STAMBA: Security Testing for Android Mobile Banking Apps

Sriramulu Bojjagani and V.N. Sastry

Abstract Mobile banking activity plays a major role for M-Commerce (Mobile-Commerce) applications in our daily life. With the increasing usage on mobile phones, vulnerabilities against these devices raised exponentially. The privacy and security of confidential financial data is one of the major issues in mobile devices. Android is the most popular operating system, not only to users but also for companies and vendors or (developers in android) of all kinds. Of course, because of this reason, it's also become quite popular to malicious adversaries. For this, mobile security and risk assessment specialists and security engineers are in high demand. In this paper, we propose STAMBA (Security Testing for Android Mobile Banking Apps) and demonstrate tools at different levels. These supported tools are used to find threats at a mobile application code level, communication or network level, and at a device level. We give a detailed discussion about vulnerabilities that help design for further app development and a detailed automated security testing for mobile banking applications.

1 Introduction

Android mobile application and operating system security has been clearly explained in [1, 8, 9], but some improvements are are to be made to the implementation of android security because versions of android operating system was started with the Cupcake 1.5, now KitKat 4.4, expected in future is KeyLimePie 5.0 [7]. Recent developments in android mobile operating system have been tested and demonstrated by drozer framework [14] and wire shark packet analyzer [5]. Android work based on

S. Bojjagani(✉) · V.N. Sastry
Centre for Mobile Banking (CMB), Institute for Development and Research in Banking Technology (IDRBT), Hyderabad, India
e-mail: sriramulubojjagani@gmail.com, vnsastry@idrbt.ac.in

S. Bojjagani
School of Computer and Information Sciences, University of Hyderabad, Hyderabad, India

S.M. Thampi et al. (eds.), *Advances in Signal Processing and Intelligent Recognition Systems*,
Advances in Intelligent Systems and Computing 425,
DOI: 10.1007/978-3-319-28658-7_57

Linux kernel operating system and its mobile applications are written in an eclipse of Java language run with built-in Application Programming Interfaces (API'S) [34]. Android uses a security framework that consists of application sandboxing, secure inter-application communication, cryptographic API's, application signing [8]. But these countermeasures for security against vulnerability may not be effective. Even though we have existing malware detection mechanisms, they are failed to eliminate the android mobile threats totally. These android malware threats change with a timeline, some common examples of malware threats found in android devices in 2014 are Torec, DroidPack, DriveGenie, OldBoot [7]. Android intents and permissions framework for security mechanism provides a guard between software and hardware resources. Intents in android is a communication model for launching the activities and services, but we should take care about the applications exported and services exported [4, 21]. Intent spoofing attack is the most common found in android, it leads to the broadcast receivers, exported applications and exported services [20], we examine the mobile applications not only at android application level, but also we examine the android applications at network level and device level. Many security threats have been found including confidential information shared unauthorized parties because of poor SSL (Secure Socket Layer) encryption, and insufficient transport layer protection, improper session handling these threats are well described in OWASP (or Open Web Application Security Project) [16]. And other unexpected behavior.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 considers threat scenario and vulnerability analysis for mobile banking apps. Section 4 describes the proposed testing strategy. Finally, Section 5 concludes the paper.

2 Related Work

Mobile banking applications based on Android, iOS, Windows platforms, have been tested by others in the last few years. Chakraborti et al. [3] proposed a security framework for mobile apps for any enterprise. This paper provides the only literature review on possible threats and vulnerabilities. Marforio et al. [18] described security indicators for detecting threats against phishing attacks in mobile platforms and possible countermeasures. This proposed framework describes application phishing, web phishing but doesn't deal with vulnerabilities in the code or app level, and communication level. Kathuria et al. [15] deals with the challenges in android application development. This proposed app mainly focuses on the user-centric level, without giving details about mobile security testing. Felt et al. [11] clearly describes android permissions and applied some automated testing techniques to android version 2.2 for determining the maximum permissions that are needed for an application and compares those permissions with actually required permissions. Likewise, he examined 940 Android apps using the tool of Stowaway and detected that 1/3rd of them are over privileged. He et al. [12] analyzes forty-seven: Android, iOS mobile apps in survey regarding SD cards, logging, Bluetooth, content provider, usage of cloud services, The internet.

