Your Voice Assistant is Mine: How to Abuse Speakers to Steal Information and Control Your Phone [*] [†]

Wenrui Diao, Xiangyu Liu, Zhe Zhou, and Kehuan Zhang
Department of Information Engineering
The Chinese University of Hong Kong
{dw013, lx012, zz113, khzhang}@ie.cuhk.edu.hk

ABSTRACT

Previous research about sensor based attacks on Android platform focused mainly on accessing or controlling over sensitive components, such as camera, microphone and GPS. These approaches obtain data from sensors directly and need corresponding sensor invoking permissions.

This paper presents a novel approach (GVS-Attack) to launch permission bypassing attacks from a zero-permission Android application (VoicEmployer) through the phone speaker. The idea of GVS-Attack is to utilize an Android system built-in voice assistant module – Google Voice Search. With Android Intent mechanism, VoicEmployer can bring Google Voice Search to foreground, and then plays prepared audio files (like "*call number 1234 5678*") in the background. Google Voice Search can recognize this voice command and perform corresponding operations. With ingenious design, our GVS-Attack can forge SMS/Email, access privacy information, transmit sensitive data and achieve remote control without any permission. Moreover, we found a vulnerability of status checking in Google Search app, which can be utilized by GVS-Attack to dial arbitrary numbers even when the phone is securely locked with password.

A prototype of VoicEmployer has been implemented to demonstrate the feasibility of GVS-Attack. In theory, nearly all Android (4.1+) devices equipped with Google Services Framework can be affected by GVS-Attack. This study may inspire application developers and researchers to rethink that zero permission doesn't mean safety and the speaker can be treated as a new attack surface.

Categories and Subject Descriptors

D.4.6 [**Operating System**]: Security and Protection—*Invasive software*

[*]Responsible disclosure: We have reported the vulnerability of Google Search app and corresponding attack schemes to Google security team on May 16th 2014.

[†]Demo video can be found on the following website: https://sites.google.com/site/demogvs/

SPSM'14, November 7, 2014, Scottsdale, Arizona, USA.

ACM 978-1-4503-2955-5/14/11 ...$15.00.
http://dx.doi.org/10.1145/2666620.2666623.

General Terms

Security

Keywords

Android Security; Speaker; Voice Assistant; Permission Bypassing; Zero Permission Attack

1. INTRODUCTION

In recent years, smartphones are becoming more and more popular, among which Android OS pushed past 80% market share [32]. One attraction of smartphones is that users can install applications (*apps* for short) as their wishes conveniently. But this convenience also brings serious problems of malicious application, which have been noticed by both academic and industry fields. According to Kaspersky's annual security report [34], Android platform attracted a whopping 98.05% of known malware in 2013.

Current Android phones are equipped with several kinds of sensors, such as light sensor, accelerometer, microphone, GPS, etc. They enhance the user experience and can be used to develop creative apps. However, these sensors could be utilized as powerful weapons by mobile malware to steal user privacy as well. Taking the microphone as an example, a malicious app can record phone conversations which may contain sensitive business information. With some special design, even credit card and PIN number can be extracted from recorded voice [46].

Nearly all previous sensor based attacks [46, 40, 13, 10, 47, 49, 30] only considered using input type of components to perform malicious actions, such as accelerometer for device position analysis and microphone for conversation recording. These attacks are based on accessing or controlling over sensitive sensors directly, which means corresponding permissions are required, such as `CAMERA` for camera, `RECORD_AUDIO` for microphone, and `ACCESS_FINE_LOCATION` for GPS. As a matter of fact, output type of permission-free components (e.g. speaker) can also be utilized to launch attacks, namely an indirect approach.

Consider the following question:

> *Q: To a zero-permission Android app that only invokes permission-free sensors, what malicious behaviors can it do?*

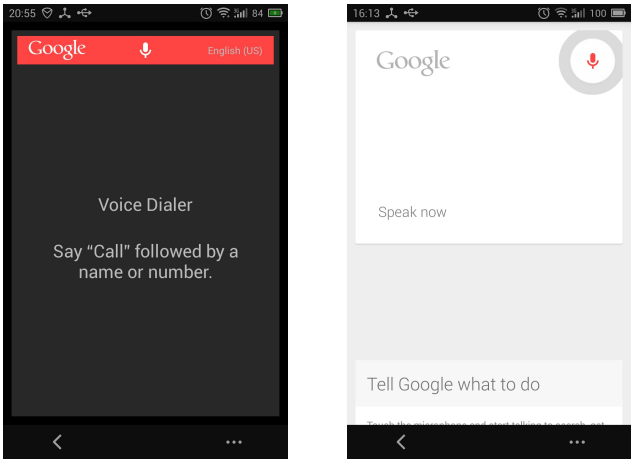
In general, zero permission means harmlessness and extremely limited functionality. However, our research results show that:

> *A: Via the phone speaker, a zero-permission app can make phone calls, forge SMS/Email, steal personal schedules, obtain user location, transmit sensitive data, etc.*

This paper presents a novel attack approach based on the speaker and an Android built-in voice assistant module – Google Voice Search (so we call it GVS-Attack). GVS-Attack can be launched

by a **zero-permission** Android malware (called VoicEmployer correspondingly). This attack can achieve varied malicious goals by bypassing several sensitive permissions.

GVS-Attack utilizes system built-in voice assistant functions. Voice assistant apps are hands-free apps which can accept and execute voice commands from users. In general, typical voice commands include dialing (like "*call Jack*"), querying (like "*what's the weather tomorrow*") and so on. Benefited from the development of speech recognition and other natural language processing techniques [16], voice assistant apps can facilitate users' daily operations and have become an important selling point of smartphones. Every mainstream smartphone platform has its own built-in voice assistant app, such as Siri [8] for iOS and Cortana [36] for Windows Phone. On Android, this function is provided by Google Voice Search[1] (see Figure 1) which has been merged into Google Search app as a sub-module starting from Android 4.1. Users can start Google Voice Search through touching the microphone icon of Google Search app widget, or the shortcut named Voice Search, etc.



(a) Voice Dialer Mode (b) Velvet Mode

Figure 1: Google Voice Search on Android

The basic idea of GVS-Attack is to utilize the capability of Google Voice Search. Through Android Intent mechanism, VoicEmployer can bring Google Voice Search to foreground, and then plays an audio file in the background. The content of this audio file could be a voice command, such as "*call number 1234 5678*", which will be recognized by Google Voice Search and the corresponding operation would be performed. By exploiting a vulnerability of status checking in Google Search app, zero permission based VoicEmployer can dial arbitrary malicious numbers even when the phone is securely locked. With ingenious design, GVS-Attack can forge SMS/Email, access privacy information, transmit sensitive data and achieve remote control without any permission. What is more, context-aware information can be collected and analyzed to assist GVS-Attack, which makes this attack more practical.

It is difficult to identify the malicious behaviors of VoicEmployer through current mainstream Android malware analysis techniques. From the aspect of permission checking [9, 22], VoicEmployer doesn't need any permission. From the aspect of app dynamic behavior analysis [21, 53], VoicEmployer doesn't perform malicious actions directly. The actual executor is Google Voice Search, a "trusted" system built-in app module. In addition, the voice commands are outputted from the speaker and captured by the microphone as input. Such inter-application communication channel is beyond the control of Android OS. We tested several famous anti-virus apps (AVG, McAfee, etc.) to monitor the process of GVS-Attack. None of them can detect GVS-Attack and report VoicEmployer as malware.

