Android Security 2015 Year In Review

April 2016

android

Contents

Overview

Google is committed to ensuring that Android is a safe ecosystem for users and developers. We do that by investing in multiple layers of protection across a large and growing ecosystem. We provide security applications and services for Android, constantly strengthen the core Android platform, and foster an ecosystem rich with security innovation. We also regularly measure the effectiveness of these efforts by collecting, analyzing, and sharing data about the security of the Android ecosystem. We consider transparency to be critical, so our second annual Android Security Year in Review is intended to share the progress we've made with regards to security in the last year, as well as provide our view of the state of security in the Android ecosystem.

Google security services for Android

To protect the Android ecosystem and its users, Google provides a complete set of endpoint security services that is included automatically as part of Google Mobile Services (GMS). These include both cloud-based services and on-device services delivered as Android applications, so users don't have to install additional security services to keep their devices safe. In 2015, these services protected over 1 billion devices, making Google one of the world's largest providers of on-device security services.

In 2015, we increased our understanding of the ecosystem using automated systems that incorporate large-scale event correlation and machine learning to run more than 400 million automatic security scans per day on devices with Google Mobile Services. Thanks in part to these scans, successful exploitation of vulnerabilities on Android devices continued to be extremely rare during 2015. The largest threat was installation of Potentially Harmful Applications (PHAs), or applications that may harm a device, harm the device's user, or do something unintended with user data. On average, less than 0.5% of devices had a PHA installed during 2015 and devices that only installed applications

from Google Play averaged less than 0.15%. Ongoing protection by Verify Apps, which scans for PHAs, and SafetyNet, which protects from network threats—as well as actions taken by the Android Security Team—helped stop the spread of PHAs like Ghost Push and reduced Russian fraudware by over 80%. We also released the SafetyNet Attest API to help developers check device compatibility and integrity.

Android platform security

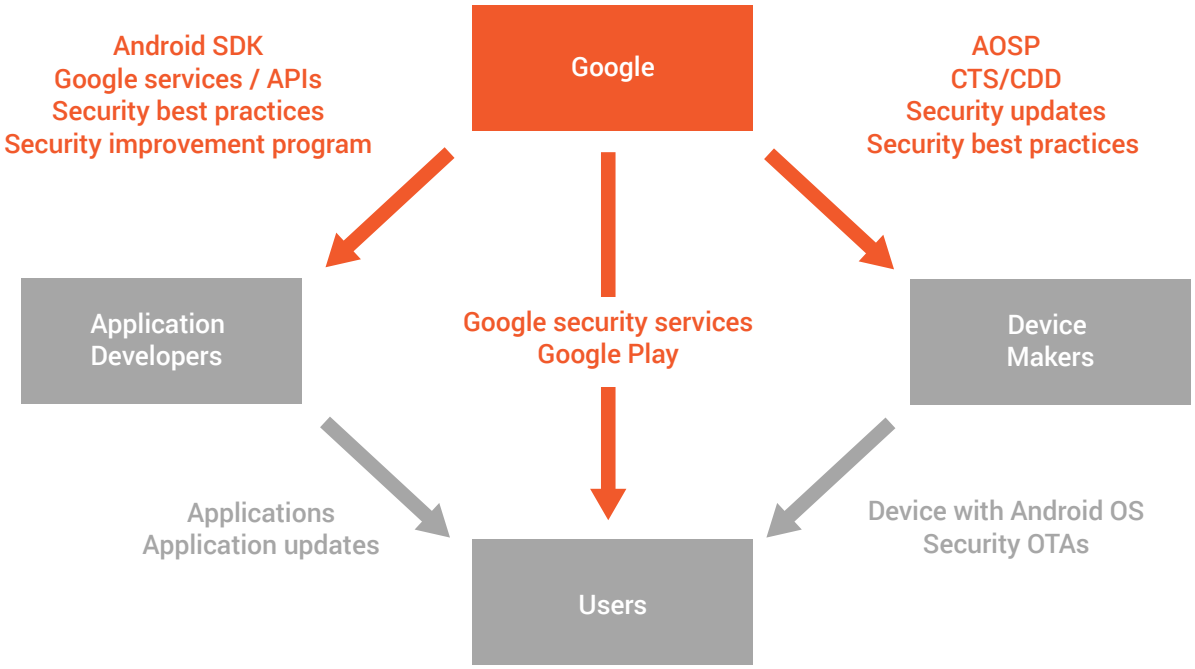
All Android devices share a common security model that provides every application with a secure, isolated environment known as an application sandbox. The Android security model has grown stronger over time, with further application isolation enabled by SELinux, enhanced exploit mitigations, and cryptographic features, such as full disk encryption and verified boot.

In 2015, Android continued to iterate and expand platform security technology with the launch of Android 6.0. Most new devices with Android 6.0 have a hardware root of trust and provide a verifiable good boot state. We introduced support for device fingerprint sensors, improving user security through ease of use. We changed the permission model so that users can see, grant, and revoke permissions for applications at a granular level, allowing for better control of the data and capabilities that each application can access. Encryption is now mandatory for all devices capable of supporting it, and has been extended to allow for encrypting data on SD cards. We continue to guide the Android ecosystem to widely adopt the strongest available security technologies.

Ecosystem security programs

Android also has a number of efforts under way to promote security best practices in the ecosystem. The Android Compatibility Definition Document and Compatibility Test Suite provide a detailed series of security requirements and tests to prove compatibility with these requirements. Google works with device manufacturers to ensure that current devices are secure, and to define a roadmap of constantly increasing security for devices (such as the requirement introduced in 2015 for most new Android devices to use encryption and verified boot). Google Play encourages application developers to adopt security best practices; we introduced policy changes that enhanced user data protection in 2015, and also notified developers about potential security issues, resulting in improvement of security for over 100,000 applications.

Google's role in Android ecosystem security



We launched the Android Vulnerability Rewards program to encourage independent security researchers to test Android's security protections and help us make the Android platform and ecosystem even safer.

We continued to provide device manufacturers with ongoing support for fixing security vulnerabilities in devices, and have expanded the program to include monthly public security bulletins with security patches released to the Android Open Source Project (AOSP). In addition to the updates that we release for Nexus devices, several device manufacturers and network providers are also working toward monthly updates of their devices and services for users. As part of this process, we introduced the Android security patch level, which makes checking if an Android device is up-to-date with all security patches as simple as knowing today's date.

Openness strengthens security

Over time, we've come to recognize that the diversity of devices is a security strength unique to the Android ecosystem. It is well known that highly uniform ecosystems are at risk of ecosystem-wide compromise. The classic real-world example of this phenomenon is crop blights, but the Internet-wide worms of the late 1990s are more recent, digital examples. Because Android is open source,

it has allowed device manufacturers to customize devices and introduce diversity. Android's varied ecosystem (with over 60,000 different device models) provides a naturally occurring defense against simple widespread exploitation, and has made it more difficult for attackers to be successful against the platform as a whole.

Android's open source model has also allowed device manufacturers to introduce new security capabilities. Samsung KNOX, for example, has taken advantage of unique hardware capabilities to strengthen the root of trust on Samsung devices. Samsung has also introduced new kernel monitoring capabilities on their Android devices. Samsung is not unique in their contributions to the Android ecosystem. Blackberry has worked to enhance the security of their devices by enabling kernel hardening and other features in the Blackberry PRIV. CopperheadOS has both introduced security improvements to their own version of Android and made significant contributions to the Android Open Source Project. These are just some of the various contributions made possible through open sourcing that improved the Android ecosystem in 2015.

To summarize, Android has multiple layers of security technology in place to protect our users. In 2015, we improved our security technology, our understanding of the threats that the ecosystem faces, and our ability to respond to those threats. Android continues to advance the state of security, while protecting our users.

Google Security Services for Android

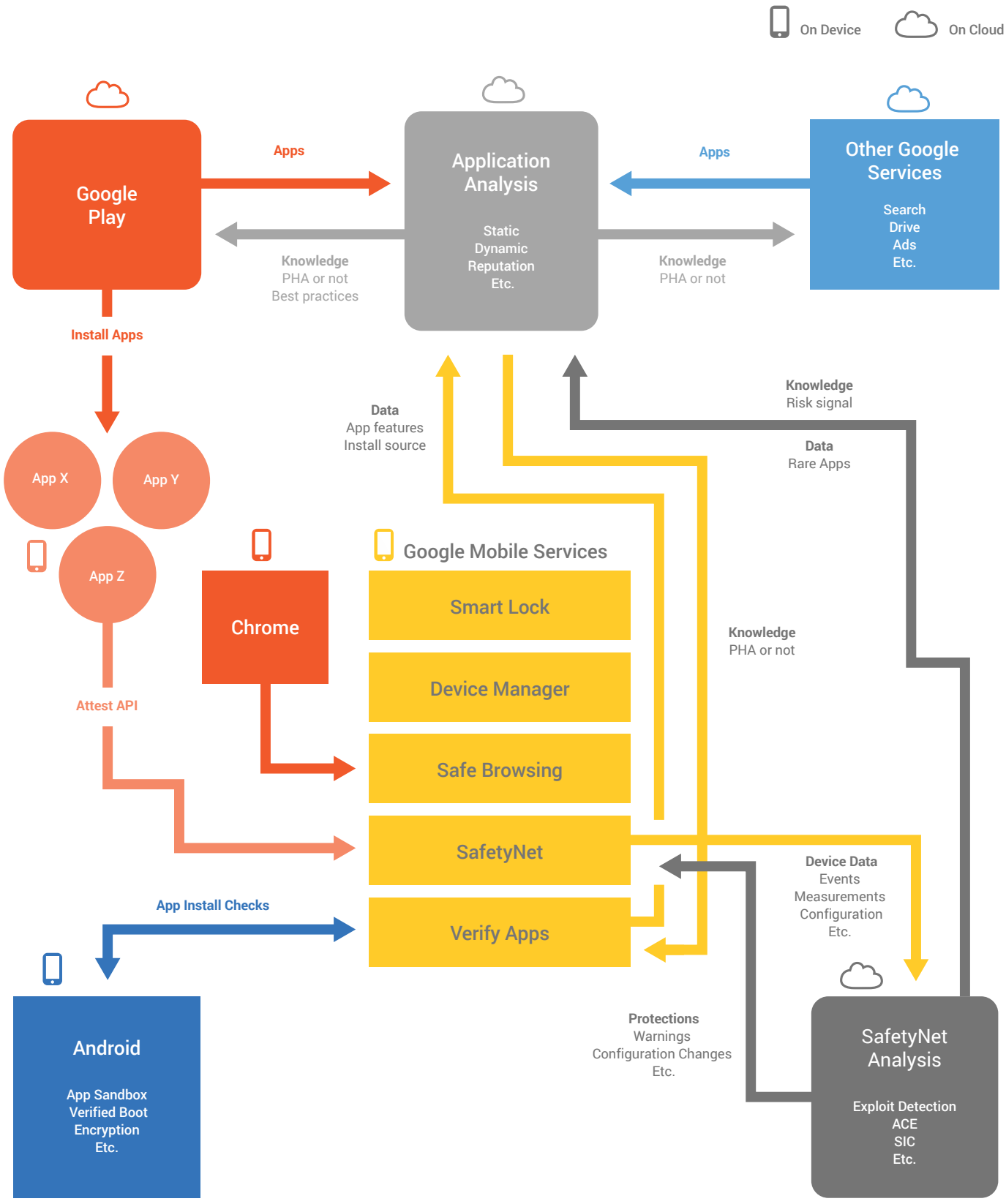
As stated earlier, Google provides security services to protect the Android ecosystem that includes both cloud-based services and on-device services delivered as Android applications. All devices with Google Mobile Services have the complete set of endpoint services that protect against a wide range of common threats including network attacks, application exploits, Potentially Harmful Applications, and physical attacks such as device theft.

Through aggregated, anonymized security data sent from user devices, we gather information and monitor the general state of the Android ecosystem. These services scan for Potentially Harmful Applications at install time, perform regular scans of installed applications, and provide user protection. The services also automatically send anonymized data back to Google, which we use to monitor the overall cleanliness of the Android ecosystem.

As of the end of 2015, there were over 1 billion devices protected by Google's security services, and over 400 million device security scans were conducted per day. We believe this makes our security services the most widely deployed and used endpoint protection in the world.

> We believe this makes our security services the most widely deployed and used endpoint protections in the world.