Related work closest to our mobile banking applications from [6, 10, 13, 17]. Hu et al. [13] propose a set of six criteria for identifying and evaluating killer apps for mobile payment, banking, emerging mobile commerce applications and services. This paper doesn't achieve certain challenges of privacy and security concerns in mobile banking, and incompatible of mobile communication. Fahl et al. [10] examine various most popular free apps and investigation of the current state of SSL/TLS (Secure Socket Layer/Transport Layer Security) for android, he used a tool MalloDroid that detect potential threats against MITM (Man-In-The-Middle) attacks. Lee at el. [17] describes the complete literature review on the investigation of features and security in mobile banking. This framework of design helps and recommends for security and privacy issues involved in current mobile banking services. However, this framework doesn't analyze the real time scenarios possible in mobile banking services. Delac et al. [6] develop an attacker-centric model for different mobile platforms such as Android and iOS. The designed threat model addresses 3 key features of mobile device security, 1. Goals of attacker's, 2. Attack vectors, 3. Mobile malware.

Apart from other studies, we focus on examining several mobile banking Android applications by static code analysis and dynamic analysis using the tools of ApkAnalyser [22], Mercury [23] or (Drozer) [24] for static analysis, Wireshark [5, 27], Burp Suite [28] for dynamic analysis and found 356 exploitable vulnerabilities. Our testing approach moves further than previous related work: because others have tested with a dynamic testing strategy with one tool, but we focus on code or app level, communication level, and device-level testing. This makes a novel study for security in mobile banking applications and it is helpful for code developers for further enhancing the security measurements.

3 Threat Scenario and Vulnerability Analysis for Mobile Banking Apps

Before testing the mobile banking app's, initially we define the context. Mobile banking apps are more securable, user-friendly, and immediate mobile payment system.

3.1 Threat Scenario

A complete modeling of threat and vulnerability analysis is beyond the scope of our effort, but we suggest some points that bring out a neat framework for secure mobile banking. We concerned on apps for code level and network level, device level, because they are most important for the banker side and at customer's point of view.

- **Untrusted party learning of bank data:** Unauthorized persons gain the bank information belonging to an individual customer. They access not only secure data but also monitor the network.
- **Tampering with bank data:** An adversary tamper or alters the bank data, by performs replay and man-in-the-middle attacks in the communication media or network.
- **Customer chooses wrong bank app** Here the end user or customer chooses a wrong app, that app is installed in his/her mobile phone and giving valid credentials to the untrusted bank. Then the adversary plays all attacks on to the original bank and end user. This type of threat is called as a phishing attack.

The above three types of threats represent the violation of the security features such as data integrity, confidentiality, authorization, authentication, and non-repudiation.

3.2 *Identifying the Attack Surface and Analysis of Vulnerabilities*

Figure 1 shows a typical threat scenario for mobile banking apps, initially we install mobile banking apps in a mobile device (smartphone). The smartphone stores the bank apps data internally in a file system, database backups.

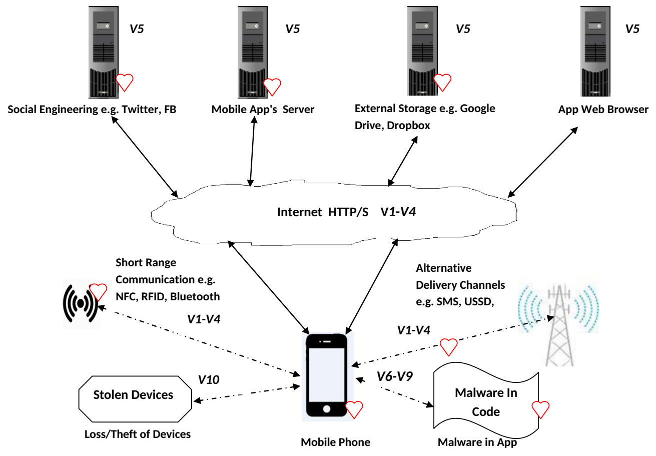

V5
V5
V5
V5
Social Engineering e.g. Twitter, FB
Mobile App's Server
External Storage e.g. Google Drive, Dropbox
App Web Browser
Internet HTTP/S V1-V4
Short Range Communication e.g. NFC, RFID, Bluetooth
V1-V4
Alternative Delivery Channels e.g. SMS, USSD,
V1-V4
V10
Stolen Devices
Loss/Theft of Devices
V6-V9
Malware In Code
Malware in App
Mobile Phone

