Vault 101:
Bypassing Photo Vault Security on Android Mobile Devices

Disclosure Information
22/05/2017 - Vulnerabilities privately disclosed to vault app teams.
21/06/2017 – Public release

Credits and Authors
Ben Marks and Rob Manning - security.disclosures@bt.com
Introduction

Since the Android operating system has exploded into the market with over 86.8% share\(^1\) of the smartphone market in 2016Q3; its users have been able to pick and choose their choice of apps from over 2.8 million available\(^2\).

Of course, these are not all original, with multiple types of app having many different variations, formats and functionalities from many different authors. With having many different authors comes many different skill sets and coding capabilities, some of who aren’t aware of the security flaws and vulnerabilities that surround Android as a mobile operating system.

---

\(^1\) [http://www.idc.com/promo/smartphone-market-share/os](http://www.idc.com/promo/smartphone-market-share/os)

1 Outline

Photo Vaults, Locker Vaults, Hide Pictures, are all generalised names for the same type of application, these apps are designed with one main function in mind, to hide photos and videos that are of a sensitive nature from the standard Android system gallery to protect the user’s privacy should they lose or share their device. Many of these have other embedded functionality, such as App Lockers, File Vaults, or are designed to look like fake calculators, however the main functionality remains the same.

After personally using these applications for a number of months, we became aware of several security issues within the implementation of these vault applications and started investigating these data locker vaults on a wider scale. This paper investigates these data locker applications, all of which in different variations, claim to hide and protect your sensitive images and are advertised as such on their Google Play Store pages.
2 Problems

When looking at these vault applications there were a number of areas we were examining, these included the images themselves, as well as any personal data saved by the application. The areas are outlined below with a brief description and the categories.

- Depth of image security – The level of protection placed onto the user’s image when it was stored in the vault app.
  - No Security – The image was copied into a folder that was hidden on the device using a dot(.) prefix.
  - Trivial Security – As above and the image extension changed or removed.
  - Basic Security – As above and/or the simple modification of a limited number of bytes within the image header.

- Credential storage – How the application stores sensitive login information via shared_prefs files.
  - Plaintext
  - Weak or easily reversed method – Base64, MD5 or SHA1
  - Encrypted or strongly hashed

- App data backups – Whether backups allowed data exfiltration.
  - The Android system allows all app created files, such as app preferences, databases, files and cache to be backed up into an Android Backup file and pulled off the device. This functionality is enabled if android:allowBackup attribute is missing, or explicitly set to true and requires device access to confirm.

- Exported Activities – Certain Android Activities, if exported or contain intent filters, can bypass authentication screens or can allow access to app functionality.

- Clear data setting – Clearing app data via the Settings menu removes saved login credentials and app configuration so acts like a new app install, thus allowing a new password/PIN to be set at next launch, previously hidden files are unaffected and remain in the app.

- Sensitive user information leakage – Any further vulnerabilities that allowed app compromise.
  - Recovery email for password/PIN reset
  - Recovery of security question/answer
  - Lock pattern hash obtainable
  - SQL injection into the database Content Providers
3 Analysis

The aim of this investigation was to determine how secure these applications were, and what, if any, protections were placed on the user images and to what level. This research was conducted due to putting user files and data at risk and so were checked to ensure sufficient protection was in place or not, in comparison to what the applications advertised on the Google Play Store website.

For this analysis ten apps were investigated, which are outlined below. Investigation in this sense attributes itself to both manual and dynamic on-device examination using tools and techniques to identify if any of the outlined problems were present in any of the applications. No brute force techniques were necessary to obtain the information within this paper, or to compromise the apps in question. Similarly, no malicious traffic was sent to the application authors’ server infrastructure or website; the only traffic generated were recovery emails for forgotten password functionality that already exists within the apps.

Table 1 – App information and statistics (Information correct at time of analysis – May 2017)