Google security services for Android, 2015



On-device protections

One of our design goals is to provide the right protection at exactly the moment when it is needed most by the user. Google's use of both on-device and cloud-based services provides Android devices using GMS with flexibility to improve security in ways that are not possible within a traditional client operating system. The endpoint protections Google provides include preventing installation of Potentially Harmful Applications, enabling users to protect a lost or stolen device, protecting users against potentially harmful websites, simplifying the user-authentication process, and even helping third-party applications check the security of a device.

Google on-device protections, 2015

Service	Protection
Verify Apps	Protection from Potentially Harmful Applications
SafetyNet	Protection from network and application-based threats
Safebrowsing	Protection from unsafe websites
Developer API	Provide applications with a way to use Google's security services
Android Device Manager	Protection for lost and stolen devices
Smart Lock	Improve user authentication and physical protection

This section provides a description of these services and details the improvements to these services in 2015.

Verify Apps

Introduced in 2012, Verify Apps uses a cloud-based service to check every application prior to install to determine if the application is potentially harmful. In 2014, these checks were extended to scan applications already on the device to ensure none of them were harmful. Verify Apps will prompt the user to remove a PHA if one is found. Verify Apps can also remove an application without requiring the user to confirm the removal. We may use this functionality in rare cases to remove PHAs we determine are purely harmful and have no possible benefit to users.

In 2015, we improved the ability of Verify Apps so that it can remove applications that register as Device Administrators. We also added the ability for Verify Apps to disable applications that have been installed onto the system partition following a compromise of the device security model.

Not all security improvements are technical in nature. Some of them come from understanding user behavior and making the easiest choice also the safest. In late 2015, we made several changes to the Verify Apps warning dialog to make it easy for users to proceed with the safe option of not installing a PHA. We added a red icon with an exclamation mark to signal to the user that this dialog needs their full attention. We also moved the option to proceed with installation under a cut to prevent cases where a user clicks OK without fully reading what the dialog says.

> Changing the user experience resulted in 50% fewer users installing PHAs.

Comparison of Verify Apps Dialog Improvements

Previous Verify Apps warning dialog

New Verify Apps warning dialog, with the option to proceed with install hidden

New Verify Apps warning dialog, showing the option to move forward with install

Verify Apps—Rare app collection
Verify Apps protects users against applications that are installed from any source—whether they come from Google Play or outside of Play—so it is important that our systems have visibility into as many applications as possible. All applications that are submitted to Google Play undergo a review. Similarly, all applications that Google's cloud-based systems are able to locate on public websites are reviewed.

Starting in 2015, users can send applications from their device to Google for review. This increases the effectiveness of the protection provided by Verify Apps for all users.

SafetyNet

SafetyNet allows devices to contribute security-related information to Google's cloud-based services. This information can include information about security events, logs, configuration information, and other security-relevant information. SafetyNet was introduced in 2013.

SafetyNet—Exploit detection
Many vulnerabilities have tell-tale characteristics associated with exploitation, such as passing a too-long string into a buffer, or receiving two different responses from a DNS server when requesting a single lookup.

Google began to use this knowledge to improve Android device security in 2013 when we added logging as part of vulnerability patches to detect exploitation. When a vulnerability is fixed, code is inserted into the platform (or application) which generates a log when a potential exploit attempt is detected. This log contains information required to track exploitation trends and better understand the effectiveness of our security improvements.

In 2015, we added exploit detection for multiple new vulnerabilities, including several related to Stagefright.

SafetyNet—Network probes
Certificate pinning and blacklisting were introduced in Android 4.2 to provide a mechanism to respond to potential compromises in the Certificate Authorities installed by default on Android devices. On devices with Android 4.4 and later, Android displays a warning if a certificate was installed locally on the device that could allow interception of SSL traffic. Starting in October 2014, SafetyNet used active network probes to identify cases where the system certificate store has been manipulated.

Throughout 2015, SafetyNet found that fewer than 2 out of every million devices had installed a local certificate to man-in-the-middle network connections to Google services. In most cases, those certificates were installed by the user, although we have seen a small number of instances where devices were compromised and had a certificate installed directly into the system certificate store, which avoids the security warning to users on newer devices. All instances appear to be part of legitimate enterprise security efforts. At this time, we have not detected any manipulation of the system certificate store efforts that we would classify as "malicious."

Android Device Manager

In 2013, Google introduced the Android Device Manager service to help users locate their lost Android devices. Users are also able to remotely set up a lock screen or erase the device entirely to protect their personal data and accounts. This is available to Android users who sign into their Google accounts on their phones. No additional downloads are required, and the service is enabled by default on devices running Android 4.4 and above.

In 2015, 17.8 million people used Android Device Manager to locate their device, representing a 43% increase in usage over 2014. Of these, 22% were using Android Device Manager for the first time. Most users find their devices with the Locate and Ring functionality. Lock and Wipe functions are used significantly less. This may indicate that in general, devices are simply lost and users are able to recover them.

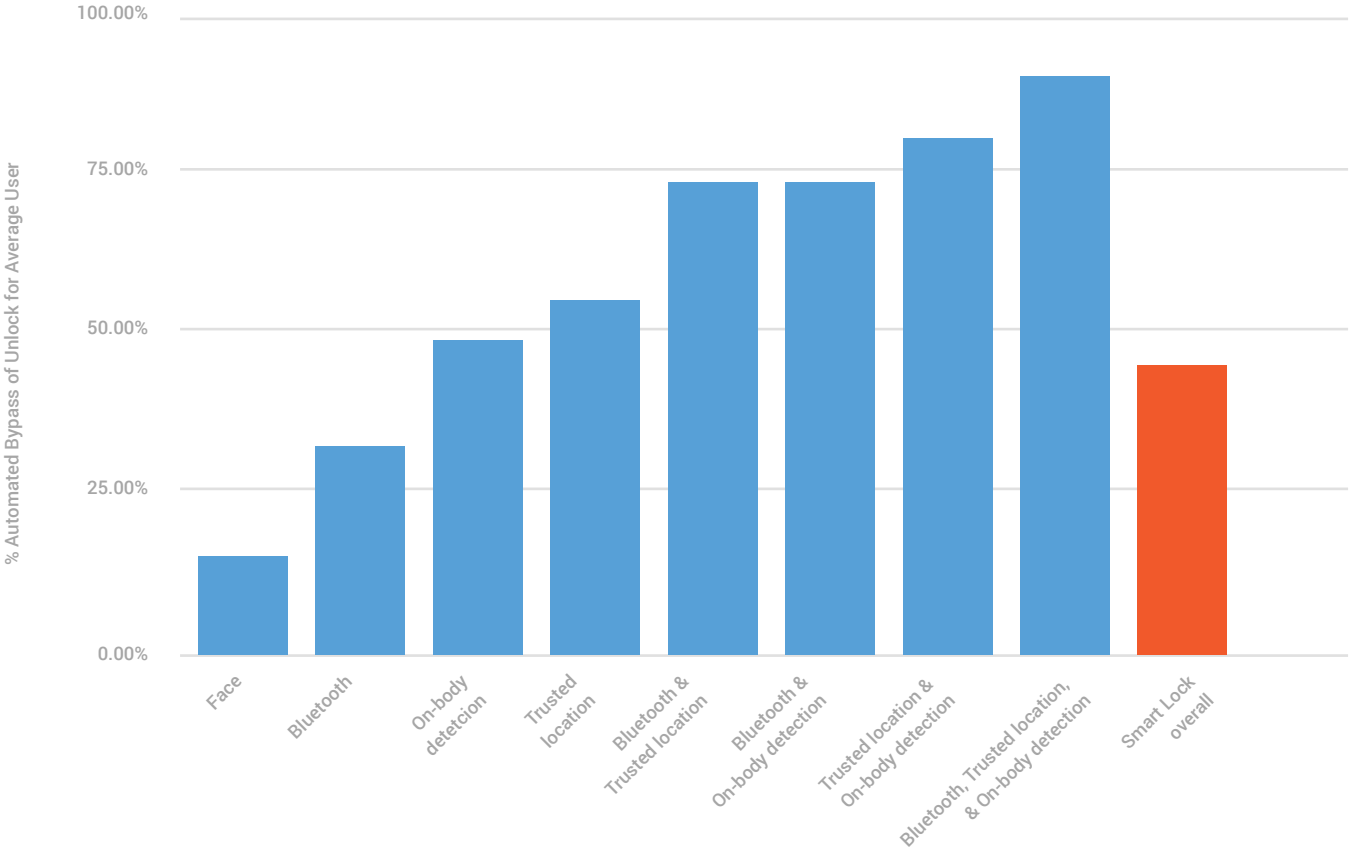
We saw a steady growth in Android Device Manager usage, with more than 200,000[1] daily users at the end of 2015.

[1] The original version of this document included a graph with incorrect daily data. We have the final count (~200,000 users per day), but do not have data for the year.

Smart Lock

Using a lockscreen greatly increases user privacy and security. Our research has found that many users of mobile devices choose not to use a lockscreen because manually unlocking their device dozens or even hundreds of times a day is too burdensome. In 2014, Android 5.0 introduced Smart Lock, which allows a user's device to remain unlocked as long as it remains in their possession, as determined by certain security signals. This reduces the number of times that a user needs to manually unlock their device and encourages adoption of a more secure lockscreen. Initially, Smart Lock supported trusted faces and trusted Bluetooth devices. In 2015, we extended Smart Lock to include on-body detection and trusted places. As the graph below shows, on average, users of Smart Lock need to unlock their device about half as often as before they enabled the feature. And users that have configured Smart Lock to use multiple unlock mechanisms have even better results—use of trusted Bluetooth devices, trusted places, and on-body detection reduces the number of manual unlocks by about 90%.

Smart Lock Usage



Cloud-based security analysis

In a diverse ecosystem with over 1 billion devices, one of the key benefits of Google's security services is that they can gather and analyze data. This allows us to provide protections that are optimized for the current environment, and in some cases even for a single device. At the end of 2015, Google provided over 400 million device security scans each day, contributing billions of pieces of new data to our analysis engine every day. This section describes some of the new analysis capabilities introduced to Google's security services for Android in 2015 to enhance our understanding of potential threats so that we can better protect users.

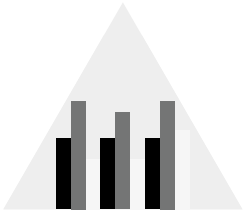
Application security analysis

Before applications become available in Google Play, they undergo an application review process to confirm that they comply with Google Play policies. We conduct similar analysis for Android applications that Google has found outside of Google Play by crawling the web for these applications, as well as drawing from other sources such as the VirusTotal database and the Rare App Collection feature in Verify Apps.

The Application Security Analysis systems at Google were built through a collaboration between the Google Safebrowsing Team and the Android Security team, leveraging the extensive experience Google Safebrowsing has developed in testing the security of websites. Google's systems use a variety of algorithms, including expert systems and machine learning, to see patterns and make connections that humans would not. The signals and results are continuously monitored and refined to reduce error rate and improve precision.

At the end of 2015, these systems were conducting ongoing automated analysis on over 35 million Android Application Packages (APKs). This includes every version of every application that has been published in Google Play and millions of APKs that were never published in Google Play. Each APK is analyzed multiple times. This analysis requires tens of thousands of CPU cores, many terabytes of RAM, and many petabytes of storage. Because this analysis has been ongoing for several years, our visibility into the application ecosystem is larger than the current install base of applications. Many APKs that we have found outside of Google Play have very few installations, and applications from Google Play are updated automatically, so older versions are replaced by newer versions. At the end of 2015, about 75% of the APKs within our system were not in active circulation (they have 0 known installations) and another 10% currently had fewer than five installations.

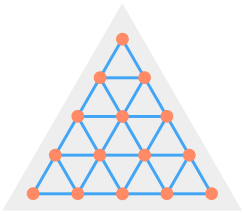
Here are some of the ways that our machines learn what is good and what is bad:



Static analysis
We analyze application code without running the app. Application features are extracted and analyzed against expected good behavior and potential bad behavior.

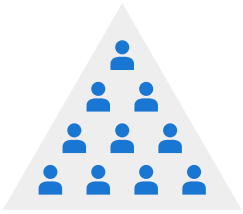
Dynamic analysis
We run applications to identify interactive behavior that cannot be seen with static analysis. This allows reviewers to identify attacks that require connection to a server and dynamic downloading of code.

