HearMeOut: Detecting Voice Phishing Activities in Android

Joongyum Kim, Jihwan Kim, Seongil Wi, Yongdae Kim, Sooel Son*
KAIST
{kjkpoi,payload,seongil.wi,yongdaek,sl.son}@kaist.ac.kr

ABSTRACT

In South Korea, voice phishing has been proliferating with the advent of voice phishing apps: the number of annual victims had risen to 34,527 in 2020, representing financial losses of approximately 598 million USD. However, the voice phishing functionalities that these abusive apps implement are largely understudied. To this end, we analyze 1,017 voice phishing apps and reveal new phishing functionalities: outgoing call redirection, call screen overlay, and fake call voice. We find that call redirection that changes the intended recipients of victims' outgoing calls plays a critical role in facilitating voice phishing; our user study shows that 87% of the participants did not notice that their intended recipients were changed when call redirection occurred. We further investigate implementations of these fatal functionalities to distinguish their malicious behaviors from their corresponding behaviors in benign apps. We then propose HearMeOut, an Android system-level service that detects phishing behaviors that phishing apps conduct in runtime and blocks the detected behaviors. HearMeOut achieves high accuracy with no false positives or negatives in classifying phishing behaviors while exhibiting an unnoticeable latency of 0.36 ms on average. Our user study demonstrates that HearMeOut is able to prevent 100% of participants from being phished by providing active warnings. Our work facilitates a better understanding of recent voice phishing and proposes practical mitigation with recommendations for Android system changes.

CCS CONCEPTS

• **Security and privacy → Mobile and wireless security**; **Malware and its mitigation**; **Intrusion/anomaly detection and malware mitigation**; **Domain-specific security and privacy architectures**.

KEYWORDS

Voice phishing, Android voice phishing apps, Phishing app detection, System security

ACM Reference Format:
Joongyum Kim, Jihwan Kim, Seongil Wi, Yongdae Kim, Sooel Son* . 2022. HearMeOut: Detecting Voice Phishing Activities in Android. In *The 20th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '22), June 25–July 1, 2022, Portland, OR, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3498361.3538939

*Corresponding author.

MobiSys '22, June 25–July 1, 2022, Portland, OR, USA

ACM ISBN 978-1-4503-9185-6/22/06…$15.00
https://doi.org/10.1145/3498361.3538939

1 INTRODUCTION

Voice phishing is a notorious scam in which fraudsters contact victims via phone calls and exploit their personal information, deceiving these victims into transferring their money to the fraudsters. The Federal Trade Commission (FTC) reports that financial losses due to voice phishing have amounted to more than 450 million USD since 2014 [45]. Especially in South Korea, voice phishing has been traumatizing a growing segment of the general public. From 2016 to 2020, the number of annual victims rose from 17,516 to 34,527 [2, 24]. The annual financial losses in 2020 increased to approximately 598 million USD [69].

Unfortunately, as the voice phishing business has become more lucrative, phishing techniques have evolved. Fraudsters have impersonated the voices of representatives of known organizations or familiar personal contacts [50, 81, 91, 108] and have changed the area code displayed [44, 108, 109]; they have manipulated victims' incoming calls [44, 50, 81, 91, 108, 109]. However, with the advent of voice phishing apps, fraudsters have changed their scamming strategies.

One recent and notable trend in voice phishing involves the redirection of victims' outgoing calls. Fraudsters entice a victim to install an Android phishing app by offering loan opportunities with lower interest rates. Once the victim installs the phishing app and initiates a call to a legitimate bank, their outgoing call is redirected to the fraudsters. Since the victim initiated the call using the legitimate phone number, they believe that the recipient of their calls is authentic, leading them to disclose their confidential and private information. This outgoing call redirection plays a key role in rendering victims susceptible to fraudsters' demands [24, 28, 57, 67, 68, 107]. The financial losses due to voice phishing campaigns that employ call redirection are ten times greater than those due to phishing campaigns that do not use such apps [52].

