MOBILE BANKING APPS ARE NOT CREATED EQUAL

ZIMPERIUM®

Executive Summary

In JPMorgan Chase's 2018 annual report, CEO Jamie Dimon states in a widely read letter to shareholders, "The threat of cybersecurity may very well be the biggest threat to the U.S. financial system." In April 2019, five of seven CEOs from the largest U.S. banks testified during House Financial Services Committee and claimed cybersecurity as the largest risk to our financial system. As banks encourage us to move our money to mobile and cashless formats for convenience and speed, so does the risk of cybercrime. Today, cybercriminals target 63 percent of the banks in our study using specific malware campaigns to trick mobile users into surrendering their money and banking credentials.

Financial regulators like the US Federal Reserve acknowledge mobile device risks as well. As our banking practices migrate to mobile and cloud, so do the concerns about security and access our devices have. Regulators warn that mobile banking may help to address some challenges consumers face, but the mobile banking channel needs to become more trustworthy.

> "A well-designed and secure mobile platform, as well as consumer access to and facility with mobile technology and the Internet, are likely needed in order for mobile banking to be a reliable banking channel." US Federal Reserve

Executive Summary

Are banks doing enough?

This research details how both iOS and Android banking applications from the top 45 US banks and mobile payment providers fared for security and privacy. The results derive from a mobile application reputation scanning service offered to Zimperium licensed customers. Zimperium is providing the anonymous results on the mobile app risks to the banks, analysts, users and the market under responsible disclosure. If you are a banking leader responsible for the mobile or digital channels, Zimperium will assist you in identifying your app in the report.

Some of the quick facts to follow in this research include:

- Despite banks increasingly encouraging customers to use mobile banking apps and acknowledging cybersecurity as the biggest threat to the financial system, banks fail to adhere to coding best practices. Not adhering to application development best practices exposes both customer and bank data and increases the chances for fraud as banks implement more third-party mobile and cloud services.
- Banks continue to use shared code in production apps that is infrequently updated or retired. If shared code is no longer supported or is vulnerable, all apps containing this code are impacted after an incident occurs. Furthermore shared code means that anyone (especially open source code) has the opportunity to review and probe the code for vulnerabilities and weaknesses to identify the attack surface and exploit it.
- Banks continue to share sensitive customer data with advertisers. This practice increases the chances of data leakage if a mobile banking app is reverse engineered or a third-party library or service vulnerability is exploited. If there is a data breach and personally identifiable information is exposed, banks suffer severe brand damage and regulatory fines.
- Most banks fail to obfuscate code, secure mobile device data, or implement runtime application self-protection for mobile apps. Failing to obfuscate code allows anyone to reverse engineer an app to identify weaknesses and identify an attack surface. One such example is how cybercriminals reverse-engineered a mobile app to identify vulnerabilities and stole millions of dollars overnight from thousands of users all without being detected.

Table of Contents

Methodology

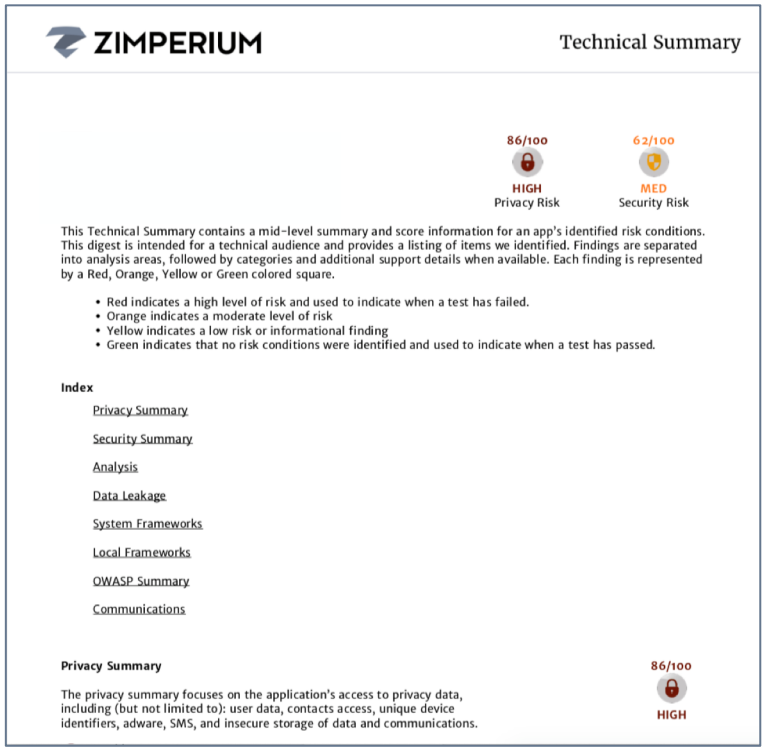
This research discloses the security and privacy risks exposure and whether or not each passes or fails the Open Web Application Security Project (OWASP) Mobile Top 10 practices for the top 48 US mobile banking and mobile payment apps.

The scores are calculated using Zimperium's z3A Advanced Application Analysis engine. Zimperium z3A is an application reputation scanning service that continually evaluates risks posed by mobile apps.

z3A provides deep intelligence about app behavior, including content (the app code itself), intent (the app's behavior), and context (the domains, certificates, shared code, network communications, and other data). z3A also provides privacy and security ratings, enabling enterprises to create security policies limit or remove risky apps from managed devices.

This research scanned and scored 90 mobile banking apps available in the Apple App Store and Google Play in April 2019 for security, privacy, and data leakage risks. Each of the banks in this research have greater than $50B in assets and are headquartered in the US.

The security summary focuses on application risks. These risks include functionality and code use, application capabilities, and critical vulnerabilities.


ZIMPERIUM

Technical Summary

86/100
HIGH
Privacy Risk

62/100
MED
Security Risk

This Technical Summary contains a mid-level summary and score information for an app's identified risk conditions. This digest is intended for a technical audience and provides a listing of items we identified. Findings are separated into analysis areas, followed by categories and additional support details when available. Each finding is represented by a Red, Orange, Yellow or Green colored square.

- Red indicates a high level of risk and used to indicate when a test has failed.
- Orange indicates a moderate level of risk
- Yellow indicates a low risk or informational finding
- Green indicates that no risk conditions were identified and used to indicate when a test has passed.

Index

Privacy Summary
Security Summary
Analysis
Data Leakage
System Frameworks
Local Frameworks
OWASP Summary
Communications

Privacy Summary

86/100
HIGH

The privacy summary focuses on the application's access to privacy data, including (but not limited to): user data, contacts access, unique device identifiers, adware, SMS, and insecure storage of data and communications.

The privacy information focuses on the application's access to private user data, unique device identifiers, SMS, communications, and unsecure data storage.

The security and privacy scores are on a 0-100 scale. Higher aggregate scores indicate apps that contain many privacy and security risk conditions.

The OWASP summary contains testing results performed on the application against the OWASP Top 10 Mobile categories. Sections receiving a passing mark are represented in green while sections failing a test are represented in red.

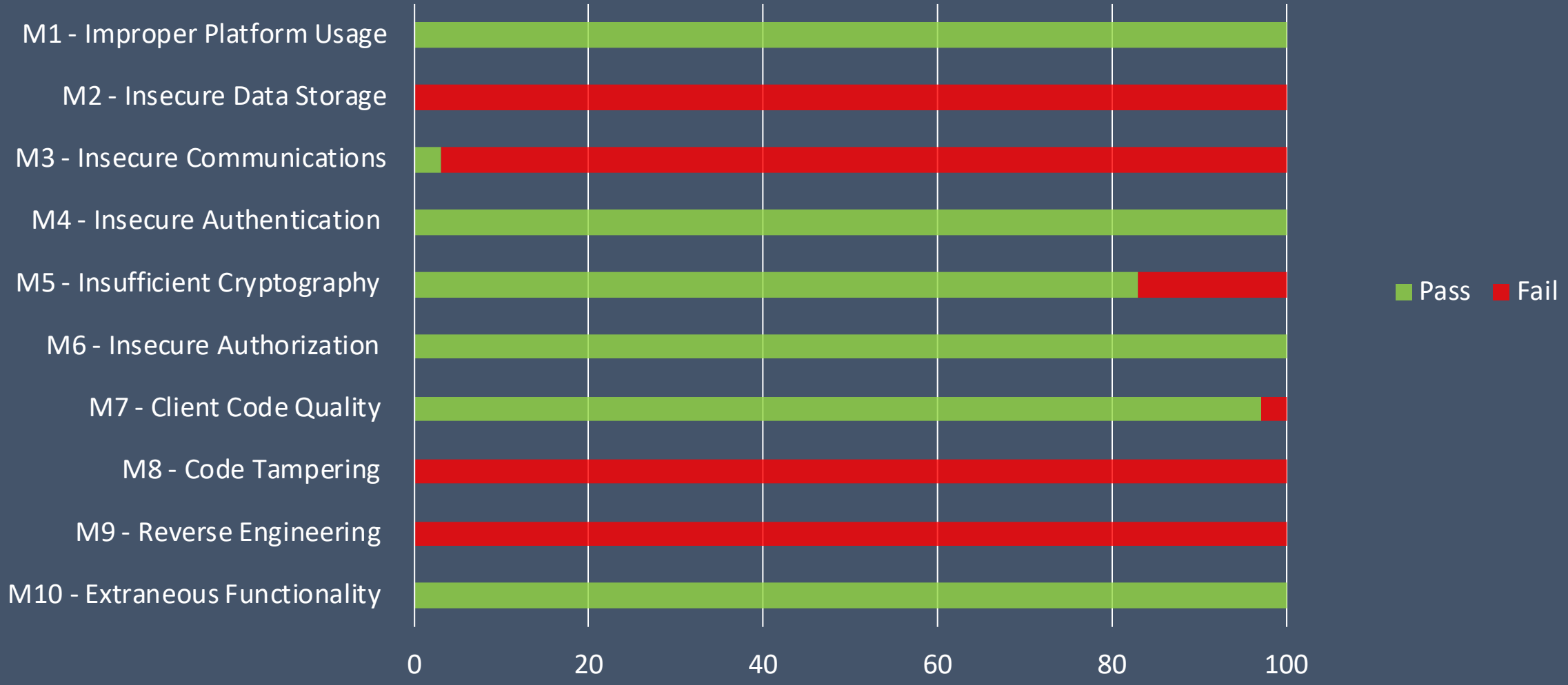
OWASP Mobile Top 10 Results

OWASP is an open community dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted. OWASP publishes a top 10 list of application development best practices applying to web applications and a set applying to mobile apps.

Part of our research into the mobile banking applications includes providing a passing or failing mark for each of the OWASP Mobile Top 10. The tables below summarize passing and failing marks collectively for all of the apps on each platform.

Over 90 percent of the apps fail exposure to reverse engineering. Someone could reverse engineer the apps to identify exploitable vulnerabilities to steal from customers or defraud banks. Nearly three of four apps use insecure data storage and potentially leak user data. More than half of all apps use insecure communication.

OWASP Mobile Top 10 Results



ZIMPERIUM

OWASP Mobile Top 10 Results

ZIMPERIUM

Security Risks

The average security score using Zimperium's z3A Advanced Application Analysis for the 48 iOS mobile banking apps is 41.

The app with the best score, scored a 9. This app consequently scored well for privacy at 30. This app is in the lower third in size of all of the iOS apps tested.

The highest score among the iOS apps we scanned, scored 64. The same app scored a 59 for the overall privacy score placing it near the average amongst all of the apps in our sample. The bank distributing this app has several million mobile users, according to official company statements.

Some of the common security issues we found during our iOS analysis are alarming. Several apps fail to secure user data, contain HTTP communication links, allow for over the air installation, and monitor users' pasteboard.

Security Risks - iOS

- **95% of apps are using NSURLAuthenticationMethodServerTrust**
  - The 'NSURLAuthenticationMethodServerTrust' can apply to any protocol but is most commonly used for overriding SSL and TLS chain validation.
- **89% of apps using Swizzling API calls**
  - These apps implement Swizzling API calls. The Swizzling API calls may impact the app's ability to trust security decisions based on untrusted inputs or manipulated/swizzled output.
- **16% of apps allow arbitrary loads**
  - These apps are configured to allow unsecure and unverified connection to servers with lower TLS versions. It will enable cipher suites that do not support forward secrecy and does not discriminate between HTTP or HTTPS connections.
- **2 apps implement an over-the-air app installation**
  - The app implements an over-the air app installation method. This implementation circumvents Apple's review process and could be utilized to install functionality not approved by Apple.
- **2 apps are monitoring the iOS pasteboard**
  - Monitoring the iOS Pasteboard is available to apps; however, it is best to limit this in mobile banking applications since apps can capture passwords or SMS security codes used in user authentication.

Security Risks - iOS

- **4 apps send query parameters with user names, passwords, and device information**
  - Sending personally identifiable information from the device increases the chances for data leakage. When passing parameters via the GET parameter (URL parameters), the data is exposed by any network packet inspection regardless if they are using HTTP or HTTPS. Using a 'POST' statement would conceal/encrypt that data when using HTTPS.
- **68% of apps implement a pin-point location functionality from the CoreLocation framework**
  - Apple only allows pin-point location functionality in navigation apps. Apple may consider Pin-point location geolocation abuse if implemented in non-navigation apps. However, a valid use for pin-point location in a mobile banking app is to locate a branch or ATM location. The recommended CoreLocation call for non-navigation apps is to use 10-meter accuracy.
- **2 apps implement a low-level IOKit device enumeration API call**
  - Implementing a low-level IOKit device enumeration API call could indicate a malicious intent to obtain data normally not available. The developer should consider using an approved Apple method to avoid this risk condition. Uber famously abused IOKit to read private data from users that had once installed and removed the app.

Security Risks - Android

The security risk scores for Android apps at the same banks are 48 percent higher. The average security risk score among the Android mobile banking apps is 61, while the app with the lowest score scored 11. This app's privacy score is also exemplary by scoring 5. It is one of the smaller apps with fewer features than most. Interestingly, the bank producing this Android app scored second best with its iOS counterpart.

Common security issues we found during our Android analysis are very similar to the banks' iOS apps. Several apps are using unsecure data storage, components not protected by a permission, WebView, and using WebKit to download information from the internet.