Third-party reports
We cultivate active relationships with industry and academic security researchers. These independent security researchers also evaluate applications in a variety of ways and will often let us know if they see something amiss.

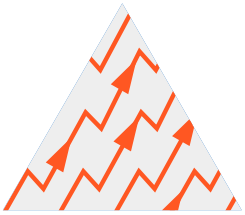


Developer relationships
We analyze non-code features to determine possible relationships between applications and to evaluate whether the developer that created the application may have previously been associated with creation of Potentially Harmful Applications.

Signatures
We use signatures to compare apps against a database of known bad apps and vulnerabilities.



SafetyNet
A privacy preserving sensor network spanning the Android ecosystem, identifying apps and other threats that cause harm to the device.



Heuristic and similarity analysis
We compare applications with each other to find trends that lead to harmful apps.

In the following sections, we provide more details about specific improvements to the way that we analyze applications, in addition to the ways that this analysis is used to protect the Android ecosystem.

Static analysis

Drawing from the experience developed by the SafeBrowsing team for detecting web-based abuse, one of the first technologies Google used to test Android applications was static analysis of the application code. Static analysis allows us to extract specific application behaviors and then express them in a manner that can be compared against our policies. For example, we can determine ifan application contains code that would allow it to send an SMS to a specific number.

In 2014, we replaced substantial parts our static analysis framework, allowing us to create rules that linked functionality that spanned multiple subcomponents in an application. In 2015, we expanded the set of rules used in static analysis.

The Application Security Improvement Program provides an example of a result that wasn't possible before the improvements made in 2015: some applications override the default SSL error handling in a way that would allow web connections to a server with an invalid certificate. This is done to simplify testing during development, but if the code remains in a production application, it is susceptible to man-in-the-middle attacks. We are now able to identify applications with this potential security problem, and we have helped thousands of application developers catch this issue in their application.

Dynamic analysis

Drawing from the experience developed by the SafeBrowsing team for detecting web-based abuse, Google also uses dynamic analysis to test Android applications in an emulated environment where we run the application and provide inputs. This allows us to monitor its behavior to detect potentially harmful behavior that may not have been apparent in static analysis.

One of the challenges of dynamic analysis is that malicious applications may attempt to detect that they are running in an analysis framework and evade detection. We use the diversity of Android ecosystem as a natural camouflage: our systems are designed to simulate a large number of different devices in different situations and monitor applications to see how they respond. If they respond differently that may be an indication the application is trying to evade detection. In 2015, we made a large number of changes that are designed to prevent detection of the analysis framework.

Dynamic analysis allows us to see how an application reacts to external variables, such as external network servers. Many applications download

functionality from servers, receive data that determines how they will behave, or both. In 2015, we introduced a more sophisticated process for deciding when an application can have access to the network, and how they would access the network. These techniques allow us to increase the coverage of our dynamic analysis while minimizing the information we reveal to malicious application developers who may want to evade our analysis.

Another challenge in dynamic analysis is exercising all of the functionality of the application. For example, if an application starts by asking for an account login, then the automated system that lacks account credentials may not be able to run very much of the application functionality. In 2015, we began to use our static analysis to identify events and/or conditions that will provide for increased code coverage in our dynamic analysis.

There are also two entirely new areas that we introduced to our dynamic analysis in 2015:

Human-powered dynamic analysis
Many Google products interact with Android applications and have policies enforced using a review process. Google Play, for example, reviews all application and content updates for compliance with the Google Play Developer Content Policy, and Google AdWords reviews apps promoted through mobile app install campaigns.

Within these review processes, automated systems are not always sufficient to conduct a complete review, so we have many analysts across Google that conduct manual reviews of Android applications. In 2015, we enhanced our automated systems so that these manual reviews could provide data directly into our automated system. Letting humans interact with the application increases coverage for our dynamic analysis, and provides another opportunity for our automated systems to detect potentially harmful behavior.

Honeypots
A honeypot is a set of data that appears to be legitimate but is actually fake data that is isolated and monitored. It appears to contain information or a resource of value to attackers that would not be of interest in a legitimate use case. In 2015, Android began to use honeypots to enhance our dynamic analysis of applications. Specifically, we create fake account data and then monitor that account for spam and other abusive interactions. These honeypots are generated in a manner that allows us to associate the abusive interaction with a set of applications, so that we can identify PHAs even if the abuse takes place long after our initial evaluation of the application.

Developer relationships
Before a developer is allowed to submit an application to Google Play, they need to create an account with Google and make a small commercial transaction. Google uses this information to perform a risk assessment of the developer before they have even uploaded an application to Google Play. Over time, our systems gain more information about the developer account and are able to make more accurate risk assessments.

Our systems have discovered that once a developer makes a single PHA, they are much more likely to produce PHAs in the future. Because of this key finding, our systems can take action against all applications associated with an individual developer. In some instances, the same individual developer may also create multiple accounts, so our system is designed to identify accounts that are created or used by the same developer.

In 2015, we updated our system for analyzing individual developers and also for analyzing the relationship between developers. These improved systems allow us to find clusters of accounts that are controlled by the same developer, and more quickly respond to applications created by the developer (both inside and outside of Google Play).

Heuristics and similarity
Starting in 2012, our systems use algorithms to detect similarity between applications.

To perform this analysis, every application is decomposed into thousands of constituent pieces and each piece is classified as a feature. Some features, like a proprietary logo for a popular application, are expected to be unique—so if we find a second application using that logo, that may be an indication of an attempt to impersonate the popular application to increase installs or trick a user. In 2015, the Safebrowsing team worked to improve our ability to detect visual similarity between assets (such as logos) that can be used to mislead or trick users to install deceptive applications or phishing applications.

In other cases, analysis of feature similarity can identify associations that are not apparent to a casual observer and allow us to to determine that two applications are related, or might have been created by the same entity. For example, two applications may include an image that is shared only by those two applications and that we have seen in no other applications (or anywhere else, for that matter), which suggests that the two applications had the same author. In 2015, we greatly expanded the number and types of features that we can extract from applications, improving our ability to identify these relationships between applications. To manage the increasing number

of features that are available, we also expanded our use of machine learning and other data analysis techniques to detect non-obvious relationships between applications.

In 2015, we also integrated VxClass, a technology that can create clusters based on similarity in the code structure. This allows us to more quickly identify applications that originate from related source code.

Signatures

Signatures allow our application analysis to detect applications that are the same as a previously identified PHA. The simplest signature that we use is a match of the entire application to a previously known PHA. As mentioned in the section on Rare App Collection, during 2015, over 90% of the time a user attempted to install an application from outside of Google Play, that app had already undergone a complete automated analysis, so this rudimentary signature analysis continues to be quite effective.

In 2014, we deployed a more flexible signature format that can be used to identify applications on the client without sending them to Google for complete analysis. Rather than checking the entire application, this approach extracts multiple features from the application and then checks for similarities to known PHAs based on features. This allows Verify Apps to quickly identify applications that have not been previously seen and warn users even if that application has not undergone a comprehensive automated analysis.

In 2015, we continued to expand our set of signatures and enhance our signature format to provide greater client-side detection rates.

SafetyNet integration

SafetyNet provides information about the security of devices in the real world. Starting in 2014, we began to use this data to identify potentially harmful behavior that might not occur within our emulated environment. For example, how a user responds to a security warning can be an indication of whether an application is potentially trying to trick a user. In 2014, we began use SafetyNet results to identify applications that tried to abuse SMS, based on users' responses to warnings about premium SMS. (See SMS Fraud for more details.)

In 2015, we began to integrate data from the Anomaly Correlation Engine to detect rooting applications and other PHAs.

Anomaly Correlation Engine

SafetyNet gathers anonymized data from over 1 billion Android devices to build a picture of the Android ecosystem. In late 2015, we created the Anomaly

Correlation Engine (ACE) to extend SafetyNet's ability to detect and identify PHAs. ACE monitors for changes in key device security indicators, then examines which applications have changed since the device was in a known secure state. By gathering this information across a large number of devices, we can determine which application is likely to have caused the security posture change and investigate. This allows us to quickly identify new PHAs and take steps to protect users by blocking and removing them from the Android ecosystem before they can spread widely.

System Integrity Check

SafetyNet checks for system integrity when a device is in idle mode, is charging or above a certain charge level, and is connected to an unmetered network. The on-device client hashes the system partition and checks it against a cloud-based service with a collection of known system partitions, called System Integrity Check (SIC). Most devices have well-known system partitions, so in most cases only one query is made, keeping the number of network queries and associated charges low.

If the query results in an unknown system partition, then the SafetyNet client will recursively search to find the source of the hash mismatch. This approach minimizes the number of queries and network traffic, while providing a precise understanding of the state of the device system partition.

Much like the Anomaly Correlation Engine, the System Integrity Check is used to detect potential anomalies and improve the quality of the SafetyNet Attest API. It also provides a way to measure the diversity of the Android ecosystem: SIC has identified over 175,000 unique system partitions (and over 60,000 system partitions that have more than 1,000 active devices).

At-risk device identification

We use data gathered by SafetyNet to identify populations of devices that have a higher risk of a potential security event. For example, our 2014 annual report showed that devices that had the Russian locale and installed applications from outside of Google Play were over 5 times more likely to install a PHA than the worldwide average: in early 2015, we identified this device population as "at risk."

Once we identify a group of at-risk devices, we can make changes to the default configuration of our services to enable stronger protection for those users. These changes may have side effects on users (such as increased bandwidth consumption or requiring the user to navigate an additional UI element), so when strengthening security, we must carefully balance the potential benefit to users.

Some example changes that we applied to at-risk devices include:

- *Increasing the frequency of device-wide security scans*. This allows changes in policies to be adopted more quickly across the target. For example, devices that are at greatest risk are scanned once per day, compared to a global average of approximately once per week.
- *More aggressive blocking of Potentially Harmful Applications.* By default, Verify Apps simply warns users about PHAs, but it can also block installation of applications. We may use automated blocking of certain types of targeted harmful applications that continue to be installed despite users being warned that the application is harmful.

Stronger security for at-risk devices allowed us to reduce the occurrence of Potentially Harmful Applications in Russia by over 80% in 2015. (See the section Russian Banking Fraud for more details.)

C&C monitoring

Some Potentially Harmful Applications (for example, botnets) are designed to receive commands from a server, guiding their actions. In 2012, we began to deploy systems to monitor the command and control servers (C&C) of known backdoors and automated systems. In 2015, we added several new C&C protocols and server instances to our C&C monitoring systems.

Our C&C monitoring systems simulate the behavior of a client application, connect to the C&C, and check to see if any commands have been initiated. This allows us to detect and quickly react to changes in behavior. For example, we might detect that a C&C is telling the members of a botnet to install a specific PHA. Our C&C monitoring systems, in collaboration with Verify Apps on users' devices, would allow us to both block the installation of that PHA and identify any existing, but previously unidentified, members of the botnet based on their attempt to install the PHA.

App Security Improvement Program

The App Security Improvement Program identifies apps in Google Play that have known security vulnerabilities (through incorrect coding practices or by using known vulnerable libraries), notifies the developers of their app's vulnerabilities, and encourages them to fix the vulnerabilities. Apps uploaded to Google Play are scanned for specific known vulnerabilities. As apps are identified, developers are alerted via email and the Play Developer Console to let them know their app contains one of these known vulnerabilities. We then provide the developer with guidance to fix the vulnerabilities.

In 2015, we launched campaigns to remediate five known types of vulnerabilities. These campaigns cover known vulnerabilities in the following libraries: Vungle, Apache Cordova, WebView SSL, GnuTLS, and Vitamio. The program has resulted in the remediation of vulnerabilities in over 100,000 apps in Google Play.

To encourage security fixes within an industry-standard timeline, we began imposing remediation deadlines in 2015. After 90 days from the first announcement, app updates and new apps containing the vulnerability are not accepted in Google Play. Any app that was already in Play and exceeds the 90-day remediation period without a fix continues to be available on Google Play. However, if the developer wants to upload a new version after the remediation period, the new version must include fixes for the vulnerabilities we alerted them about.