Previous studies have focused on analyzing voice phishing involving incoming calls. They examined user studies of victims [109], honeypots [50], and crime reports [23, 79]. However, to the best of our knowledge, no previous study has investigated voice phishing apps and their new phishing strategies that involve outgoing call redirection. We believe that answering the following questions is paramount to devise a defense against voice phishing threats: *What distinctive functionalities do voice phishing apps support? Are existing defenses effective in mitigating new threats that voice phishing apps pose?*

Our contributions. We conducted an investigative study to identify the distinctive characteristics of Android voice phishing apps. We collected 1,017 Android voice phishing apps from AhnLab [3], a well-respected anti-virus corporation in South Korea, and the Financial Security Institute (FSI) [46].

We identified three new phishing functionalities: *call redirection*, *call screen overlay*, and *fake call voice*. Our analysis results show

that these functionalities work together to make outgoing call redirection seamless, thus facilitating voice phishing campaigns. Of the phishing apps studied, 978 apps (96.1%) redirect the outgoing call to fraudsters when victims call public directory assistance services (i.e., 114), banks, loan businesses, and government branches.

We emphasize that call redirection plays a critical role in deceiving victims especially when all three functionalities work together. Our user study results show that 87% of participants were deceived by voice phishing campaigns that involve call redirection. Half of these participants were well aware of voice phishing threats due to their prior experiences of receiving voice phishing calls. However, the high success rate of 87% manifests the efficacy of call redirection. These results also explain the motivation of most phishing apps implementing call redirection. Unfortunately, the voice phishing threats that these new functionalities pose have been largely understudied.

To understand the current mitigation of voice phishing app threats, we analyzed defense apps from Google Play that are specifically designed to detect voice phishing apps. We observed that these defense apps mainly support the detection of call redirection and voice phishing apps in victims' devices. However, all the defense methods fail to completely block voice phishing threats and disregard to detect other types of voice phishing behaviors, such as call screen overlay. These failures stem from known signature- and blacklist-based detection approaches, which fraudsters are able to bypass with ease, namely by mutating their APKs. Also, the existing defense apps that warn of apps using the `PROCESS_OUTGOING_CALLS` permission often generate false positives.

To address these shortcomings, we propose HearMeOut, an Android system-level detection method to notify the user of phishing activities, including outgoing call redirection in runtime. HearMeOut is a voice phishing notification service that detects suspicious activities at the Android system level and notifies the users of detected activities. Based on our observation that all phishing apps abuse the Android APIs of five service managers (e.g., Telephony and Media), we implement detectors at the Android API framework layer to identify phishing activities that abuse Android APIs. We identify the distinctive characteristics of phishing activities that abuse Android APIs in runtime versus those of benign activities. We leverage the invocation times, actual parameters, and caller apps of the APIs to detect phishing activities, which play a decisive role in differentiating benign from phishing behaviors. HearMeOut also blocks identified phishing behaviors. Especially when call redirection occurs, HearMeOut displays an active warning dialog that notifies this possible phishing behavior.

We evaluate the efficacy of HearMeOut in detecting phishing behaviors. For various types of benign and phishing activities from 53 benign and 12 phishing apps, HearMeOut achieved an accuracy of 100% with zero false positives. Also, the execution overhead of HearMeOut was negligible: the average additional latency that HearMeOut entailed for detection was 0.36 ms.

We further conducted user studies with 45 participants to measure the efficacy of voice phishing that involves outgoing call redirection and HearMeOut's effectiveness in preventing such voice phishing. The experimental results show that 87% of participants were tricked by phishing campaigns that involved voice phishing and that 69.6% disclosed their personal information. By contrast, 22 participants (100%) with HearMeOut acknowledged outgoing call redirection warnings and thus avoided falling victims to voice phishing.

Our contributions are summarized as follows:

(1) We analyze the distinctive functionalities of Android voice phishing apps of which the threats have been understudied.
(2) We identify the limitations of existing voice phishing app detection methods that stem from inaccessible system information and blacklist-based detection approaches.
(3) We design and implement HearMeOut, the first system-level phishing behavior detection service and propose practical changes to the current Android system.

