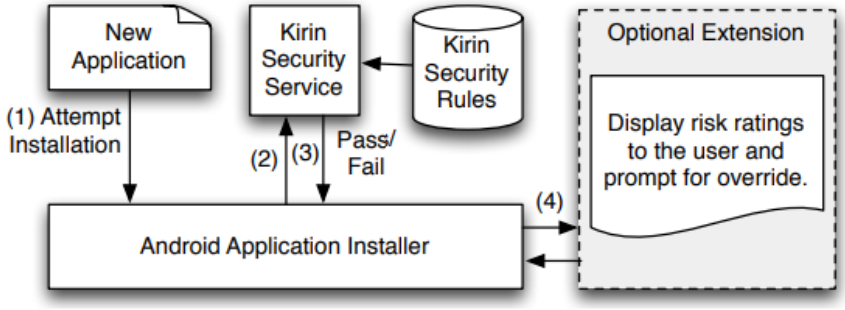
Second assignment

Kirin structure

Kirin is based on security rules, and the rules represent templates of undesirable security properties. It also provides a means of defining dangerous combinations and automating analysis at install time, by looking at the permission labels that the application asks for.

If we look at these properties alone we cannot indicate malicious potential, but if we look at the specific combinations that Kirin is searching for, we could understand the risks that applications with these permission labels have.

For example, an application that can start on boot, read geographic locations, and access the Internet is potentially a tracker installed as premeditated spyware.




In this figure we can see that the installer extracts the permissions from the manifest, then passes those to the Kirin security service to evaluate the risks. If it cannot pass all the rules it could either reject the installation or it could warn the user.

The paper " On Lightweight Mobile Phone Application Certification" (Link) sets Kirin's rules and explains each one of them as follows(4.2.X in the paper):

1. An application must not have the SET_DEBUG_APP permission label. It allows an application to turn on debugging for another application.
2. An application must not have PHONE_STATE, RECORD_AUDIO, and INTERNET permission labels. It protects against the voice call eavesdropper.
3. An application must not have PROCESS_OUTGOING_CALL, RECORD _AUDIO, and INTERNET permission labels. It protects against the voice call eavesdropper.
4. An application must not have ACCESS_FINE_LOCATION, INTERNET, and RECEIVE_BOOT_COMPLETE permission labels. It protects against a location tracker.
5. An application must not have ACCESS_COARSE_LOCATION, INTERNET, and RECEIVE_BOOT_COMPLETE permission labels. It protects against a location tracker
6. An application must not have RECEIVE_SMS and WRITE_SMS permission labels. protects against malware hiding or otherwise tampering with incoming SMS messages. (also now days can block 2FA hijacking).

7. An application must not have SEND_SMS and WRITE_SMS permission labels. It mitigates mobile bots sending SMS spam.
8. An application must not have INSTALL_SHORTCUT and UNINSTALL_ SHORTCUT permission labels. It ensures that a third-party application cannot have both. it could  redirect the shortcuts for frequently used applications to a malicious one.
9. An application must not have the SET_PREFERRED_APPLICATION permission label and receive Intents for the CALL action string. it prevents malware from replacing the default voice call dialer application without the user's knowledge.

## Improvement and Additions

We could add more rules to the nine suggested by the paper:

1. Access to the device's storage and ability to modify system settings (permission.WRITE_SETTINGS). This could allow an app to delete or modify files on the device, or even change the device's settings without the user's knowledge.
2. Access to the device's camera and microphone, as well as the ability to access the internet and send or receive data. This could allow an app to record video or audio without the user's knowledge and potentially send it to third parties.
3. Access to the device's calendar and ability to access the internet and send or receive data. This could allow an app to track and potentially modify the user's schedule without their consent.
4. Access to device's SMS (permission.READ_SMS) and the ability to access the internet(INTERNET) by sending data could lead to 2FA hijacking and reading personal data.
5. Access to install new applications (permission.INSTALL_PACKAGES) this could lead to malicious applications installage without the user's knowledge or acceptance.
6. Access to call log of the device, mainly to read(READ_CALL_LOG) and write(WRITE_CALL_LOG), this could lead to the changing of the users call log without the user's knowledge.

How we got to the ideas suggested is by searching the internet for malicious combinations in android permission website (Link). Also, we opened random malicious APK files that we got, and by reading the manifest's xml file, then thinking about the combinations that we saw in the permissions.

Another option to improvements of Kirin is to make it an independent program, that gets an APK or a folder and returns an answer about its content. That option will remove the need to use another program like APKTOOL that could cause a problem in the supply chain.