Fig. 1 A Typical Threat Scenario for mobile banking app in context. The heart represents fine-grained banking data

In Figure 1 shows several types of individual servers that are connected to the network. Some apps connect to the social engineering server such as Twitter, Facebook (FB) etc. Many fake mobile banking apps are connected directly to the payment gateway to a dedicated app web server for allowing the transactions of upload valid credentials to the bank server, backup, and data synchronization. The app's connect to external storage services e.g.Google Drive, Dropbox. Finally, app web browser is used to search the rate and amount payable for apps. This framework of threat scenario helps to identify the attack surface for various threats and suggests where attacker look for loopholes existing in the network. The mobile risks or vulnerabilities (denoted as V1-V10) are:

- **V1. Masquerade:** This attack is caused because of not establishing a secure connection between client and server, and improper SSL/TLS i.e., poor transport layer protection. Because of this reason the adversary performs session hijacking, and stealing session ID's.
- **V2. Man-In-The-Middle:** He is able to tap the secure data, manipulate, integrate with own data on a network and send to victims.
- **V3. Replay:** Attacker sends subsequent retransmission or delayed messages in the network.
- **V4. Traffic Analysis, Wi-Fi Sniffing:** The attackers always monitor the network traffic and observe packets are in encrypted format or not, sometimes they capture the packets.
- **V5. Browser Exploits:** It is also called as SSL BEAST (Browser Exploit Against SSL/TLS), he will be able to leverage weaknesses implemented in cipher block chaining to exploit the SSL protocol. Based on this reason the attacker easily read the encryption form data in plain text.
- **V6. SQL Injection:** The attacker identifies weak tables in the database possibly by SQLite can be subject to inject in web applications.
- **V7. Lack of binary protection:** Typically, an attacker will analyze and apply some reverse engineering tools for getting the original source code, then modify that code and perform hidden functionality.
- **V8. Broken Cryptography:** The code contains weak cryptography algorithms, easily breakable passwords, and poor key management process leads adversary easily steal the algorithms and theft secure data.
- **V9. Data Leakage:** Agents easily exploit the vulnerabilities by malware exist in the mobile of legitimate apps, or physical access of the device by an adversary from the victim's mobile device.
- **V10. Insecure Data Storage:** An Attacker that has attained a theft of mobile device, malware or another re-modified app acting on the attacker's behalf that executes on the smartphone.

Other possible attacks such as DoS (Denial of Service), phishing, pharming, protocol attacks, etc.) are currently not included.

4 Testing Strategy

Our testing mechanisms is divided into 4 parts: i. Static analysis ii. Dynamic analysis iii. Web app server security, iv. Device forensic. Apart from the four mechanisms of testing, we consider three levels of security testing 1. App Level, 2. Communication Level, and 3. Device Level. Table 1 shows how our testing three levels addresses four mechanisms. The possible vulnerabilities from V1-V10 represented in the previous section are shown in a table. For each vulnerability, the countermeasures or precautions to be taken by the developer, end-user, and banker is shown in Table 1.

- **Application Level or App Level:** This level identifies the vulnerabilities in code, for android applications all the files are .apk only, so we put this .apk file on ADB (Android Debugging Bridge) and possible vulnerabilities identified at attack surface. For this level, we consider both static and dynamic analysis testing mechanism.
- **Network or Communication Level:** An attacker captures or alters the packets in the network. So in this level we used dynamic analysis.
- **Device Level:** Sometimes the devices e.g. mobile device, SD card, personal log files stolen by an adversary, compromised devices, and physical threats comes under this category.

Table 1 Strategy for Security Testing of Mbank Apps

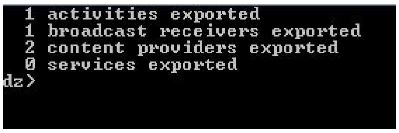
Testing Levels	Testing Mechanism	Supported Tools	Vulnerabilities	Countermeasures
App Level	Static and Dynamic	Drozer ApkAnalyser virustotal	V6-V9	Check Uniform Resource Locator (URL) Caching for both HTTP request and response, Application backgrounding, Use strong cryptography algorithms with appropriate key lengths.
Communication Level	Dynamic Analysis Web app server security	Drozer WireShark Burp Suite TCPDUMP	V1-V5	Apply SSL/TLS for transport channels. Use digital certificates signed by a trusted Certificate Authority (CA) provider. Do not send confidential data over alternate delivery channels (e.g, SMS, MMS, or notifications).
Device Level	Device Forensic	Sleuth Kit	V10	Never store personal credentials on the SD card, For databases concern SQLcipher for SQLite data encryption mechanism.

