Detecting Android Malware with Supervised Machine Learning

Protecting Android users is a persistent and constantly adapting challenge as over a half million new instances of malware are detected daily (Palatty, 2023). The review and approval for device applications often includes both a statistical evaluation and human evaluation (*About Google Play Reviews - Google Play Help*, n.d.). Modeling through supervised machine learning may allow for or improve automation used in this evaluation. Feature importance may also assist in informing human-reviewer practices to better protect users as well as informing internal warning systems of user devices. To accomplish this, the analysis will seek to answer:

1.Can application features be used to accurately predict malicious applications?

2.What application features or permissions put users at the highest risk?

Background/History

For Android device users, the only barrier between their sensitive information and application malware is their device's permissions system. Users may grant or deny permissions as they wish and the risks of granting permissions are mitigated a couple of ways. Primarily this is done through a devices internal security software which may warn a user of the risky permissions (Ehsan et al., 2022). Because users may assume these permissions are what allow applications to operate properly, this decision ultimately comes down to trust. Beyond warnings, there is the "vetting" of applications through platform markets like Google Play, but malicious applications still manage to get approved (Newman, 2017).

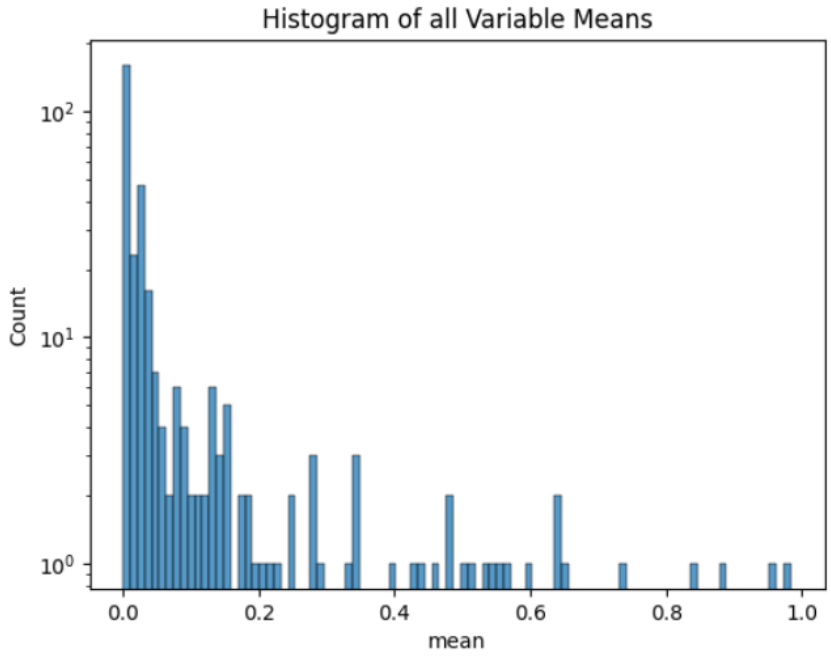
Data Explanation

My dataset was uploaded from kaggle.com and is licensed under MIT. It contains data on over 4 thousand Android applications as rows, each with over 300 columns of attributes including a target label of “benign” or “malware”. All features are binary expressions of 0 or 1 to indicate the presence of each attribute. Nearly all the attributes can be categorized into one of the following groups: permissions, system functions, security, communication, data access, application lifecycle, and device controls. A description for each group is detailed in Appendix Table 1 and a complete list of all columns are documented in Appendix Table 2.

Data Preparation

The data for this analysis was clean and required no major transformations to begin examining. A review of summary statistics across variables revealed no missing values. I chose to plot a histogram of all variables means, which in the case of binary data, offered me a quick way to see the distribution of all variables. Most application features were present 20% or less of the time across applications. Forty-three features in total were found with 0 variance which were

Figure 1



removed from the dataset prior to multivariate analysis.

Due to the large volume of features, I chose to run a simple point-biserial correlation between my target variable and each remaining feature. I filtered for the strongest coefficient results that also had a 0.05 or less p-value. The two variables with the strongest relationship to my target were permissions called “READ_PHONE_STATE” with a coefficient of 0.76 and “RECEIVE_BOOT_COMPLETED” with a 0.59 coefficient. The first allows an application to know the users phone number, network information, and the status of current calls. The second allows an application to know when a user boots their device (Alshehri et al., 2017).