Contributions. We summarize this paper's contributions here:

- *New Attack Method and Surface.* To the best of our knowledge, GVS-Attack is the first attack method utilizing the speaker and voice assistant apps on mobile platforms. Besides, this attack can be launched by a totally zero-permission Android malware that can perform many malicious actions through sensitive permission bypassing, such as malicious number dialing, SMS/Email forging and privacy stealing.

- *New Vulnerability.* We found a vulnerability of status checking in Google Search app (over 500 million installations), which can be utilized by GVS-Attack to dial arbitrary malicious numbers even when the phone is securely locked with password.

- *Prototype Implementation and Evaluation.* We implemented a VoicEmployer prototype and carried out related experiments to demonstrate the feasibility of GVS-Attack. In addition, We designed and tested some attack assisting schemes, such as context-aware information analysis, sound volume setting, etc.

Roadmap. The rest sections are organized as follows: Section 2 introduces Google Voice Search related contents, including backgrounds, vulnerability analysis and the adversary model. The details of GVS-Attack are described in Section 3, which contains three different levels of attacks. In Section 4, a VoicEmployer prototype was implemented and related experiments were carried out. Section 5 and Section 6 discuss corresponding defense strategies and some related in-depth topics respectively. Previous research about sensor based attacks and Android app security analysis are reviewed in Section 7. Section 8 concludes this paper.

2. GOOGLE VOICE SEARCH ON ANDROID

2.1 Backgrounds

Google Services Framework. Google Services Framework / Google Mobile Services are pre-installed on nearly all brands of Android devices. It can be treated as a suit of pre-installed apps developed by Google, including Google Play, Gmail, Google Search, etc. [26] These killer apps (Google Search app has over 500 million installations just on Google Play) are so popular that even customized Android firmwares would keep them. Taking CyanogenMod[2] as an example, due to licensing restrictions, Google Services Framework cannot come pre-installed with CyanogenMod, but these apps can still be installed via separate Google Apps recovery package [17].

Android Intent Mechanism. In Android OS, Intent mechanism allows an app to start an activity / service in another app by describing a simple action, like "view map" and "take a picture" [1]. An Intent is a messaging object which declares a recipient (and

[1]Some people confuse Google Now with Google Voice Search. In fact, Google Voice Search module can work independently when Google Now is off. See http://www.google.com/landing/now/

[2]http://www.cyanogenmod.org/

contains data). VoicEmployer utilizes this mechanism to invoke Google Voice Search module of Google Search app. From the view of Android OS, it only executes a normal operation to invoke a system built-in app module.

An intent filter is an expression in an app's manifest file that specifies the type of intents that the component would like to receive [1]. For example, the manifest file of Google Search app defines that hands-free function (voice commands) related components can handle two kinds of actions, namely `ACTION_VOICE_COMMAND` [2] and `ACTION_VOICE_SEARCH_HANDS_FREE` [4].

2.2 Google Search App Vulnerability Analysis

Google Voice Search is a voice assistant module of Google Search app. It is designed for hands-free operations and can accept several kinds of voice commands, such as "*call Jack*" and "*what's the weather tomorrow*". Google Voice Search runs in two modes: Voice Dialer – Figure 1(a) and Velvet – Figure 1(b). Voice Dialer mode only accepts voice dialing commands and Velvet mode is the fully functional mode. Google Voice Search can be invoked through Android Intent mechanism. First, a third-party app constructs an Intent based on `ACTION_VOICE_COMMAND` or `ACTION_VOICE_SEARCH_HANDS_FREE`, and then passes it to `startActivity()`. Android OS resolves this Intent and finds Google Search app can handle it. Then this Intent is passed to Google Search app. According to the current phone status[3], different modes will be started by Google Search app:

1. If the phone is unlocked and the screen is on, Velvet mode will be started.
2. If the phone is insecurely locked, namely sliding to unlock the screen without authentication, Voice Dialer mode will be started.
3. If the phone is securely locked, such as using password and pattern password, a voice warning will be played: "*please unlock the device*".

Vulnerability in Google Search App. In fact, Voice Dialer mode can be started even if the phone is securely locked. But this function is only designed for the Bluetooth headset hands-free mode. When a user presses the button on his connected Bluetooth headset for certain time, this event can trigger passing an `ACTION_VOICE_COMMAND` based Intent to the OS if the phone is not in call process. Then Voice Dialer mode will be activated. This usage scenario is reasonable, because the user must first confirm the connection (pairing) request of the Bluetooth headset on his phone, which is a process of authorization, and only after that, this Bluetooth headset is treated as a trusted device. In Android source codes, the file `HeadsetStateMachine.java` defines and handles related operations [6].

However, we found a vulnerability that Google Search app doesn't strictly check whether the phone is connected with a Bluetooth headset. This vulnerability results in that a third-party app can pass an `ACTION_VOICE_COMMAND` based Intent to the OS and activate Voice Dialer mode of Google Voice Search, even when the phone is securely locked. Through decompiling the apk file of Google Search app (version 3.3.11.1069658.arm[4], released on March 14th 2014), we analyzed related codes and located this vulnerability. Figure 2 shows the simplified logical flow of Google Search app handling voice assistant type of Intents. The area bounded by the dotted line is the location of this vulnerability.

[3]The code implementations of corresponding status checking are: `KeyguardManager.isKeyguardLocked()`, `KeyguardManager.isKeyguardSecure()` and `PowerManager.isScreenOn()`.

[4]Following versions have been applied code obfuscation techniques, and this vulnerability still exists.

2.3 Adversary Model

GVS-Attack needs to be launched by an Android malware (called VoicEmployer) which has been installed on the user's phone. This malware could disguise as a normal app and contains attack modules. Since VoicEmployer doesn't need any permission, disguising should be quite easy. After VoicEmployer is opened by the user (namely the victim) once, subsequent attack processes need no interaction with the victim. The attack scene is when the victim is not using his phone.

Assumption. The victim's Android (4.1+) phone should contain complete Google Services Framework.

The reason is that VoicEmployer needs to invoke Google Voice Search (and Google speech recognition / synthesis service) to execute attacks. This assumption is quite weak, because most smartphone vendors pre-install Google Services Framework on their products. In order to make GVS-Attack work well, the victim's phone should use Android 4.1 or higher versions. Google Search app running on Android 4.0 or lower versions cannot be updated to the newest version and lacks many new features of voice commands. In August 2014, Android 4.1 or upper versions had more than 75% device share distribution [3].

Secure Lock. The victim's Android phone could be securely locked or not. GVS-Attack contains three different levels: Basic Attack, Extended Attack and Remote Voice Control Attack. The secure lock status corresponds to Basic Attack and the insecure lock status corresponds to all levels of attacks.

Basic Attack achieves arbitrary number dialing. In order to launch Extended Attack and Remote Voice Control Attack, there is an additional requirement – the victim doesn't enable secure screen lock functions, such as password and pattern password. Google Voice Search cannot run in Velvet mode without unlocking the screen. In fact, for convenience, many users (more than 30%) would give up secure screen lockers [41].

Zero Permission. GVS-Attack can be launched by VoicEmployer **without any permission**. Android OS uses a permission mechanism to enforce restrictions on the specific operations. The permission abusing problem has been noticed long before. Malicious apps utilize some sensitive permissions (such as `CAMERA`, `RECORD_AUDIO` and `READ_CONTACTS`) to collect user privacy and achieve other illegal targets. Since this problem is so widespread, some users will pay attention to unknown or new apps requiring sensitive permissions [23].