4.1 Static Analysis

Static analysis does not involve opening a file, or running the code or reverse engineering it is based on data contained in the APK (Android application package file). Static analysis involves identifying and querying cryptographic hash values, such as MD5 (Message Digest), metadata, strings, extract the apps permissions. The tools virustotal [30], androguard [25] is used for this analysis.

- **Antivirus Scans and Aliases:** Antivirus scans and aliases help to analyze a threat, identify date and time, comments, votes, and a list of aliases. Aliases acknowledge that stamp are other common related names attributed to the same code. Antivirus scan results, samples, blogs, and so on. In our test of various apps, the results are shown in Figure 3 and Figure 4 respectively. In Figure 4 reveals one Trojan horse detected out of 54 engines.

- **Broadcast Receivers:** Attackers often hold needed data about an apps attack surface and offer attackers to perform many wrong things, from arbitrary code execution to proliferating information. Broadcast receivers respond for both software and hardware-level events. They get notifications for these events through intents. The Drozer [24] tool identifies these receivers at the attack surface.
- **Activities and Services** These are the application components, and services that facilitate user interaction. It is useful for identifying which application, services can be released without permissions during an application security assessment because any of them provide access to confidential data or cause an app to crash if launched in the wrong context.
- **Content Providers (V6, V8, V9) :** In any database architecture, content providers have the ability to perform malicious operations into their SQLite databases or any file stores. They identify weak URI (Uniform Resource Identifier) in the database and perform all Structure Query Language (SQL) operations. For this content providers, we use the same tool drozer. The test app results on the attack surface of activities, broadcast receivers, content providers, services exported is shown in the Figure 2.
- **Certificate Information (V1-V5):** All apps must be signed because android uses x.509 certification otherwise apps will not install. And tester verifies the given certificates valid by the CA (Certification Authority), time period, and serial no. To verify these certificate procedures we use [29].



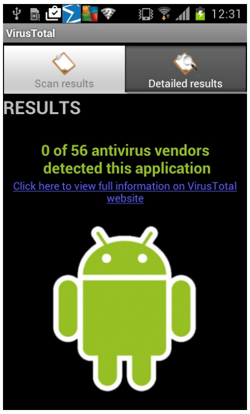
```
  1 activities exported
  1 broadcast receivers exported
  2 content providers exported
  0 services exported
dz>
```

Fig. 2 An attack surface contains vulnerabilities in the app

4.2 *Dynamic Analysis*

Dynamic or behavioral analysis is mostly manual testing. In this analysis we create a dummy account for testing the credentials of user Account no, MPIN (Mobile PIN), user ID, Password, OTP (One Time Password) all these are invalid. From this account details, we can perform transactions of transferring an amount from one account to another account.

- **Insufficient Transport Layer Protection (V1-V5, V9) :** When designing any application, the information is exchanged in a client and server environment. To transmit data, it must traverse the device carrier or network devices (intermediate

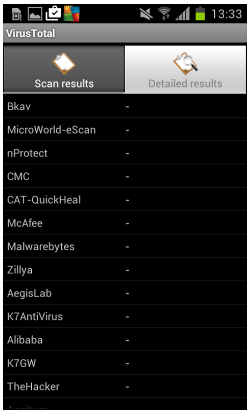


RESULTS

0 of 56 antivirus vendors detected this application

Click here to view full information on VirusTotal website

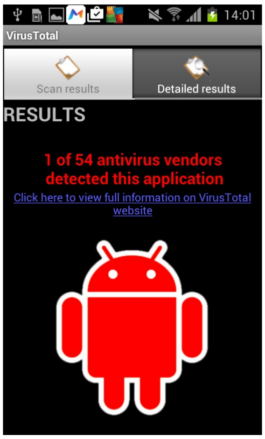
(a) Scan Results



Bkav	-
MicroWorld-eScan	-
nProtect	-
CMC	-
CAT-QuickHeal	-
McAfee	-
Malwarebytes	-
Zillya	-
AegisLab	-
K7AntiVirus	-
Alibaba	-
K7GW	-
TheHacker	-