Figure 2

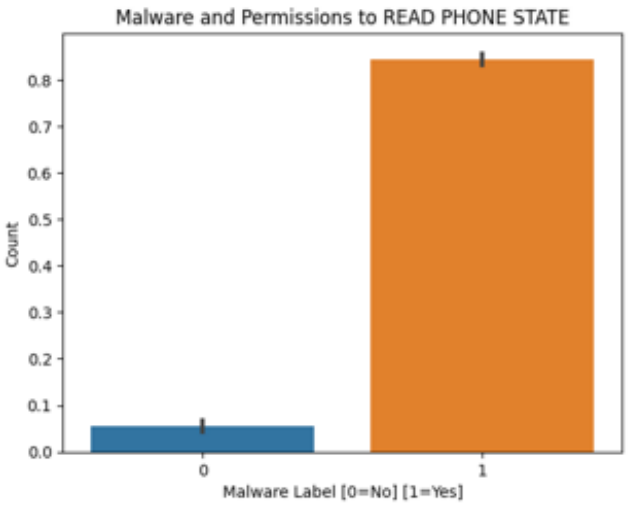
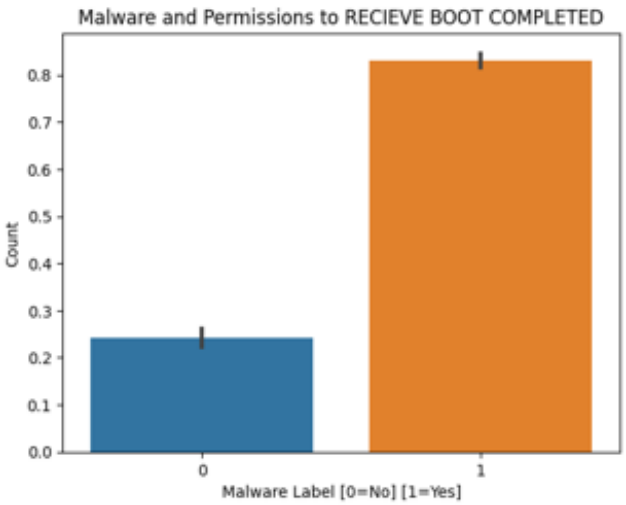


Figure3



Methods

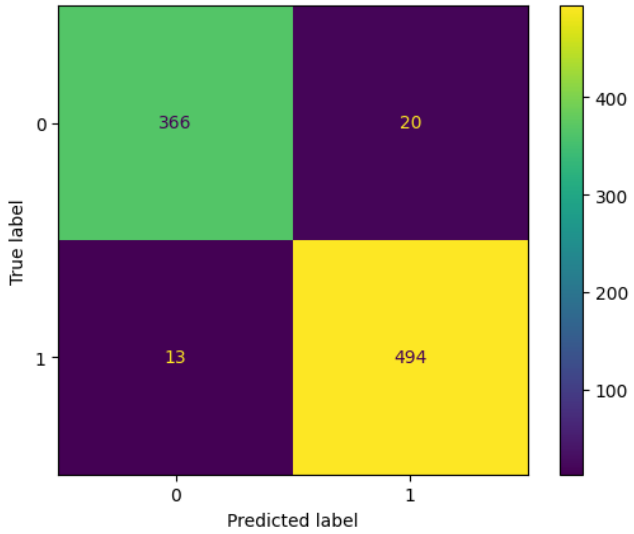
Prior to modeling, the data was split at an 80:20 ratio for training and testing, and a null accuracy score was calculated at 0.567 as a baseline to improve upon. A tree-based model was desirable for this project to manage the large feature count. A gradient boosting classifier was chosen for this project for its ability to learn from wrong predictions. More importantly, it was chosen for its ability to learn from mistakes without being prone to overfitting. For evaluation, accuracy score

and confusion matrix were used. Rates for true positive, true negatives, false positives, and false negatives were also calculated. After modeling, the SHAP library was imported to create visualizations to interpret how this model was influenced by the data features.