Instead of requesting these permissions explicitly, our attacks only use the speaker to perform sensitive actions protected by permissions. According to the current Android permission mechanism, playing audio doesn't require any permission.

Context-aware Information Collection and Analysis. Since voice commands played may be heard and interrupted by the victim, VoicEmployer needs to analyze the current environment in order to decide whether to launch attacks or not, that is to collect information through sensors and legal Android API invoking. On mobile platforms, the feasibility of context-aware information analysis has been demonstrated in several previous research [25, 35, 46]. As a matter of fact, this target could be achieved **without any permission**. For example, light sensor and accelerometer can be used to analyze the surrounding environment. And the internal

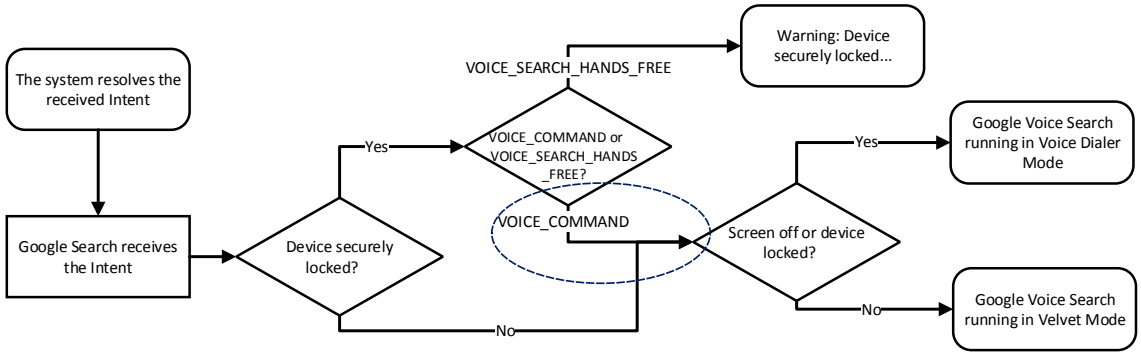


Figure 2: The Vulnerability of status checking in Google Search app

status of the phone can be checked by analyzing CPU workload, memory workload, etc.

Typical Attack Scene. Due to job requirements or merely personal habits, many people don't turn off their phones when sleeping [48]. In the early morning (before dawn, such as 3 AM), the victim is likely to be in deep sleep and the environment is quiet. The voice commands could be played in a very low volume and still be recognized by Google Voice Search. According to related medical research, nocturnal awakenings usually occur with noise levels greater than 55 dB[5] [39]. It means voice commands in low volume will not awaken the victim. This situation provides an ideal scenario for GVS-Attack launching.

3. ATTACKS

The basic idea behind GVS-Attack is to utilize the capability of Google Voice Search and bypass Android permission checking mechanism. There are two highlights in our attacks:

Attack in Any Situation. By exploiting the vulnerability of status checking in Google Search app, zero-permission VoicEmployer can dial arbitrary malicious numbers even when the phone is securely locked.

Remote Data Transmission and Voice Control. Through call channel, sensitive data can be transmitted without the INTERNET permission. Also the attacker can control the victim's Android phone remotely.

GVS-Attack provides a **new inter-application communication channel** for mobile malware attacks. In the process of GVS-Attack, the input of microphone comes from the output of speaker (VoicEmployer). This information transmission process is beyond the control of Android OS, which can be treated as an uncontrolled physical communication channel or a kind of covert channel. Figure 3 shows this communication channel. This new attack channel is totally different from previous research on security of application communications [33, 15].

In the following subsections, three types of attacks are described in details. Basic Attack can dial arbitrary numbers. Extended Attack can bypass several sensitive permissions. Combined previous two attacks, the attacker can interact with the phone, which leads to Remote Voice Control Attack.

[5]Actually the noise-induced nocturnal awakening is affected by many factors, such as sleep stages, age, gender, smoking, etc. Also sensitivity to noise may vary greatly from one individual to another. These topics are beyond the scope of this paper. More details see the guideline document [31] of World Health Organization (WHO).

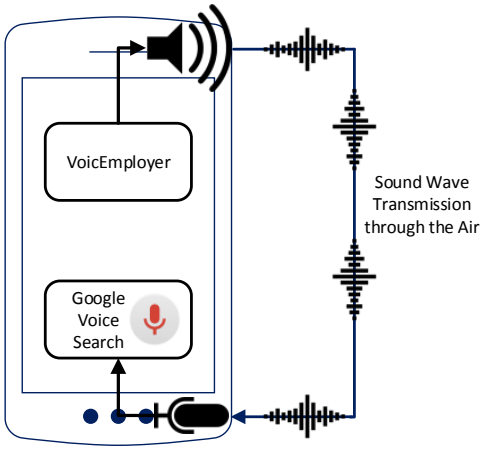


Figure 3: Inter-Application Communication Channel of GVS-Attack

3.1 Basic Attack: Malicious Number Dialing

Basic Attack achieves arbitrary malicious number dialing. That is to say the `CALL_PHONE` permission is bypassed. The vulnerability of Google Search app makes this attack still valid when the phone is securely locked. Without unlocking the screen keyguard, Google Voice Search would run in Voice Dialer mode which only accepts dialing commands, as Figure 1(a) shows. Before launching attacks, we answer two questions first: how to invoke Google Voice Search and how to prepare voice command files.

Google Voice Search Invoking. When both the surrounding environment and phone internal status reach the predefined threshold, VoicEmployer will invoke Google Voice Search through Android Intent mechanism. In fact, VoicEmployer only needs to construct an Intent based on `ACTION_VOICE_COMMAND`, and passes it to `startActivity()`.

Voice Command Files Preparation. After Google Voice Search starting, VoicEmployer plays an audio file (voice commands) as a background service. These files could be packaged in VoicEmployer as a part of apk resource, but this method would increase the size of installation package. A more covert solution is to utilize Google Text-to-Speech (TTS) [5] service which is a system built-in service used for translating text contents to speech. TTS service is based on Internet, but it doesn't require that the app invoking TTS declares

the INTERNET permission. Therefore VoicEmployer can execute speech synthesis and save results as an audio file in its own data folder (internal storage) dynamically.

Attack Launching. In previous preparations, VoicEmployer synthesizes two audio records like "*call 1234 5678*" and "*OK*". 1234 5678 could be a malicious number.

When this attack is launched, Voice Dialer mode of Google Voice Search is activated and VoicEmployer plays the audio record – "*call 1234 5678*". Then Voice Dialer recognizes this command and returns a voice feedback "*do you want to call 1234 5678, say OK or cancel*". VoicEmployer plays another audio record "*OK*". After receiving this confirmation, Voice Dialer makes a call to 1234 5678. Since 1234 5678 is a malicious number, the victim will be charged with premium-rate service fee. What is more, his phone number is leaked as well.

3.2 Extended Attack: Sensitive Permission Bypassing

Extended Attack needs to invoke Velvet mode of Google Voice Search, as Figure 1(b) shows. Therefore, the precondition is that the victim doesn't enable secure screen lock functions (password, pattern password, etc.). Nonetheless, VoicEmployer must wake the phone and turn on the screen itself.

In general, an app requires the WAKE_LOCK permission to wake the phone from sleep status, and then sets flag LayoutParams.FLAG_DISMISS_KEYGUARD to unlock the screen keyguard. Because WAKE_LOCK is not a sensitive permission and doesn't correlate with user data, so this requirement is not exorbitant. Actually there is a tricky implementation to bypass this permission. That is, VoicEmployer invokes Google Voice Search once and does nothing until it exits for timeout. At this moment, the phone has been in waked status (the screen is on) and the keyguard has be dismissed. As a result, VoicEmployer utilizes the WAKE_LOCK permission of Google Search app to wake the phone indirectly.

