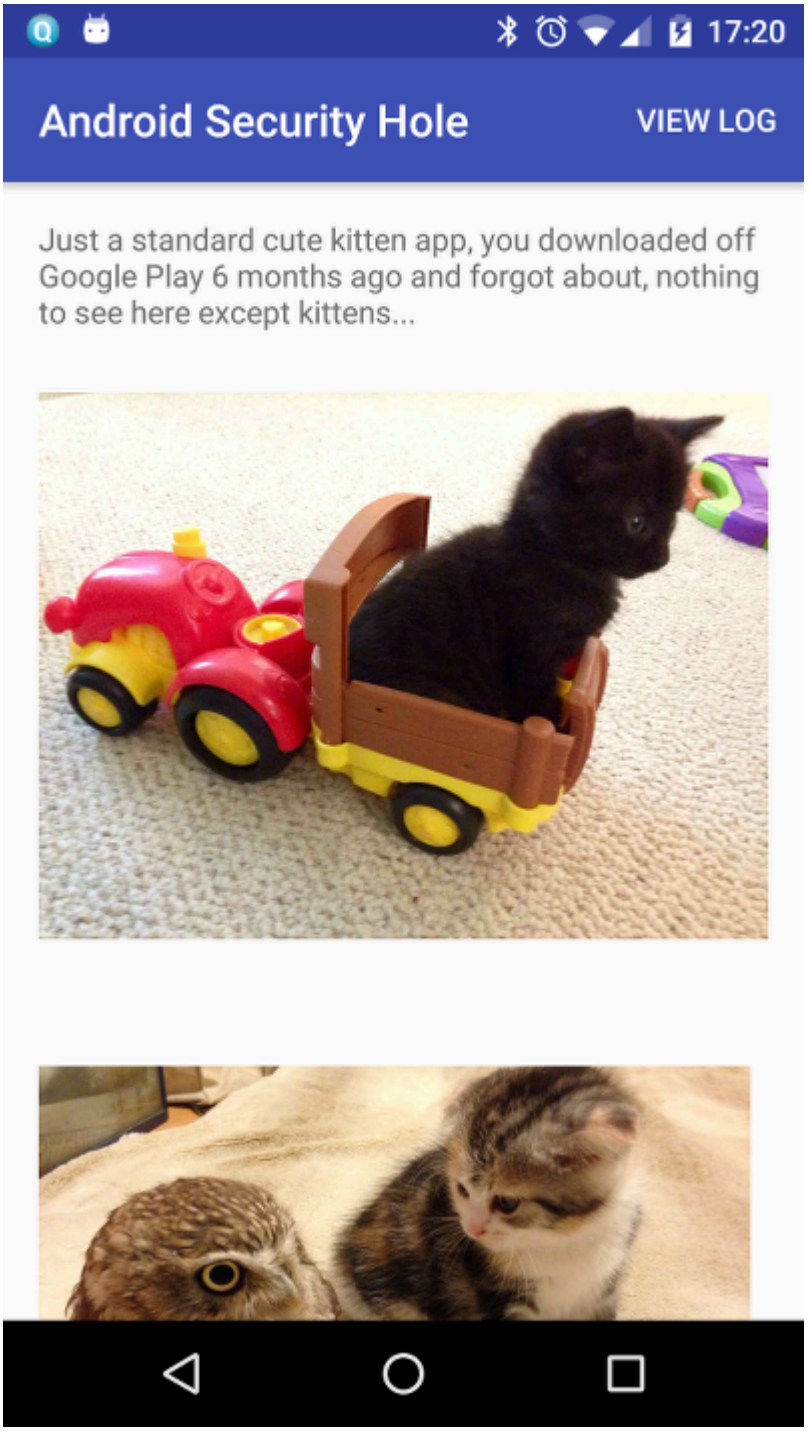
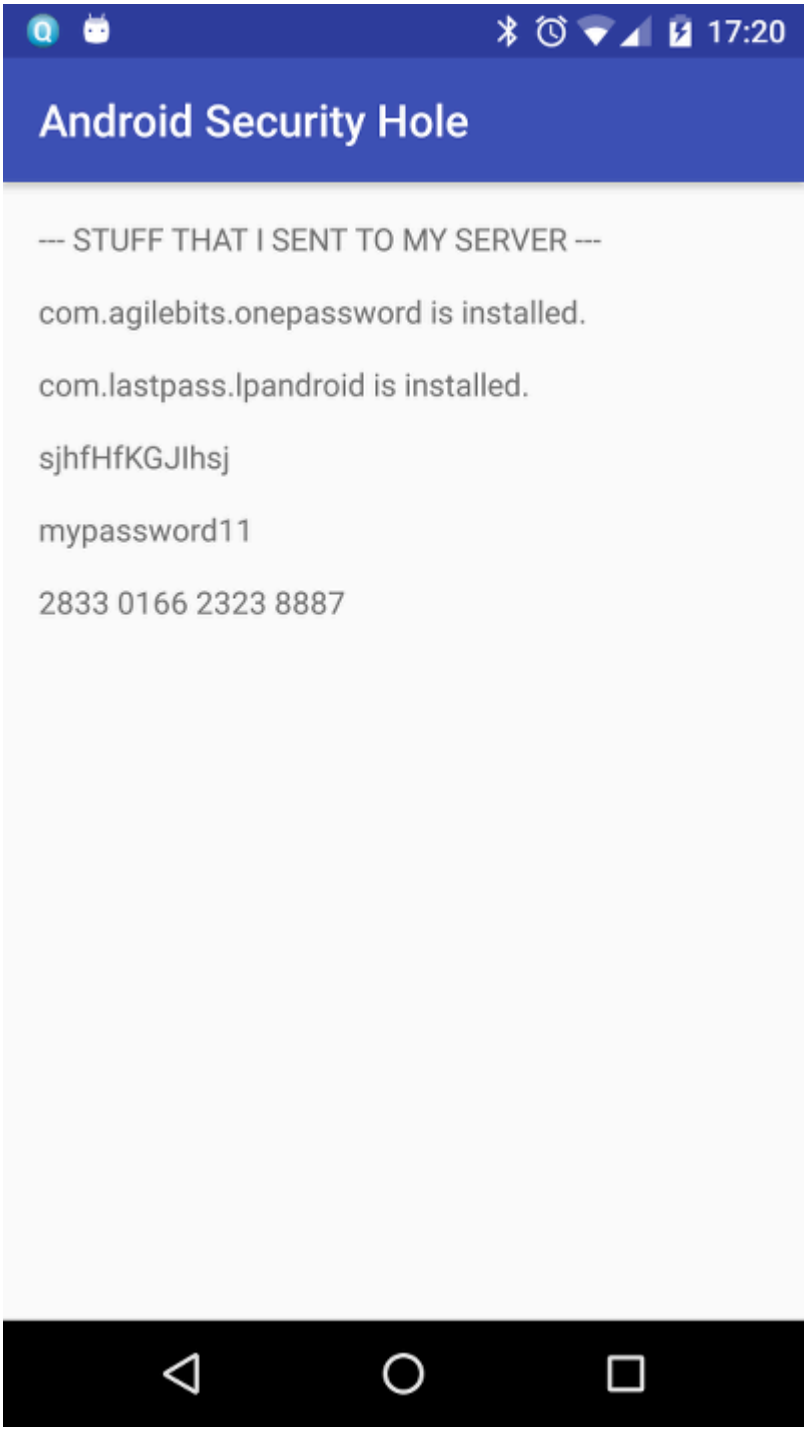
GitHub - grepx/android-clipboard-security: A project demonstrating a security hole In Android's API that allows any installed application to listen to changes to the clipboard (listen to everything that you copy and paste).