Potentially Harmful Application identification

We use the term Potentially Harmful Application (PHA) to describe any application that we determine may harm a device, harm the device's user, or do something unintended with user data through the device. This definition includes intentionally malicious apps like phishing apps or ransomware, but it also includes non-malicious apps. For example, a game that transmits a list of a device's installed apps without express user consent is classified as a PHA. All PHAs are prohibited on Google Play by policy and Verify Apps will warn users about PHAs if they are installed from outside of Google Play. We conduct the same analysis for applications that Google has found outside of Google Play to deliver the Verify Apps feature. For users who have enabled protection for applications that are downloaded from outside Google Play, Verify Apps warns them based on the application's classification.

We modified the PHA warnings slightly in the past year to make the them clearer and easier for users to understand. The current list of warnings presented to users is:

> The Security Improvement program has resulted in remediation of vulnerabilities in over 100,000 apps in Google Play.

Potentially Harmful Application (PHA) classifications

Classification	Description to users
Backdoor	This app lets hackers control your device, giving them unauthorized access to your data.
Call fraud	This app can add charges to your mobile bill by making costly calls without informing you first.
DDOS	This app can be used to perform denial of service attacks against other systems and resources.
Generic PHA	This app can damage your device, add hidden charges to your mobile bill, or steal your personal information.
Harmful site	This app comes from a website that distributes Potentially Harmful Apps.
Non-Android	This app can harm non-Android devices.
Phishing	This app is fake. It can steal your personal data, such as passwords.
Privilege escalation	This app can permanently damage your device or cost you money.
Ransomware	This app can restrict access to your device until a sum of money is paid.
Rooting malware	This app contains code that attempts to bypass Android's security protections.
Rooting (non-malware)	This app contains code that attempts to bypass Android's security protections.
SMS Fraud	This app can add charges to your mobile bill by sending costly SMS messages without informing you first.
Spam	This app can be used to flood targeted tablets, PCs, and mobile phones with messages.
Spyware	This app can spy on you by sending your personal data to unauthorized parties.
Trojan	This app is fake. It can damage your device and steal your data.
Windows	This app can harm a device running Windows.
WAP Fraud	This app can add charges to your mobile bill without asking you first.

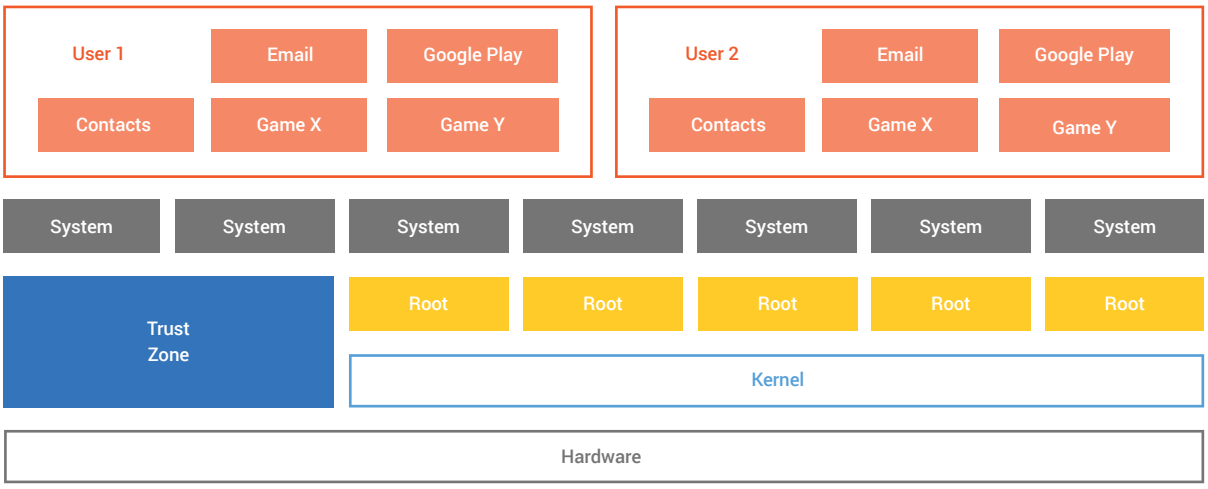
In 2015, we added two new types of warnings, Privilege Escalation and Spam, bringing the total number of warning categories to 17. For more information, see our report on classifying Potentially Harmful Applications.

The vast majority of applications are not classified as potentially harmful, so in general, users of Verify Apps will see nothing displayed at the time of install. If an application is classified as potentially harmful, then in addition to displaying the warning, Verify Apps allows the user to decide whether to proceed with the installation. In very rare cases, such as for the Russian Banking Fraud discussed later, we may judge that the app is extremely harmful, and will block the installation.

Android Platform Security

Since Android was launched over seven years ago, all Android devices have shared a common security model that provides every application with a secure, isolated environment known as an application sandbox. Android was one of the first operating systems to introduce the idea of sandboxing to both protect applications from attacks and protect the device from applications. Sandboxing is used for all applications on the device, including system-level applications. The Android security model has grown stronger over time, with further application isolation enabled by SELinux, enhanced exploit mitigations, and cryptographic features such as full disk encryption and verified boot. Many Android devices also take advantage of unique hardware security features such as integration of key storage and cryptographic routines into TrustZone, and the creation of a hardware root of trust.

The Android security model



This section covers 2015 updates to the Android platform and its features, as well as how we rate the severity of platform vulnerabilities and the programs we have in place to reward researchers who find platform-level vulnerabilities.

Updates and features

In 2015, there was one major new release for Android: version 6.0, named Marshmallow. This section summarizes major security features included in the Android platform and highlights where they were updated in Android 6.0. For a list of more features, see Security Enhancements in Android 6.0.

Application sandbox

The application sandbox is the fundamental technology for the Android security model. Starting with the earliest versions of Android, the Android platform used Linux user-based protection as a means of identifying and isolating application resources. Each Android application is assigned a user ID (UID) and is run as that user in a separate process. This approach is different from other operating systems (including the traditional Linux configuration), where multiple applications run with the same user permissions. SELinux was first used in Android 4.4 in conjunction with the Linux UID to define the application sandbox.

The combination of UID boundaries and SElinux creates a kernel-level application sandbox. The kernel enforces security between applications and the system at the process level through standard Linux facilities, such as user and group IDs that are assigned to applications. By default, applications cannot interact with each other and applications have limited access to the operating system.

Because the application sandbox is in the kernel, this security model extends to native code and to operating system-level applications. All of the software above the kernel including operating system libraries, application framework, application runtime, and all applications run within their own application sandbox.

With Android 6.0, these boundaries were further enhanced with a few important changes. Android's SELinux implementation now supports ioctl filtering, which restricts the set of ioctls available to applications, reducing the size of the potential kernel attack surface. We also enhanced multi-user separation by utilizing Multi-Level Security (MLS) to further enforce file access limits. This provides operating system enforced guarantees that data isn't accessible across user boundaries, or between user and enterprise data.

Permissions

Because of the application sandbox, an Android application can only access a limited range of system resources. These restrictions are implemented in a variety of different forms. Some capabilities are restricted by an intentional lack of APIs to the sensitive functionality (e.g. there is no Android API for directly manipulating the SIM card). In some instances, separation of roles provides a security measure, as with the per-application isolation of storage. In other instances, the sensitive APIs are intended for use by trusted applications and protected through a security mechanism known as Permissions.

Prior to Android 6.0, users had to accept all permission requests from an application at the time they installed it. This all-or-nothing approach required users to make a trade off between accepting all requested permissions or not installing the application. Android 6.0's granular permissions structure gives users more fine-grained control of what resources installed applications are allowed to access. With granular permissions, users are able to only grant permissions that they want to the app. In addition, permissions are granted at application run time, which allows the user to grant permissions as they are needed.

Verified Boot

Verified Boot, introduced in Android 4.4, provides a hardware-based root of trust, and confirms the state of each stage of the boot process. During boot, Android warns the user if the operating system has been modified from the factory version, provides information about what the warning means, and offers solutions to correct it. Depending on device implementation, Verified Boot will either allow the boot to proceed, stop the device from booting so the user can take action on the issue, or prevent the device from booting up until the issue is resolved. Starting from Android 6.0, device implementations with Advanced Encryption Standard (AES) crypto performance above 50MiB/seconds support Verified Boot for device integrity.

Details on Android Verified Boot implementation and features can be found in the Verified Boot section on source.android.com.

Full disk encryption

Encryption was introduced to Android in version 3.0, and has continuously evolved since that time. Starting with Android 5.0, it was strongly recommended that manufacturers enable encryption for all devices. With Android 6.0, devices that use a lockscreen and have Advanced Encryption Standard (AES) crypto performance above 50MiB/second are required to always encrypt private app data and shared data storage partitions, by default. This requirement has been added to the Android Compatibility Definition Document.

In addition to requiring that the device main storage be encrypted at all times, Android 6.0 also allows removable storage media, such as SD cards, to be encrypted as well. Data on SD cards can't be read if the card is removed from the device, providing protection for extensible storage as well as internal storage.

User authentication

Using a lockscreen greatly increases user privacy and security. All versions of Android provide a variety of lockscreen methods to authenticate the user prior to allowing access to the device, including PIN, Password, and Pattern. Starting with Android 5.0 in 2014, Android introduced TrustAgents, which allows for more flexible lockscreen mechanisms provided by an application on the device (Google's Smart Lock was built on this technology).

Starting with version 6.0, Android supports fingerprint scanners. This allows applications to use biometrics for authentication, reducing the number of times a user needs to enter their password or unlock pattern, thus decreasing friction around lockscreen use. Lockscreen use is higher on devices with a fingerprint scanner. For example, 55.8% of Nexus 5 and 6 devices (which have no fingerprint scanner) have a lockscreen, compared to 91.5% on fingerprint-enabled Nexus 5X and 6P devices. We are seeing an increase in lockscreen usage for other Android devices that provide fingerprint scanner support.

Android security patch level

The Android security patch level is a user-visible date that allows consumers and enterprise customers to verify they are using a version of Android that contains the most recent security updates. Our monthly public security bulletins document newly patched security vulnerabilities and the security patch level that contains all of these fixes. By checking the security patch level, users can verify their device has the fixes for the issues described in our bulletins.

Businesses can use the security patch level in their enterprise mobile management platform to require up-to-date security to access corporate resources. This will become a key new tool and best practice for enterprises to protect their infrastructure.

Android Security Patch Level

The Android Security Patch Level is available for devices running Android 4.4 and above. The patch level is required for all Android 6.0 and above devices, and this is tested with the Android Compatibility Test Suite (CTS). Google is also requiring that all Android 5.0 and above devices with Google Mobile Services provide a patch string.

KeyStore and lockscreen

With Android 6.0, Lockscreen verification now occurs in the Trusted Execution Environment (TEE) for devices that support a TEE (such as the majority of new devices that launched with Android 6.0). This provides brute force protection with exponentially increasing delays on verification of the user's lockscreen challenge.

The KeyStore in Android 6.0 includes the ability to tie successful unlocks using a PIN, pattern, password, or fingerprint to KeyStore keys such that certain keys are only available within a certain time window of the unlock. In addition, KeyStore now supports TEE-based AES and HMAC keys. These improvements, along with a host of other smaller enhancements, provide app developers more options to secure their data and communications.

Vulnerabilities and programs

A platform vulnerability represents the possibility of a bad actor bypassing built-in security features in order to steal information, or cause harm to a device. It's important to note that a vulnerability only represents the potential for security control bypass. In order to use a vulnerability, an attacker must be able to construct an exploit that takes advantage of the vulnerability. Actual exploitation of a vulnerability may be complicated or prevented by other security controls. As we assesses a vulnerability to assign a severity, we take its potential exploitability into account. For example, a vulnerability that would normally be rated as Critical could have the severity reduced to Low if there is no way to reach the vulnerable code. We err on the side of caution, so we consider a vulnerability exploitable unless we can prove that it can't be exploited.

Vulnerability severity rating system

The Android Security Team uses a 4-tier system to rate the severity of vulnerabilities. The system used in 2015 is similar to what was presented in last year's Android Security Year In Review report, but we have made some changes. The most important change to the rating system is that we readjusted the Critical rating to remove the requirement that there be active exploitation detected in the wild. The effect of this was to shift a number of vulnerabilities that once would have been rated as High into the Critical category.