Analysis

The gradient boosting classifier was successful with over 96% accuracy with my test dataset which was much higher than the null accuracy baseline at 57%. Given the importance of not

Figure 4



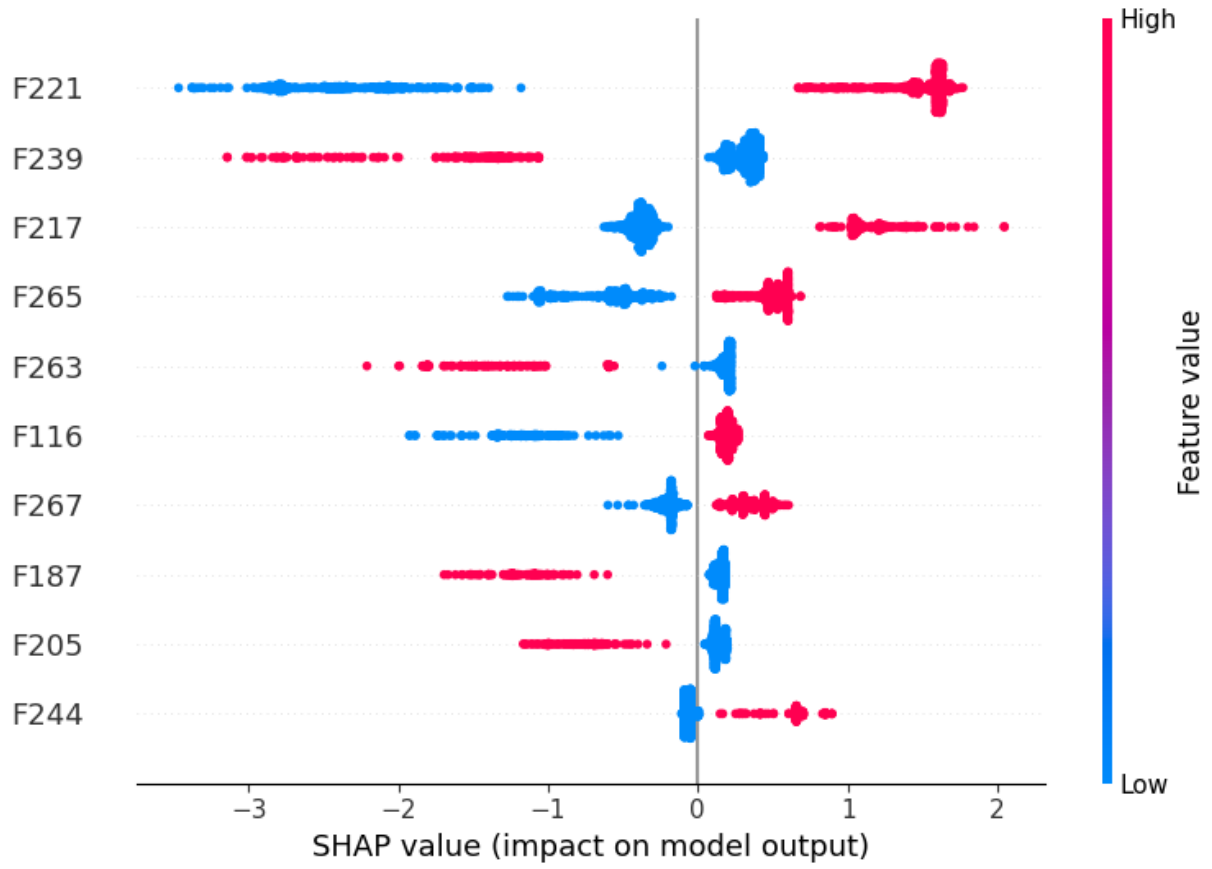
allowing malware to go undetected, it was important to view the results in a confusion matrix. With a test sample of 893, it was assuring to see only 23 missed predictions. A false negative rate of 2% was calculated along with a false positive rate of 5%.