User Sensitive Data Collection. Running in Velvet mode, Google Voice Search can accept more types of voice commands [27], not just dialing commands. Different voice commands cause different information leakages. Action commands lead to that specific operations are performed, such as sending SMS. Other querying commands will trigger voice feedbacks, such as "*what is the time?*" will get "*the time is 9:39 pm*". Typical information leakages include (*sentences in italic* are voice commands or voice feedbacks):

- "*Email to [contacts], subject "meeting cancel", message "tomorrow's meeting has been canceled".*" This command results in sending an Email to the contacts with the above subject and message. It can be used to forge Emails with any contents.
- "*Send SMS to number 1234 5678 "confirm subscribe to weather forecast service".*" This command results in sending an SMS to number 1234 5678. It can be used to forge SMS and subscribe to premium-rate services.
- "*What is my next meeting?*" ⇒"*Your next calendar entry is tomorrow 10 AM. The tile is "Meet with boss".*" The victim's calendar schedule is leaked through voice feedback.
- "*What is my IP address?*" ⇒ "*Your public IP address is 111.222.111.222.*" The victim's IP address is leaked through voice feedback.
- "*Where is my location?*" ⇒ "*Here is a map of Brooklyn District.*" The victim's location (district level) is leaked through voice feedback.

It is equivalent to that VoicEmployer utilizes the capability of Google Voice Search and bypasses Android permission checking mechanism. From Android OS's view, VoicEmployer just passes an Intent and plays some audio files. Table 1 shows the permissions which can be bypassed by GVS-Attack. Some voice commands are not listed, because they need interaction (touching the screen) with the user, such as "*create a calendar event ...*" and "*post to Google plus ...*".

Table 1: Permissions Bypassed by GVS-Attack

Voice Command	Bypassed Permission(s)
Call ...	READ_CONTACTS, CALL_PHONE
Listen to voicemail	WRITE_SETTINGS, CALL_PHONE
Browse to Google dot com	INTERNET
Email to ...	READ_CONTACTS, GET_ACCOUNTS, INTERNET
Send SMS to ...	READ_CONTACTS, WRITE_SMS, SEND_SMS
Set alarm for ...	SET_ALARM
Note to myself ...	GET_ACCOUNTS, RECORD_AUDIO, INTERNET
What is my next meeting?	READ_CALENDAR
Show me pictures of ...	INTERNET
What is my IP address?	ACCESS_WIFI_STATE, INTERNET
Where is my location?	ACCESS_COARSE_LOCATION, INTERNET
How far from here to ...?	ACCESS_FINE_LOCATION, INTERNET

Remote Data Transmission. GVS-Attack can dial a malicious number through playing "*call ...*". When the call is answered by an auto recorder, the data transmission channel has been built. Any audio data can be transferred through this channel instead of using Internet connection.

Google Voice Search running during phone call period. In Android OS, the audio recording method is synchronized, which means multiple apps cannot access the microphone at the same time on API level. However, as an essential system module, the phone call function of Android is based on hardware-level implementations instead of invoking MediaRecorder / AudioRecord class. Android OS simply needs to send control signals and related hardware (GSM module, microphone, speaker, etc.) will complete audio data processing functions directly [7]. Therefore during the phone call period, another app can also access the microphone through Android API. It means Google Voice Search can still be brought to the foreground to accept voice commands[6].

As a result, in terms of querying commands, the feedback voice can be obtained by the attacker through this call channel. Based on speech recognition techniques, the attacker can obtain corresponding text information. To be specific, VoicEmployer

[6]Google Voice Search is an Internet based service. With 3G/4G or Wi-Fi data connection, the phone can make phone call and access the Internet at the same time. To 2G data connection (GPRS and EDGE), the two functions are conflicted.

invokes Google Voice Search and then plays voice command "*call number 1234 5678*". After connected, VoicEmployer invokes Google Voice Search and plays querying commands again, like "*where is my location*". The feedback voice – "*here is a map of Sha Tin district*" is transmitted to the attacker through the call channel.

If VoicEmployer could get more permissions, more sensitive information would be leaked. A natural idea is to play audio records on external storage directly. These audio records are recorded by the victim and probably contain sensitive information, like a business negotiation. This operation requires the `READ_EXTERNAL_STORAGE` or `WRITE_EXTERNAL_STORAGE` permission. Both of them allow an app to read all data on external storage. For text files, they can be converted to audio data through TTS service. For example, with the `READ_CONTACTS` permission, VoicEmployer can get phone contacts of the victim. Then it utilizes TTS service to convert the text into voice. So the attacker can obtain a complete copy of the victim's contacts through the call channel. Similar risks happen in `READ_CALL_LOG`, `READ_SMS`, etc.

The voice channel is not used for transferring text and voice data only. Actually, any type of files can be translated to hex coding format, so any file could be transmitted in the form of audio coding (or even read hex codes directly) in theory. When the attacker receives all hex codes of a photo through the call channel, he can restore it easily. If this photo contains sensitive information, such as selfie, it will be quite harmful to the victim. In practical attacks, compression encoding and error checking mechanism should be considered.

3.3 Remote Voice Control Attack: Real World Case Study

Remote Voice Control Attack combines Basic Attack and Extended Attack. This attack also needs to invoke Velvet mode of Google Voice Search.

In Remote Voice Control Attack, after VoicEmployer triggering a malicious number dialing, the attacker answers this call directly. During the calling period, VoicEmployer invokes Google Voice Search periodically. When the attacker speak some voice commands, these commands will be played by the speaker (headset) of victim's phone. Then Google Voice Search recognizes the commands and perform corresponding operations. It means that the attacker can interact with the victim's phone through Google Voice Search, namely remote voice control.

Previous research [55] showed more than 90% of Android malware would turn the compromised phones into a botnet controlled through network or short messages. These controls were based on corresponding communication permissions (such as `INTERNET` and `SEND_SMS`) or even root privilege. But our remote control method utilizes the capability of Google Voice Search to build the communication channel and to access sensitive data. These targets are completed without any permission.

With special design, Remote Voice Control Attack could become quite powerful. One example of voice interaction is the command "*How far from here to Lincoln Memorial by car?*". Google Voice Search provides a voice feedback like "*The drive from your location to Lincoln Memorial is 17.6 kilometers*". Then the attacker can use the successive approximation method to ask similar questions of different locations until he gets a feedback like "*The drive from your location to White House is 120 meters*". At this moment, the attacker gets the accurate location of the victim, which is quite dangerous.

Another example is that the attacker can leave a note to the victim using the command "*Note to self: You have been hacked*". Google Voice Search would make a note (send an Email to the victim) with the content "*You have been hacked*". In the morning, after the victim get up, he will find such a strange note on his phone. Based on this thought, some complex social engineering attacks [29] could be designed and launched.

4. IMPLEMENTATION AND ANALYSIS

4.1 Overall Structure and Attack Experiments

We implemented a VoicEmployer prototype to demonstrate our attack schemes. Our implementation contains 5 modules: MainActivity, AlarmReceivor, EnvironmentService, WakedActivity and VoiceCommandService.