By grepx

Archived: 2026-04-05 16:15:23 UTC

Android Clipboard Security Hole Demonstration App


17:20
Android Security Hole
VIEW LOG
Just a standard cute kitten app, you downloaded off Google Play 6 months ago and forgot about, nothing to see here except kittens...



This project demonstrates a security hole In Android's API that allows any installed application to listen to changes to the clipboard (listen to everything that you copy and paste).

If it isn't clear why this is a genuine security hole consider that sensitive information is copied and pasted all of the time:

- Many less security conscious users will copy and paste credit card numbers, bank details and passwords from other apps such as notes.
- More security conscious users will often use password managers such as LastPass and 1Password and will often copy and paste passwords from the password manager app. How to protect against this attack is

discussed below.
- Personally, I've known about this hole for some time and have always been careful, but recently I entered my credit card number into the wrong field on a form and quickly copied and pasted it across to the correct field before realising what I was doing.

There are probably many more examples that can be given of reasons why sensitive information can pass through the clipboard.

How it works

The important code for attack is contained within `ClipboardWatcherService.java` and the real meat of the attack is this line of code:

```
((ClipboardManager) getSystemService(CLIPBOARD_SERVICE)).addPrimaryClipChangedListener(listener)
```

`ClipboardManager` is a system service that allows you to register a listener for when the clipboard changes. **There is no permission required to access this service despite the fact that exposing it is far more dangerous than many of the things on Android that require permission to be granted.**

It would be difficult to accurately detect whether an app is using this API. The `CLIPBOARD_SERVICE` string could be generated at runtime, and the references to `ClipboardManager` and `ClipboardManager.OnPrimaryClipChangedListener` could be constructed using reflection at runtime. Besides, its currently a valid API that has some valid use cases in certain apps such as looking up words in a dictionary or providing additional clipboard functionality.

When combined with the internet access via the `android.permission.INTERNET` permission attackers can transmit your clipboard to a remote server. Almost every app has a good reason to request this permission and it won't appear suspicious.

Deploying the attack

The attack is designed to be incorporated into a simple malicious app that is then distributed via the Play store or elsewhere. There are over 2.4 million apps in the Play store, many of which are very simple apps that offer funny pics, food recipes, wallpapers etc. They are cheap to make and require very little effort, template apps can be bought online. Google is quite good at removing malware from the store and the vast majority of apps are not malicious. However, the risk remains and I believe that very few users realise how much access a seemingly innocent app will have to their device, even without complex attack vectors or suspicious permissions.

From my own experience watching user behaviour for my own apps on the Play store, many users keep apps installed that they haven't used for *years*. They forget they have them installed, and never bother to uninstall them.

Improving the basic attack

To increase the potential of the attack I combined it with an alarm broadcast receiver that will run once a day and make sure that the `ClipboardWatcherService` is still running (the user could have killed it or some battery manager application might have done). This does **not** require any additional permissions.

In a real implementation I would probably batch the harvested clipboard data together and also use this alarm to upload it in the middle of the night when detection is less likely. The device is also likely be plugged in and charging so it won't show up in the battery usage statistics, and the device is also likely connected to wifi so it won't show up in the data usage statistics (more advanced optimisations are possible for checking both of those).

Start on device bootup

I also added the `android.permission.RECEIVE_BOOT_COMPLETED` permission, so that I can automatically start the `ClipboardWatcherService` when the device boots up. Without this, I would have to wait for the user to open my app again after a system restart before I can start the service, and since I'm hoping the user has forgotten about my app this will never happen.

This permission will not be listed by the Play store during install, since it is not considered a "special permission". The only way a paranoid user could detect that a random recipe/wallpaper pic app is suspiciously requesting this permission is by scrolling down to the bottom of the Play store profile page and clicking "Permission Details" to see the full set of permissions (nobody does this).

With this permission I can capitalise for many years harvesting clipboard data from users who have forgotten that my app is installed or haven't bothered to uninstall it.

Log which apps are installed on the device

Android has a few ways to learn about which apps are installed on the device, none of them require any additional permissions. The simplest is to just use `PackageManager`:

```
 final PackageManager pm = getPackageManager();
List<ApplicationInfo> packages = pm.getInstalledApplications(PackageManager.GET_META_DATA);
for (ApplicationInfo packageInfo : packages) {
if (packageInfo.packageName.equals("com.agilebits.onepassword") |
         packageInfo.packageName.equals("com.lastpass.lpandroid")) {
                  log.append(packageInfo.packageName);
                  log.append(" is installed.\n\n");
         }
}
```

Here I have narrowed the search to just see if 1Password or LastPass is installed, but I could target bitcoin or banking related apps or just send the entire list to a remote server.

I could also have executed `pm list packages` in a background shell to get this information.