Figure 5: Table for True Positive Rate, True Negative Rate, False Positive Rate, and False Negative Rate

Metric	Result
TPR	0.974359
TNR	0.948187
FPR	0.051813
FNR	0.025641

With this high accuracy, the feature SHAP values were calculated and visualized to see what permissions had the largest impact on the model. Below we can see a summary plot of feature SHAP values, which are listed in descending importance, and limited to 10 features.

Figure 6



The top 3 features here F221,F239, and F217 refer to permissions for READ_PHONE STATE, RECEIVE, and INSTALL_SHORTCUT. Specifically, we can see that applications with permissions for READ_PHONE_STATE and INSTALL_SHORTCUT influenced the model to predict malware. Meanwhile permission to RECEIVE influenced the model to predict benign applications. The permission RECEIVE in this context is related specifically to Google Cloud messaging or push notifications.

Conclusion

Overall, I would consider this model highly effective for predicting Android application malware based on permissions with 96% accuracy. False negatives were of high concern for this project and the model performed well with a 2% false negative rate. The SHAP visualizations also proved capable of revealing how features influenced the model which could easily lend to the review process. Permissions for READ_PHONE_STATE and RECEIVE_BOOT_COMPLETED were most influential on our model's prediction for malware. Inversely, the permission for RECEIVE was the strongest feature related to benign applications.

Assumptions & Limitations

The largest assumption this modeling-solution makes is that permissions are enough and that legitimate permissions can't, also, be abused. The model remains limited while it operates without consideration or context of an application's purpose. While some permissions or combinations of permissions might be associated with malware, there are scenarios where they are required for an application to simply run as designed. For example, gathering information on call status might appear invasive for a music app, but not when you consider that the app uses this information to mute itself when calls are active. Finally, depending how this solution is implemented, may not account for the fact that permissions may be changed and added later by developers.

Challenges

The resulting challenge of this model still becomes its accuracy, even at 96%. This is because rates for false negatives and false positives at 2% or 5% still means a lot considering nearly 3 thousand applications are added daily to Google Play (*Must-Know Google Play Store Statistics*

[Current Data] • *GitNux*, 2023). It is also important to consider the rapid rate of malware development, where obfuscation techniques are always improving ways to mask malicious operations. These issues relate to false negatives, but false positives also come at a cost. Any modeling lacking context for what applications are designed to perform will continue to muddy the waters; wasting resources as some applications are flagged in error.

Additional Applications & Recommendations

Because of the limitations, previous research on permissions-based detection has proposed its use for external services that detect hardware. In this scenario, user application data is uploaded for analysis. When conducted externally, computational costs for a user's device are no longer a concern. Recommendations to strengthen this model, would be to include data for "intents". Intents are similar to permissions except rather than allowing access to information, they allow interactions between applications.

Implementation Plan

The current review process includes an algorithmic scan of application code without executing the code. For implementation, this classifier would ideally be used to update or supplement scanning algorithms for malware. This should involve little interference in the current flow and structures within the review process. In addition, a more thorough report using the SHAP library would be created to further explore how the model handles permissions for classification. If proposed for device-based internal detection, follow-up research would have to be conducted for best privacy practices. This is because permissions used to execute malware analyses are themselves, invasive to user data.

Ethical Assessment

Currently, I do not believe this project solution creates any undue burden on developers in terms of time and effort to have applications reviewed. The overarching dilemma in this malware detection rather, is how to balance protecting user devices and data, without introducing further risk. This issue does not arise in the code review process for applications to be approved for store release. If this algorithm is implemented by services that scan devices for malware via uploading data or used for internal scanning software, privacy precautions will be critical.

Questions

1. How successful is the current review process at keeping malicious apps out?
2. How exactly do permissions relate to malware?
3. How do applications with malware go undetected?
4. What types of malware are there?
5. How are legitimate permissions abused?
6. Why is a gradient boosting algorithm less prone to over-fitting?
7. What are SHAP values?
8. How are SHAP values calculated?
9. How do I interpret a SHAP summary visualization?
10. Why does malware detection at the user-level, introduce its own privacy risks?