(b) Detailed Results

Fig. 3 Virus total scan without virus results and detailed tesults

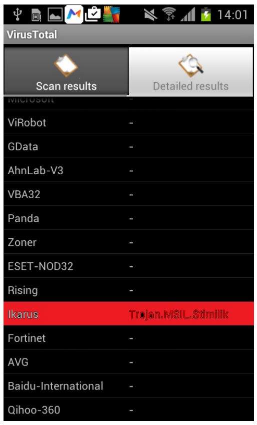


RESULTS

1 of 54 antivirus vendors detected this application

Click here to view full information on VirusTotal website

(a) Scan Results



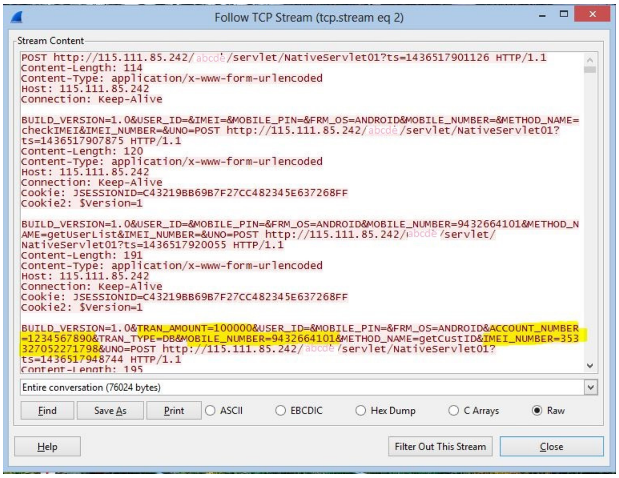
ViRobot	-
GData	-
AhnLab-V3	-
VBA32	-
Panda	-
Zoner	-
ESET-NOD32	-
Rising	-
Ikarus	Trojan.MSIL.Stimilik
Fortinet	-
AVG	-
Baidu-International	-
Qihoo-360	-

(b) Detailed Results

Fig. 4 Virus total scan with virus results and detailed results

routers or nodes, proxy's, cell towers etc.). Threat agents might exploit vulnerabilities such as monitor Wi-Fi (Wi-Fi Sniffing), compromised network, capture sensitive data or malware in mobile etc. These threats are because of insufficient transport layer protection. So we test this by Wireshark [27], tPacketCapture [31], Burp Suite [28]. The results are shown in Figure 5, and Figure 6 respectively. These two figures shows that the highlighted with yellow color data is not in encrypted format, all the sensitive information is represented as a plain text.

- **Secure backup directories, files, and logging:** Here we test backup directories, files or log information contained encrypted data or not using the tool of adb [26].



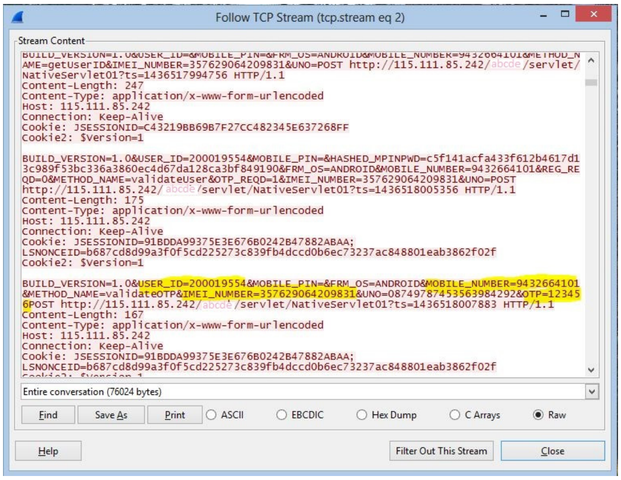
```
POST http://115.111.85.242/abcde/servlet/NativeServlet01?ts=1436517901126 HTTP/1.1
Content-Length: 114
Content-Type: application/x-www-form-urlencoded
Host: 115.111.85.242
Connection: Keep-Alive

BUILD_VERSION=1.0&USER_ID=&IMEI=&MOBILE_PIN=&FRM_OS=ANDROID&MOBILE_NUMBER=&METHOD_NAME=
checkIMEI&IMEI_NUMBER=&UNO=POST http://115.111.85.242/abcde/servlet/NativeServlet01?
ts=1436517907875 HTTP/1.1
Content-Length: 120
Content-Type: application/x-www-form-urlencoded
Host: 115.111.85.242
Connection: Keep-Alive
Cookie: JSESSIONID=C43219BB69B7F27CC482345E637268FF
Cookie2: $Version=1

BUILD_VERSION=1.0&USER_ID=&MOBILE_PIN=&FRM_OS=ANDROID&MOBILE_NUMBER=9432664101&METHOD_N
AME=getUserList&IMEI_NUMBER=&UNO=POST http://115.111.85.242/abcde/servlet/
NativeServlet01?ts=1436517920055 HTTP/1.1
Content-Length: 191
Content-Type: application/x-www-form-urlencoded
Host: 115.111.85.242
Connection: Keep-Alive
Cookie: JSESSIONID=C43219BB69B7F27CC482345E637268FF
Cookie2: $Version=1

BUILD_VERSION=1.0&TRAN_AMOUNT=100000&USER_ID=&MOBILE_PIN=&FRM_OS=ANDROID&ACCOUNT_NUMBER
=1234567890&TRAN_TYPE=DB&MOBILE_NUMBER=9432664101&METHOD_NAME=getCustID&IMEI_NUMBER=353
327052271798&UNO=POST http://115.111.85.242/abcde/servlet/NativeServlet01?
ts=1436517948744 HTTP/1.1
Content-Length: 195
```