- **`MainActivity`** shows the normal starting UI. Its main functions are registering an alarm and preparing audio files of voice commands.
- **`AlarmReceivor`** is used to receive alarm events and start EnvironmentService.
- **`EnvironmentService`** is used for context-aware information collection and analysis. Range of analysis include light sensor, accelerometer, `/proc/`, system workloads, etc. If the analysis results reach a predefined threshold, WakedActivity will be started. Related implementation details are described in Section 4.2.
- **`WakedActivity`** is used to unlock insecure screen keyguard through setting `LayoutParams.FLAG_DISMISS_KEYGUARD`. This function can only be implemented in Activity components. Without dismissing screen keyguard, Velvet mode cannot be started. *Note: This module is not necessary for Basic Attack.*
- **`VoiceCommandService`** is used to invoke Google Voice Search and play audio files. Playing Voice command will be delayed a short while in order to wait Google Voice Search getting ready. In terms of Remote Voice Control Attack, Google Voice Search will be invoked periodically.

Taking Remote Voice Control Attack as an example, Figure 4 shows the workflow of VoicEmployer implementation. It utilizes the alarm mechanism to trigger attack related modules to avoid continuous background services.

Attack Results. The experimental versions of Google Search app were 3.3.11.1069658.arm and 3.4.16.1149292.arm, which were released on March 14th 2014 and May 6th 2014 respectively. We tested VoicEmployer prototype on Samsung Galaxy S3 GT-I9300, Meizu MX2 and Motorola A953. GVS-Attack can be launched on all of them successfully. On Samsung Galaxy S3 with official firmware, S Voice app [44] (a built-in voice assistant developed by Samsung) would be started, not Google Voice Search, but attack processes are similar. This change derives from that Samsung sets a higher priority in the manifest file of S Voice app to respond `ACTION_VOICE_COMMAND` and `ACTION_VOICE_SEARCH_HANDS_FREE`. Table 2 summarizes the experiment results.

Speech Recognition Accuracy. We consider Command Success Rate (CSR) instead of Word Error Rate (WER) [45], because the executing results are what we only care. Results showed CSR was nearly 100%. The reasons are as follows:: 1. the attack happens in a quiet environment without interfering noise; 2. voice command sentences are simple, which don't lead to ambiguity;

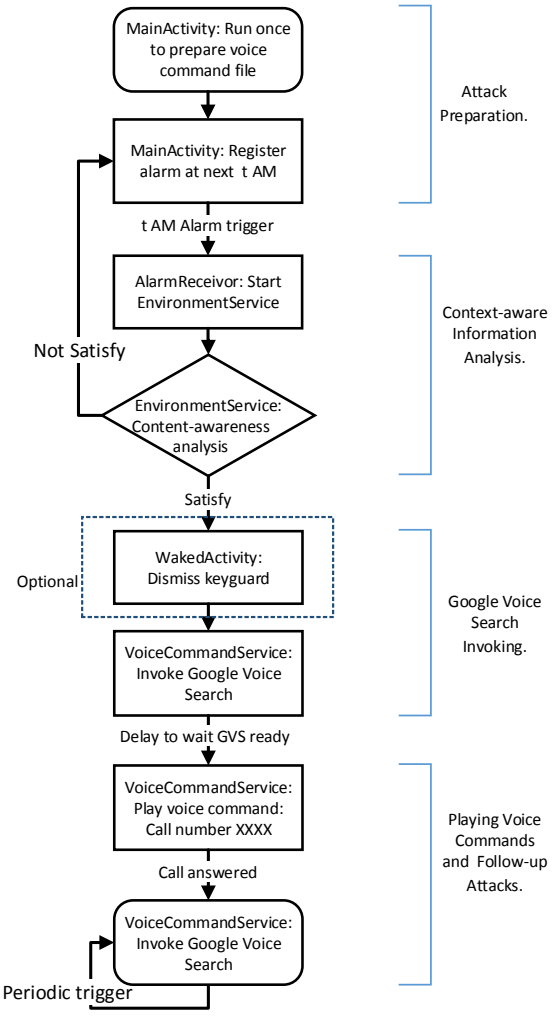


Figure 4: Workflow of VoicEmployer Implementation

3. VoicEmployer utilizes Google TTS to synthesize the command speech, which removes accent and pronunciation problems.

Detection by Anti-Virus Apps. We tested the following anti-virus apps for Android platform: AVG AntiVirus[7], McAfee Antivirus & Security[8], Avira Antivirus Security[9], ESET Mobile Security & Antivirus[10] and Norton Mobile Security[11]. After executing threat scanning, none of them reported VoicEmployer prototype as a malware. Also in the attack processes, none of their real-time monitoring modules can detect GVS-Attack's malicious behaviors.

4.2 Context-aware Information Collection and Analysis Experiments

The occasion of GVS-Attack is when the victim is not using his phone. Especially in the early morning (before dawn, such as 3 AM), the victim is likely to be in deep sleep. To avoid the attack being found and interrupted by the victim, VoicEmployer needs to analyze the current environment (including the surroundings and phone status) before deciding whether to launch attacks. This target can be achieved through sensors and legal Android API invoking without any permission. If the analysis result shows the victim is using his phone, VoicEmployer will keep inactive. Otherwise, GVS-Attack could be launched. Range of analysis and criteria include:

- Light sensor. Get the current brightness: if the brightness is very low (such as less than 5% of max), it means the phone is probably in pockets / bags or the victim has turned off the light at night.
- Accelerometer. Get the current position information of the phone: if the result shows the phone is in static status (namely stable readings), it means the victim is probably not holding / taking the phone.
- Android SDK class - `PowerManager`. Check the screen is on or off: if the screen is off, it means the victim is probably not using the phone.
- Android SDK class - `SimpleDateFormat`. Obtain the current system time: for example, 3 AM means the victim is likely to be in deep sleep.
- Read `/proc/stat` and `top` command. Analyze the current CPU workload: if the workload is low (such as less than 50%), it means the victim is probably not using his phone.
- Read `/proc/meminfo` and `top` command. Analyze the current RAM usage: if the usage is low (such as less that 50%), it means the victim is probably not using his phone.

Based on above analyses, we designed and carried out context-aware information analysis experiments using EnvironmentService module in 3 typical scenes:

1. The user is using his phone to play games (take Angry Birds Rio[12] as an example) with the light on.
2. The user is walking on the street, while the phone (screen-off) is in his trouser pocket.
3. [*Target Scene*] The phone (screen-off) is put on a horizontal table with the light off.

The experiment was based on Meizu MX2 which has current mainstream hardware configurations (ARM Cortex-A9 quad-core 1.6 GHz, 2 GB LPDDR2 RAM). Before every scene's test, the phone was restarted to cut out distractions of irrelevant factors. Table 3 shows test results in 30 seconds. We can find that some item values of Scene 3 (target scene) are significantly different from those of Scene 1 and Scene 2. It demonstrates that context-aware information analysis can be used to detect the current environment and assist GVS-Attack practically.