References

About Google Play Reviews - Google Play Help. (n.d.). https://support.google.com/googleplay/answer/13870172?hl=en

Alshehri, A., Hewins, A., McCulley, M., Alshahrani, H., Fu, H., & Zhu, Y. (2017). Risks behind Device Information Permissions in Android OS. *Communications and Network*, 09(04), 219–234. https://doi.org/10.4236/cn.2017.94016

Danny Revaldo. (2024). Android Malware Detection Dataset [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/7689244

Ehsan A, Catal C, Mishra A. Detecting Malware by Analyzing App Permissions on Android Platform: A Systematic Literature Review. *Sensors* (Basel). 2022 Oct 18;22(20):7928. doi: 10.3390/s22207928. PMID: 36298282; PMCID: PMC9609682.

Must-Know Google Play Store Statistics [Current Data] • GitNux. (2023, December 16). GITNUX. https://gitnux.org/google-play-store-statistics/

Newman, L. H. (2017, September 22). Why Google Play Store malware is so hard to stop. *WIRED*. https://www.wired.com/story/google-play-store-malware/

Palatty, N. J. (2023, December 21). 30+ malware statistics you need to know in 2024. *Astra Security Blog*. Retrieved March 21, 2024, from https://www.getastra.com/blog/security-audit/malware-statistics/

Papaioannou, P. (2021, September 2). *How malicious applications abuse Android permissions*. NVISO Labs. https://blog.nviso.eu/2021/09/01/how-malicious-applications-abuse-android-permissions/

Appendix

Table 1: Feature Groups & Descriptions

Permission	Various permissions requested by Android applications, such as access to location (coarse and fine), camera, microphone, contacts, SMS, calendar, storage, and more.
System	Features related to system functions and controls, including access to device hardware (e.g., sensors, Bluetooth, NFC), system settings (e.g., changing network state, WiFi settings), and system services (e.g., managing accounts, managing documents).
Security-related	Features related to security functionalities and behaviors, encompassing permission management, authentication, encryption (e.g., cryptographic operations), and security policy enforcement.
Communication	Features related to communication functionalities, including sending and receiving SMS messages, making phone calls, accessing network state, and managing network connections.
Data Access	Features related to accessing and manipulating data, such as reading and writing to various data sources (e.g., external storage, databases), accessing user information (e.g., contacts, call logs), and accessing app-specific data.
App Lifecycle	Features related to managing the application lifecycle, including app installation and uninstallation, app startup and shutdown, app updates, and app permissions.

Device Control	Features related to controlling device behavior and settings, such as changing system settings, modifying audio settings, controlling device display, and managing device power.
Miscellaneous	Other miscellaneous features including accessing system logs, system services and components (e.g., camera, location manager), handling system events (e.g., incoming calls, boot completed), and interacting with system UI components.