- **60 % of apps contain exported components not protected by a permission**
  - If a component such as a service is exported and not protected with strong permissions, then any application can start and bind to the service. Depending on the duties of a particular service, it may leak information or perform unauthorized tasks.
- **40% of apps enables WebView to execute JavaScript code**
  - Mobile apps should avoid Javascript when possible. That said, if the app is connecting to the bank's website to show pages in the app, and the bank's website uses javascript, then the app also has to be configured to 'allow' javascript. However, it is best not to use Javascript. The banks should use APIs to pull in required data without javascript as an alternative.
- **29% of apps implement Java runtime with exclusions**
  - These applications can execute commands at the OS level, such as launching other applications and processes.
- **3 apps incorrectly store data**
  - These apps use insecure storage data mode (WORLD_READABLE, WORLD_WRITABLE).
- **1 app does not check the validation of the CN(Common Name)of the SSL certificate**
  - This is a critical vulnerability and allows attackers to perform MITM attacks with a valid certificate without the user's knowledge.

Privacy Risks

The average privacy score using Zimperium's z3A Advanced Application Analysis for the 48 iOS mobile banking apps is 56. The lowest score is 21 and highest score is 86. Areas for improvement for the best in class app are related to system logs and access to a devices album image picker functionality.

The app scoring 86 has numerous issues for privacy and data leakage. This app has numerous advertising frameworks installed, communicating over HTTP, may be leaking data via the UIText auto-correction functionality, allows pinpoint geolocation, and is vulnerable to CVE-2015-5208 and CVE-2015-5207. This app implements a direct Photo Library enumeration functionality that can access EXIF metadata from stored photos and videos. The developer should consider using an approved Apple method to avoid this risk condition.

There are several common privacy and data leakage risks for many of the iOS mobile banking applications. Our research indicates 15 apps are using the open-source, Plausible Labs Crash analytic framework. The company develops and markets a platform for analyzing consumer interactions with mobile apps, solutions for marketers to advertise in-apps, as well as a service for applying monetization structures to mobile apps.

Other apps are using the Crashalytics SDK. Crashlytics is used for crash reporting, application logging, online review and statistical analysis of application logs. Some PII data can be collected such as the UDID of the device, the IP Address which is then geo-coded to a city and in some instances the user's name and email address are also collected. Crashlytics is a Google-owned software company purchased from Twitter in 2017.

Privacy Risks - iOS

Some of the notable privacy risks identified in the sample of iOS apps include:

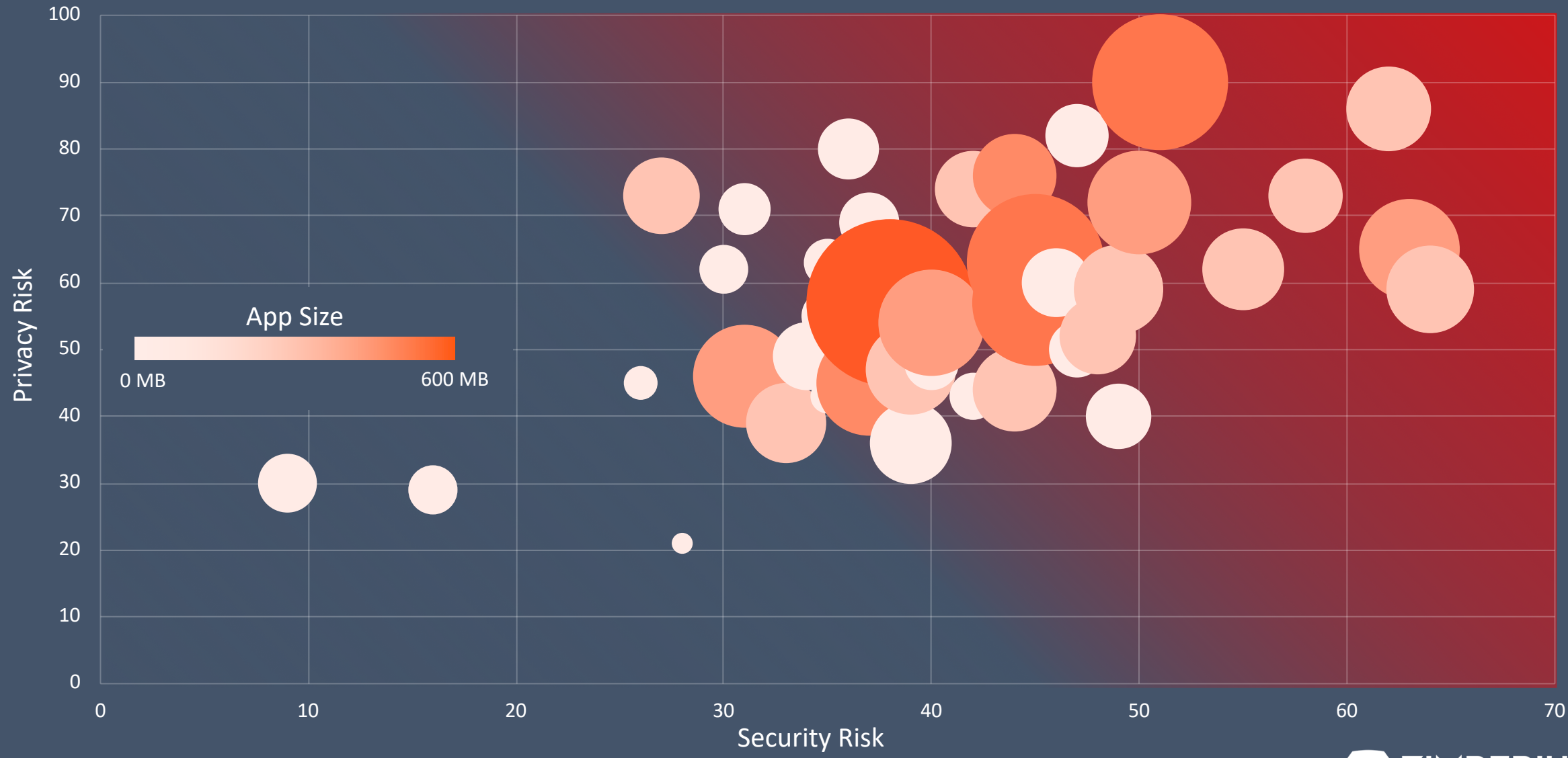
- **38% of apps are capable of taking screenshots**
  - This could possibly lead to data leakage by removing unencrypted text and private data.
- **35% of apps implement the MFMessageComposeViewController**
  - This enables the app to send SMS messages programmatically. Unintentional data leakage, SMS spam and trojan behavior are risk considerations.
- 59**% of apps access 'canSendMail', 'MFMailComposeViewController', and have the ability to send email**
  - Data leakage is the biggest risk concern with email functionality. However, many apps have email functions for support and general questions about the app or banking services.
- **11% of apps send query parameters with private information such as usernames, passwords, device IDs and IP addresses**
  - This could potentially lead to data leakage if not implemented correctly or exploited.

Privacy Risks - Android

Some of the notable privacy risks identified in the sample of Android apps include:

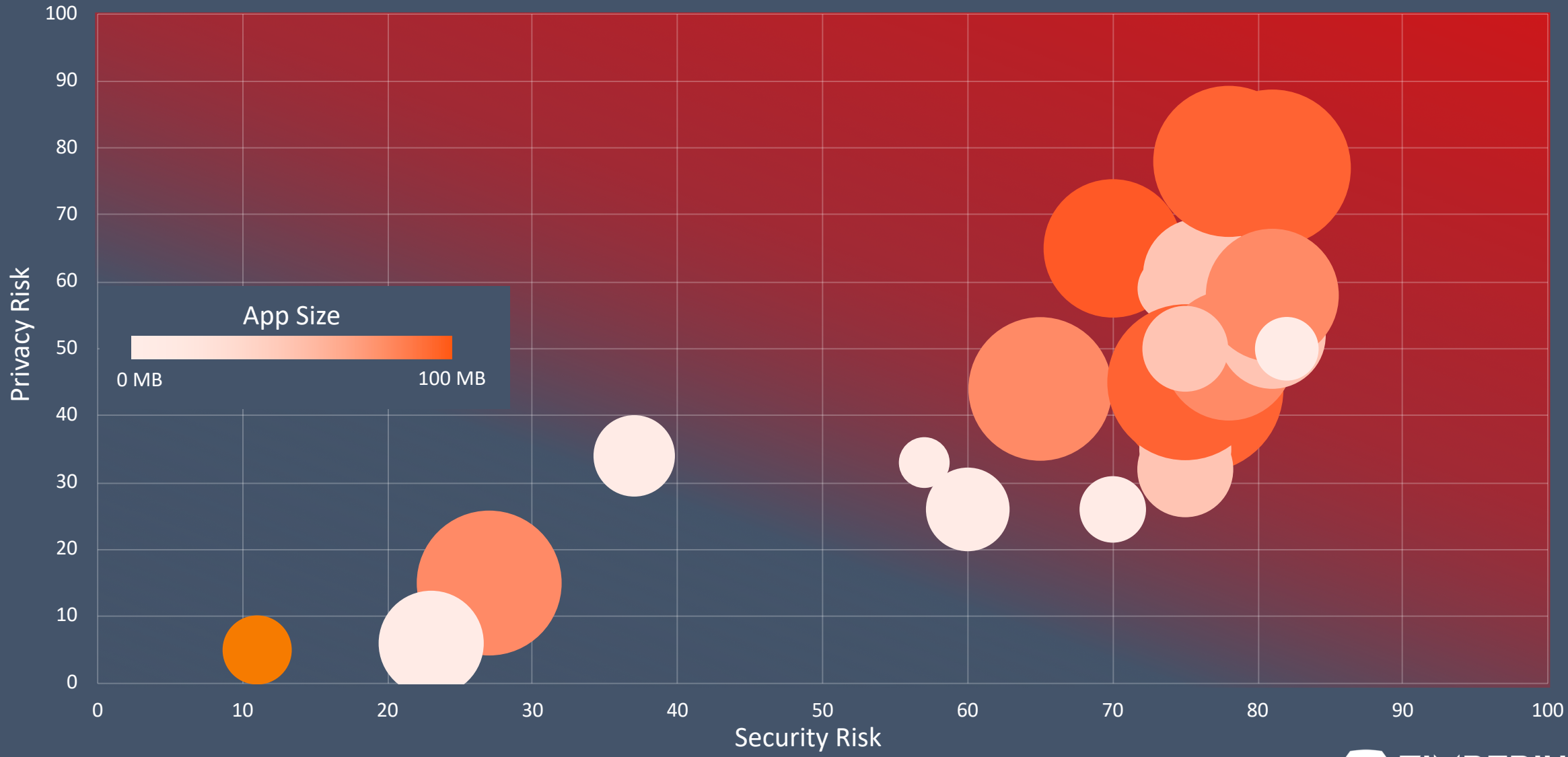
- **29% of apps grant access to content provider data**
  - The permission to one or more content provider's data is granted. This could potentially lead to the disclosure of information.
- **3 apps capture the device serial number**
  - This application requests the device serial number information from the system build properties. The US Federal Trade Commission considers serial numbers a persistent identifier that can be used to recognize a user over time and across different Web sites or online services.
- **27% of apps seek to identify the device manufacturer**
  - This application requests the device manufacture information from the system build properties and looks for specific HTC models.
- **33% of apps query the device model number**
  - This app is reading the device model number or manufacturer.
- **53% of apps access android.permission. ACCESS_FINE_LOCATION**
  - The application can retrieve the device's precise location using GPS or network based locations such as cell towers position and Wi-Fi. This level of accuracy is recommended only for navigation apps.
- **27% of apps access contacts**
  - These applications are accessing your personal contacts.

