get('http://localhost:5000/second_command/')
7 print(r.text)
```

Abbildung 4.8: rpc\_server.py

```
1 $ python3 rpc_server.py
2 Hello World!
3 Goodbye World!
```

Auf der Client-Seite erfolgt folgende Ausgabe:

```
1 $ python3 rpc_client.py
2 * Running on http://127.0.0.1:5000/
3 Execute command!
4 127.0.0.1 - - [18/May/2016 19:10:56] "GET / HTTP/1.1" 200 -
5 Execute second command!
6 127.0.0.1 - - [18/May/2016 19:10:56] "GET /second_command/ HTTP
  /1.1" 200 -
```

Aufgrund dieser Basis wurde das RPC-Tool implementiert.

#### **4.6.4 Technisches Detail 1**

#### **4.6.5 Technisches Detail 2**

### **4.7 Abgleich mit Anforderungen**



## **5 Anwendung der Umgebung**

### **5.1 Pentest Anwendung 1**

### **5.2 Pentest Anwendung 2**



## 6 Fazit