Table 2: Data Set Columns of All Android Permissions

ACCESS_ALL_DOWNLOADS	CONFIGURE_DISPLAY_COLOR_TRANSFORM	PROVIDE_TRUST_AGENT
ACCESS_BLUETOOTH_SHARE	CONFIGURE_WIFI_DISPLAY	QUERY_DO_NOT_ASK_CREDENTIALS_ON_BOOT
ACCESS_CACHE_FILESYSTEM	CONFIRM_FULL_BACKUP	READ_BLOCKED_NUMBERS
ACCESS_CHECKIN_PROPERTIES	CONNECTIVITY_INTERNAL	READ_DREAM_STATE
ACCESS_CONTENT_PROVIDERS_EXTERNALLY	CONTROL_INCALL_EXPERIENCE	READ_FRAME_BUFFER
ACCESS_DOWNLOAD_MANAGER	CONTROL_KEYGUARD	READ_INPUT_STATE
ACCESS_DOWNLOAD_MANAGER_ADVANCED	CONTROL_LOCATION_UPDATES	READ_INSTALL_SESSIONS
ACCESS_DRM_CERTIFICATES	CONTROL_VPN	READ_LOGS
ACCESS_EPHEMERAL_APPS	CONTROL_WIFI_DISPLAY	READ_NETWORK_USAGE_HISTORY
ACCESS_FM_RADIO	COPY_PROTECTED_DATA	READ_OEM_UNLOCK_STATE
ACCESS_INPUT_FLINGER	CREATE_USERS	READ_PRECISE_PHONE_STATE
ACCESS_KEYGUARD_SECURE_STORAGE	CRYPT_KEEPER	READ_PRIVILEGED_PHONE_STATE
ACCESS_LOCATION_EXTRA_COMMANDS	DELETE_CACHE_FILES	READ_PROFILE
ACCESS_MOCK_LOCATION	DELETE_PACKAGES	READ_SEARCH_INDEXABLES
ACCESS_MTP	DEVICE_POWER	READ_SOCIAL_STREAM
ACCESS_NETWORK_CONDITIONS	DIAGNOSTIC	READ_SYNC_SETTINGS
ACCESS_NETWORK_STATE	DISABLE_KEYGUARD	READ_SYNC_STATS
ACCESS_NOTIFICATIONS	DISPATCH_NFC_MESSAGE	READ_USER_DICTIONARY
ACCESS_NOTIFICATION_POLICY	DISPATCH_PROVISIONING_MESSAGE	READ_WIFI_CREDENTIAL
ACCESS_PDB_STATE	DOWNLOAD_CACHE_NON_PURGEABLE	REAL_GET_TASKS
ACCESS_SURFACE_FLINGER	DUMP	REBOOT
ACCESS_VOICE_INTERACTION_SERVICE	DVB_DEVICE	RECEIVE_BLUETOOTH_MAP
ACCESS_VR_MANAGER	EXPAND_STATUS_BAR	RECEIVE_BOOT_COMPLETED
ACCESS_WIFI_STATE	FACTORY_TEST	RECEIVE_DATA_ACTIVITY_CHANGE
ACCESS_WIMAX_STATE	FILTER_EVENTS	RECEIVE_EMERGENCY_BROADCAST
ACCOUNT_MANAGER	FLASHLIGHT	RECEIVE_MEDIA_RESOURCE_USAGE
ALLOW_ANY_CODEC_FOR_PLAYBACK	FORCE_BACK	RECEIVE_STK_COMMANDS
ASEC_ACCESS	FORCE_STOP_PACKAGES	RECEIVE_WIFI_CREDENTIAL_CHANGE
ASEC_CREATE	FRAME_STATS	RECOVERY
ASEC_DESTROY	FREEZE_SCREEN	REGISTER_CALL_PROVIDER
ASEC_MOUNT_UNMOUNT	GET_ACCOUNTS_PRIVILEGED	REGISTER_CONNECTION_MANAGER
ASEC_RENAME	GET_APP_GRANTED_URI_PERMISSIONS	REGISTER_SIM_SUBSCRIPTION
AUTHENTICATE_ACCOUNTS	GET_APP_OPS_STATS	REGISTER_WINDOW_MANAGER_LISTENERS