2 BACKGROUND: VOICE PHISHING

Voice phishing is a traditional scam that exploits the non-face-to-face nature of telephone communication to deceive victims [50, 81, 91, 108]. In recent years, voice phishing has been proliferating, distorting victims' monetary assets. In South Korea, the related annual financial losses increased from 216 million USD in 2017 to a record high of approximately 598 million USD in 2020 [69, 96, 97]. Voice phishing apps using call redirection have been identified as a critical factor in increasing these financial losses. The financial losses due to voice phishing using phishing apps are ten times greater than that due to phishing campaigns without these apps [52].

Voice phishing victims suffer agonizing losses both financially and psychologically; they find themselves having lost their savings, paying off loans that they did not take out, or being blackmailed using exfiltrated privacy-sensitive information [22, 23, 25, 26, 84, 112]. Recently, a voice phishing group consisting of 93 scammers was prosecuted for crimes committed while impersonating a local prosecutor; they had extorted over 10 million USD from 300 victims over the last five years. As a consequence of their constant blackmailing and extortion, a young victim was driven to take his own life [71]. These incidents call for a better understanding of phishing victims and practical mitigation of new voice phishing threats.

Call redirection. Voice phishing campaigns consist of four steps: (1) deceiving a victim into installing a voice phishing app, (2) inducing the victim to call a bank or government agency, (3) redirecting the victim's outgoing call to the attacker, and (4) extorting money. Fraudsters start by sending a target victim an SMS text with a URL. The SMS text describes an opportunity for the victim to refinance an existing loan with one that has a lower interest rate, notifies the victim about shipment tracking information, or notifies the victim about a health checkup schedule. The text further instructs the victim to install a mobile app by clicking on the URL. Clicking this URL causes the victim's device to install a voice phishing app. Note that Android offers an option for users to install the app from other app markets besides Play and unknown sources [13]. To install a voice phishing app, victims need to explicitly click "Yes" in a popup window that asks the user's consent to install apps from "Unknown sources." Unfortunately, the increasing number of phishing victims indicates that naive users tend to grant the permission to install phishing apps.

These phishing apps redirect the victim's outgoing call to a legitimate bank or government agency to the fraudsters' number instead in an attempt to deceive the victim into misplacing their

trust. In the final step, the fraudsters ask the victim to transfer money to one of their bank accounts. They then withdraw the transferred money before the victim realizes that they have been phished.

We emphasize the critical role of outgoing call redirection, which contributes to convincing victims to misplace their trust. According to our user study (§8), 87% of participants were deceived when their outgoing calls to a legitimate bank were redirected to our staff member who was impersonating a fraudster; they provided their personal information, including social security numbers, bank accounts, and home addresses.

3 VOICE PHISHING FUNCTIONALITY

We describe the representative functionalities of voice phishing apps in the wild. Specifically, we explain how these apps implement each phishing functionality using Android APIs.

Phishing apps. We collected 1,017 phishing apps from the FSI [46] and AhnLab [3], a well-respected anti-virus corporation in South Korea; we obtained 198 and 819 phishing apps from the FSI and the anti-virus corporation, respectively. These mobile apps had been reported by phishing victims and confirmed by the organizations above from 2018 to 2021.

We acknowledge that our collection of voice phishing apps could be biased due to the limited number of apps it contains. However, we emphasize that our objective is to understand the behaviors of mobile voice phishing apps to devise a practical countermeasure.

Analysis method. We analyzed the functionalities of 1,017 phishing apps. We investigated declared permissions, Intent filters, and Android components in the manifest files of these apps. We also statically analyzed the decompiled code and their Android API invocations to implement phishing functionalities using a JADX APK decompile tool [101].

3.1 Disguising Apps

We investigated the app names, image resources, and icons of 1,017 phishing apps to identify target apps into which they are disguised. The most popular apps disguised as phishing apps are banking apps; 824 phishing apps (81%) took on the appearance of banking apps, displaying the logo and images of actual banks. The main role of these apps is to entice victims to make loan inquiries using official bank phone numbers. These apps display the official phone number of the organization they are disguised as in most of their activities. When the victim calls the bank's official phone number, the attacker intercepts the call using outgoing call redirection. In addition, 89 phishing apps were disguised as package delivery tracking apps, and 23 apps were disguised as health checkup apps.