Privacy and Security Risk Scores - iOS



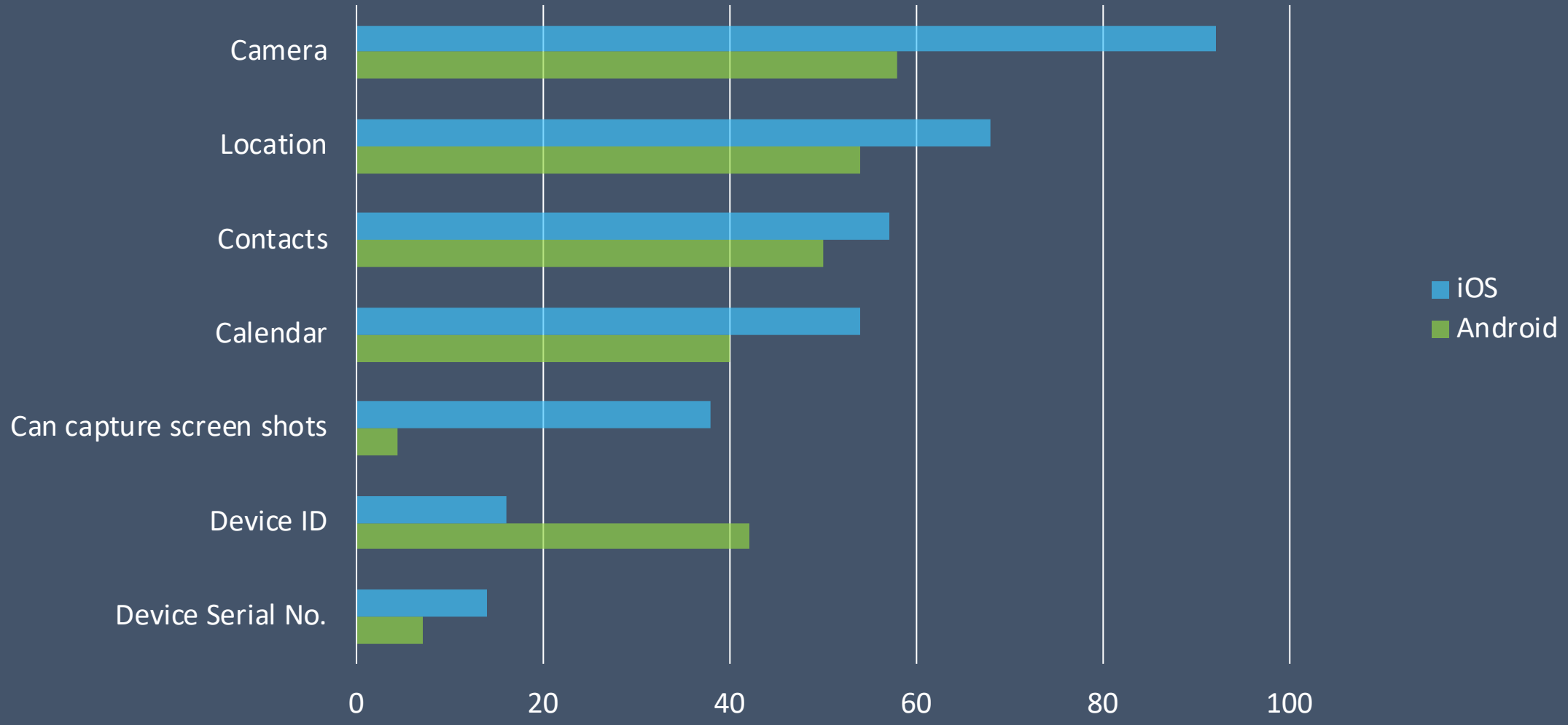
ZIMPERIUM

Privacy and Security Risk Scores - Android



ZIMPERIUM®

Permissions



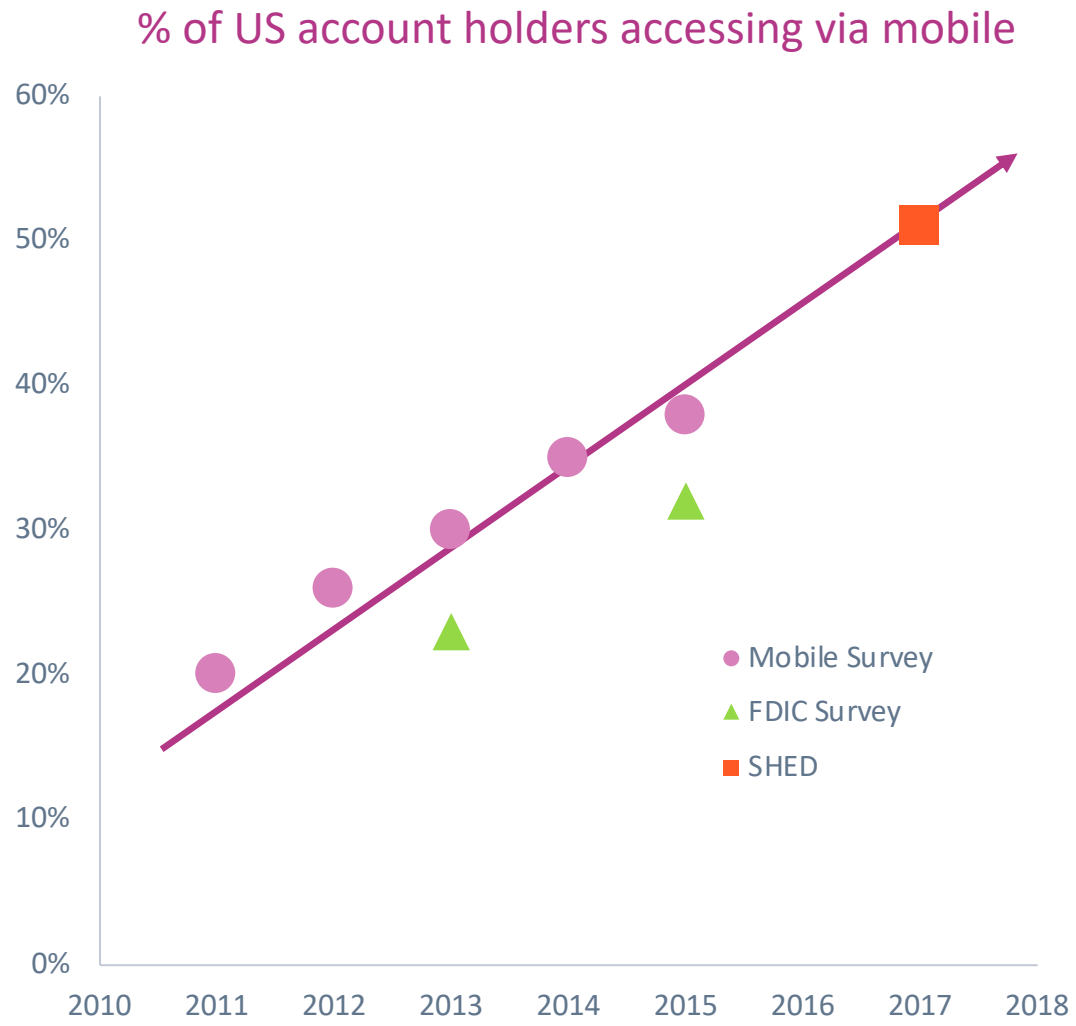
ZIMPERIUM®

Advertising Frameworks

According to the US Federal Reserve, more than half of households with checking accounts, access those accounts via a mobile device. Mobile devices offer extreme conveniences for services like banking and shopping. However, these services could be overextending their reach into private user data mobile devices generate and contain.

Advertisers like Facebook, and Google (Adwords, Admob, and DoubleClick) need user data to serve relevant ads to users. Many mobile apps generate revenue from serving ads to users or incent app developers in order to ingest user data.

In late 2018, Facebook urged banks to integrate services to increase user engagement on its platform. Most banks refused to share user data, but a few still integrate Facebook to offer services in their mobile apps.



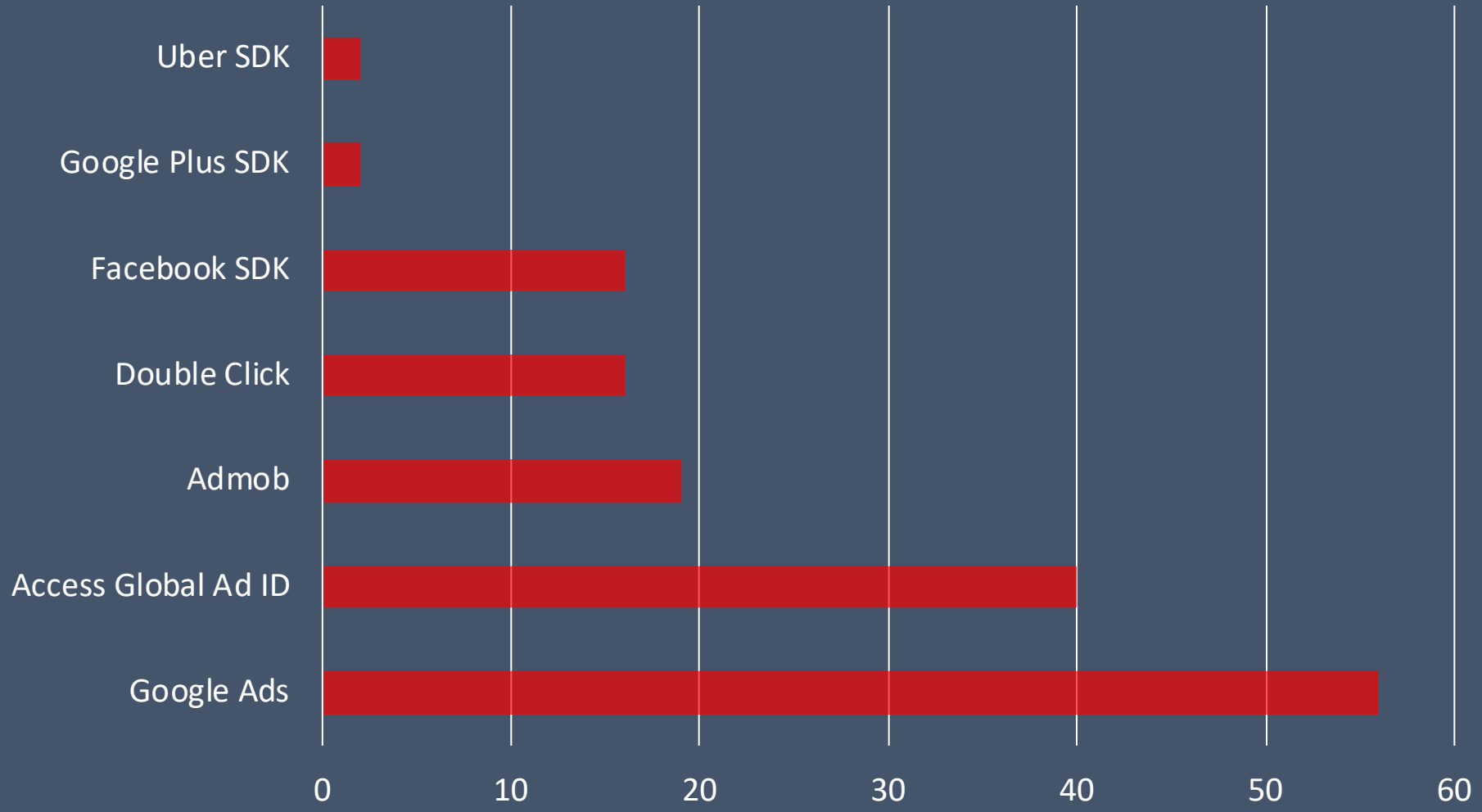
US Federal Reserve

Advertising Frameworks

Our research indicates 16 percent of the banks in our dataset integrate the Facebook SDK into their mobile apps. Using the Facebook SDK by itself is not a vulnerability, but could be considered data leakage depending on what data Facebook is capturing to uniquely identify the user. This would apply not only in the banking apps we reviewed, but any app and platforms embedding Facebook. Other ad platforms and device calls we identified while researching mobile banking apps include:

- 56% of banks integrate Google Ads into their consumer mobile banking apps.
- 19% of iOS apps implement the AdMob advertising framework. AdMob is one of the world's largest mobile advertising platforms and claims to serve more than 40 billion mobile banner and text ads per month across mobile Web sites and handset apps.
- 40% of banks access the iOS 'ASIdentifierManager' and implement low-level API calls to retrieve the device global Ad identifier key which can track the user across any app that also captures this identifier.
- 16% of banks integrate the Facebook SDK.
- 16% of banks install code from DoubleClick for advertising data.
- One app still integrates the Google Plus SDK. This SDK allows users to sign in with their Google account, customize the user experience with Google+ info, pull people into the app with interactive posts, and add +1 buttons so users can recommend the content. On March 7, 2019, Google deprecated Google Plus APIs.
- One app installs the Uber SDK, to presumably, share information on rewards programs.

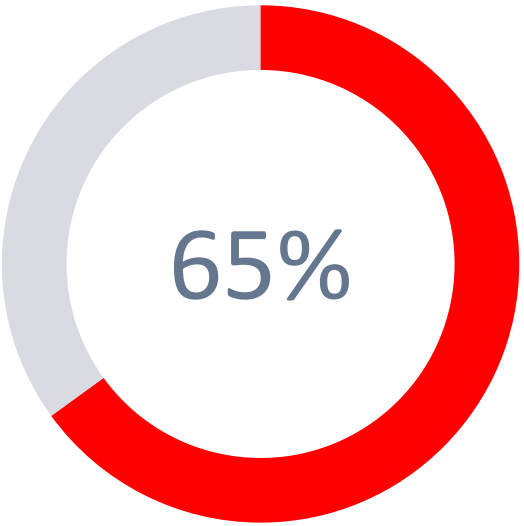
Advertising



ZIMPERIUM®

Malware Campaigns

During our analysis of mobile banking apps, Zimperium sought to discover which of the banks are actively targeted by specific mobile malware campaigns. Our research indicates the BankBot, Anubis II, and Marcher campaigns target 31 of the 48 (65%) banks in our subset.



BankBot is an Android-targeting malware using fake overlay screens to mimic actual banking apps to steal user credentials. Distributed as benign apps in Google Play, BankBot-infected apps pose as entertainment and mobile banking apps. These apps surveil devices for specific banking apps and wait for users to log into their banking apps. The first version of BankBot focused on a small number of institutions but has since evolved to target several hundred banks all over the world, including most in our test.

How BankBot works:

1. An infected app checks package information of other apps installed on the device for one of the targeted bank apps.
2. If one is found, BankBot connects to its command and control server, uploads the target's package name and label and receives an URL to the library containing files used for the overlay webpage.
3. BankBot monitors an infected device for a target bank app to launch. The malware then displays the overlay page on top of the legitimate app when the app runs.
4. The overlay tricks the user into believing they are using the legitimate app, and phishes/steals the user's credentials.
5. The user is none the wiser when surrendering banking credentials to the cybercriminals via the fake overlay screens. Current versions of BankBot also defeat two-factor authentication by capturing a user's secret SMS codes sent to the same device.

Malware Campaigns

Zimperium's research into current malware campaigns identified the following banks as targets:

- Ally Bank
- American Express
- Bank of America
- Barclays
- BBVA Compass
- BMO Harris Bank
- Capital One
- Charles Schwab Corporation
- Citigroup
- Citizens Bank
- Discover Financial
- Fifth Third Bank
- Huntington Bank
- JP Morgan Chase
- Key Bank
- M&T Bank
- Morgan Stanley
- MUFG Union Bank
- PayPal
- PNC Bank
- RBC Bank
- Regions Bank
- Santander Bank
- Silicon Valley Bank
- State Street Corporation
- SunTrust Banks
- TD Bank, N.A.
- U.S. Bancorp
- UBS
- USAA
- Wells Fargo & Co.

Solution

To combat fraud and mobile threats, developers need to arm their apps with self-protection. zIAP™ is a secure and straightforward SDK enabling organizations to deliver self-protecting iOS and Android apps to detect mobile threats and remediation on unmanaged devices. zIAP embeds Zimperium's award-winning z9™ engine within applications by using an easy-to-implement software development kit. zIAP ensures mobile app threat detection and remediation to reduce fraud via your mobile channel and to safeguard sensitive user data.

> "Combating fraud is a major focus for our bank. We selected Zimperium's zIAP to protect millions of online customers, and billions of transactions, against malicious apps like BankBot, device exploits and rogue WiFi networks."
>
> -CSO, Global 100 Financial Institution

Example automated remediations include but are not limited to the following models:

- When a man-in-the-middle (MITM) attack is occurring, the app can automatically establish a VPN to create a secure tunnel.
- When a device has malware like BankBot installed, the app can trigger immediate steps to freeze access until the user resets their password online.
- When a user Jailbreaks or Roots a device, the app can allow the session to continue but raise the user's fraud score to account for the additional risk.
- When an external actor compromises a device, the app can display a dialog box asking the user to complete their transaction offline.