BACKUP	GET_DETAILED_TASKS	REMOTE_AUDIO_PLAYBACK
BATTERY_STATS	GET_INTENT_SENDER_INTENT	REMOVE_DRM_CERTIFICATES
BIND_ACCESSIBILITY_SERVICE	GET_PACKAGE_IMPORTANCE	REMOVE_TASKS
BIND_APPWIDGET	GET_PACKAGE_SIZE	REORDER_TASKS
BIND_CARRIER_MESSAGING_SERVICE	GET_PASSWORD	REQUEST_IGNORE_BATTERY_OPTIMIZATIONS
BIND_CARRIER_SERVICES	GET_PROCESS_STATE_AND_OOM_SCORE	REQUEST_INSTALL_PACKAGES
BIND_CHOOSER_TARGET_SERVICE	GET_TASKS	RESET_FINGERPRINT_LOCKOUT
BIND_CONDITION_PROVIDER_SERVICE	GET_TOP_ACTIVITY_INFO	RESET_SHORTCUT_MANAGER_THROTTLING
BIND_CONNECTION_SERVICE	GLOBAL_SEARCH	RESTART_PACKAGES
BIND_DEVICE_ADMIN	GLOBAL_SEARCH_CONTROL	RETRIEVE_WINDOW_CONTENT
BIND_DIRECTORY_SEARCH	GRANT_RUNTIME_PERMISSIONS	RETRIEVE_WINDOW_TOKEN
BIND_DREAM_SERVICE	HARDWARE_TEST	REVOKE_RUNTIME_PERMISSIONS
BIND_INCALL_SERVICE	HDMI_CEC	SCORE_NETWORKS
BIND_INPUT_METHOD	INJECT_EVENTS	SEND_CALL_LOG_CHANGE
BIND_INTENT_FILTER_VERIFIER	INSTALL_GRANT_RUNTIME_PERMISSIONS	SEND_DOWNLOAD_COMPLETED_INTENTS
BIND_JOB_SERVICE	INSTALL_LOCATION_PROVIDER	SEND_RESPOND_VIA_MESSAGE
BIND_KEYGUARD_APPWIDGET	INSTALL_PACKAGES	SEND_SMS_NO_CONFIRMATION
BIND_MIDI_DEVICE_SERVICE	INTENT_FILTER_VERIFICATION_AGENT	SERIAL_PORT
BIND_NFC_SERVICE	INTERACT_ACROSS_USERS	SET_ACTIVITY_WATCHER
BIND_NOTIFICATION_LISTENER_SERVICE	INTERACT_ACROSS_USERS_FULL	SET_ALWAYS_FINISH
BIND_NOTIFICATION_RANKER_SERVICE	INTERNAL_SYSTEM_WINDOW	SET_ANIMATION_SCALE
BIND_PACKAGE_VERIFIER	INTERNET	SET_DEBUG_APP
BIND_PRINT_RECOMMENDATION_SERVICE	INVOKE_CARRIER_SETUP	SET_INPUT_CALIBRATION
BIND_PRINT_SERVICE	KILL_BACKGROUND_PROCESSES	SET_KEYBOARD_LAYOUT
BIND_PRINT_SPOOLER_SERVICE	KILL_UID	SET_ORIENTATION
BIND_QUICK_SETTINGS_TILE	LAUNCH_TRUST_AGENT_SETTINGS	SET_POINTER_SPEED
BIND_REMOTEVIEWS	LOCAL_MAC_ADDRESS	SET_PREFERRED_APPLICATIONS
BIND_REMOTE_DISPLAY	LOCATION_HARDWARE	SET_PROCESS_LIMIT
BIND_ROUTE_PROVIDER	LOOP_RADIO	SET_SCREEN_COMPATIBILITY
BIND_RUNTIME_PERMISSION_PRESENTER_SERVICE	MANAGE_ACCOUNTS	SET_TIME
BIND_SCREENING_SERVICE	MANAGE_ACTIVITY_STACKS	SET_TIME_ZONE
BIND_TELECOM_CONNECTION_SERVICE	MANAGE_APP_OPS_RESTRICTIONS	SET_WALLPAPER
BIND_TEXT_SERVICE	MANAGE_APP_TOKENS	SET_WALLPAPER_COMPONENT
BIND_TRUST_AGENT	MANAGE_CA_CERTIFICATES	SET_WALLPAPER_HINTS
BIND_TV_INPUT	MANAGE_DEVICE_ADMINS	SHUTDOWN
BIND_TV_REMOTE_SERVICE	MANAGE_DOCUMENTS	SIGNAL_PERSISTENT_PROCESSES
BIND_VOICE_INTERACTION	MANAGE_FINGERPRINT	START_ANY_ACTIVITY
BIND_VPN_SERVICE	MANAGE_MEDIA_PROJECTION	START_PRINT_SERVICE_CONFIG_ACTIVITY
BIND_VR_LISTENER_SERVICE	MANAGE_NETWORK_POLICY	START_TASKS_FROM_RECENTS
BIND_WALLPAPER	MANAGE_NOTIFICATIONS	STATUS_BAR
BLUETOOTH	MANAGE_PROFILE_AND_DEVICE_OWNERS	STATUS_BAR_SERVICE
BLUETOOTH_ADMIN	MANAGE_SOUND_TRIGGER	STOP_APP_SWITCHES
BLUETOOTH_MAP	MANAGE_USB	STORAGE_INTERNAL