The rating system used in 2015 is as follows:

2015 severity rating system

Rating	Consequence of successful exploitation
Critical	— Remote privileged code execution (execution at a privilege level that third-party apps cannot obtain) — Local permanent device compromise (device cannot be repaired without re-flashing the entire operating system, such as a verified boot or Trusted Execution Environment/TEE compromise) — Remote permanent denial of service (inoperability, either completely permanent or requiring re-flashing the device)
High	— Remote unprivileged code execution (execution at a privilege level that third-party apps can obtain through installation) — Local access to system/signature-level permission data or capabilities without permission — Local permanent denial-of-service (inoperability, either completely permanent or requiring re-flashing the device) — Remote temporary denial-of-service (remote hang or reboot)
Moderate	— Access to "dangerous" level permission data or capabilities without permission with an app installed on the device — Local temporary denial-of-service (can be resolved only through a factory reset)
Low	— Access to "normal" level permission capabilities without permission with an app installed on the device — Local temporary denial-of-service (can be resolved by booting the device into Safe Mode and removing the problem application)

New vulnerabilities are discovered through a combination of internal efforts by the Android Security Team and reports from external security researchers. Google supports and encourages responsible disclosure of vulnerabilities through the Android Security Vulnerability Rewards Program.

Android Security Rewards program

On June 16, 2015, Google expanded its existing Vulnerability Reward program to encourage and reward researchers who find, fix, and prevent vulnerabilities on Android. The Android Security Rewards program covers security vulnerabilities discovered in the latest available Android versions for Nexus phones and tablets currently available for sale in the Google Store in the United

States. The program rewards reported vulnerabilities rated as Critical, High, and Moderate. At our discretion, we may reward Low severity vulnerabilities as well, per the rules of the program.

During 2015, we awarded $210,161 for vulnerabilities submitted to the program. This total breaks down to 30 Critical, 34 High, 8 Moderate, and 33 Low severity issues.

> We would like to acknowledge 35 security researchers and their colleagues for their contributions to help improve Android Security. Thank you.

Android platform monthly security updates program

The Android Security Team regularly provides security patches to manufacturers for Android 4.4.4 and higher so they can provide security updates to their devices. 70.8% of all active Android devices are on a version that we support with patches. We have provided these regular updates directly to manufacturers since the release of these versions of Android. On August 5th, 2015, we expanded these releases to include a monthly public security update program to the Android Open Source Project (AOSP), as well as a security update lifecycle for Nexus devices. Since then, we've released monthly security updates to AOSP and for Nexus devices, as have many of our partners such as Samsung, LGE, and Blackberry.

In 2015, we first began to see device manufacturers publicly document their commitment to provide security updates. For example, we will provide security patches to Nexus devices for a minimum of 3 years from time of device launch or 18 months from last sale on Google Play. Samsung and Blackberry have also made statements about updates they provide for devices.

In total, in 2015 we released patches for 69 Critical, 54 High, 34 Moderate, and 16 Low severity fixes. Of these, 7 Critical, 2 High, and 6 Moderate severity fixes were released directly to partners in the January through July timeframe, prior to the start of public bulletins. Although no public security bulletin was provided prior to August 2015, all of the patches provided privately to partners are available in AOSP.

Overall, we provided patches for 94 more vulnerabilities in 2015 than 2014. As mentioned above, due to the significant changes to our severity ratings for vulnerabilities, doing year-over-year comparisons on the severity of the vulnerabilities is not possible. The largest factor that contributed to the rise in the number of vulnerability patches in 2015 was the introduction of the Android Security Rewards program, which encouraged researchers to take a closer look at Android. Of the patches issued, 42% of the Critical, 22% of the High, and 9% of the Moderate severity issues were found internally by Google.

The monthly public security patches are available in AOSP and details on the fixes included in the patches are available in the corresponding Android security bulletin for that month. As of this writing, all patches created in open source code in 2015 are available in AOSP.

Ecosystem Data

This section provides data on the overall state of the Android ecosystem in 2015 with details and trends for categories of Potentially Harmful Applications (PHAs) and platform vulnerabilities.

At a high level, PHAs affected fewer than 1% of devices with GMS for nearly all of 2015, and on average approximately 0.5% of devices had a PHA installed. The rate of PHA install attempts in Google Play dropped, and there was an increase in some categories of PHA install attempts. Noteworthy increases in PHAs were quickly addressed. We also saw that devices that allow apps from outside of Google Play are around 10 times more likely to have PHAs than those that only install from Play.

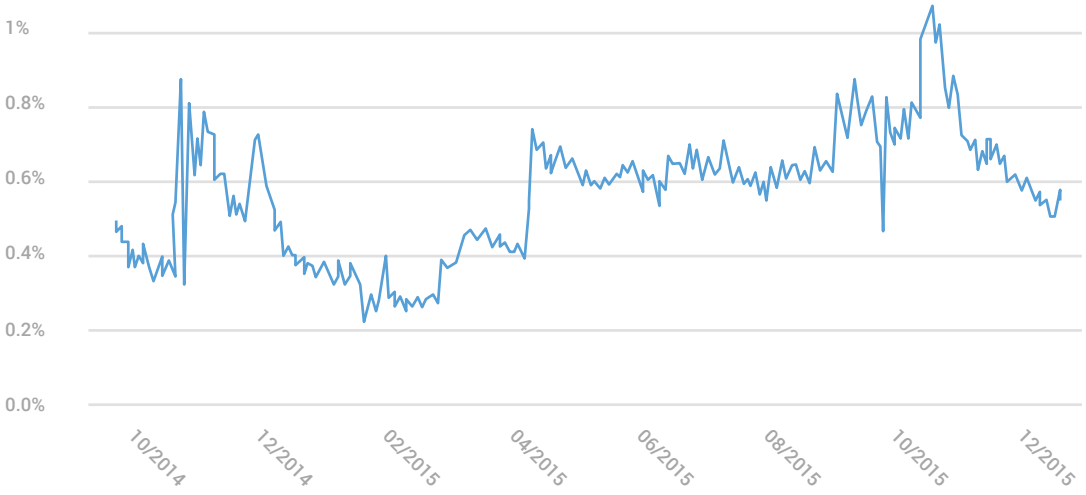
Potentially Harmful Applications

The broadest statistic we track is the frequency with which PHAs are detected during a full-device scan. We refer to this statistic as “device hygiene.” The graph below shows the level of device hygiene starting where the Android Security 2014 Year In Review left off, showing the trend through November and December 2014 and continuing on through all of 2015.

> As the graph illustrates, with the exception of a two week period, over 99% of all Android devices were free of known PHAs[1].

[1] Many of the graphs show an increase in the number of PHAs downloaded and installed in devices in the time period between mid-August through mid-October. This was caused by a family of PHAs known as “Ghost Push”, which was disrupted by the Android security team. (See Ghost Push for more details)

Percentage of devices with PHA installed (except Rooting)



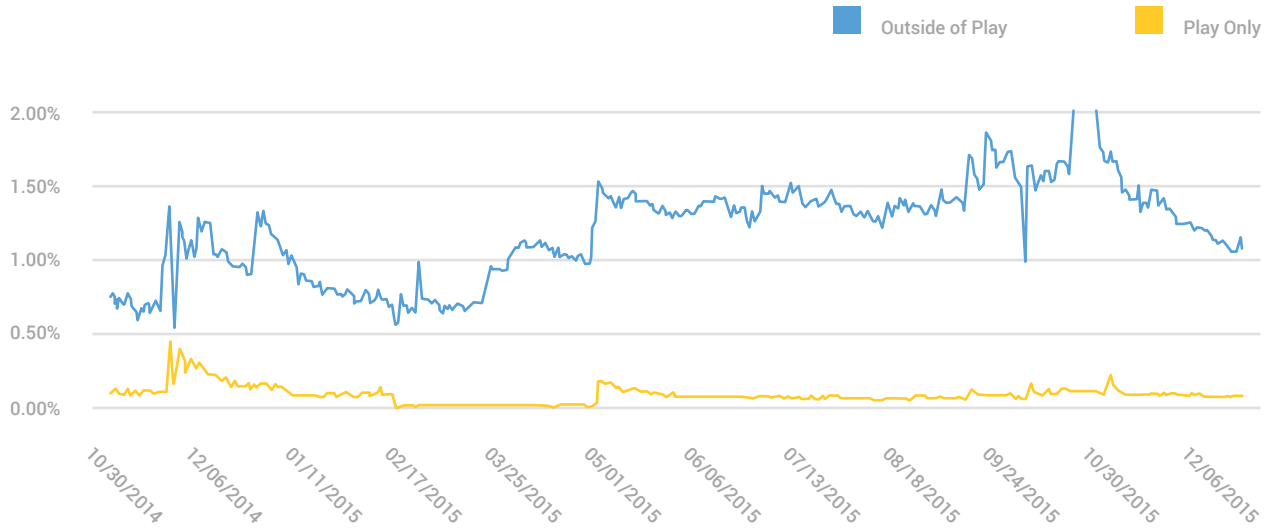
Note that the term “except Rooting” indicates that we have omitted non-malicious rooting applications. We differentiate hostile rooting apps from non-malicious ones based on two factors: the app identifies itself as a rooting application and it gets explicit user consent to root the device. The basic shape of the graph is similar when including non-malicious rooting to the overall graph.

Taking a different view, we perform a related analysis of the percentage of devices that have known PHAs installed. The yellow line indicates the percentage of devices with one or more known PHAs that have only installed apps from Google Play[2]. The blue line represents the percentage of PHAs found on devices that have unknown sources enabled and have installed applications from outside of Google Play. As with the previous graph, the data shown goes back to November 2014 in order to show the trends since the last Android Year In Review report.

> Overall, the trend shows devices that allow installing apps from outside Google Play are around 10 times more likely to have PHAs than those that only install from Play.

[2] For a period starting in February through mid-April, an incorrectly applied update caused Verify Apps to not show alerts for a number of known PHAs. In the graph, this period can clearly be seen by a flat spot on the yellow Play Only line. The sharp upward line in April resulted from correcting this issue. We have done a full analysis on what caused the problem and have made changes to our procedures to ensure this won’t happen again, and also our monitoring system to ensure that we catch issues like this quickly. Data outside this timeframe better reflects normal trends.

Fraction of devices with PHA installed (Except Rooting)



Historical trends

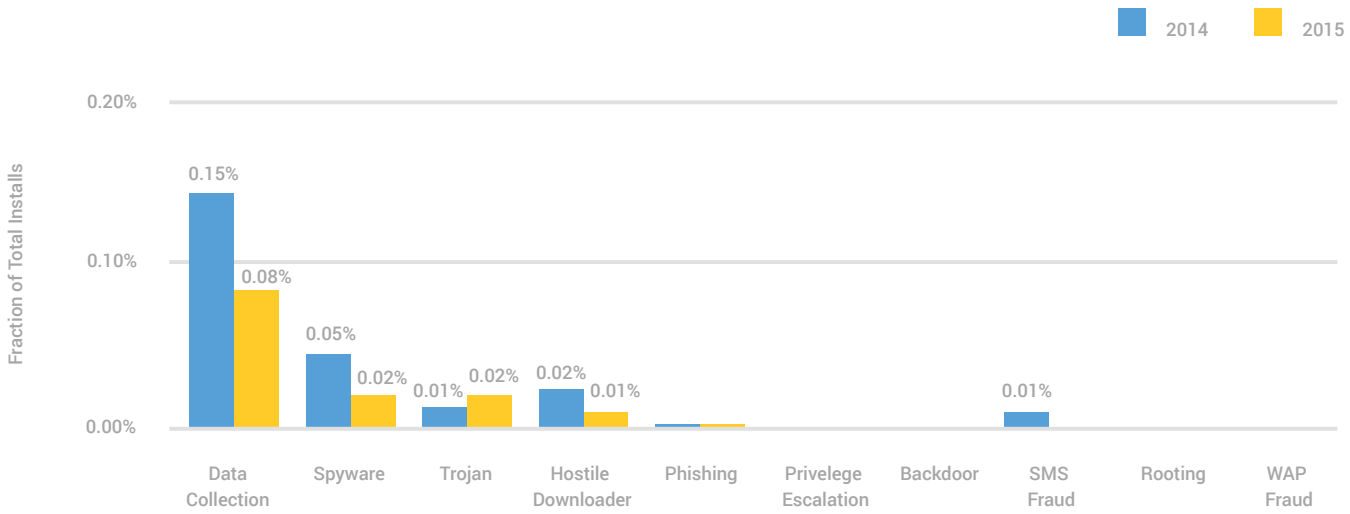
Last year we reported on several classifications of Potentially Harmful Applications. In this section, we compare the state of activity of these PHAs in 2015 to previous years.