Mobile threat detection and mitigation

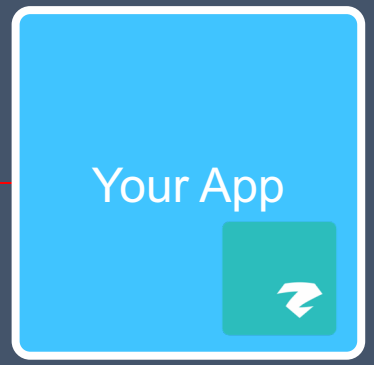
Route to Compromise

Abnormal Process Activity
Device Tampering
App Tampering
Malicious Processes
File system changed
System Tampering

Internal Network Access
MITM
MITM - ARP
MITM - Fake SSL certificate
MITM - ICMP Redirect
MITM - SSL Strip
Rogue Access Point
SSL Downgrades
TCP Scans
Unsecured WiFi

No Encryption
No Device Pin
Malicious Processes
SELinux Disabled
Sideloaded App(s)
Stagefright Vulnerability
Third Party App Store Enabled
USB Debugging Mode Enabled
Device jailbreaking / rooting
Debugger Detection
Emulator Detection

Malware



App Tampering

Device jailbreaking / rooting
Debugger Detection
Emulator Detection

Mitigation (Examples)

Deny Authentication / Lock account

Delete Data / Clear Cache

Increase user risk score

Restrict access until password reset

Establish VPN connection

Restrict or limit mobile transactions

Custom Rules and Remediations

ZIMPERIUM

Conclusion

Mobile banking transactions will continue to increase as services reach more underbanked users and more mobile commerce services become available. Given the growth and the acknowledged risks from all of the stakeholders, banks need to increase mobile banking security ensuring safe and secure banking.

- Despite banks increasingly encouraging customers to use mobile banking apps and acknowledging cybersecurity as the biggest threat to the financial system, banks fail to adhere to coding best practices. Not adhering to application development best practices exposes both customer and bank data and increases the chances for fraud as banks implement more third-party mobile and cloud services.
- Banks continue to use shared code in production apps that is infrequently updated or retired. If shared code is no longer supported or is vulnerable, all apps containing this code are impacted after an incident occurs. Furthermore shared code means that anyone (especially open source code) has the opportunity to review and probe the code for vulnerabilities and weaknesses to identify the attack surface and exploit it.
- Banks continue to share sensitive customer data with advertisers. This practice increases the chances of data leakage if a mobile banking app is reverse engineered or a third-party library or service vulnerability is exploited. If there is a data breach and personally identifiable information is exposed, banks suffer severe brand damage and regulatory fines.
- Most banks fail to obfuscate code, secure mobile device data, or implement runtime application self-protection for mobile apps. Failing to obfuscate code allows anyone to reverse engineer an app to identify weaknesses and identify an attack surface. One such example is how cybercriminals reverse-engineered a mobile app to identify vulnerabilities and stole millions of dollars overnight from thousands of users all without being detected.

To implement runtime self protection into your mobile app or to identify your bank in our report, please contact us today for a consultation.

About the authors

Scott King

Director Embedded Security

Scott has over 20 years experience providing customized software solutions to enterprise customers in mobile, supply chain and DevOps. Scott invests his time researching mobile app security and worldwide mobile threat events.

Ken Lloyd

VP of Risk

Ken Lloyd is a highly accomplished Senior Global Tech Executive and Board Member with more than 20 years of success in cyber security sector focusing in on the areas of Anti-Malware/Virus technologies and Mobile Security product solutions.

Matteo Favaro

Malware Analyst

Matteo is a malware analyst and researcher at Zimperium. He spends his time studying new novel attack vectors and finding ways to defend against new threats. He has a soft spot for (de-)obfuscation and software protection topics.

Appendix

The following pages contain the high-level summaries from the z3A application scans. Each bank's iOS and Android apps are scanned independently. Each summary page greatly condenses the information from each report into a simple picture. Many z3A technical reports contain great detail and are more than 70 pages when printed. Each bank name is obfuscated in order to not identify the bank or its mobile apps. If you would like more information about your mobile application score, please contact us for a consultation.

Bank - US 51b

The iOS app sends query parameters with private information such as the password. The iOS app implements an over-the air app installation and links to several non-HTTPS link including a personal email address. The app was compiled without implementing Stack smashing protection. The Android app enables WebView to execute JavaScript code. The app grants one or more permissions to content provider's data. This could potentially lead to the disclosure of information. The Android app is actively targeted by the BankBot malware campaign.

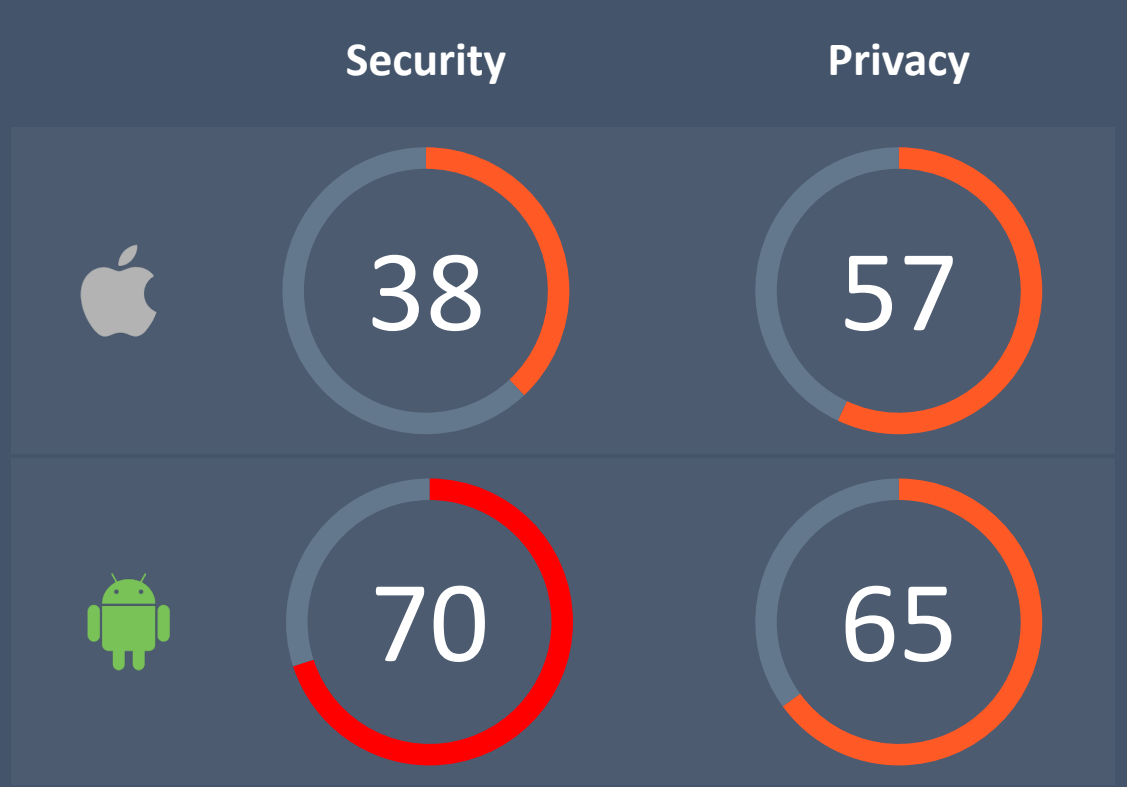
OWASP Mobile Top 10		
Improper Platform Usage	🟢	🟢
Insecure Data Storage	🔴	🔴
Insecure Communications	🔴	🟢
Insecure Authentication	🟢	🟢
Insufficient Cryptography	🟢	🟢
Insecure Authorization	🟢	🟢
Client Code Quality	🟢	🟢
Code Tampering	🔴	🟢
Reverse Engineering	🔴	🔴
Extraneous Functionality	🟢	🟢

	Security	Privacy
	64	59
	78	78

ZIMPERIUM

Bank - US 76r

The iOS app sends query parameters with private information such as the IP address. The app is using a non-encrypted HTTP connection. The app implements low-level API calls to retrieve the device global Ad identifier key and implements low-level telephony enumeration API calls. The Android app uses synthetic methods to access private class entries which are normally not accessible. This is a suspicious and unusual coding practice that should be reviewed. The permission to one or more content provider's data is granted. This could potentially lead to the disclosure of information. The app is actively targeted by the BankBot malware campaign.

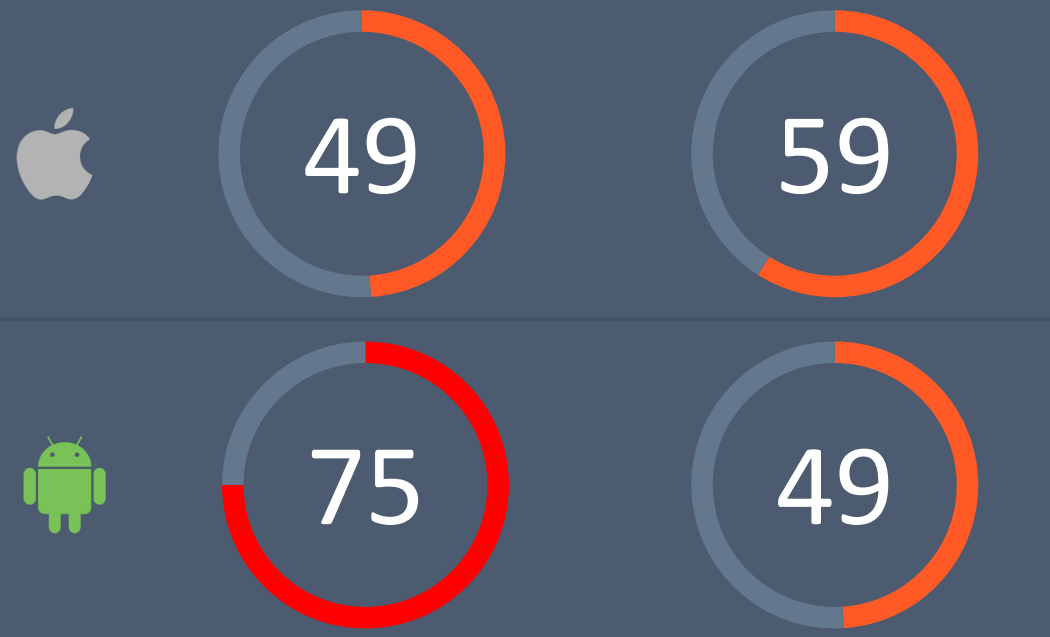


ZIMPERIUM

Bank - US 11c

The iOS app implements Swizzling API calls. This could indicate malware activity and may impact the app ability to trust security decisions which is based on untrusted inputs or manipulated/swizzled output. The Android app uses the WebKit to download a file from the Internet. The application's methods of injected Java objects are enumerable from JavaScript. The app enables WebView to execute JavaScript code. This could potentially allow an attacker to introduce arbitrary JavaScript code to perform malicious actions.

OWASP Mobile Top 10		
Improper Platform Usage	🟢	🟢
Insecure Data Storage	🔴	🔴
Insecure Communications	🔴	🔴
Insecure Authentication	🟢	🟢
Insufficient Cryptography	🟢	🟢
Insecure Authorization	🟢	🟢
Client Code Quality	🟢	🟢
Code Tampering	🔴	🟢
Reverse Engineering	🔴	🔴
Extraneous Functionality	🟢	🟢

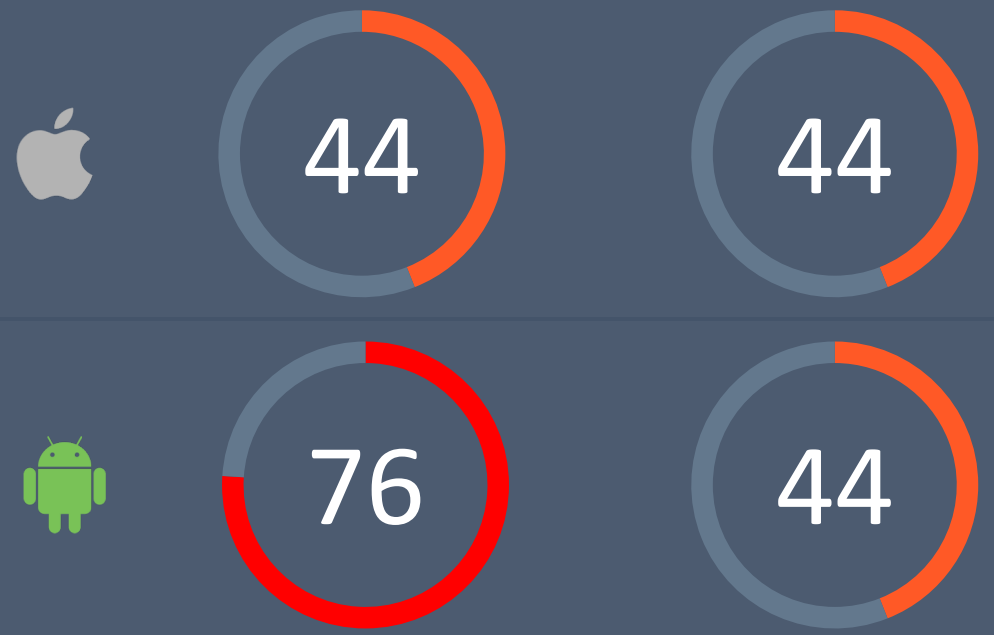


ZIMPERIUM

Bank - US 70d

The iOS app uses Inter-App Audio, which allows the app to send and receive audio from other Inter-App Audio-enabled apps. The app is using a non-encrypted HTTP connection and implements low-level telephony enumeration API calls, this includes carrier, cell-tower, Cell ID, signal Strength and provider. The Android app is not performing active checks and validating SSL certificates. It can allow self-signed, expired or mismatch CN certificates for SSL connections. It is actively targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●



ZIMPERIUM

Bank - US 59v

The iOS app attempts to retrieve a contact from the AddressBook from the depredated ABAddressBookGetPersonWithRecordID. The app retrieves the device global Ad identifier and implements AdMob advertising framework. This app uses Inter-App Audio which is a privacy risk and allows the app to send and receive audio from other Inter-App Audio-enabled apps. The Android app grants permissions to one or more content provider's data and is actively targeted by the BankBot malware campaign.