Fig. 5 TCP Stream in unencrypted format: Amount, Mobile no, Account no



```
BUILD_VERSION=1.0&USER_ID=&MOBILE_PIN=&FRM_OS=ANDROID&MOBILE_NUMBER=9432664101&METHOD_N
AME=getUserID&IMEI_NUMBER=357629064209831&UNO=POST http://115.111.85.242/abcde/servlet/
NativeServlet01?ts=1436517994756 HTTP/1.1
Content-Length: 247
Content-Type: application/x-www-form-urlencoded
Host: 115.111.85.242
Connection: Keep-Alive
Cookie: JSESSIONID=C43219BB69B7F27CC482345E637268FF
Cookie2: $Version=1

BUILD_VERSION=1.0&USER_ID=200019554&MOBILE_PIN=&HASHED_MPINPWD=c5f141acfa433f612b4617d1
3c989f53bc336a3860ec4d67da128ca3bf849190&FRM_OS=ANDROID&MOBILE_NUMBER=9432664101&REG_RE
QD=0&METHOD_NAME=validateUser&OTP_REQD=1&IMEI_NUMBER=357629064209831&UNO=POST
http://115.111.85.242/abcde/servlet/NativeServlet01?ts=1436518005356 HTTP/1.1
Content-Length: 175
Content-Type: application/x-www-form-urlencoded
Host: 115.111.85.242
Connection: Keep-Alive
Cookie: JSESSIONID=91BDDA99375E3E676B0242B47882ABAA;
LSNONCEID=b687cd8d99a3f0f5cd225273c839fb4dccd0b6ec73237ac848801eab3862f02f
Cookie2: $Version=1

BUILD_VERSION=1.0&USER_ID=200019554&MOBILE_PIN=&FRM_OS=ANDROID&MOBILE_NUMBER=9432664101
&METHOD_NAME=validateOTP&IMEI_NUMBER=357629064209831&UNO=087497874535639842928OTP=12345
6POST http://115.111.85.242/abcde/servlet/NativeServlet01?ts=1436518007883 HTTP/1.1
Content-Length: 167
Content-Type: application/x-www-form-urlencoded
Host: 115.111.85.242
Connection: Keep-Alive
Cookie: JSESSIONID=91BDDA99375E3E676B0242B47882ABAA;
LSNONCEID=b687cd8d99a3f0f5cd225273c839fb4dccd0b6ec73237ac848801eab3862f02f
Cookie2: $Version=1
```

Fig. 6 TCP Stream in unencrypted format: OTP

4.3 *Web App Server Connection*

In this testing mechanism apps are directly connected to dedicated web server to backup files or synchronize bank data. For this analysis again we used a dummy account of customer on the web site.

- **Web server connection (V1-V4):** Normal HTTP connection allows an adversary easily read the customer sensitive data; by history of URLs (Uniform Resource Locater) check the connection with secure transport of (HTTPS:)
- **Web server authentication and authorization (V1-V2):** Further we investigate the effectiveness of the server, e.g. an attacker understands how the authentication procedure is vulnerable, they create bypass or fake authentication by submitting transaction service requests to the mobile application backend server and bypass any direct interaction with the mobile application. This submission process is typically done via mobile malware within the device or botnets owned by the attacker. For this analysis we use again, Wireshark [27], and tPacketCapture [31].