The problem with password managers

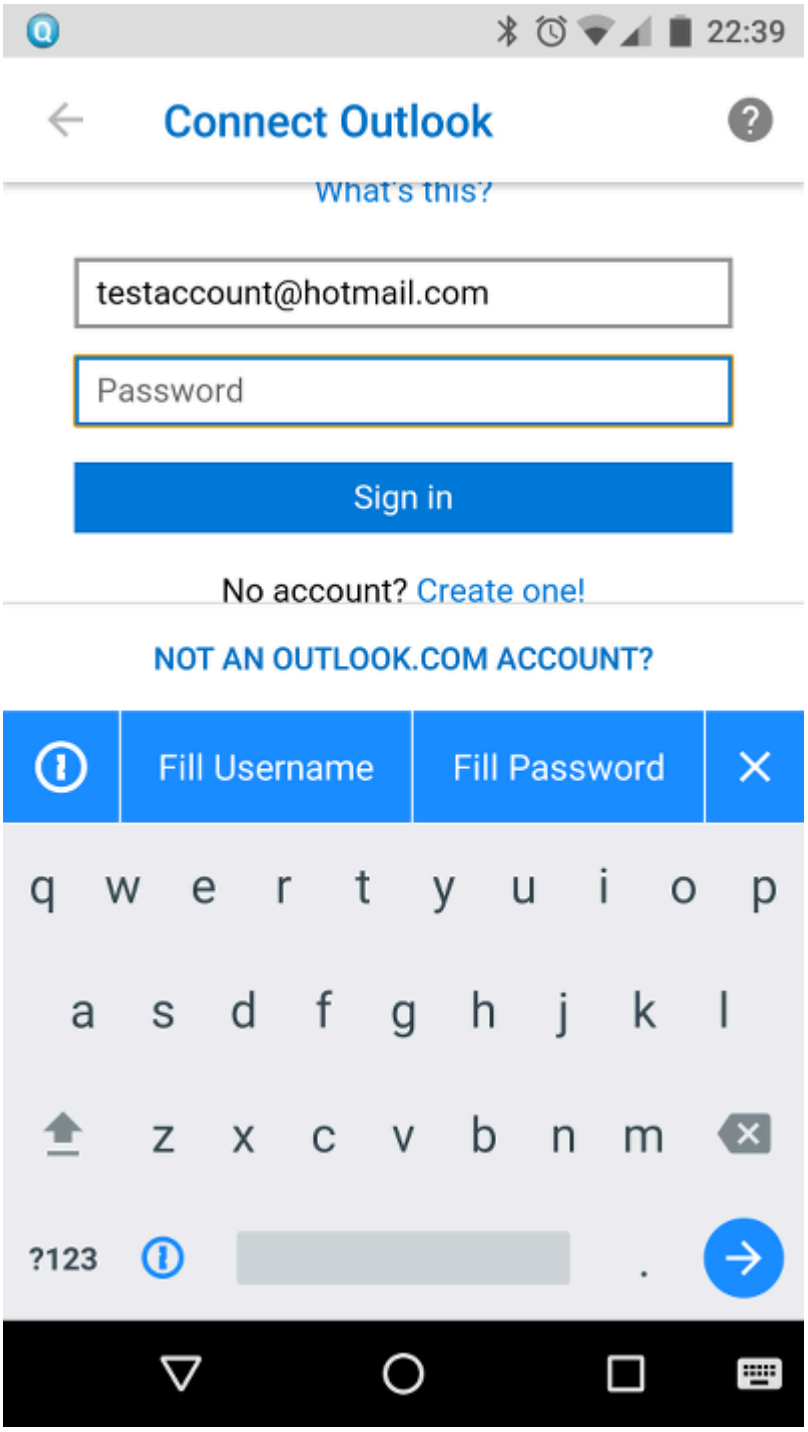
Password managers are particularly vulnerable to this attack since many encourage the user to copy and paste passwords out of the app and into the password box for the app in question and many users will use the app like this, trusting the clipboard to be secure.

Both 1Password and LastPass include the ability to copy and paste passwords directly out of the app (as well as credit card numbers etc.). In my opinion, this option should be removed from both password managers if they really care about their user's security. The users who are not savvy enough to know better are probably the same ones who install random apps off the Play store and keep them installed for years.

Thankfully, both also have ways to avoid using the clipboard and they should educate their users to only use these methods.

Both are essentially identical to configure:

1. Enable auto-fill by going to Settings → Accessibility → LastPass / 1Password. This will enable the password manager to quickly autofill the forms in many apps.
2. Enable the password manager's custom keyboard by going to Language & input → Current keyboard → Choose keyboards and switching it on. To enter the password, you need to switch from your usual keyboard to the custom keyboard each time. It will allow you to insert the password without going via the clipboard. This is used when auto-fill fails, which is quite often. **It's not very convenient, but it's the only secure way** (except for typing it).



With these 2 methods available, **you should never copy and paste passwords or other important information via the clipboard on Android.**

Conclusion

In my opinion, clipboard manager access in Android should at least require the user to give permission, and it should be a "special permission" that's clearly listed during installation on the Play store. Also, the `android.permission.RECEIVE_BOOT_COMPLETED` should be bumped up to a "special permission" that's listed during installation. Perhaps an additional permission should be added for requesting to see which apps are

installed on the device. The permission system is still abused by many apps though, and users are increasingly blind to permission requests, so this might not be enough.

The next level of protection would require apps to show a persistent notification while they are listening to the clipboard. All of the apps that have a valid use case for listening to the clipboard such as dictionary lookups and clipboard helpers do this anyway.