<table>
<thead>
<tr>
<th>Name</th>
<th>Author</th>
<th>Version</th>
<th>App Store URL</th>
<th>Downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gallery Vault - Hide Pictures</td>
<td>ThinkYeah Mobile</td>
<td>2.9.17</td>
<td><a href="https://play.google.com/store/apps/details?id=com.thinkyeah.galleryvault">https://play.google.com/store/apps/details?id=com.thinkyeah.galleryvault</a></td>
<td>10,000,000 - 50,000,000</td>
</tr>
<tr>
<td>AppLock</td>
<td>DoMobile Lab</td>
<td>2.22.2</td>
<td><a href="https://play.google.com/store/apps/details?id=com.domobile.applock">https://play.google.com/store/apps/details?id=com.domobile.applock</a></td>
<td>100,000,000 - 500,000,000</td>
</tr>
<tr>
<td>Hide Pictures &amp; Videos - Vaulty</td>
<td>Squid Tooth LLC</td>
<td>4.3.3</td>
<td><a href="https://play.google.com/store/apps/details?id=com.theronrogers.vaultyfree">https://play.google.com/store/apps/details?id=com.theronrogers.vaultyfree</a></td>
<td>5,000,000 - 10,000,000</td>
</tr>
<tr>
<td>Private Photo, Video Locker</td>
<td>Kohinoor Apps</td>
<td>28.0</td>
<td><a href="https://play.google.com/store/apps/details?id=secret.hide.calculator">https://play.google.com/store/apps/details?id=secret.hide.calculator</a></td>
<td>1,000,000 - 5,000,000</td>
</tr>
<tr>
<td>Calculator Vault- Gallery Lock</td>
<td>Sure Applications</td>
<td>9.3</td>
<td><a href="https://play.google.com/store/apps/details?id=com.calculator.vault">https://play.google.com/store/apps/details?id=com.calculator.vault</a></td>
<td>1,000,000 - 5,000,000</td>
</tr>
<tr>
<td>PicLock - Hide Photos &amp; Videos</td>
<td>XCS Technologies</td>
<td>2.0</td>
<td><a href="https://play.google.com/store/apps/details?id=com.xcs.piclock">https://play.google.com/store/apps/details?id=com.xcs.piclock</a></td>
<td>5,000,000 - 10,000,000</td>
</tr>
<tr>
<td>Hide Photos, Video-Hide it Pro</td>
<td>ANUJ TENANI</td>
<td>5.9</td>
<td><a href="https://play.google.com/store/apps/details?id=com.hideitpro">https://play.google.com/store/apps/details?id=com.hideitpro</a></td>
<td>10,000,000 - 50,000,000</td>
</tr>
<tr>
<td>All-in-One Vault</td>
<td>NewSoftwares.net</td>
<td>1.2</td>
<td><a href="https://play.google.com/store/apps/details?id=net.newsoftwares.folderlockadvanced">https://play.google.com/store/apps/details?id=net.newsoftwares.folderlockadvanced</a></td>
<td>500,000 - 1,000,000</td>
</tr>
<tr>
<td>PhotoSafe</td>
<td>Slickdroid</td>
<td>2.0.4</td>
<td><a href="https://play.google.com/store/apps/details?id=com.slickdroid.vaultypro">https://play.google.com/store/apps/details?id=com.slickdroid.vaultypro</a></td>
<td>1,000,000 - 5,000,000</td>
</tr>
</tbody>
</table>

3 https://play.google.com/store/apps
These applications were selected to provide a good functional reflection of the security quality of these vault apps using the numbers of downloads as a factor (ranging from 500,000 – 500 million) which allowed a good overall picture of the security in place on apps ranging from unknown to popular.

The issues that were found, and summarised in Table 2 allowed for complete compromise of the application in 100% of cases. The term ‘compromise’ in this situation refers to, breaking the role of the application, which is to protect photos in a safe and secure manner. That is, for every application that was investigated, the protections in place on the user photos were able to be reverse engineered to obtain the original photo.

Table 2 - App vulnerabilities

<table>
<thead>
<tr>
<th>App</th>
<th>Image Protection</th>
<th>Credential Storage</th>
<th>App Backups</th>
<th>Exported Activities</th>
<th>Clear data setting</th>
<th>Sensitive user information leakage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gallery Vault - Hide Pictures</td>
<td>Basic</td>
<td>Encrypted</td>
<td></td>
<td>Yes</td>
<td></td>
<td>Plaintext email recovery Debug log feature</td>
</tr>
<tr>
<td>AppLock</td>
<td>Trivial</td>
<td>Weak</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Plaintext email recovery Pattern hash Content Provider injection</td>
</tr>
<tr>
<td>Hide Pictures &amp; Videos - Vaulty</td>
<td>Basic</td>
<td>Weak (Root req)</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Plaintext security questions (Root req)</td>
</tr>
<tr>
<td>Private Photo, Video Locker</td>
<td>None</td>
<td>Plaintext</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Plaintext email recovery Pattern hash</td>
</tr>
<tr>
<td>Calculator Vault- Gallery Lock</td>
<td>None</td>
<td>Plaintext</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Plaintext security questions Plaintext email recovery Pattern hash</td>
</tr>
<tr>
<td>PicLock - Hide Photos &amp; Videos</td>
<td>None</td>
<td>Plaintext</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Plaintext email recovery</td>
</tr>
<tr>
<td>Hide Photos, Video-Hide it Pro</td>
<td>Trivial</td>
<td>Plaintext</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Plaintext email recovery</td>
</tr>
<tr>
<td>All-in-One Vault</td>
<td>Basic</td>
<td>Plaintext</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Plaintext email recovery</td>
</tr>
</tbody>
</table>
Not noted in the above table, 80% of the applications stored the ‘secured’ user data in the Android external storage location that can be read by any app with the appropriate permissions. Any user with physical access to the device can also view the contents of the external storage location by simply using a File Manager application which are readily and legitimately available on the Play Store.