The charts in this section show a different statistic than the previous data—these describe the per-day fraction of all installs that were classified as PHAs. It is a count of *installation attempts*, rather than a count of *devices* with an installed application. Due to limitations of our data collection before 2015, this is the only statistic we have available to track historical trends. Unfortunately, we believe that it may overstate the prevalence of certain PHAs outside of Google Play. For example, we frequently see repeated install attempts of the same application onto a device, which increases the number of installation attempts without actually increasing the number of installs. However, it's the best data that we have available now and we think it is useful for showing overall trends.

Note that several categories were defined midway through 2014, which prevents making a complete year-over-year comparison between 2014 and 2015 for those categories. For these areas, we present the partial 2014 data to illustrate trends over time.

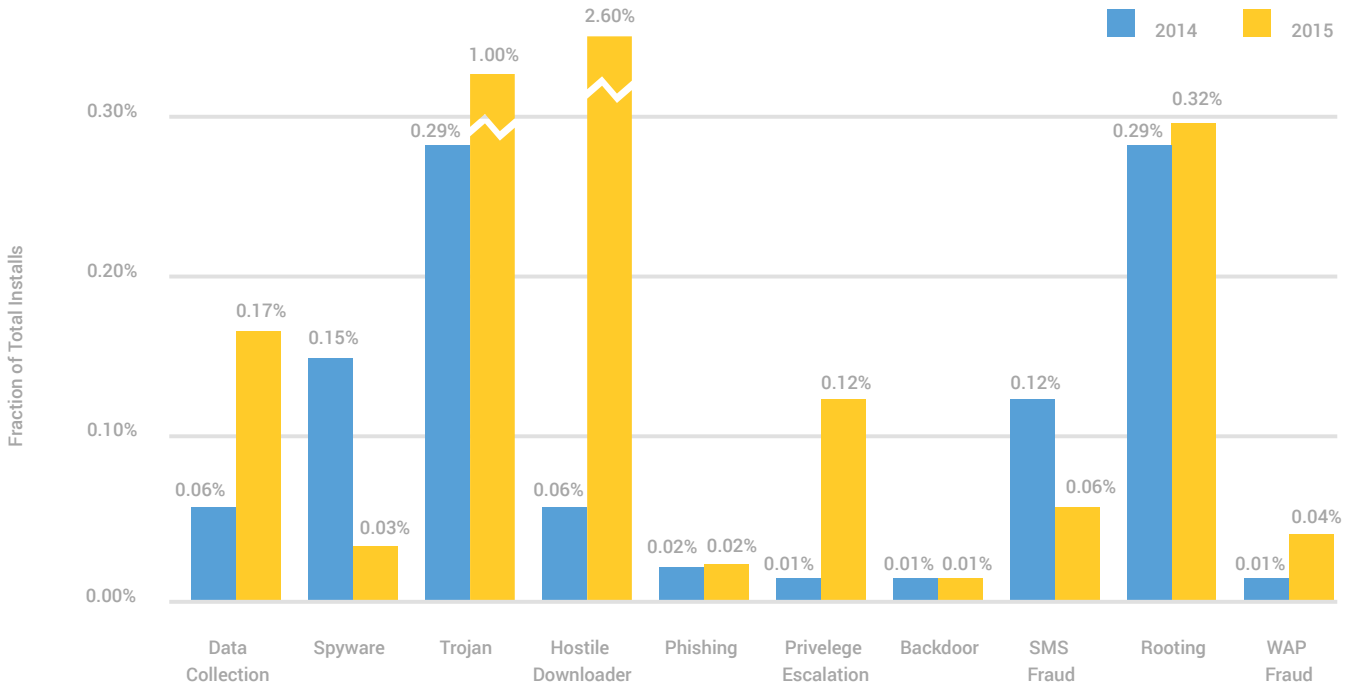
The chart below shows the trends since 2014 for top PHA categories we see in Google Play. Overall, within Google Play we saw a reduction of over 40% in the number of PHA installation attempts in 2015.

Year-Over-Year PHA Comparison—Google Play



The following graph shows the occurrence rates for user attempts to install a PHA from a source outside of Google Play, broken down by category of the PHA. Overall, outside of Google Play the worldwide rate of installation attempts of any PHA increased in 2015 versus 2014.

Year-Over-Year PHA Comparison—Outside of Google Play



Data Collection

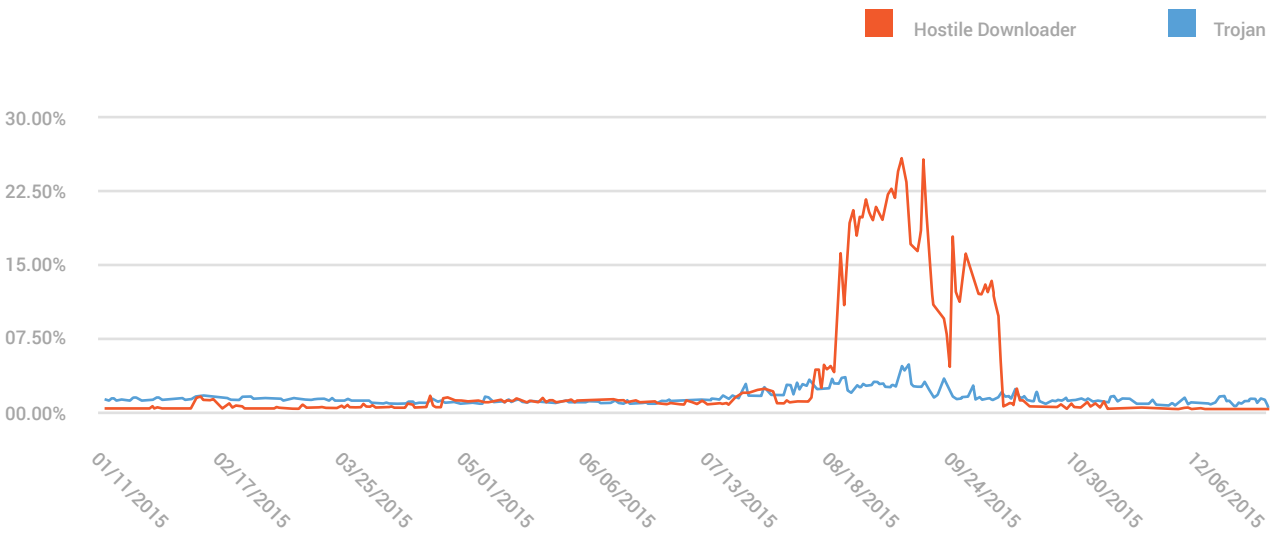
In late 2014, we created the Data Collection category to track applications that copy lists of package names off of devices. Previously, this behavior was not considered to be a PHA, so there were initially a significant number of applications that were classified as Data Collection, but that decreased throughout the year.

Hostile Downloader and Trojan

The increase in Hostile Downloader in 2015 was almost entirely due to a single family of PHAs known as Ghost Push, which is described in more detail in the Noteworthy PHAs section. This family used a multistage process that installed a hostile downloader app (an app that downloads other applications without the user's permission), then used this to download other applications, primarily Trojans.

The following graph shows how nearly all installs of Hostile Downloader and Trojans occurred in the third quarter. Once Ghost Push was identified, it was promptly stopped and we began to remove these applications from users' devices.

2015: Hostile Downloader and Trojan trends



Privilege Escalation

In late 2014, we introduced Privilege Escalation as a category, so the year-over-year increase comes from having a full year's worth of data in 2015, versus having only a partial year in 2014. We classify apps as Privilege Escalation apps if they disable Android security measures like SELinux, or abuse Android APIs in

ways that are harmful to users. The most common case we saw was abuse of device administrator privileges to prevent users from uninstalling applications. We changed Verify Apps to more effectively remove apps that abuse device administrator privileges in mid-2015.

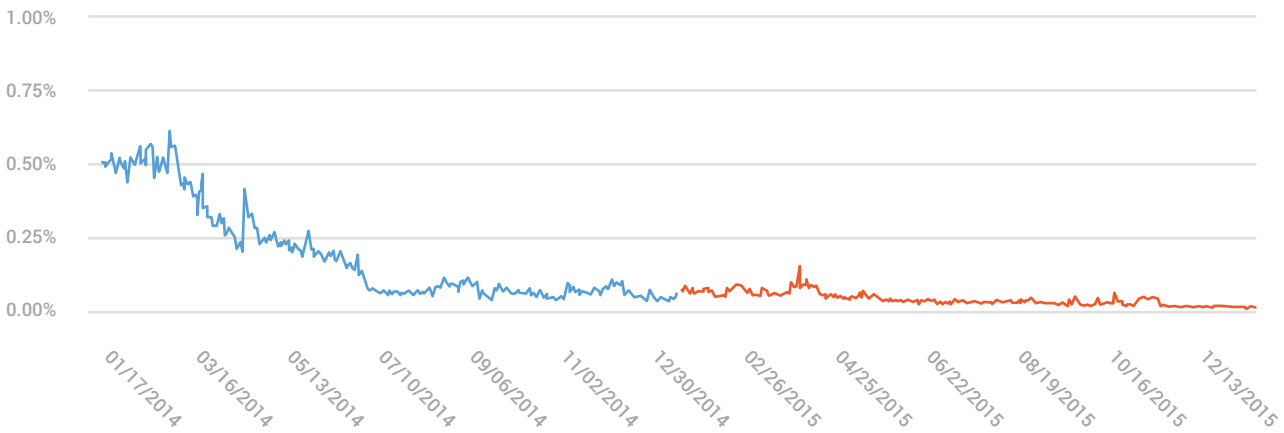
Spyware

Spyware is a category used to describe applications that attempt to take policy-protected pieces of information from a device and send it off the device without adequate consent from the user. Non-consensually collected information can lead to possible privacy violations, surveillance, or reputational harm to users. The number of installs of Spyware declined in 2015. A couple of factors contributed to this decline.

First, the Google Play policy team began enforcing more restrictive policies on advertising and tracking SDKs that collect information about users or devices. Applications and developer SDKs that collect information are required to comply with the Google Play disclosure and consent regulations if they want to stay on Google Play. To comply with Google Play policy, many developers have reduced user data collection, improved disclosure to the user about what they are collecting, or both. As a result, these SDKs are now compliant with Google Play policy and no longer classified as spyware.

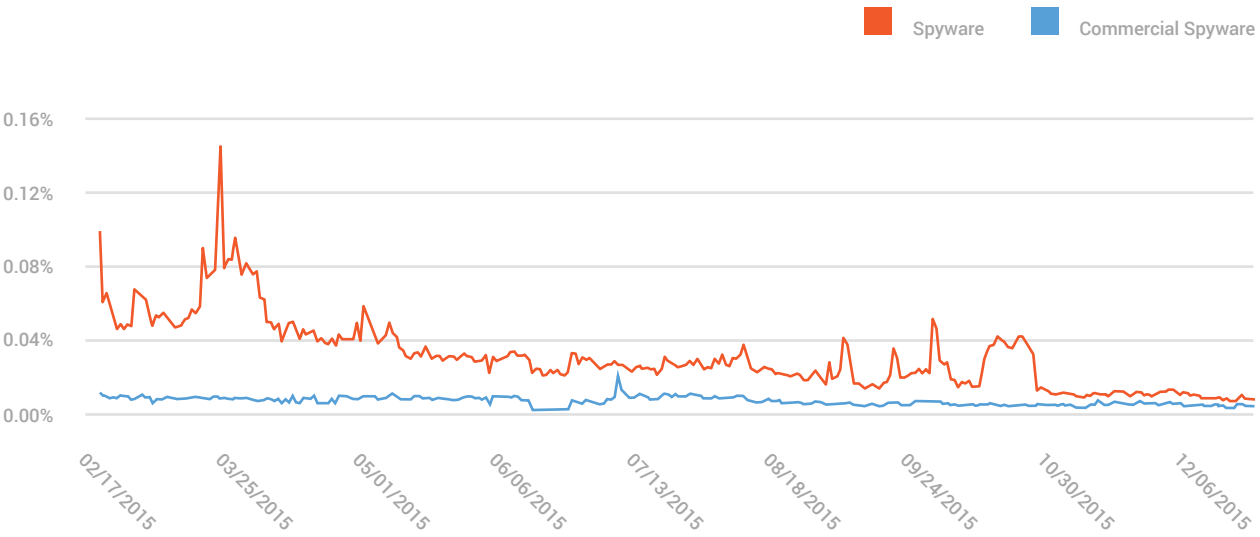
The other contributing factor may be a specialization of previous spyware apps that move them into other PHA categories. For example, we classify spyware that focuses on monitoring the spouses and acquaintances of the PHA installer as Commercial Spyware. Likewise we classify spyware that focuses on stealing login credentials as Phishing applications, and spyware that allows for remote access on devices as Backdoors. Spyware that simply collects a broad range of sensitive information is becoming exceedingly rare.