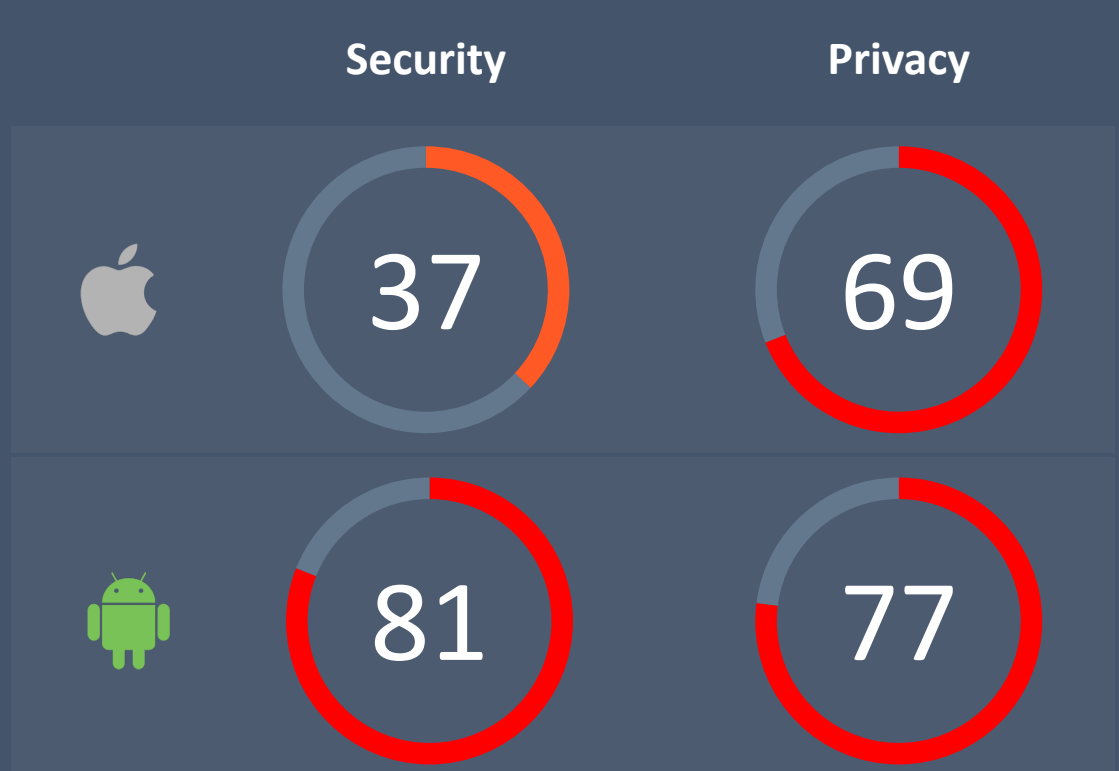
OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

	Security	Privacy
	31	71
	75	47

ZIMPERIUM

Bank - US 36x

The iOS app implements a low-level IOKit device enumeration API calls. Implementing a low-level IOKit device enumeration API call could indicate a malicious intent to obtain data normally not available. The developer should consider using an approved Apple method to avoid this risk condition. The Android app uses the 'proceed()' method during SSL error handling. This can allow the connection to be established even if the SSL Certificate is invalid. This is not a recommended practice. The app can send SMS messages and can capture the receipt of an SMS message. This app is actively targeted by the BankBot malware campaign.



ZIMPERIUM

Bank - US 81k

The app is using a non-encrypted HTTP connections and implements low-level API call to retrieve the device global Ad identifier key. The app was compiled without implementing Stack smashing protection nor ObjectiveC Automatic Reference Counting. The Android app modifies its user agent string. It is recommended the developer use the properties derived from System.getProperty("http.agent") or WebView(this).getSettings().getUserAgentString() to set the user agent string. This app is actively targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage		
Insecure Data Storage		
Insecure Communications		
Insecure Authentication		
Insufficient Cryptography		
Insecure Authorization		
Client Code Quality		
Code Tampering		
Reverse Engineering		
Extraneous Functionality		

	Security	Privacy
	48	52
	75	35

ZIMPERIUM

Bank - US 62d

The iOS app attempts to retrieve a contact from the AddressBook from the depredated ABAddressBookGetPersonWithRecordID. The app retrieves the device global Ad identifier and implements AdMob advertising framework. The app is configured to allow unsecure connection to servers. It will allow cipher suites that do not support forward secrecy and does not discriminate between HTTP or HTTPS. The Android application refers to SSL/TLS hosts with self-signed certificate and other apps also use this same certificate. The app is targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

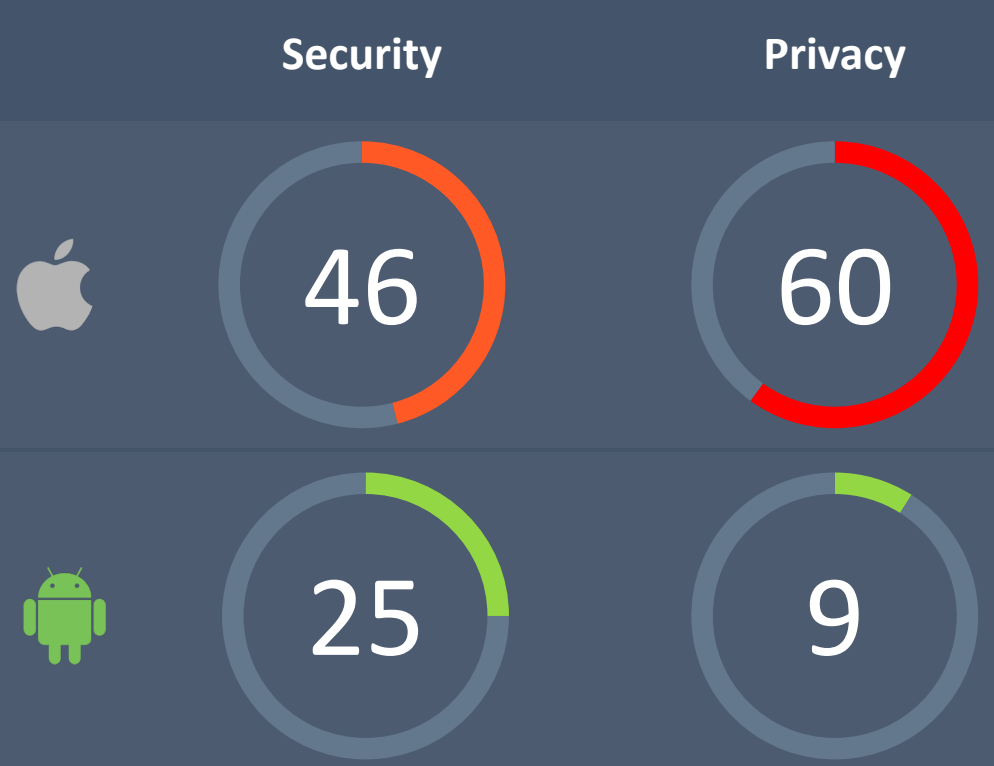
	Security	Privacy
	62	86
	76	61

ZIMPERIUM

Bank - US 56p

The iOS app implements a low-level IOKit device enumeration API call. Because enumeration is currently prohibited by Apple it could indicate trojan or malware activity. The developer should consider using an approved Apple method to avoid this risk condition. The app may be vulnerable to local or remote SQL injection attack. The Android app stores inline API keys and values. The app is not implementing Certificate Pinning. Both apps fail the Static Data Exposure and Source Code Reverse Engineering Exposure test and expose source level metadata symbols as outlined by OWASP Mobile Top 10.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●



ZIMPERIUM

Bank - US 33g

This app sends query parameters with device information and private information such as the first name, last name, and password. This app does not protect its jailbreak detection functionality from manipulation and fails the OWASP Top 10 Binary Protection testing. The Android app is not performing active checks and validating SSL certificates. It can allow self-signed, expired or mismatch CN certificates for SSL connections. The application uses PendingIntent. It grants the right to perform the operation specified with the same permissions and identity of the calling app

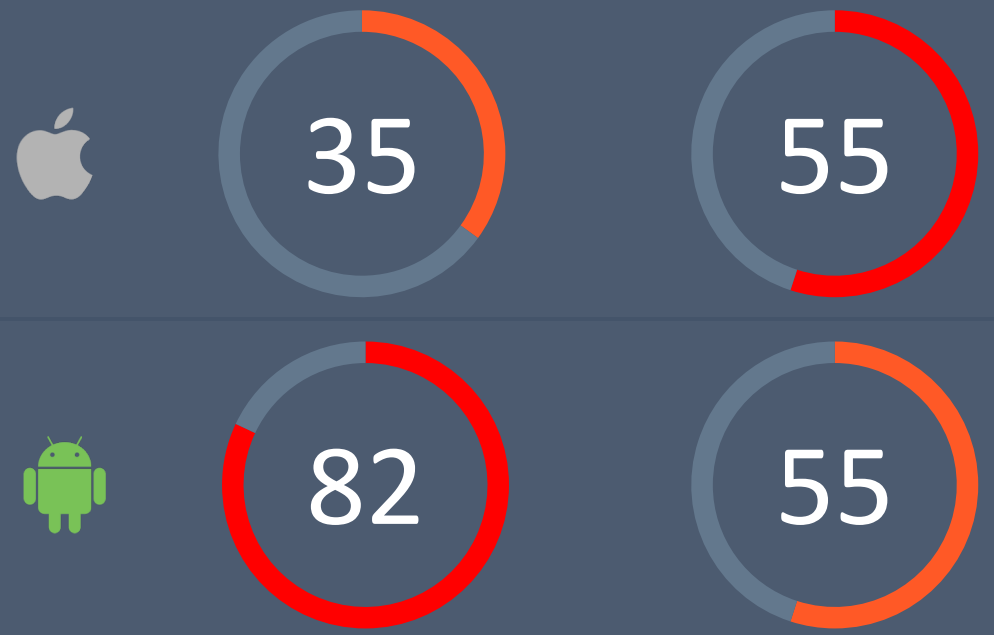
OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

ZIMPERIUM

Bank - US 18j

The iOS app is implementing the EventKit Framework which enables it to gain access to calendar data. 'Bearer' related oauth tokens were found. An adversary could potentially gain access to these tokens using the Juice Jacking approach if they are not encrypted. The app implements Swizzling API calls. This could indicate malware activity and may impact the app ability to trust security decisions. The Android app This application can execute commands at the OS level such as launching other applications and processes and can access system commands that require a rooted device. The app uses unsecure storage data mode (WORLD_READABLE, WORLD_WRITABLE).

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●



ZIMPERIUM

Bank - US 98y

The iOS app is actively monitoring the pasteboard and sends query parameters with private information such as the username. The app is was compiled without implementing Stack smashing protection. This app implements AdMob advertising network. The Android app uses the WebKit to download a file from the Internet. The app enables WebView to execute JavaScript code. This could potentially allow an attacker to introduce arbitrary JavaScript code to perform malicious actions. This app is actively targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

	Security	Privacy
	47	82
	76	47

ZIMPERIUM

Bank - US 83s

This app uses Inter-App Audio allowing the app to send and receive audio from other Inter-App Audio-enabled apps. There is a privacy risk of audio being captured and being sent to other Inter-App audio-enabled apps. The app implements low-level API call to retrieve the device global Ad identifier key and the app implements the AdMob advertising framework. The app uses Swizzling API calls and uses HTTP connections. The Android app is not doing active checks and validating SSL certificates. It can allow self-signed, expired or mismatch CN certificates for SSL connections and is actively targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

	Security	Privacy
	44	67
	74	59

ZIMPERIUM

Bank - US 48k

The iOS app can access the device unique identifier without notification to the user. The user should receive notification of this activity to protect privacy. The application stores inline API keys and values. The app is implementing the EventKit Framework which enables it to gain access to calendar data. The Android application requests the device manufacture information from the system build properties and is looking for specific Samsung models. The app has the ability to send an email via an Intent. This is an unusual method to create an email. The Android app is actively targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	🟢	🟢
Insecure Data Storage	🔴	🔴
Insecure Communications	🔴	🔴
Insecure Authentication	🟢	🟢
Insufficient Cryptography	🟢	🟢
Insecure Authorization	🟢	🟢
Client Code Quality	🟢	🟢
Code Tampering	🔴	🟢
Reverse Engineering	🔴	🔴
Extraneous Functionality	🟢	🟢

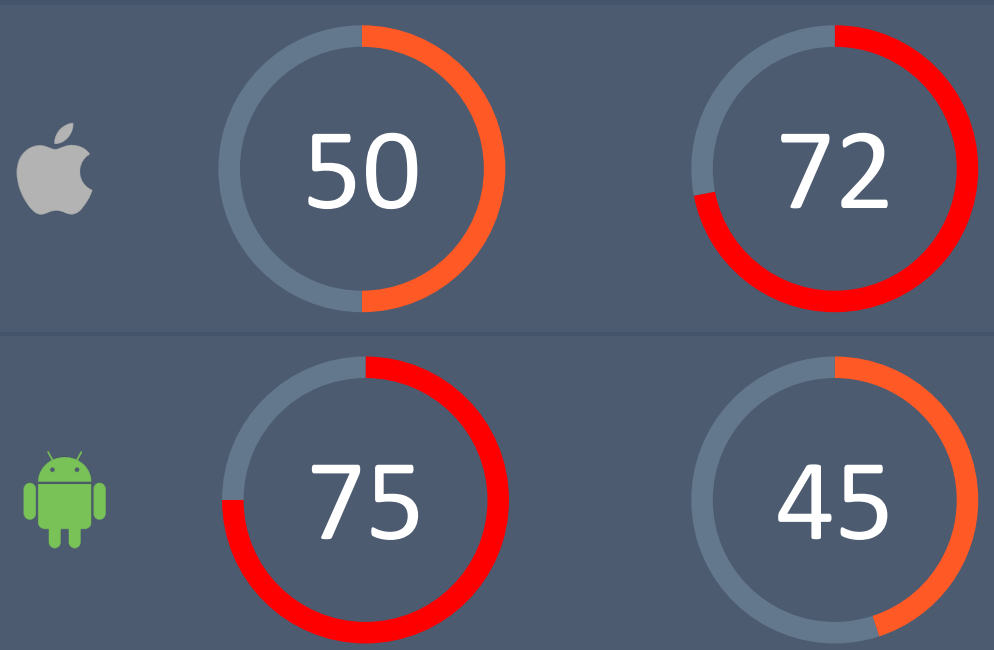
	Security	Privacy
iOS	45	63
Android	65	44

ZIMPERIUM

Bank - US 34f

The iOS app implements low-level API call to retrieve the device global Ad identifier key and implements the AdMob advertising framework. It uses a non-encrypted HTTP connection sends query parameters with private information such as the first name, last name, and the password. The app source code is exposed to reverse engineering. The Android app is reading all of the contact data on the device. The app contains exported components that are not protected by a permission. If a component, such as a service, is exported and not protected with strong permissions, then any application can start and bind to the service.