4.4 Device Forensic

Using this testing mechanism, suggests a complete investigation of file structures of the android operating system, services, data storage, and external devices [32]. For complete forensic analysis, no tool extracts all the possible information when the device is locked [33]. We conduct forensic analysis on mobile devices using the sleuth kit [2].

5 Case Study

Case study section deals with an overview of our testing for android mobile banking apps. For testing the bank apps we need a .apk files. And these apps which are directly available from the Google's Play Store.The mobile devices for testing on two mobile platforms, android and J2ME.

5.1 Test Bed and Other Tools Used

For Android Environment

- Mobile Phone Used: HTC Desire X (1.2 GHz Dual Core, 768 MB RAM)
- Samsung Galaxy wave- II (1GHz Dual Core, 1 GB RAM)
- Android version "KitKat"
- Eclipse ADT IDE integrated with Android SDK
- Drozer Agent [24], VirusTotal, ApkAnalyser, androguard
- Wireshark, Burp Suite, TCPDUMP, tpacketcapture, Sleuth Kit

We connect a mobile device with the laptop running Ubuntu operating system version 12.04. Install above static tools on mobile device and for dynamic testing, tools installed on the laptop.

';

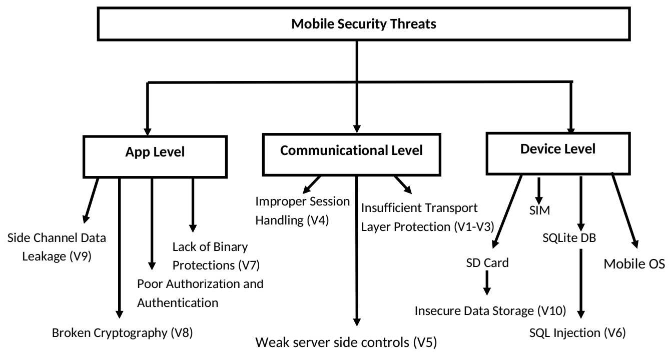

Mobile Security Threats
App Level
Communicational Level
Device Level
Side Channel Data Leakage (V9)
Lack of Binary Protections (V7)
Poor Authorization and Authentication
Broken Cryptography (V8)
Improper Session Handling (V4)
Insufficient Transport Layer Protection (V1-V3)
Weak server side controls (V5)
SIM
SQLite DB
SD Card
Mobile OS
Insecure Data Storage (V10)
SQL Injection (V6)

Fig. 7 Coverage vulnerabilities with OWASP

5.2 *App Preparation*

We tested several financial institute apps like banking and other apps. For static analysis testing, verify the vulnerabilities at attack surface but for dynamic analysis test banks create dummy accounts with some amount. These dummy account credentials are same for the original account, in the sense if customer account no is 9 digits, then dummy account no is also nine digits. Other valid information for doing transactions such as OTP (One Time Password), login id, password, and an amount in payer's account.

5.3 *Discussion*

The verification and validation of our testing strategy are demonstrated by identifying several cryptographic parameters, and privacy issues. Our testing strategy is compared with OWASP (Open Web Application Security Project) [19] and it covers entirely or partly 6 of 8 static and 4 of the 7 dynamic categories. This cross compare testing mechanism is shown in Figure 7.

6 Conclusions and Future Work

Our security testing strategy for mobile banking applications gives clear evidence of cryptography features such as authentication, data integrity, confidentiality, non-repudiation. We test and analyze many apps and found 356 exploitable vulnerabilities. This paper is helpful for those who are in the android app development and android malware threat analysis. The contribution of our work suggests, the app developer should concern the cryptography features, then implement the app in a secure environment. Our work shows that automated security testing for mobile banking applications is expensive, because the large amount of manual work is needed. Some tools for automated security testing cover only a minimum number of vulnerabilities, and every year the new versions of device OS launch into the market. So, many updations are needed and fully automated testing is desirable.

As future work, we will focus on testing mobile banking apps in the environment of iOS, Windows. Enhancing the security and safety parameters for mobile banking apps will be needed to concern the fully automated spectrum of issues.