2014–2015: Spyware trends, outside of Google Play



The following graph shows a comparison between generic spyware and what we term Commercial Spyware. Commercial Spyware is when a person other than the device owner uses temporary access to the device to install spyware. Our research suggests that this is likely someone who has a personal relationship with the device owner, so we have also seen use of the phrase "spouseware" for this category of application.

2015: Spyware and Commercial Spyware trends, outside of Google Play



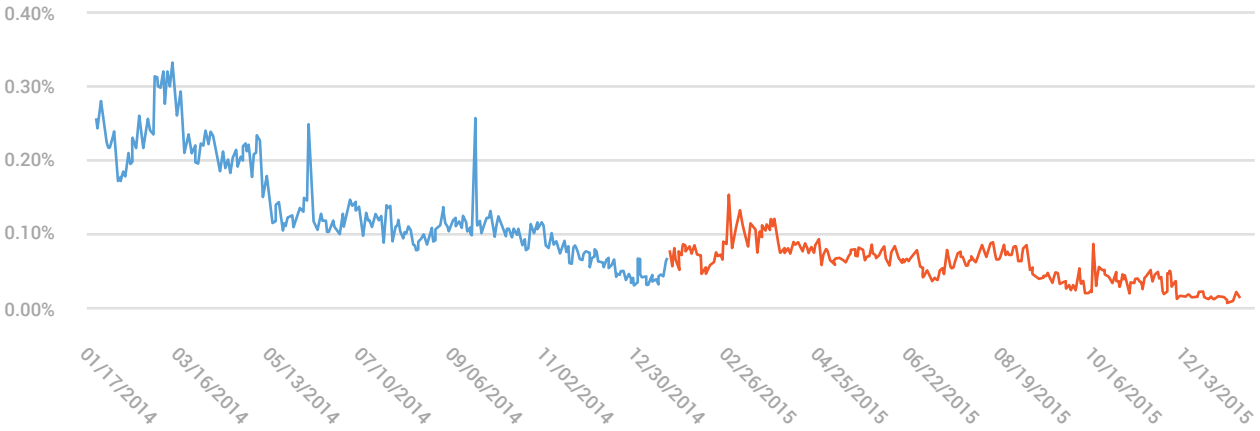
SMS and WAP Fraud

In the early years of Android, SMS fraud was the most common type of abuse that Android users were exposed to. It was a quick way to steal money from users that granted applications the SEND_SMS permission. In 2014, we introduced a new system dialog that warns users when apps want to send premium SMS, and saw the decline of SMS fraud apps. This trend continues throughout 2015. Now SMS fraud is very rare on Google Play and other app sources in nearly all parts of the world, except for a few countries. Because SMS-based payments are more common in these few remaining countries (such as Vietnam) than in other countries, users are more likely to agree to app requests to send premium SMS.

The introduction of runtime permissions in Android 6.0 added an additional hurdle to SMS fraud. In addition to the premium SMS warning dialog, there is now another dialog that asks users if they want to allow an app to send any kind of SMS. At this point there are two runtime dialogs between an attempted SMS fraud and a successful SMS fraud. We expect SMS fraud to decline further in 2016.

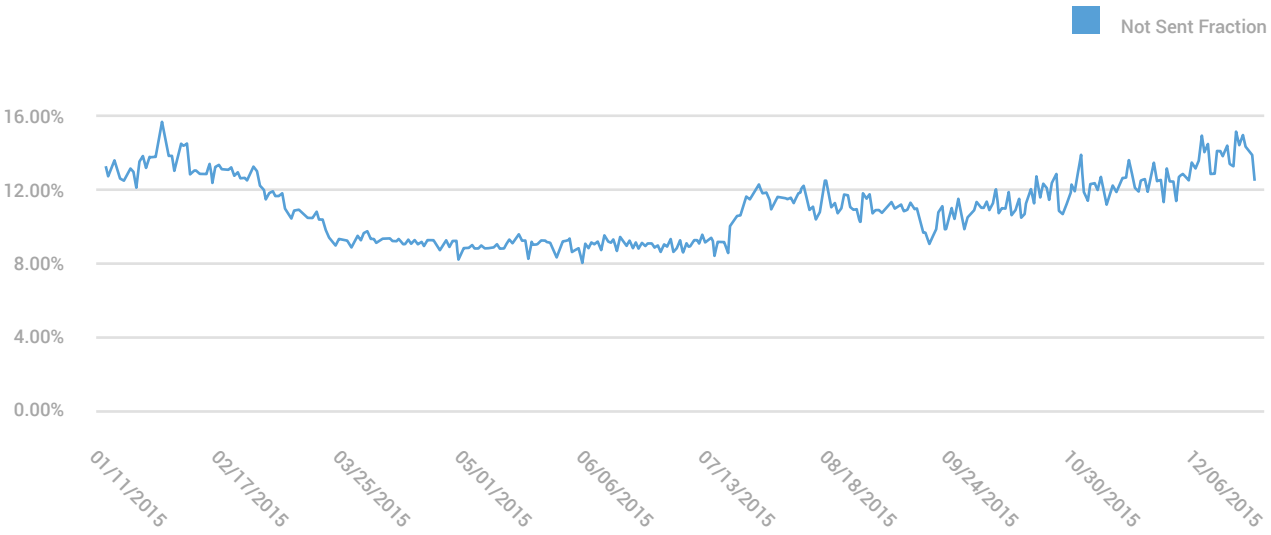
The chart below clearly illustrates the decline in SMS fraud between 2014, shown in blue, and 2015, shown in red.

2014–2015: SMS Fraud trends, outside of Google Play



As the graph below shows, worldwide around 12% of premium SMS requests were blocked by the user. This number is consistent with what we saw in 2014.

2015: Percentage of blocked SMS requests

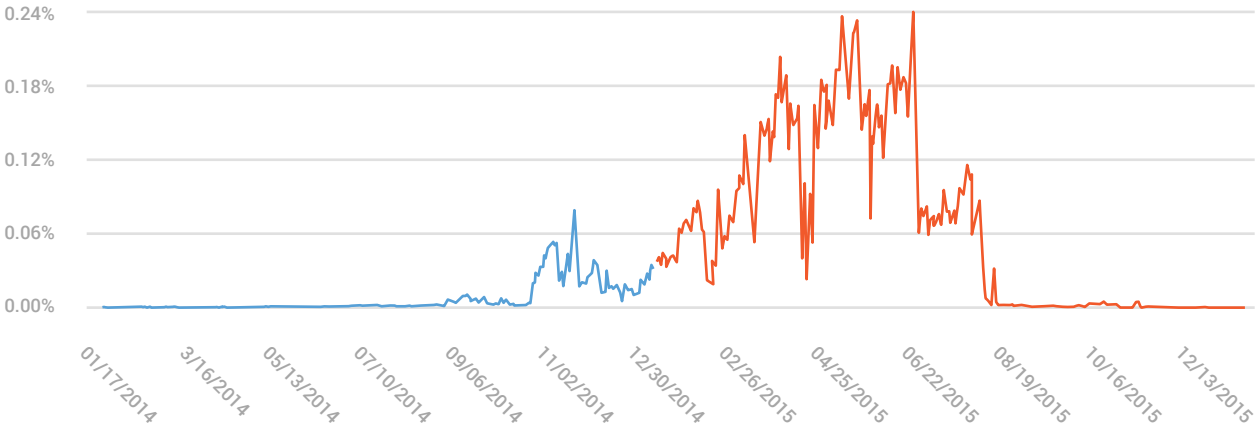


SafetyNet aggregates data that is anonymously collected about the applications that are more likely to have requests to send premium SMS be rejected by a user. This is used to identify PHAs, which are subsequently blocked by Verify Apps, or removed from Google Play.

As SMS fraud declined in popularity, another kind of toll fraud first appeared. WAP fraud is a method of abusing carrier billing services to make charges to user phone bills without user consent. Only a few countries, most notably Russia and Spain, have carriers that expose an easy way to make WAP charges from Android apps.

The charts below show a comparison of WAP fraud between 2014 and 2015. Note that the numbers tracking WAP fraud only began in October, 2014, so only Q4 2014 is represented. Because the data is lacking for Q1–Q3 of 2014, it is difficult to establish clear trends year over year, but there was a significant decline in WAP Fraud in the second half of 2015.

2014–2015: WAP Fraud trends, outside of Google Play



Ransomware

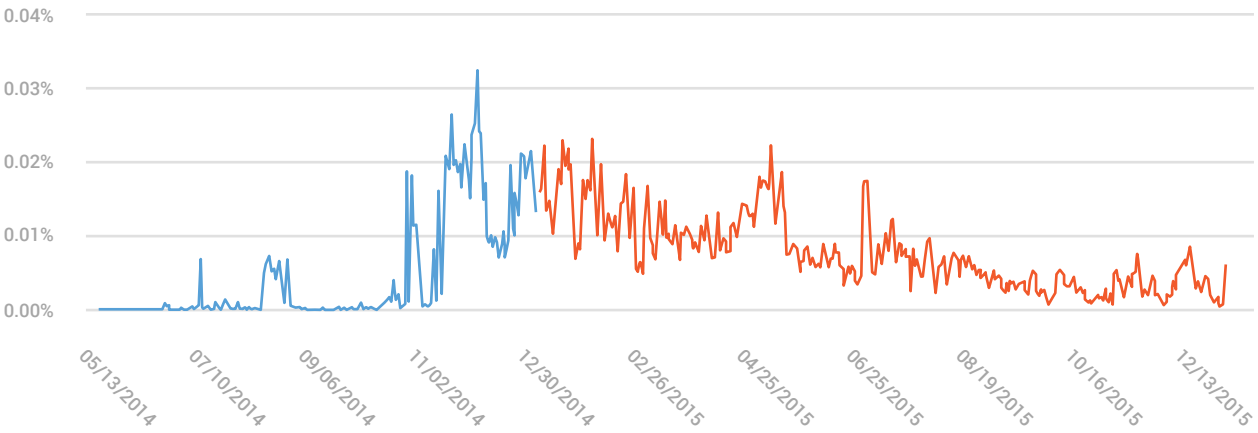
Ransomware first came to Android around the beginning of 2014. Ransomware applications take two dominant forms:

- Applications that encrypt data on the device external storage (such as an SD Card) and then demand payment to decrypt the data.
- Applications that prevent normal functioning of the device and then demand payment to regain access to the device.

At this time, ransomware remains a PHA category that is almost exclusively distributed outside Google Play. The most common distribution schemes involve tricking users into installing apps they believe are legitimate offerings. Often these are pornography apps, fake Flash player apps, or fake media player apps. A lot of ransomware also target Russian-speaking users with instructions for payment only given in Russian or using common Russian online payment methods. However, some ransomware families have localized their code to target users in other parts of the world.

Verify Apps began tracking incidents of Ransomware in mid-June 2014. Overall, Ransomware installs are less than .01% of all installs.