OWASP Mobile Top 10		
Improper Platform Usage	🟢	🟢
Insecure Data Storage	🔴	🟢
Insecure Communications	🔴	🔴
Insecure Authentication	🟢	🟢
Insufficient Cryptography	🟢	🔴
Insecure Authorization	🟢	🟢
Client Code Quality	🟢	🟢
Code Tampering	🔴	🟢
Reverse Engineering	🔴	🔴
Extraneous Functionality	🟢	🟢



ZIMPERIUM

Bank - US 21h

The iOS app implements low-level API call to retrieve the device global Ad identifier key. The app implements Cloud-Base Storage through the CloudKit framework. This app sends query parameters with device information and private information such as the IP address. The Android app send a text based SMS message. This application has access to the device microphone and can record audio. It is important to determine how and when this feature is implemented within the app to then quantify this risk finding. The application stores inline API keys and values and requests the device serial number information from the system build properties.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

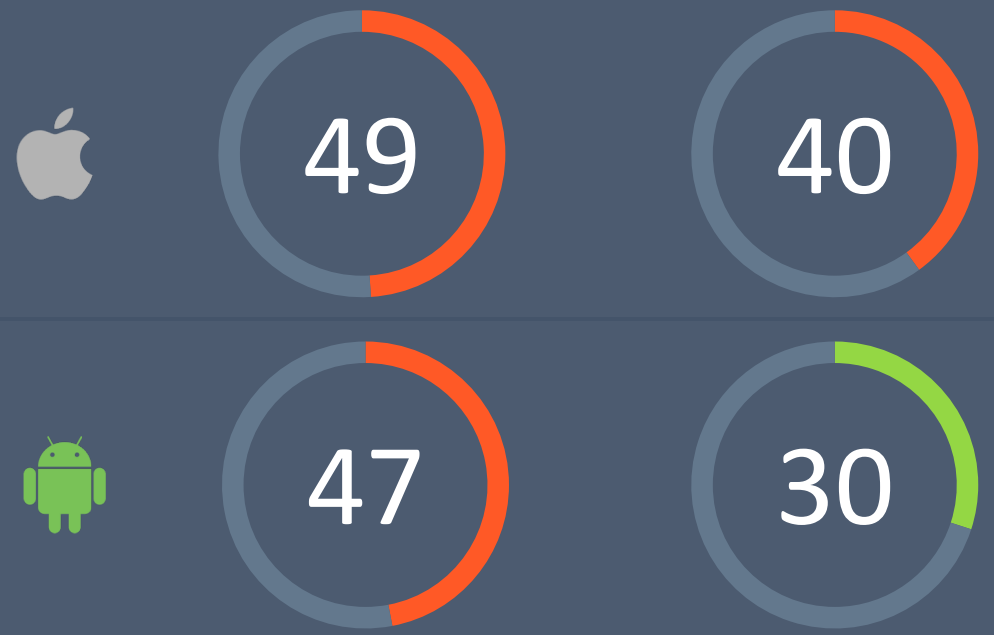
	Security	Privacy
	45	57
	78	49

ZIMPERIUM

Bank - US 14i

The iOS app was compiled without implementing Stack smashing protection and without ObjectiveC Automatic Reference Counting (ARC) Flag. The app is configured to allow unsecure and unverified connection to servers with lower TLS versions. The Android app implements checkCallingOrSelf(Uri)Permission. This method is used to determine the permissions of the app being called. This method also has the potential to grant the calling application the same permissions as this app. It is recommended to use checkCalling(Uri)Permission instead. It is actively targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●



ZIMPERIUM

Bank - US 92t

The iOS app sends query parameters with private information such as the IP address. The app implements a Listening Server over TCP Sockets and implements network sockets. The app opens a networking listening port. The app backend may be vulnerable to OWASP Web Top 10 Mobile weaknesses. The Android app enables WebView to execute JavaScript code. This could potentially allow an attacker to introduce arbitrary JavaScript code to perform malicious actions. This application can execute commands at the OS level such as launching other applications and processes.

OWASP Mobile Top 10		
Improper Platform Usage	🟢	🟢
Insecure Data Storage	🔴	🟢
Insecure Communications	🔴	🟢
Insecure Authentication	🟢	🟢
Insufficient Cryptography	🟢	🟢
Insecure Authorization	🟢	🟢
Client Code Quality	🟢	🟢
Code Tampering	🔴	🟢
Reverse Engineering	🔴	🔴
Extraneous Functionality	🟢	🟢

	Security	Privacy
	42	43
	57	33

ZIMPERIUM®

Bank - US 85a

Both applications refer to endpoints that are currently exposed to the FREAK vulnerability. The iOS app implements low-level API call to retrieve the device global Ad identifier key. The Android app grants permission to data from one or more content providers. This could potentially lead to the disclosure of information. The application stores inline API keys and values and is actively targeted by the BankBot malware campaign. This app can load compiled code in APK and JAR files. This can include files located in external storage and potentially on the Internet.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

	Security	Privacy
	55	62
	81	52

ZIMPERIUM®

Bank - US 61v

The iOS app implements low-level API call to retrieve the device global Ad identifier key and is using non-encrypted HTTP connections. The app implements low-level telephony enumeration API calls, including, cell-tower, Cell ID, signal Strength and provider. This app sends query parameters with private information such as the IP address. Application has implemented the Rollout SDK which enables it to push code changes dynamically, This behavior is restricted by Apple guidelines, as it enables external entities to affect application run-time behavior. The Android app grants permission to one or more content provider's data and is actively targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

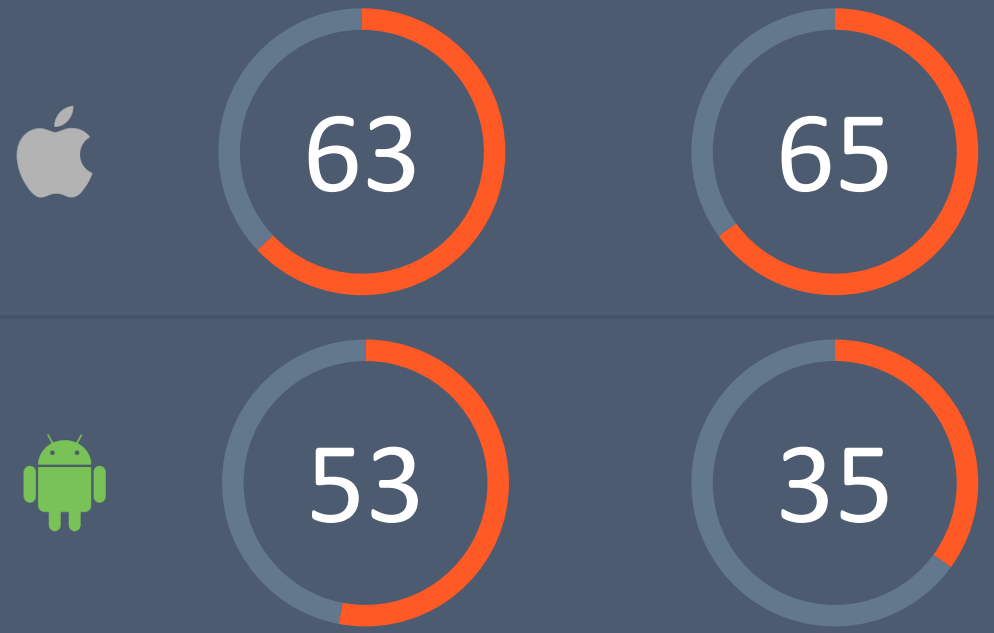
	Security	Privacy
	58	73
	81	58

ZIMPERIUM

Bank - US 45j

This app uses Inter-App Audio allowing the app to send and receive audio from other Inter-App Audio-enabled apps. There is a privacy risk of audio being captured and being sent to other Inter-App audio-enabled apps. The app implements low-level API call to retrieve the device global Ad identifier key. Application has implemented the Rollout SDK enabling it to dynamically push code changes. This behavior is restricted by Apple guidelines, as it enables external entities to affect application run time behavior. The Android app refers to SSL/TLS hosts with self-signed certificate and the app is not implementing Certificate Pinning. It is actively targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

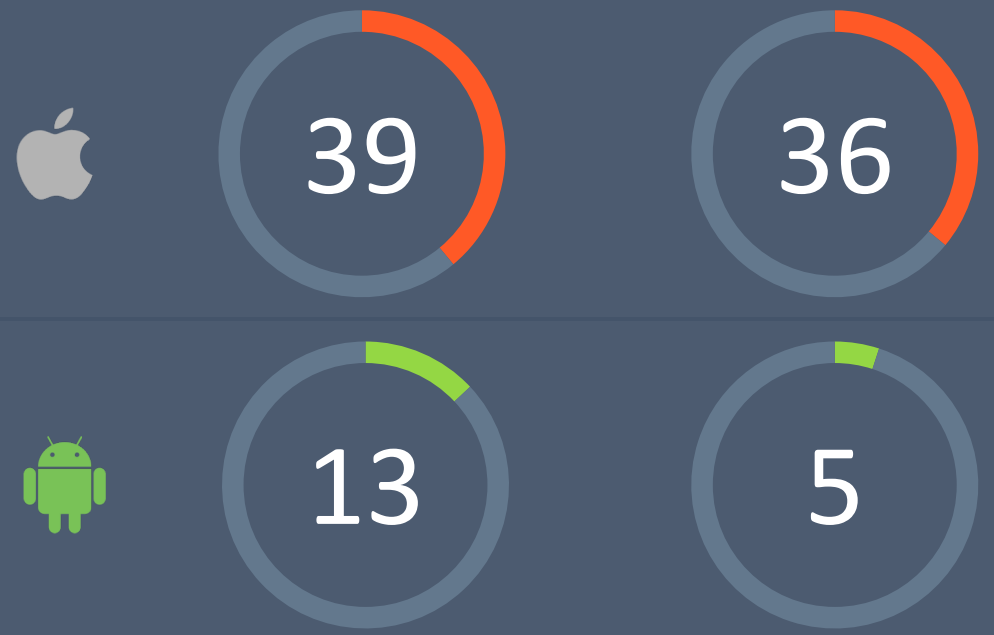


ZIMPERIUM

Bank - US 28x

Bearer' related oauth tokens were found in the iOS App. An adversary could potentially gain access to these tokens. This application is implementing the PhoneGap SDK, which may permit the developer to remotely modify application functionality. The app implements Swizzling API calls. This app is reading the device serial number. The Android app is not implementing Certificate Pinning. The app contains exported components that are not protected by a permission. If a component, such as a service, is exported and not protected with strong permissions, then any application can start and bind to the service.

OWASP Mobile Top 10		
Improper Platform Usage	🟢	🟢
Insecure Data Storage	🔴	🟢
Insecure Communications	🔴	🟢
Insecure Authentication	🟢	🟢
Insufficient Cryptography	🟢	🟢
Insecure Authorization	🟢	🟢
Client Code Quality	🟢	🟢
Code Tampering	🔴	🟢
Reverse Engineering	🔴	🔴
Extraneous Functionality	🟢	🟢

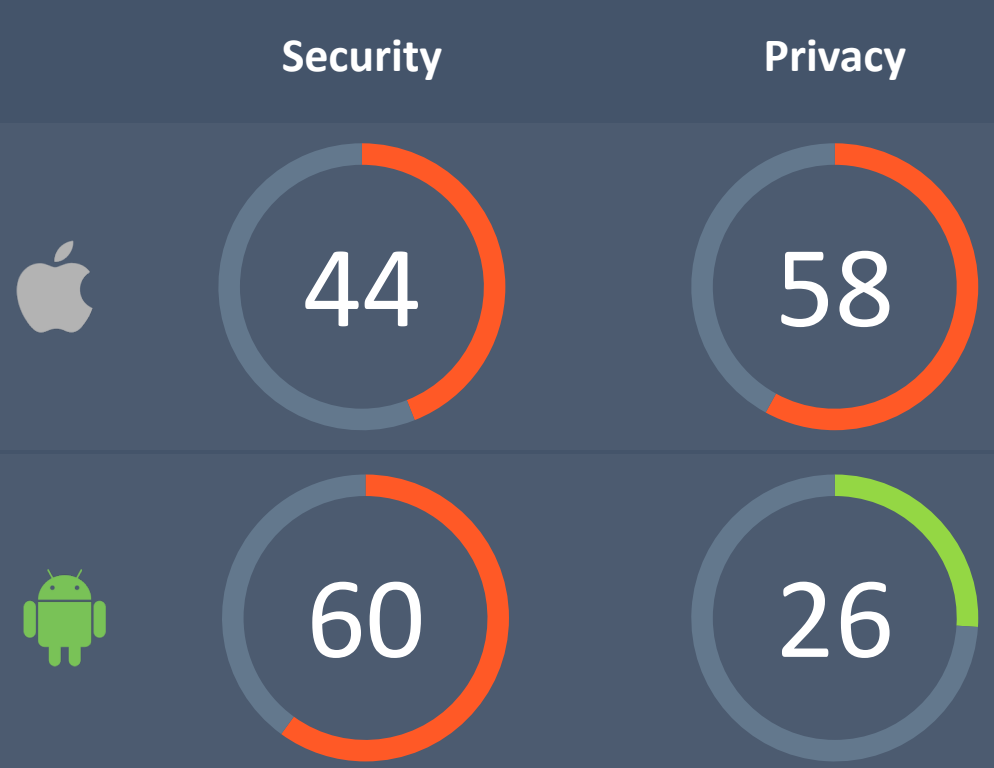


ZIMPERIUM

Bank - US 19p

The iOS app implements an over-the air app installation. This implementation could be utilized to install a drive-by malicious apps.The Android app uses Java code reflection. Reflection enables a Java program to analyze and modify itself. An attacker can create unexpected control flow paths through the application, potentially by-passing security checks. Exploitation of this weakness can result in a limited form of code injection. The Android app is actively targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	🟢	🟢
Insecure Data Storage	🔴	🔴
Insecure Communications	🔴	🔴
Insecure Authentication	🟢	🟢
Insufficient Cryptography	🟢	🟢
Insecure Authorization	🟢	🟢
Client Code Quality	🟢	🟢
Code Tampering	🔴	🟢
Reverse Engineering	🔴	🔴
Extraneous Functionality	🟢	🟢



ZIMPERIUM

Bank - US 49p

Bearer' related oauth tokens were found in the iOS App. An adversary could potentially gain access to these tokens. This app does not protect its jailbreak detection functionality from manipulation and fails the OWASP Top 10 Binary Protection testing. The Android app uses unsecure storage data mode (WORLD_READABLE, WORLD_WRITABLE). The app utilizes the Java.io.File delete() method to perform a file deletion. This method is not a secure file deletion method. This application uses synthetic methods to access private class entries, which are normally not accessible. This is a suspicious and unusual coding practice.

