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, 18. Mai 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, 18. Mai 2016

Inhaltsverzeichnis

1 Einleitung

Smartphones verbreiten sich immer stärker. Dies Zeigt auch eine Statistik von Gardner, nach welcher die Absätze von Mobilen Geräten die von herkömmlichen Rechnern und Laptops weiter übertreffen werden [4], siehe Tabelle 1. Mit diesem Fortschritt im Bereich der Verkäufe steigt natürlich auch die Nutzung von Smartphones. Diese können genutzt werden um im Internet zu ßurfen", 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 bereit stellen. 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)[4]

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-Allianzëntwickelt. 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 ğ203 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 ğ203STGB

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.[1] 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

4.2 Aktuelle Situation und Vergleich

4.2.1 IOS

Emulation vs. Hardware

Die Emulation von iOS-Geräten ist derzeit mit der Verwendung von XCode möglich. XCode wiederum ist nur unter Mac OSX erhältlich. Da Max OSX laut EULA nicht auf nicht Äpple-branded computers"verwendet werden darf [2], ist die Simulation von iOS-Geräten nur unter Apple-Hardware möglich.
TODO warten auf MacBook

Debugging

XCode

[1] https://github.com/ajinabraham/Mobile-Security-Framework-MobSF

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 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.[3]

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 werde kann. Es enthält wichtige Kommandozeilen-Tools wie *adb*, *fastboot* oder *logcat*, auf welche im weiteren Verlauf noch detailliert eingegangen wird.

Compatibility Testing Suite

[3] Seite 18

Emulation vs. Hardware

Android SDK

Debugging

Android Debug Bridge[1]

Logcat

Android Debug Bridge[1]

4.3 Anforderungen und Abgleich mit MobSF

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

4.4 Laboraufbau

4.5 Entwicklung der Umgebung

4.5.1 Aufbau

4.5.2 Schnittstellen

4.5.3 Technisches Detail 1

4.5.4 Technisches Detail 2

4.6 Abgleich mit Anforderungen

5 Anwendung der Umgebung

5.1 Pentest Anwendung 1

5.2 Pentest Anwendung 2

6 Fazit

Literatur

[1] *Android Debug Bridge*. URL: http://developer.android.com/tools/help/adb.html (besucht am 22. 02. 2016).

[2] Inc. Apple. *OSX 10.11 EULA*. URL: http://images.apple.com/legal/sla/docs/OSX1011.pdf (besucht am 16. 05. 2016).

[3] Joshua J. Drake u. a. *Android Hacker's Handbook*. John Wiley & Sons, Inc., Indianaposil, Indiana, 2014.

[4] Gartner. *Gartner Sales PC/Mobile Phones*. URL: http://www.gartner.com/newsroom/id/3270418 (besucht am 16. 05. 2016).