2014–2015: Ransomware trends, outside of Google Play



Noteworthy PHAs and Vulnerabilities

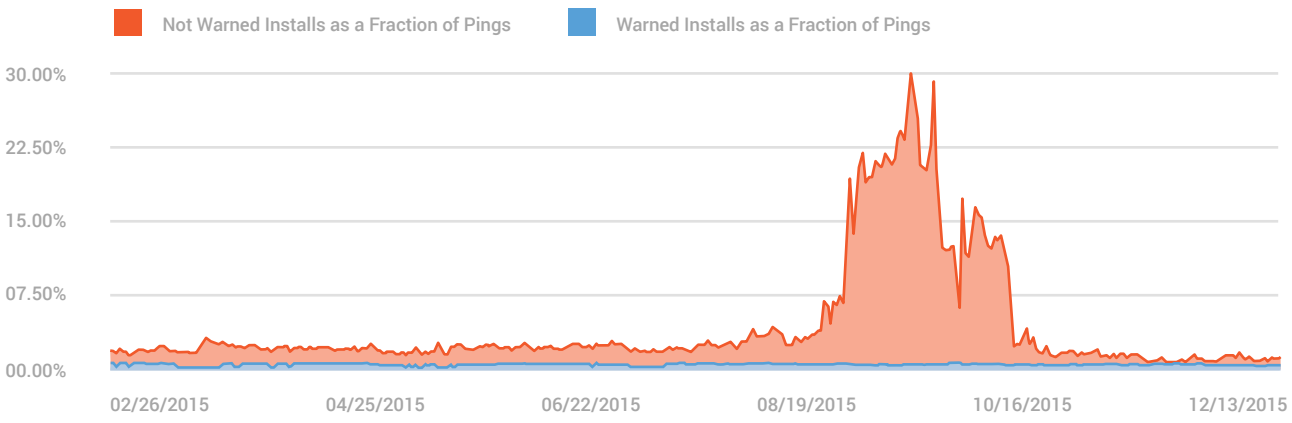
The following section includes data that was collected with Verify Apps and SafetyNet about particular security events that were prominent in 2015.

Ghost Push

Ghost Push is a family of PHA we have been monitoring since October 2014. This type of PHA is a hostile downloader that once installed attempts to download other PHAs to the device. In the summer of 2015, we saw a sudden large spike in the number of variants being deployed, which contributed to a significant overall rise in install attempts of this particular PHA family.

The shaded red portion of the graph below clearly illustrates the impact of this family. For roughly seven weeks, Ghost Push installation attempts contributed up to 30% of all installation attempts worldwide. In total, we found more than 40,000 apps that we categorized into this family and we logged more than 3.5 billion installation attempts for these apps.

2015: PHA installs—Percentage of users warned vs. not warned, outside of Google Play

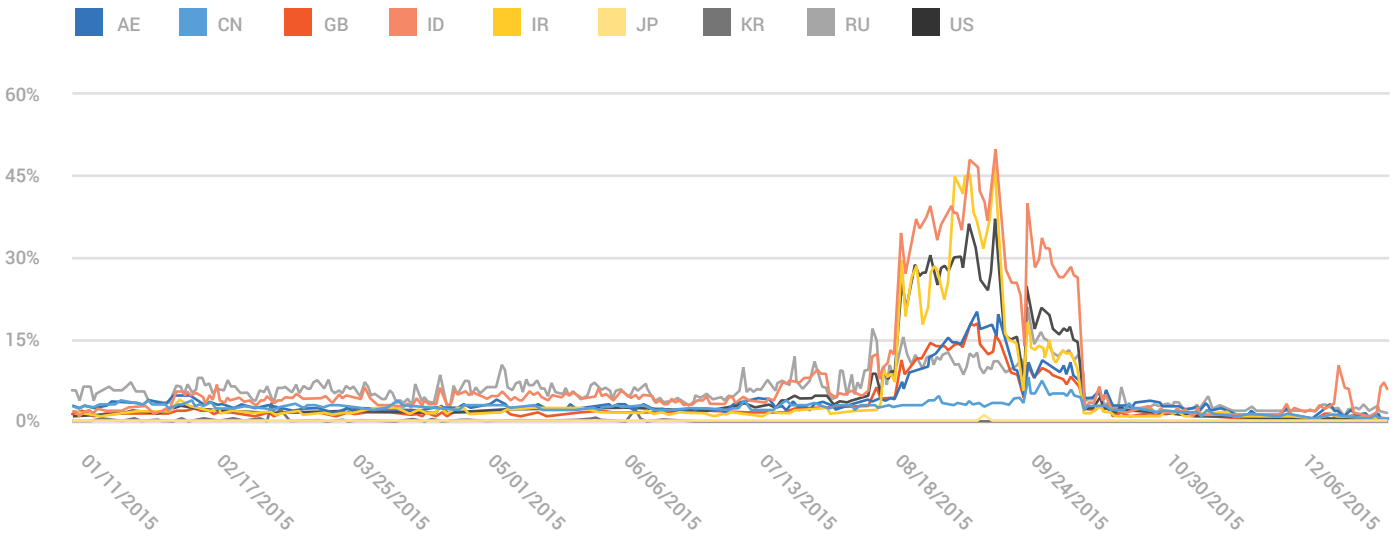


Due to the significant rate of attempted installations, we investigated the source of the installations. We discovered that many Ghost Push installation attempts pass through a company that provides OTA update infrastructure and OTA updates as a service for Android device manufacturers and carriers primarily in the Southeast Asia region. In addition to their core services around OTA updates, this company also provides an app installation service. Application developers and advertisers can pay the OTA company to remotely install applications on devices. This company installed Ghost Push onto many devices. We were able to determine that the large number of installation attempts we saw were caused by the OTA company continuously trying to install Ghost Push applications on user devices. In some instances, bugs in the application installation software caused the OTA company to try to install the same application hundreds of times onto a single device—with all but one installation attempt failing. We are working with the OTA company to develop a better security process to scan the applications they send out to devices.

Although we observed billions of installation attempts the number of affected devices was far lower than the number of installation attempts. We estimate the maximum number of affected devices to be around four million. As a result of our cleanup efforts and collaboration with other partners, the number of affected devices was quickly reduced, and has been reduced by over 90%.

The following graph shows the occurrence of Ghost Push install attempts across the top ten countries. We saw biggest impact in India and Indonesia. We attribute this to the prevalence of devices in those regions getting updates from the previously mentioned OTA company.

2015: PHA installation attempts by country



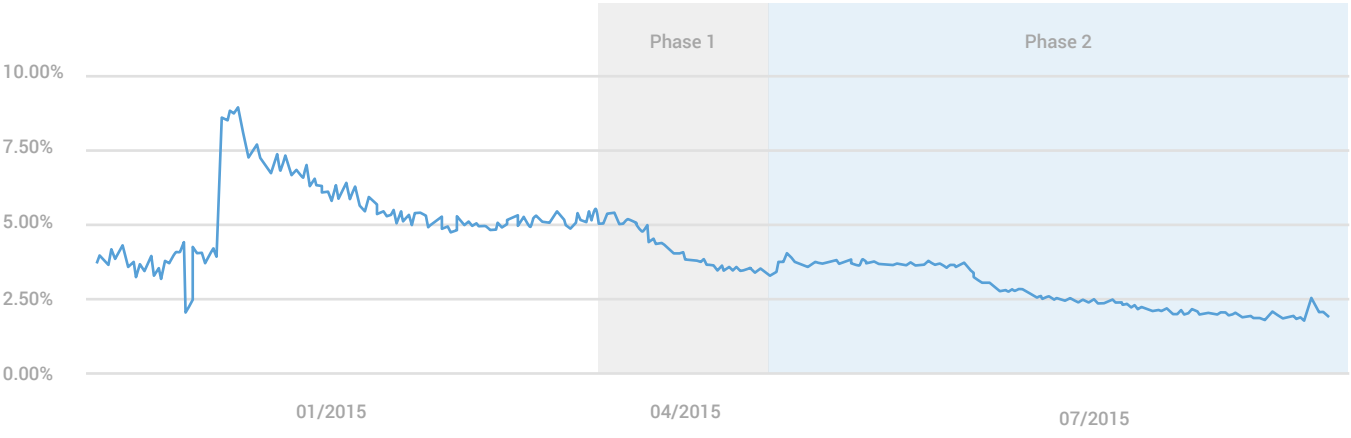
Russian banking fraud

In 2015, we focused on a family of phishing applications that targeted Russian users, specifically customers of a large Russian bank. Once installed, the applications of this family lay dormant in the background waiting for two-factor login tokens sent through SMS to user phones. We estimate the number of affected devices to have been under 100,000 at the beginning of our investigation.

In the following weeks and months, we teamed up with security engineers at the bank to identify samples from this family and remove them from users' phones. This investigation gave us the first opportunity to work with an external partner to clean up PHAs from user devices. It also allowed us to closely monitor multiple components of our user protections in one end-to-end run from application scanning in the backend to device cleanup with Verify Apps. We used a two-phased approach to target removal of this family: in the first phase, we increased the frequency of scans. In the second phase, we changed the behavior of Verify Apps to remove the applications and then notify the user of the device (the default behavior is to warn the user and allow them to determine whether to remove the application.) These two changes were very effective; after 11 weeks of focusing on this family, the number of affected devices had dropped by 80%. This decline occurred despite ongoing promotion and distribution of these PHAs on many different websites.

The following graph shows the percentage of Russian language devices that had a PHA installed, and also overlays the two phases of our targeted removal operation. In phase 1, we increased the rate of security scans. In phase 2, we maintained the elevated rate of security scans and automatically removed known PHAs in this family.

2015: Percentage of Russian devices with PHA installed



Remote vulnerabilities (Stagefright)

In 2015, we did not observe (nor did we receive reports of) any significant or widespread exploitation of remote vulnerabilities on Android devices. We did observe multiple instances of attempted exploitations of earlier remote code execution vulnerabilities affecting WebView (CVE 2012-2871). These issues were patched in 2012, but exploit code included in tools produced by the company Hacking Team became public following the unauthorized release of a large number of Hacking Team internal documents in July 2015. We believe that these exploit attempts may be successful against devices running unpatched versions of Android 4.1 and earlier.

There were several new remote code execution vulnerabilities identified in 2015, the most prominent of which were the Stagefright vulnerabilities. In late 2015, device-specific exploitation demonstration code was produced and released by the security company Zimperium. Since then we have seen a number of successful exploit proof-of-concepts demonstrated by security researchers, including Google's own Project Zero. We have also received several reports that exploits for Stagefright are included in exploit toolkits. As of this writing, we have not observed, nor are we aware of, any successful attempts to exploit the Stagefright vulnerabilities against actual user devices. We continue to monitor multiple channels for signs of widespread or targeted exploitation against user devices.

Rooting vulnerabilities

We monitor rooting vulnerabilities closely, due to their high level of potential harm if they are used maliciously. The most noteworthy non-malicious rooting application was PingPong Root, which uses a vulnerability to permanently root the device.

Note that per Google Play's policy, all rooting apps are not allowed as they compromise the device's security. SafetyNet Attest now provides an API to detect if a device is rooted.

Application vulnerabilities

At PacSec 2015, Qihoo 360 researcher Guang Gong demonstrated a vulnerability that allowed an attacker using Chrome on Android to visit an attacker-controlled website to download and install an arbitrary application on an Android device. We have made changes in Chrome and Google Play in

response to this issue, and we have not seen widespread exploitation of this issue.

At Black Hat in Las Vegas, Check Point Software revealed an exploit against the authentication methodology used by several third-party mobile Remote Support Tools (mRSTs). mRSTs are not a part of the core Android OS, and are not provided by Google. This vulnerability was named Certifigate. The vulnerability occurs with apps improperly validating the serial number on certificates, which was used to grant remote access to the device. mRSTs are frequently pre-installed on devices by manufacturers and others as a way to take remote control of a device to provide support for issues. Once we were alerted to the potential unauthorized use of this feature, we removed the apps from Google Play. We also added checks in Verify Apps to prevent potential exploitation by applications outside of Google Play. We have seen no exploitation of this vulnerability to date.

Appendix

Below is a list of links referenced in this report. These web sites provide more detailed information about the topics covered than is possible in this report.

Android 6.0 changes
http://developer.android.com/about/versions/marshmallow/android-6.0-changes.html

Android Compatibility Definition Document
http://source.android.com/compatibility/android-cdd.pdf

Android Compatibility Test Suite
http://source.android.com/compatibility/cts/index.html

Android Security Rewards program
https://www.google.com/about/appsecurity/android-rewards/

Checking device compatibility with SafetyNet
http://developer.android.com/training/safetynet/index.html

Chrome Safe Browsing on Android
https://googleonlinesecurity.blogspot.com/2015/12/protecting-hundreds-of-millions-more.html

Full disk encryption information
https://source.android.com/security/encryption/index.html

Google Vulnerability Reward program
https://www.google.com/about/appsecurity/reward-program/

How We Keep Harmful Apps Out Of Google Play and Protect Your Android Device paper
https://www.source.android.com/security/reports/Android_WhitePaper_Final_02092016.pdf

Nexus security bulletins
https://source.android.com/security/bulletin/index.html

Verified Boot information
https://source.android.com/security/verifiedboot/index.html

android