These applications use a hardcoded storage path for their protected directories. Due to being hardcoded this will remain the same for all application installs across all user devices. With this knowledge of hardcoded paths, and how to reverse engineer the protected images, a malicious party could create an Android app that can search the external storage locations for these paths, and once found, upload all contents to an attacker-controlled server. An attack like this would bypass the need to have physical access to the victim’s device. The malicious app would masquerade as a legitimate app, such as a game or similar that would be installed by the victim themselves, unaware of the consequences.

This attack vector would run on both stock and rooted devices and would be practically undetectable once installed on the device. Subsequently this app would only need the android.permission.READ_EXTERNAL_STORAGE permission* to be granted by the user at installation time, which would seem normal for a game to request due to storing game files in an accessible location.

*Android.permission.INTERNET permission is also required in the manifest, however this is a ‘Normal’ permission and thus granted by the system automatically without notifying the user.4

This attack vector can be seen as a proof of concept in Image 1 and 2. In a real application all of the code for the below actions would be run automatically in the background and not visible to the user.

The target attack surface for this type of attack is incredibly large. Using the upper and lower estimates of the Android Play Store downloads count, and excluding the two apps that did not store the data on the external storage, this attack vector could impact a lower estimate of 141,500,000 users and an upper estimate of 676,000,000 users for only eight of these vault type apps. This, however, does not account for multiple downloads from the same user on different devices, nor does it allow for users who are no longer actively using the app.

However, there are theoretically hundreds of this type of app on the Android Play Store significantly increasing the attack vector and at risk users.

Despite these two estimation stipulations the potential impact of malware of this type would generate a huge amount of private user images, videos, files, notes, and in some cases possibly credit card details, login details, bank details or other sensitive identifying information depending on the application functionality and what was stored.

The only mitigating factor is that Google’s Play Store protection mechanism, Bouncer, may be able to detect the intentions of the malware and thus decline it from appearing on the Play Store; however, Bouncer has previously been bypassed by a number of different methods so
should not be seen as a one-stop security solution\textsuperscript{5,6}. Equally, the malicious app could be uploaded onto other app stores, or app repositories which have no protection in place at all.

\textsuperscript{5} http://www.theinquirer.net/inquirer/news/2426864/poisoned-iq-test-app-evades-google-bouncer-to-infect-up-to-one-million-android-users
\textsuperscript{6} https://www.helpnetsecurity.com/2015/10/09/fake-android-apps-bypass-google-play-store-security
4 Solutions

These findings are all easily remedied and should not require much extra development time to patch the vulnerabilities found. When handling user data in any context, security should be implemented into the development phase.

In the context of these vault applications, the following recommendations should be borne in mind.

- Ensure adequate protection on sensitive user files using a strong encryption or hashing mechanism, which are available within the Android development environment, on the bytes that make up the image file.
  - Further to this, storing user files within the private app directory `/data/data/app_name/files` in which only the running app has access to is more secure than the external storage location.

- Backup functionality should be disabled, which will remove the ability for users to backup app data files, databases and any files created by the app in the `/data/data/app_name` directory, see above, that expose user login information and recovery data.

- Any data stored relating to the application, login information, security, or email should be hashed using a strong and secure method before being stored in the app’s `shared_preferences` file. Android supports SHA256 and higher as secure hashing functions. Base64, MD5 and SHA1 should be avoided as they are considered weak and easily reversed to obtain the original value.

- All internal app components, especially Activities, should be explicitly set not to be exported in the manifest. If the components need to be exported for operability, adequate permissions should be put in place to secure against unintended apps creating Intent calls.

- Make use of the `manageSpaceActivity` declaration to avoid a user with physical device access from clearing app settings and configuration by utilising the Android system settings app manager.

Depending on the app in question some of the above solutions may affect usability, however implementing other proposed solutions should mitigate shortfallings.
5 Summary

This paper has investigated the security of Android vault style applications to gauge the strength and depth of how secure users’ private images and videos really were. It was found that all ten applications examined did not appropriately secure user images, or sensitive data, allowing for the original images to be obtained from the app in every scenario via a variety of different methods.

An example of a proof of concept malware app and associated attack surface was detailed demonstrating that this would be a severe issue if it was exploited in the wild, potentially leading to the leakage of millions of user’s private data.

With this in mind, we, the security researchers and on behalf of the millions of Android users hope the development of Android applications still moves towards a more secure practise of coding and development.