References

1. Blasco, J.: Introduction to android malware analysis (2012)
2. Carrier, B.: The sleuth kit (TSK) (2010). http://www.sleuthkit.org/sleuthkit/
3. Chakraborti, S., Acharjya, D., Sanyal, S.: Application security framework for mobile app development in enterprise setup (2015). arXiv preprint arXiv:1503.05992
4. Chin, E., Felt, A.P., Greenwood, K., Wagner, D.: Analyzing inter-application communication in android. In: Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, pp. 239–252. ACM (2011)
5. Combs, G.: Wireshark: Go deep (2009). homepage for wireshark
6. Delac, G., Silic, M., Krolo, J.: Emerging security threats for mobile platforms. In: MIPRO, 2011 Proceedings of the 34th International Convention, pp. 1468–1473. IEEE (2011)
7. Dunham, K., Hartman, S., Quintans, M., Morales, J.A., Strazzere, T.: Android Malware and Analysis. CRC Press (2014)
8. Enck, W., Octeau, D., McDaniel, P., Chaudhuri, S.: A study of android application security. In: USENIX Security Symposium, vol. 2, p. 2 (2011)
9. Enck, W., Ongtang, M., McDaniel, P.: Understanding android security. IEEE Security & Privacy **1**, 50–57 (2009)
10. Fahl, S., Harbach, M., Muders, T., Baumgärtner, L., Freisleben, B., Smith, M.: Why eve and mallory love android: an analysis of android ssl (in) security. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 50–61. ACM (2012)
11. Felt, A.P., Chin, E., Hanna, S., Song, D., Wagner, D.: Android permissions demystified. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 627–638. ACM (2011)
12. He, D.: Security threats to android apps. Ph.D. thesis, Masters thesis, University of Illinois at Urbana-Champaign (2014)
13. Hu, X., Li, W., Hu, Q.: Are mobile payment and banking the killer apps for mobile commerce? In: Proceedings of the 41st Annual Hawaii International Conference on System Sciences, pp. 84–84. IEEE (2008)
14. Hunt, R.: Security testing in android networks-a practical case study. In: 2013 19th IEEE International Conference on Networks (ICON), pp. 1–6. IEEE (2013)
15. Kathuria, A., Gupta, A.: Challenges in android application development: A case study (2015)
16. King, J.: Android application security with owasp mobile top 10 2014. Ph.D. thesis, Masters thesis, Luleå University of Technology (2014)
17. Lee, H., Zhang, Y., Chen, K.L.: An investigation of features and security in mobile banking strategy. Journal of International Technology and Information Management **22**(4), 2 (2013)
18. Marforio, C., Masti, R.J., Soriente, C., Kostiainen, K., Capkun, S.: Personalized security indicators to detect application phishing attacks in mobile platforms (2015). arXiv preprint arXiv:1502.06824

19. Mobile Security Testing Guide: https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=M-Security_Testing//
20. Nauman, M., Khan, S., Zhang, X.: Apex: extending android permission model and enforcement with user-defined runtime constraints. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, pp. 328–332. ACM (2010)
21. Ongtang, M., McLaughlin, S., Enck, W., McDaniel, P.: Semantically rich application-centric security in android. Security and Communication Networks **5**(6), 658–673 (2012)
22. https://github.com/sonyxperiadev/ApkAnalyser
23. https://www.labs.mwrinfosecurity.com/tools/2012/03/16/mercury
24. https://www.mwrinfosecurity.com/products/drozer/
25. https://code.google.com/p/androguard//
26. https://developer.android.com/tools/help/adb.html
27. https://www.wireshark.org (accessed February 20, 2015)
28. https://portswigger.net/burp/ (accessed February 20, 2015)
29. https://www.opnessl.org/ (accessed March 11, 2015)
30. https://www.virustotal.com/ (accessed May 10, 2015)
31. https://play.google.com/store/apps/details?id=jp.co.taosoftware.android.packetcapture (accessed May 10, 2015)
32. Walnycky, D., Baggili, I., Marrington, A., Moore, J., Breitinger, F.: Network and device forensic analysis of android social-messaging applications. Digital Investigation **14**, S77–S84 (2015)
33. Wang, Y., Alshboul, Y.: Mobile security testing approaches and challenges. In: 2015 First Conference on Mobile and Secure Services (MOBISECSERV), pp. 1–5. IEEE (2015)
34. Wei, X., Gomez, L., Neamtiu, I., Faloutsos, M.: Permission evolution in the android ecosystem. In: Proceedings of the 28th Annual Computer Security Applications Conference, pp. 31–40. ACM (2012)