Table 2: GVS-Attack Experiments

Phone Model	Android Version	Attack Result
Samsung Galaxy S3	CyanogenMod 4.4.2	success
	Samsung Official 4.3	success
Meizu MX2	Meizu official 4.2.1	success
Motolora A953	CyanogenMod 4.1.1	success

[7]https://www.avgmobilation.com/
[8]https://www.mcafeemobilesecurity.com/
[9]http://www.avira.com/en/free-antivirus-android
[10]http://www.eset.com/us/home/products/mobile-security-android/
[11]https://mobilesecurity.norton.com/
[12]http://www.rovio.com/en/our-work/games

Table 3: Content-aware Information Analysis

Method / Tool		Scene 1	Scene 2	Scene3
Light Sensor	Max	304	1	0
	Min	126	0	0
	Avg	227.95	0.5	0
Accelerometer	X-axis Max	7.78	18.19	0.15
	X-axis Min	5.62	-0.06	0.01
	X-axis Avg	6.54	8.61	0.10
	Y-axis Max	-0.50	14.56	0.11
	Y-axis Min	-1.97	-2.15	-0.10
	Y-axis Avg	-1.10	4.78	0.01
	Z-axis Max	9.46	6.70	10.34
	Z-axis Min	5.88	-12.00	10.04
	Z-axis Avg	7.60	-0.18	10.22
CPU Workload	Max	59%	43%	40%
	Min	32%	22%	18%
	Avg	43%	29%	28%
Memory Usage	Max	66.2%	53.4%	52.3%
	Min	66.0%	53.2%	52.0%
	Avg	66.0%	53.3%	52.0%
Screen Status		on	off	off

4.3 Sound Volume and Sound Pressure Level Experiments

One issue we concerned is what the volume (`STREAM_MUSIC`) should be set for playing voice commands. The volume level should be loud enough to be recognized by Google Voice Search and may not be noticed by the victim. We call it the minimal available sound volume (*MASV* for short). In this section, we only consider the volume setting of the speaker (`MODE_NORMAL`) in Extended Attack, not the headset (`MODE_IN_CALL`). In Remote Voice Control Attack, the attacker can adjust his own speaking volume directly.

The hardware design and components are quite different in different models of Android phones. Mixed with background white noise, it is impossible to give a fixed sound volume setting. One solution is using the method of successive approximation to pretest, which needs the help of Google Speech-to-Text (STT). STT is a system built-in speech recognition service, which uses the same engine as Google Voice Search and can be invoked through `ACTION_RECOGNIZE_SPEECH` [4]. The key point is the app using STT service can get the text content of voice input directly. To be specific, VoicEmployer sets the sound volume to 1 (the minimum) at first, then invokes SST service and plays a prepared test audio file (the content could be "*This is a test message*"). Text results from SST will be returned to VoicEmployer and be compared with the correct texts. If the two sets of texts are the same, VoicEmployer sets system volume to 1 and launches GVS-Attack. If the two sets of texts are different, VoicEmployer adjusts the volume to 2 and carries out the same test again. This process will be repeated until MASV is found.

We carried out MASV test experiments in a quiet meeting room of about 8 m^2 (see Table 4 as result). The environment background sound pressure levels (SPL) [51] was about 48 dB. Another relationship we concerned is SPL vs. distance. In the same experiment environment, we recorded the transient peak SPL in different distances (0.5 m, 1 m and 2 m) from the phone using the above MASV value. Test phones were put on a flat table face up without shelter. Table 4 shows corresponding experiment results. We can find that when the distance exceeds 1 m, the SPL is quite low – only 8 dB higher than the background SPL.

Table 4: Quiet Meeting Room, SPL vs. Distance, unit: dB

Phone Model		0.5 m	1 m	2 m
Samsung Galaxy S3	– volume 6/15	57	56	54
Meizu MX2	– volume 5/15	58	54	53
Motolora A953	– volume 6/15	58	56	54

5. DEFENSE

There exist some possible (but not perfect) schemes to defend GVS-Attack. The vulnerability of status checking in Google Search app should be fixed first. When Google Search app receives an `ACTION_VOICE_COMMAND` based Intent, if the phone is securely locked, it must strictly check whether a Bluetooth headset is connected with the phone. If the connect status is true, Voice Dialer can be started. Otherwise, a warning should be given, like "*please unlock the device first*". Therefore, in the situation of secure screen lock, GVS-Attack cannot be launched and the phone is safe.

From the aspect of app development, Google Voice Search must check the status of speaker in real time, not just at the moment of initialization. If other apps are accessing the speaker, Google Voice Search should suspend that app immediately. But this solution may affect other apps' user experience. For example, an IM app may not play its notification sound when new message reach.

From the aspect of Android Intent mechanism, system built-in apps / services should also add customized permissions in the manifest file. if a third-party app wants to invoke a system built-in app / service, it must declare these requirements before installation. As a result, the user needs to confirm the authorization or cancel the installation. Nonetheless, this defense method may result in similar abuse problems like system permissions.

From the aspect of user identification, speaker recognition (or so called voiceprint recognition) techniques [11] should be deployed to verify the identity of the speaker. If voiceprint authentication fails, Google Voice Search will not accept the next voice commands. Touchless Control [38] (a variant of Google Voice Search, only for Motorola phones) has added this function to authenticate the current user through speaking "*OK, Google*" [37]. Recently Google has announced that Android L (5.0) will use voice fingerprinting for context-based security [43]. Potential problems may be how to defend voice replay attacks.

6. DISCUSSION

Soundless Attack. One potential limitation of GVS-Attack is that the victim may notice the voice command played by VoicEmployer and interrupt it. So one direct idea is whether soundless GVS-Attack could be launched. That is, VoicEmployer imports an audio file to the microphone directly without playing it, like creating a loopback. But in Android OS, the audio recording method is synchronized. It means that the microphone cannot support multi-progress accessing at the same time. So when Google Voice Search is accessing microphone to accept voice commands, VoicEmployer cannot access microphone at the same time at least on API level. In terms of audio files as the input of microphone, this feature needs the support of kernel / drivers. On Windows and Linux platforms, there exist such drivers (such as VB-Audio Virtual Cable [50]) to simulate a virtual microphone device. But on Android platform, an app cannot modify kernel or install drivers directly. One feasible

solution is to prepare a customized Android version with modified audio drivers. One recent research [54] has pointed out the security risks in Android device driver customizations. Android inherits the driver management methods of Linux and devices are placed under `/dev` (or `/sys`) as files. Zhou et al. found the vulnerability that certain important devices become unprotected (permission setting) during a customization. An unauthorized app can get access to sensitive devices, namely user data. Based on this method, similar vulnerability may occur on audio drivers (`/dev/snd`), but we have not found such a case of unprotected writing privileges on our test devices.

Another perspective of soundless attacks is high-frequency sound (such as higher than 20 kHz), which could be played by the phone and is difficult to hear by humans [42]. Unfortunately (fortunately for security), Google Voice Search only accepts reasonable human sound frequency and filters out other ranges, so the idea of high-frequency sound is not impracticable.

Since Google Voice Search is an Internet based service, we ever tried to analyze the feasibility of connection hijacking and data package tampering. After tests, the difficulties lie in that the connection is TLS protected and transmitted voice data are compressed by an unclear compression encoding algorithm.

Quiet vs. Noisy. The scene of launching GVS-Attack is a quiet environment. The volume of voice commands could be very low and still be recognized by Google Voice Search. Actually we also tested the performance of attacks in noisy environments, such as on the subway and in the canteen. The expected result would be changed, namely the volume of voice commands could be very loud and still be hidden in the background noisy. But test results showed the background noisy (especially human voice) affected the accuracy of speech recognition, to some degree. In addition, context-aware analysis will become complex and may need the `RECORD_AUDIO` permission.

Scope of Attack. Since Google Services Framework is pre-installed on nearly all brands of Android devices, most Android devices can be affected by GVS-Attack, especially these equipped with Android 4.1 or higher versions. To voice assistant apps using Google Speech-to-Text (STT) service, similar attacks could be launched. Even these using independent speech recognition engines should also be reviewed carefully, such as Samsung S Voice app. Furthermore, similar attacks may occur on iOS, Windows Phone platforms and other smart devices supporting voice control. However, for the lack of experimental devices, we haven't tested them temporarily and left them for future research.