BLUETOOTH_PRIVILEGED	MANAGE_USERS	SUBSCRIBED_FEEDS_READ
BLUETOOTH_STACK	MANAGE_VOICE_KEYPHRASES	SUBSCRIBED_FEEDS_WRITE
BRICK	MASTER_CLEAR	SUBSTITUTE_NOTIFICATION_APP_NAME
BROADCAST_CALLLOG_INFO	MEDIA_CONTENT_CONTROL	SYSTEM_ALERT_WINDOW
BROADCAST_NETWORK_PRIVILEGED	MODIFY_APPWIDGET_BIND_PERMISSIONS	TABLET_MODE
BROADCAST_PACKAGE_REMOVED	MODIFY_AUDIO_ROUTING	TEMPORARY_ENABLE_ACCESSIBILITY
BROADCAST_PHONE_ACCOUNT_REGISTRATION	MODIFY_AUDIO_SETTINGS	TETHER_PRIVILEGED
BROADCAST_SMS	MODIFY_CELL_BROADCASTS	TRANSMIT_IR
BROADCAST_STICKY	MODIFY_DAY_NIGHT_MODE	TRUST_LISTENER
BROADCAST_WAP_PUSH	MODIFY_NETWORK_ACCOUNTING	TV_INPUT_HARDWARE
CACHE_CONTENT	MODIFY_PARENTAL_CONTROLS	TV_VIRTUAL_REMOTE_CONTROLLER
CALL_PRIVILEGED	MODIFY_PHONE_STATE	UPDATE_APP_OPS_STATS
CAMERA_DISABLE_TRANSMIT_LED	MOUNT_FORMAT_FILESYSTEMS	UPDATE_CONFIG
CAMERA_SEND_SYSTEM_EVENTS	MOUNT_UNMOUNT_FILESYSTEMS	UPDATE_DEVICE_STATS
CAPTURE_AUDIO_HOTWORD	MOVE_PACKAGE	UPDATE_LOCK
CAPTURE_AUDIO_OUTPUT	NET_ADMIN	UPDATE_LOCK_TASK_PACKAGES
CAPTURE_SECURE_VIDEO_OUTPUT	NET_TUNNELING	USER_ACTIVITY
CAPTURE_TV_INPUT	NFC	USE_CREDENTIALS
CAPTURE_VIDEO_OUTPUT	NFC_HANDOVER_STATUS	VIBRATE
CARRIER_FILTER_SMS	NOTIFY_PENDING_SYSTEM_UPDATE	WAKE_LOCK
CHANGE_APP_IDLE_STATE	OBSERVE_GRANT_REVOKE_PERMISSIONS	WRITE_APN_SETTINGS
CHANGE_BACKGROUND_DATA_SETTING	OEM_UNLOCK_STATE	WRITE_BLOCKED_NUMBERS
CHANGE_COMPONENT_ENABLED_STATE	OVERRIDE_WIFI_CONFIG	WRITE_DREAM_STATE
CHANGE_CONFIGURATION	PACKAGE_USAGE_STATS	WRITE_GSERVICES
CHANGE_DEVICE_IDLE_TEMP_WHITELIST	PACKAGE_VERIFICATION_AGENT	WRITE_MEDIA_STORAGE
CHANGE_NETWORK_STATE	PACKET_KEEPALIVE_OFFLOAD	WRITE_PROFILE
CHANGE_WIFI_MULTICAST_STATE	PEERS_MAC_ADDRESS	WRITE_SECURE_SETTINGS
CHANGE_WIFI_STATE	PERFORM_CDMA_PROVISIONING	WRITE_SETTINGS
CHANGE_WIMAX_STATE	PERFORM_SIM_ACTIVATION	WRITE_SMS
CLEAR_APP_CACHE	PERSISTENT_ACTIVITY	WRITE_SOCIAL_STREAM
CLEAR_APP_GRANTED_URI_PERMISSIONS	PROCESS_CALLLOG_INFO	WRITE_SYNC_SETTINGS
CLEAR_APP_USER_DATA	PROCESS_PHONE_ACCOUNT_REGISTRATION	WRITE_USER_DICTIONARY