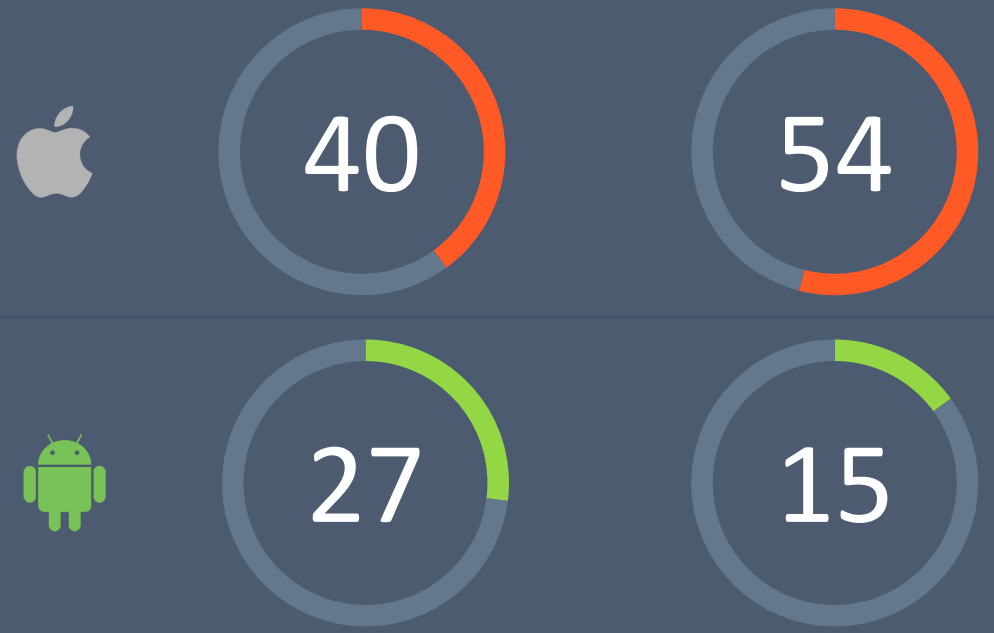
OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

	Security	Privacy
	35	43
	82	50

ZIMPERIUM

Bank - US 13n

The iOS app was compiled without implementing Stack smashing protection or ObjectiveC Automatic Reference Counting (ARC) Flag. Content providers are implicitly not secure on the Android app. They allow other applications on the device to request and share data. If sensitive information is accidentally leaked to a content provider, an attacker can call the content provider and the sensitive data is exposed to the attacker by the application. The application stores inline API keys and values. The Android app is actively targeted by the BankBot malware campaign.

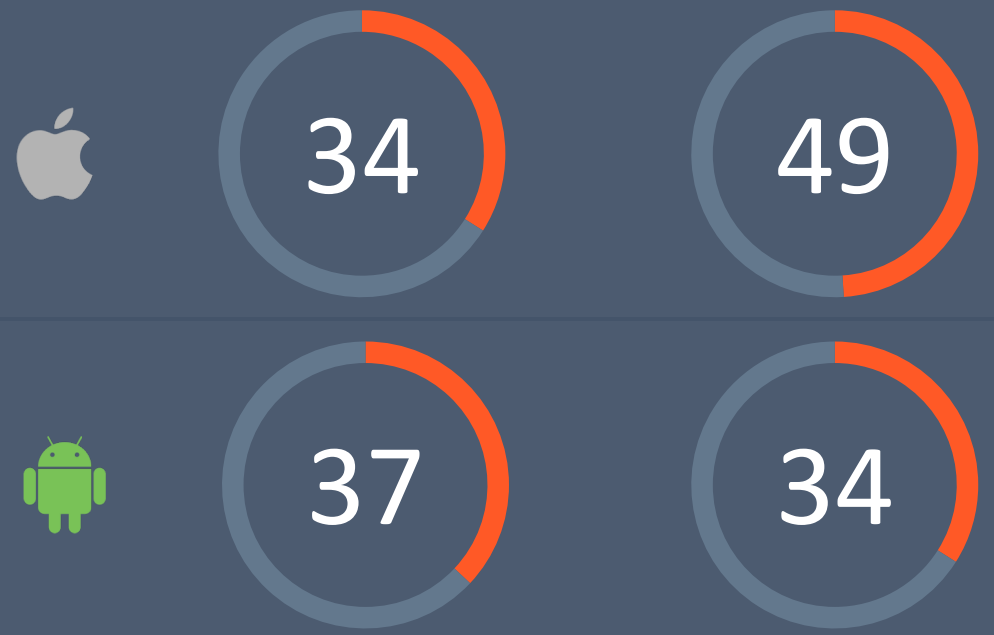


ZIMPERIUM

Bank - US 68d

The IOS app is using a non-encrypted HTTP connection and implements low-level telephony enumeration API calls including carrier, cell-tower, Cell ID, signal Strength and provider and to retrieve the device global Ad identifier key. This app sends query parameters with private information such as the password. The Android app is not performing active checks and validating SSL certificates. It can allow self-signed, expired or mismatch CN certificates for SSL connections. It is actively targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	🟢	🟢
Insecure Data Storage	🔴	🔴
Insecure Communications	🔴	🔴
Insecure Authentication	🟢	🟢
Insufficient Cryptography	🟢	🟢
Insecure Authorization	🟢	🟢
Client Code Quality	🟢	🟢
Code Tampering	🔴	🟢
Reverse Engineering	🔴	🔴
Extraneous Functionality	🟢	🟢

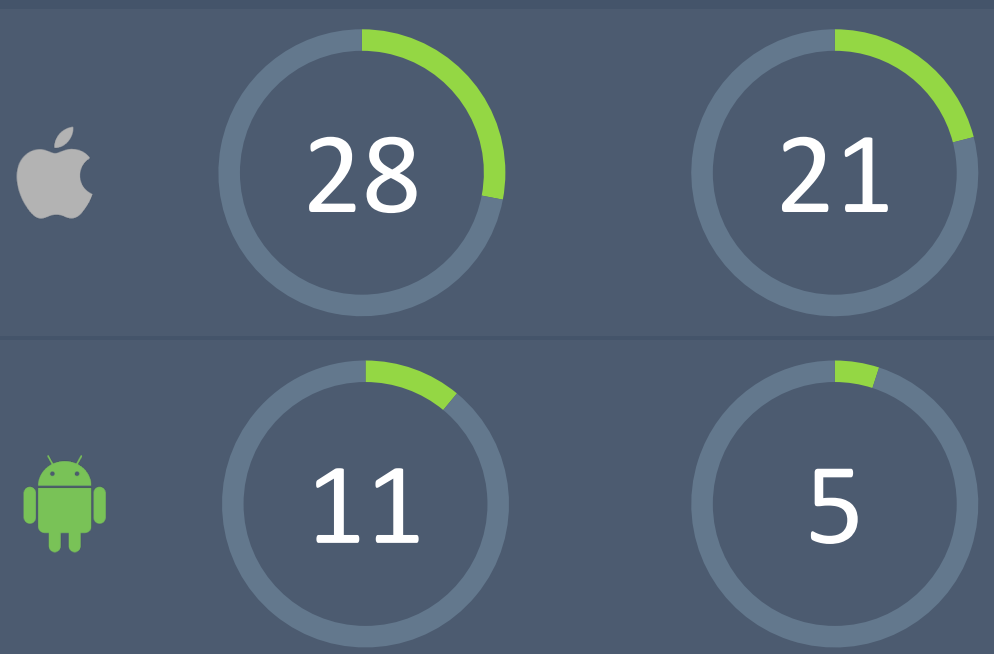


ZIMPERIUM

Bank - US 73q

The iOS app implements a Listening Server over TCP Sockets. The app implements Swizzling API calls. This could indicate malware activity and may impact the app ability to trust security decisions based on untrusted inputs or manipulated/swizzled output. This app does not protect its jailbreak detection functionality from manipulation and fails the OWASP Top 10 Binary Protection testing. The Android app contains exported components that are not protected by a permission. This application fails the Source Code Reverse Engineering Exposure test and is actively targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

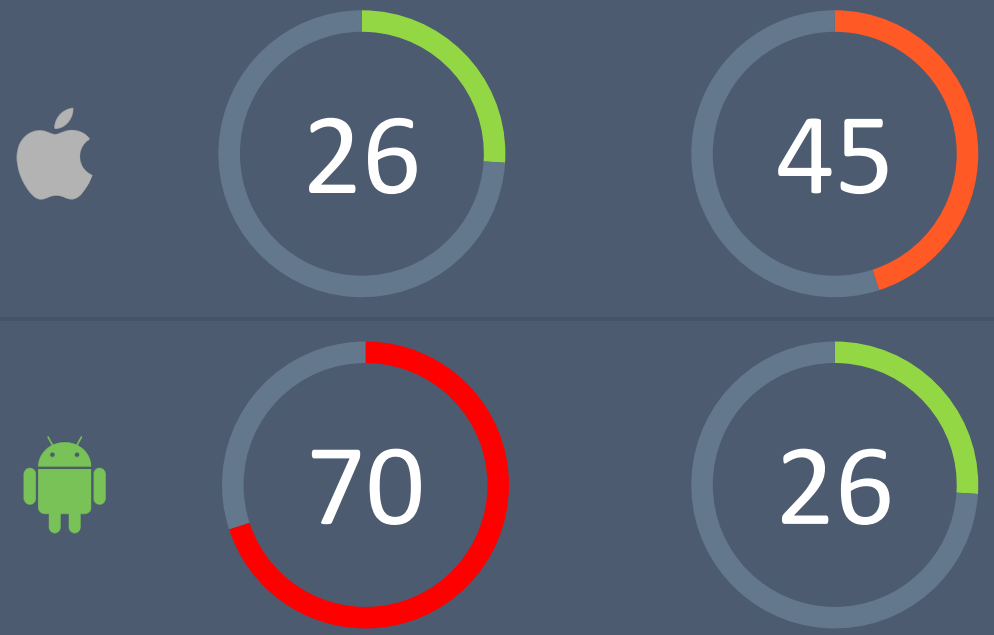


ZIMPERIUM

Bank - US 80w

Bearer' related oauth tokens were found in the iOS App. An adversary could potentially gain access to these tokens. This application is implementing the PhoneGap SDK, which may permit the developer to remotely modify application functionality. The app implements Swizzling API calls. The app is reading the device serial number. The Android app uses the WebKit to download a file from the Internet. The app enables WebView to execute JavaScript code. This could potentially allow an attacker to introduce arbitrary JavaScript code to perform malicious actions. This application has functionality to retrieve apps, Java code and DEX files from remote locations.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●



ZIMPERIUM

Bank - US 53c

The iOS application was compiled without implementing Stack smashing protectionand without ObjectiveC Automatic Reference Counting (ARC) Flag. 'Bearer' related oauth tokens were found. An adversary could potentially gain access to these tokens using the Juice Jacking approach if they are not encrypted. The app implements Swizzling API calls. This could indicate malware activity. The Android app is not implementing Certificate Pinning. The app contains exported components that are not protected by a permission. If a component or service is exported and not protected with strong permissions, then any application can start and bind to the service.

OWASP Mobile Top 10		
Improper Platform Usage		
Insecure Data Storage		
Insecure Communications		
Insecure Authentication		
Insufficient Cryptography		
Insecure Authorization		
Client Code Quality		
Code Tampering		
Reverse Engineering		
Extraneous Functionality		

	Security	Privacy
	47	50
	23	6

ZIMPERIUM

Bank - US 91r

The iOS application is implementing the PhoneGap SDK, which may permit the developer to remotely modify application functionality. This app does not protect its jailbreak detection functionality from manipulation and fails the OWASP Top 10 Binary Protection testing. The Android app uses the 'proceed()' method during SSL error handling. This can allow the connection to be established even if the SSL Certificate is invalid. This is not a recommended practice. The app is actively targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage		
Insecure Data Storage		
Insecure Communications		
Insecure Authentication		
Insufficient Cryptography		
Insecure Authorization		
Client Code Quality		
Code Tampering		
Reverse Engineering		
Extraneous Functionality		

	Security	Privacy
	9	30
	44	16

ZIMPERIUM

Bank - US 46h

The app implements a direct Photo Library enumeration functionality that can access EXIF metadata from stored photos and videos. The developer should consider using an approved Apple method to avoid this risk condition. The app implements low-level API call to retrieve the device global Ad identifier key, and implements the AdMob advertising framework. The Android app exposes objects to the WebView's Javascript. This can allow code injection or indirect access to internal objects and methods. CVE-2013-4710, CVE-2012-6636. The app contains exported components that are not protected by a permission and is actively targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

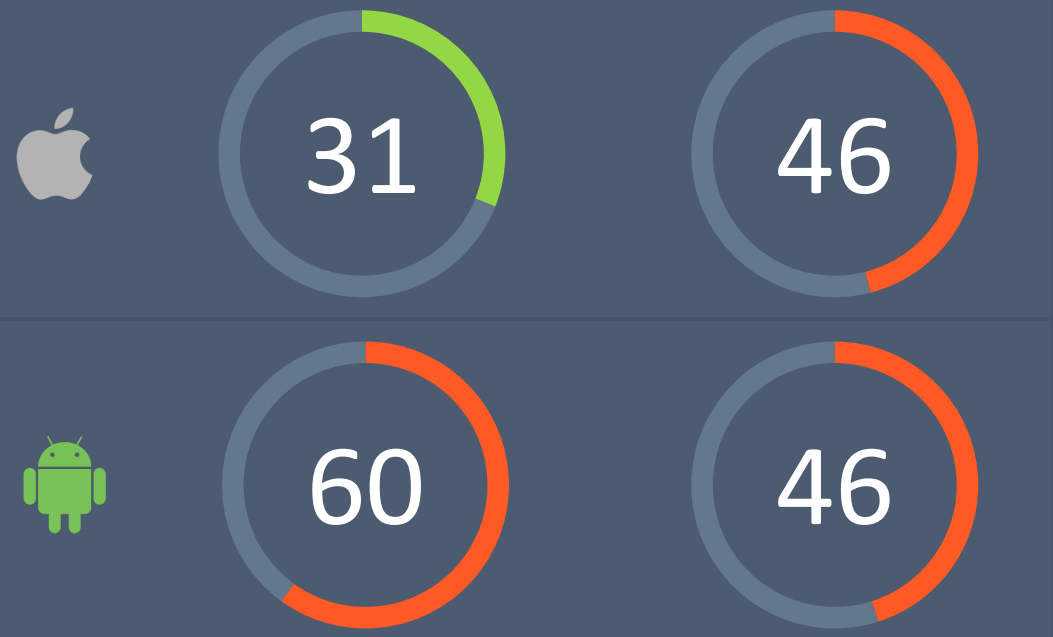
	Security	Privacy
	35	63
	75	50

ZIMPERIUM

Bank - US 78e

The iOS app uses Inter-App Audio, which allows the app to send and receive audio from other Inter-App Audio-enabled apps. The app is using a non-encrypted HTTP connection and implements low-level telephony enumeration API calls, this includes carrier, cell-tower, Cell ID, signal Strength and provider. The app is implementing the EventKit Framework which enables it to gain access to calendar data. The Android app enqueues chunks of data from various sources, such as application crashes and kernel log records. is actively targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	🟢	🟢
Insecure Data Storage	🔴	🔴
Insecure Communications	🔴	🔴
Insecure Authentication	🟢	🟢
Insufficient Cryptography	🟢	🟢
Insecure Authorization	🟢	🟢
Client Code Quality	🟢	🟢
Code Tampering	🔴	🟢
Reverse Engineering	🔴	🔴
Extraneous Functionality	🟢	🟢