Limitation. We didn't carry out a user study about the minimum available sound volume (Section 4.3), because the noise-induced nocturnal awakening is affected by many other factors, such as sleep stages, age, gender, smoking, etc [31]. Content-aware information analysis (Section 4.2) may be affected by hardware configurations. For example, high memory usage may derive from small capacity RAM. Light sensors and accelerometers from different vendors may have different precision. So it is difficult to create a general detector to handle all cases.

7. RELATED WORK

Sensor based Attacks. On mobile platforms, sensor based attacks have been designed and analyzed in several previous papers [46, 40, 13, 10, 47, 49, 30]. One typical example is Soundcomber [46]. Schlegel et al. designed a Trojan with few and innocuous permissions, that can extract targeted private information (such as credit card and PIN number) from the audio sensor of the phone. Besides, it proved that smartphone based malware can easily achieve targeted, context-aware information discovery from sound recordings. In another research project, through completely opportunistic use of the phone's camera and other sensors, PlaceRaider [49] can construct three dimensional models of indoor environments and steal virtual objects.

The work of [40] showed that accelerometer readings are a powerful side channel which can be used to extract entire sequences of entered text on a smartphone touchscreen keyboard. In [47], their side-channel attack utilizes the video camera and microphone to infer PINs entered on a number-only soft keyboard on a smartphone. The microphone is used to detect touch events, while the camera is used to estimate the smartphone's orientation, and correlate it to the position of the digit tapped by the user. The work of [30] studied environmental sensor-based covert channels in mobile malware. Out-of-band command and control channels could be based on acoustic, light, magnetic and vibrational signaling. This research is a bit like our GVS-Attack in the aspect of voice command transmission. The differences are that, in GVS-Attack, the command receiver (Google Voice Search) is not a part of malware (VoicEmployer) and the `RECORD_AUDIO` permission is not necessary. So our attack scheme is more insidious.

Beyond that, Sensors could be used for fingerprinting devices. A mechanism was proposed by [20] that smartphone accelerometers possess unique fingerprints, which can be utilized for tracking users. Similar fingerprinting methods were designed with microphone and speaker in [18, 57]. Through playbacking and recording audio samples, they could uniquely identify an individual device based on sound feature analysis.

Inter-Application Communication. In [15], Chin et al. focused on Intent-based attack surfaces. It analyzed unauthorized Intent receipt can leak user information. Data can be stolen by eavesdroppers and permissions can be accidentally transferred between apps. Another attack type is Intent spoofing, that a malicious app sends an Intent to an exported component. If the victim app takes some action upon receipt of such an Intent, the attack can trigger that action. In their following work [33], Kantola et al. proposed modifications to the Android platform to detect and protect inter-application messages that should have been intra-application messages. The target is to automatically reduce attack surfaces in legacy apps.

Application Analysis Android permission-based security has been analyzed from many aspects. For example, permission specification and least-privilege security problems were studied in [22] and [9]. Both of them designed corresponding static analysis tools to detect over-privilege problems. Permission escalation and leakage problems are also hot research topics. Related research include [24, 19, 12, 14, 28]. Permission-based behavioral footprinting was used to detect known malware in Android market at large-scale in [56]. The work of [52] noticed the problem of pre-installed apps. Wu et al. found a lot of those apps were overly privileged for vendor customizations.

To system code level analysis, dynamic analysis technique is an efficient solution. TaintDroid [21] is a system-wide dynamic taint tracking and analysis system capable of simultaneously tracking multiple sources of sensitive data. It automatically labels (taints) data from privacy-sensitive sources and transitively applies labels as sensitive data propagates through program variables, files, and interprocess messages. DroidScope [53] is an emulation based Android malware analysis engine that can be used to analyze the Java and native components of Android apps. Unlike current desktop

malware analysis platforms, DroidScope reconstructs both the OS-level and Java-level semantics simultaneously and seamlessly.

But these analysis methods are useless to our GVS-Attack, because VoicEmployer doesn't touch sensitive data and perform sensitive operations directly. The voice commands are transferred out of the phone, that is, an uncontrolled physical communication channel. In addition, the remote data transmission function is based on the call channel and the transmission media is sound. These brand-new features break previous checking and protecting mechanism.

8. CONCLUSION

This paper proposes a novel permission bypassing attack method based on Android system built-in voice assistant module - Google Voice Search and the phone speaker. The app launching attacks doesn't require any permission. But achieved malicious targets could be quite dangerous and practical, from privacy stealing to remote voice control. Through utilizing a vulnerability found by us in Google Search app, this attack can dial arbitrary malicious numbers even when the phone is securely locked. Besides, related in-depth topics are discussed, including context-aware analysis, minimal available sound volume, soundless attack, etc. The feasibility of our attack schemes has been demonstrated in the real world. This research may inspire application developers and researchers to rethink that zero permission doesn't mean safety and the speaker can be treated as a new attack surface.

Acknowledgements

We thank anonymous reviewers for their valuable comments.

9. REFERENCES

[1] Android Developers. Common Intents. http://developer.android.com/guide/components/intents-common.html.

[2] Android Developers. Intent. http://developer.android.com/reference/android/content/Intent.html.

[3] Android Developers. Platform versions. https://developer.android.com/about/dashboards/index.html#Platform.

[4] Android Developers. RecognizerIntent. http://developer.android.com/reference/android/speech/RecognizerIntent.html.

[5] Android Developers. TextToSpeech. http://developer.android.com/reference/android/speech/tts/TextToSpeech.html.

[6] AOSP. HeadsetStateMachine. https://android.googlesource.com/platform/packages/apps/Bluetooth/.

[7] AOSP. Telephony. https://android.googlesource.com/platform/packages/services/Telephony/.

[8] Apple. Siri. http://www.apple.com/ios/siri/.

[9] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie. Pscout: analyzing the android permission specification. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 217–228. ACM, 2012.

[10] A. J. Aviv, B. Sapp, M. Blaze, and J. M. Smith. Practicality of accelerometer side channels on smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 41–50. ACM, 2012.

[11] H. Beigi. *Fundamentals of speaker recognition*. Springer, 2011.

[12] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, A.-R. Sadeghi, and B. Shastry. Towards taming privilege-escalation attacks on android. In *19th Annual Network & Distributed System Security Symposium (NDSS)*, volume 17, pages 18–25, 2012.

[13] L. Cai and H. Chen. Touchlogger: inferring keystrokes on touch screen from smartphone motion. In *Proceedings of the 6th USENIX conference on Hot topics in security*, pages 9–9. USENIX Association, 2011.

[14] P. P. Chan, L. C. Hui, and S.-M. Yiu. Droidchecker: analyzing android applications for capability leak. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, pages 125–136. ACM, 2012.

[15] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner. Analyzing inter-application communication in android. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 239–252. ACM, 2011.

[16] A. Clark, C. Fox, and S. Lappin. *The handbook of computational linguistics and natural language processing*, volume 57. John Wiley & Sons, 2010.

[17] CyanogenMod. Google Apps. http://wiki.cyanogenmod.org/w/Google_Apps.

[18] A. Das, N. Borisov, and M. Caesar. Do you hear what I hear? fingerprinting smart devices through embedded acoustic components. In *(To appear) Proceedings of the 2014 ACM conference on Computer and communications security*. ACM, 2014.

[19] L. Davi, A. Dmitrienko, A.-R. Sadeghi, and M. Winandy. Privilege escalation attacks on android. In *Information Security*, pages 346–360. Springer, 2011.

[20] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi. Accelprint: Imperfections of accelerometers make smartphones trackable. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium, NDSS*, 2014.

[21] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *OSDI*, volume 10, pages 1–6, 2010.

[22] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. Android permissions demystified. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 627–638. ACM, 2011.

[23] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner. Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 3. ACM, 2012.

[24] A. P. Felt, H. J. Wang, A. Moshchuk, S. Hanna, and E. Chin. Permission re-delegation: Attacks and defenses. In *USENIX Security Symposium*, 2011.

[25] H. W. Gellersen, A. Schmidt, and M. Beigl. Multi-sensor context-awareness in mobile devices and smart artifacts. *Mobile Networks and Applications*, 7(5):341–351, 2002.

[26] Google. Google Apps for Android. https://www.google.com/mobile/android/.

[27] Google. Use your voice on Android. https://support.google.com/websearch/answer/2940021?hl=en&ref_topic=4409793.

[28] M. Grace, Y. Zhou, Z. Wang, and X. Jiang. Systematic detection of capability leaks in stock android smartphones. In *Proceedings of the 19th Annual Symposium on Network and Distributed System Security*, 2012.

[29] C. Hadnagy. *Social engineering: The art of human hacking*. John Wiley & Sons, 2010.

[30] R. Hasan, N. Saxena, T. Haleviz, S. Zawoad, and D. Rinehart. Sensing-enabled channels for hard-to-detect command and control of mobile devices. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pages 469–480. ACM, 2013.

[31] C. Hurtley. *Night noise guidelines for Europe*. WHO Regional Office Europe, 2009.

[32] IDC. Worldwide smartphone shipments edge past 300 million units in the second quarter; android and ios devices account for 96% of the global market, according to IDC. http://www.idc.com/getdoc.jsp?containerId=prUS25037214, August, 2014.

[33] D. Kantola, E. Chin, W. He, and D. Wagner. Reducing attack surfaces for intra-application communication in android. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*, pages 69–80. ACM, 2012.

[34] Kaspersky Lab. Kaspersky security bulletin 2013. http://media.kaspersky.com/pdf/KSB_2013_EN.pdf, December 2013.

[35] Y.-S. Lee and S.-B. Cho. Activity recognition using hierarchical hidden markov models on a smartphone with 3d accelerometer. In *Hybrid Artificial Intelligent Systems*, pages 460–467. Springer, 2011.

[36] Microsoft. Cortana. http://www.windowsphone.com/en-us/features#Cortana.

[37] Motorola. How do I setup and use touchless control? https://motorola-global-portal.custhelp.com/app/answers/prod_answer_detail/a_id/94881/p/30,6720,8696/action/auth.

[38] Motorola. Touchless Control. http://www.motorola.com/us/Moto-X-Features-Touchless-Control/motox-features-2-touchless.html.

[39] A. Muzet. Environmental noise, sleep and health. *Sleep medicine reviews*, 11(2):135–142, 2007.

[40] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang. Accessory: password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, page 9. ACM, 2012.

[41] Ponemon Institute. Smartphone security survey of U.S. consumers. http://aa-download.avg.com/filedir/other/Smartphone.pdf, March, 2011.

[42] S. Rosen and P. Howell. *Signals and systems for speech and hearing*, volume 29. BRILL, 2011.

[43] B. P. Rubin. Google previews “Android L” at I/O. http://www.besttechie.com/2014/06/25/google-previews-android-l-at-io/.

[44] Samsung. S Voice. http://www.samsung.com/global/galaxys3/svoice.html.

[45] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Kamvar, and B. Strope. "Your word is my command": Google search by voice: A case study. In *Advances in Speech Recognition*, pages 61–90. Springer, 2010.

[46] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang. Soundcomber: A stealthy and context-aware sound trojan for smartphones. In *Proceedings of the 18st Annual Network and Distributed System Security Symposium, NDSS*, 2011.

[47] L. Simon and R. Anderson. Pin skimmer: inferring pins through the camera and microphone. In *Proceedings of the Third ACM workshop on Security and privacy in smartphones & mobile devices*, pages 67–78. ACM, 2013.

[48] A. Smith. Americans and their cell phones. *Pew Internet & American Life Project*, 15, 2011.

[49] R. Templeman, Z. Rahman, D. Crandall, and A. Kapadia. PlaceRaider: Virtual theft in physical spaces with smartphones. In *Proceedings of the 20th Annual Network and Distributed System Security Symposium, NDSS*, 2013.

[50] VB-Audio Software. VB-Audio Virtual Cable. http://vb-audio.pagesperso-orange.fr/Cable/.

[51] Wikipedia. Sound pressure. http://en.wikipedia.org/wiki/Sound_pressure.

[52] L. Wu, M. Grace, Y. Zhou, C. Wu, and X. Jiang. The impact of vendor customizations on android security. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 623–634. ACM, 2013.

[53] L. K. Yan and H. Yin. Droidscope: seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis. In *Proceedings of the 21st USENIX Security Symposium*, 2012.

[54] X. Zhou, Y. Lee, N. Zhang, M. Naveed, and X. Wang. The peril of fragmentation: Security hazards in android device driver customizations. In *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 2014.

[55] Y. Zhou and X. Jiang. Dissecting android malware: Characterization and evolution. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 95–109. IEEE, 2012.

[56] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang. Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets. In *Proceedings of the 19th Annual Network and Distributed System Security Symposium*, pages 5–8, 2012.

[57] Z. Zhou, W. Diao, X. Liu, and K. Zhang. Acoustic fingerprinting revisited: Generate stable device id stealthily with inaudible sound. In *(To appear) Proceedings of the 2014 ACM conference on Computer and communications security*. ACM, 2014.

APPENDIX

A. PERMISSIONS STATISTICS

According to Android API Level 19, Google Search app (version 3.4.16.1149292.arm) declares / possesses the following system defined permissions:

1. ACCESS_COARSE_LOCATION
2. ACCESS_FINE_LOCATION
3. ACCESS_NETWORK_STATE
4. ACCESS_WIFI_STATE
5. BIND_APPWIDGET

6. BLUETOOTH
7. BROADCAST_STICKY
8. CALL_PHONE
9. GET_ACCOUNTS
10. GLOBAL_SEARCH
11. INTERNET
12. MANAGE_ACCOUNTS
13. MEDIA_CONTENT_CONTROL
14. MODIFY_AUDIO_SETTINGS
15. READ_CALENDAR
16. READ_CONTACTS
17. READ_EXTERNAL_STORAGE
18. READ_HISTORY_BOOKMARKS
19. READ_PROFILE
20. READ_SMS
21. READ_SYNC_SETTINGS
22. RECEIVE_BOOT_COMPLETED
23. RECORD_AUDIO
24. SEND_SMS
25. SET_ALARM
26. SET_WALLPAPER
27. SET_WALLPAPER_HINTS
28. STATUS_BAR
29. USE_CREDENTIALS
30. VIBRATE
31. WAKE_LOCK
32. WRITE_CALENDAR
33. WRITE_EXTERNAL_STORAGE
34. WRITE_SETTINGS
35. WRITE_SMS

Non-public permissions:

1. CAPTURE_AUDIO_HOTWORD
2. STOP_APP_SWITCHES
3. PRELOAD