ZIMPERIUM

Bank - US 67e

The iOS app attempts to retrieve a contact from the AddressBook from the depredated ABAddressBookGetPersonWithRecordID. The app retrieves the device global Ad identifier and implements AdMob advertising framework. The app is configured to allow unsecure connection to servers. It will allow cipher suites that do not support forward secrecy and does not discriminate between HTTP or HTTPS. The Android application refers to SSL/TLS hosts with self-signed certificate and other apps also use this same certificate. The app is targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

	Security	Privacy
	51	90
	35	7

ZIMPERIUM

Bank - US 71a

The app can access the device unique identifier without notification to the user. The user should receive notification of this activity to protect privacy. The Android app implements ACRA. ACRA is a library enabling an Android application to automatically post their crash reports to a Google Doc form. It is targeted to Android application developers to help them get data from their applications when they crash or behave erroneously. Ensure that the crash data does not contain privacy information. The Android app is actively targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

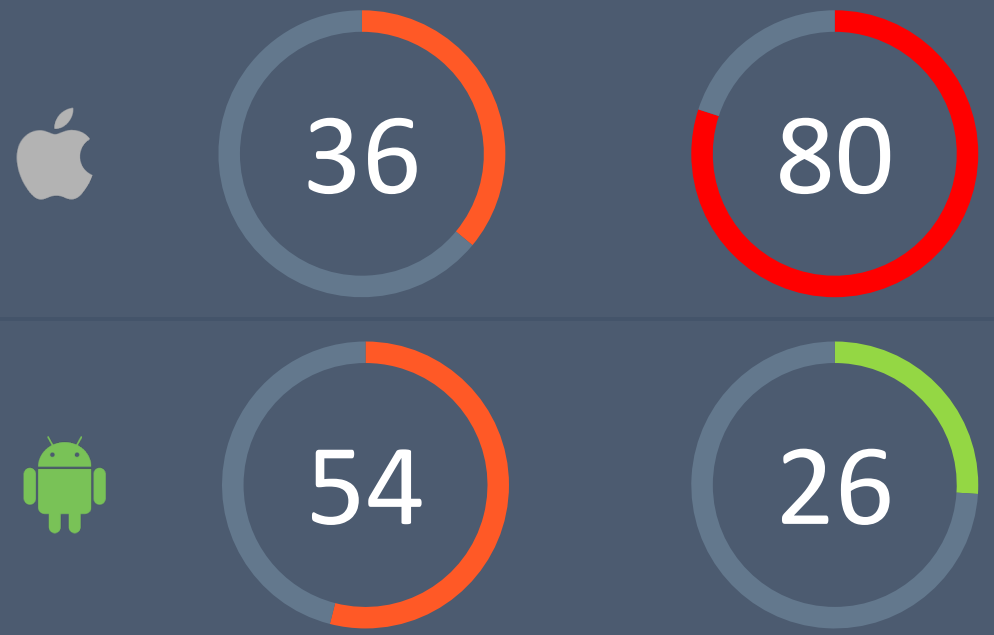
	Security	Privacy
	37	45
	75	47

ZIMPERIUM

Bank - US 74z

The iOS app attempts to retrieve a contact from the AddressBook from the depredated ABAddressBookGetPersonWithRecordID and is monitoring the iOS Pasteboard. The Android app does not check the validation of the CN of the SSL certificate. This is a critical vulnerability and allows attackers to do MITM attacks with a valid certificate without the user's knowledge. The app enables WebView to execute JavaScript code. This could potentially allow an attacker to introduce arbitrary JavaScript code to perform malicious actions. The Android app is actively targeted by the Bankbot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	🟢	🟢
Insecure Data Storage	🔴	🟢
Insecure Communications	🔴	🟢
Insecure Authentication	🟢	🔴
Insufficient Cryptography	🟢	🔴
Insecure Authorization	🟢	🟢
Client Code Quality	🟢	🟢
Code Tampering	🔴	🟢
Reverse Engineering	🔴	🔴
Extraneous Functionality	🟢	🟢



ZIMPERIUM

Bank - US 35t

The iOS app attempts to retrieve a contact from the AddressBook from the depredated ABAddressBookGetPersonWithRecordID. The app retrieves the device global Ad identifier and implements AdMob advertising framework. The app can monitor when a mobile user is in an active call and could trigger additional functionality such as recording. The Android app stores inline API keys and values and is accessing device information including serial number. The Android app uses unsecure storage data mode (WORLD_READABLE,WORLD_WRITABLE) and is actively targeted by the BankBot malware campaign.

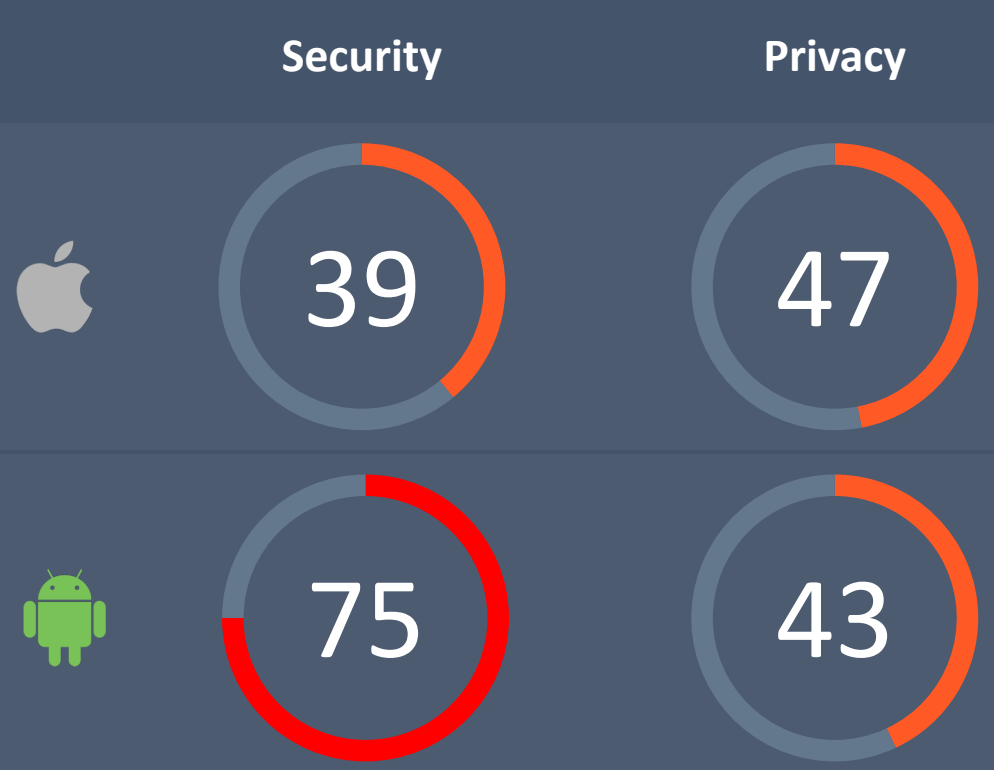
OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

	Security	Privacy
	44	76
	77	62

ZIMPERIUM

Bank - US 79t

The iOS app is using a non-encrypted HTTP connection and implements low-level telephony enumeration API calls including carrier, cell-tower, Cell ID, signal Strength and provider and to retrieve the device global Ad identifier key. The Android app uses unsecure storage data mode (WORLD_READABLE, WORLD_WRITABLE). This app can load compiled code in APK and JAR files. This can include files located in external storage and potentially on the Internet. This application can execute commands at the OS level such as launching other applications and processes and is actively targeted by the BankBot malware campaign.

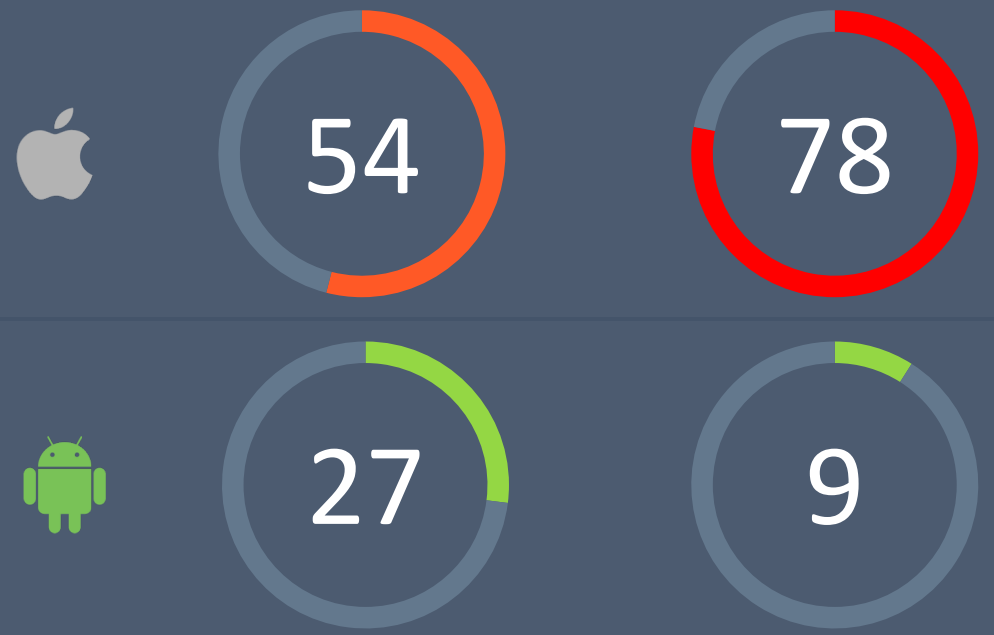


ZIMPERIUM

Bank - US 39m

The iOS app attempts to retrieve a contact from the AddressBook from the depredated ABAddressBookGetPersonWithRecordID. The app retrieves the device global Ad identifier and implements AdMob advertising framework. The app is configured to allow unsecure and unverified connection to servers with lower TLS versions. It will allow cipher suites that do not support forward secrecy and does not discriminate between HTTP or HTTPS connections. The Android app contains the Facebook SDK. The SDK is a version that is vulnerable to session hijacking. The application stores inline API keys and values. The app is targeted by the BankBot malware campaign.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●



ZIMPERIUM

Bank - US 32g

The iOS app implements low-level API call to retrieve the device global Ad identifier key and implements the AdMob advertising framework. The app source code and function names are exposed. The Android app stores inline API keys and values and is also exposed to reverse engineering. This application requests the device serial number information from the system build properties.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●

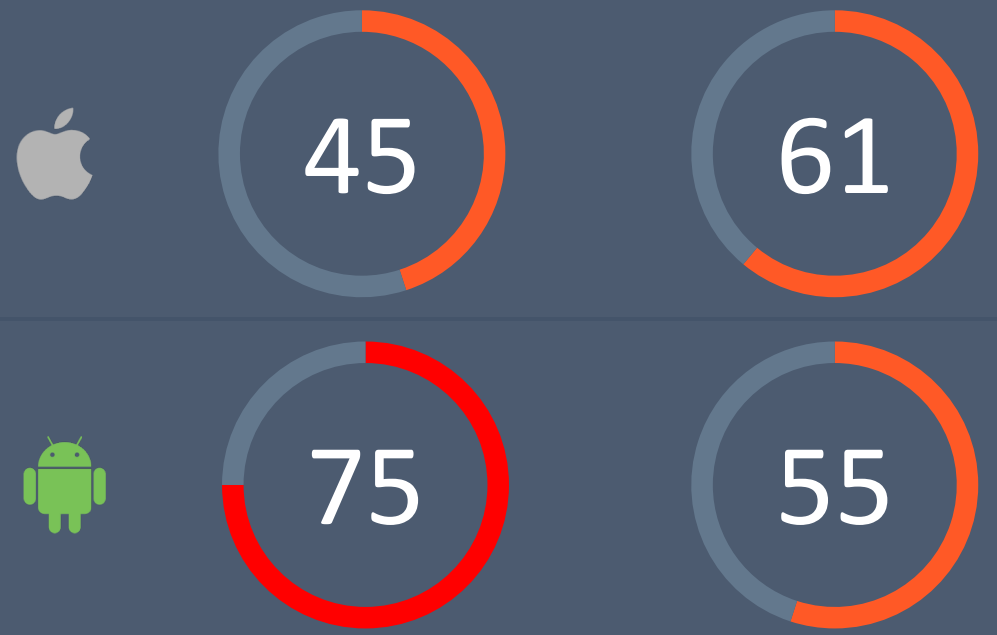
	Security	Privacy
	45	61
	20	13

ZIMPERIUM®

Bank - US 41t

The iOS app attempts to retrieve a contact from the AddressBook, photo metadata, and calendar access. The app implements an over-the air app installation. This implementation could be utilized to install a drive-by malicious apps. The Android app stores data preferences insecurely by not setting the private access mode. The behavior makes the data potentially accessible by non-authorized parties. The app contains exported components that are not protected by a permission. If a component, such as a service, is exported and not protected with strong permissions, then any application can start and bind to the service. The Android app is actively targeted by BankBot.

OWASP Mobile Top 10		
Improper Platform Usage	●	●
Insecure Data Storage	●	●
Insecure Communications	●	●
Insecure Authentication	●	●
Insufficient Cryptography	●	●
Insecure Authorization	●	●
Client Code Quality	●	●
Code Tampering	●	●
Reverse Engineering	●	●
Extraneous Functionality	●	●



ZIMPERIUM

Secure your app