3.2 Call Redirection

Call redirection refers to the functionality that redirects a victim's outgoing calls to specific phone numbers that the attackers choose. Fraudsters exploit this functionality to deceive victims into misplacing their trust in the recipients of their outgoing calls. Victims do not realize that their actual outgoing calls have been redirected and believe that the recipients of their calls are the intended recipients. For instance, fraudsters entice a victim to install their mobile phishing app by offering loan refinancing at a low-interest rate. Once the app is installed, it offers authentic phone numbers with banker photos and explanations of their financial products [25, 26, 71]. Once the victim calls one of these numbers, the outgoing call is redirected to the fraudsters.

After the victim's outgoing call is redirected to the fraudsters, the fraudsters ask the victim to send cash for the initial payment of the original loan. A victim in doubt may check these phone numbers by searching for numbers on the Internet or calling the directory assistance service in South Korea (i.e.,114). However, fraudsters redirect the outgoing calls to both 114 and legitimate bank numbers to their own numbers instead, thereby deceiving the victim into believing in their authenticity.

To implement call redirection, a voice phishing app is required to register a BroadcastReceiver with an Intent filter for the `ACTION_NEW_OUTGOING_CALL` Intent in its manifest file. When an outgoing call occurs, the Android system broadcasts an `ACTION_NEW_OUTGOING_CALL` Intent, which allows any apps with the Intent filter above to change an outgoing call number in this Intent. The final recipient of this Intent (i.e., the Android system) makes an outgoing call to this changed number. For this, the phishing app requires a victim to grant the `PROCESS_OUTGOING_CALLS` permission to receive an `ACTION_NEW_OUTGOING_CALL` Intent and change the outgoing call number in this Intent.

Among the 1,017 phishing apps, we found 978 apps (96.2%) that implement the call redirection functionality. For each app, we searched for an Intent receiver invoked upon receiving the `ACTION_NEW_OUTGOING_CALL` Intent and analyzed all Java code in this receiver that changes outgoing call numbers using `SetResultData` [36]. We then extracted all constants representing phone numbers, which are target numbers from which phishing apps redirect calls to fraudsters' phone numbers.

We observed that each app redirects an average of 791 outgoing call numbers that exist in its source code. For each identified number, we sent a query for information about it to https://114.co.kr, the largest public directory assistance service in South Korea. We identified the number's associated corporation name and industry sector. Figure 1 shows the average number of observed phone numbers for each industry sector. We observed that phishing apps targeted secondary and tertiary banks, law enforcement agencies, and government branches, including the FSI and the public directory assistance service (i.e., 114).

3.3 Call Screen Overlay

Call screen overlay refers to a functionality that covers the default call screen with another call screen when a victim's device has an incoming or outgoing call. We observed that 572 phishing apps implemented this feature of overlaying fake call screens over authentic ones.

The goal is to display an authentic phone number when victims have an incoming/ outgoing call from/to a fraudster's number, thus relieving suspicion that victims may have about the call recipient. Therefore, all call screen overlay functions were implemented together with the call redirection function.

Fake outgoing/incoming call screens. We further investigated how these phishing apps implement fake call screens. We observed that 572 phishing apps generated a call screen based on the current

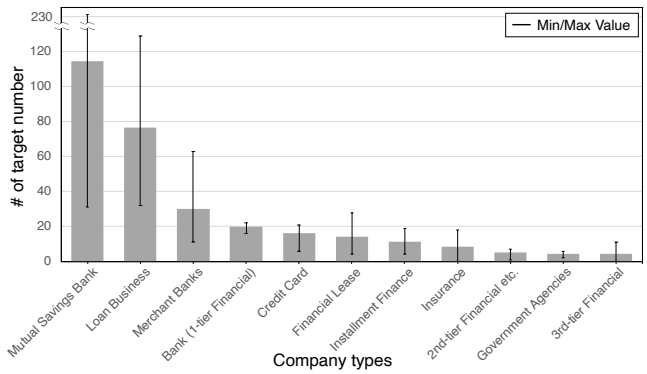

Figure 1: The average number of phone numbers that each app redirects by industry sector; the black line in each bar represents the min and max numbers.

call state (e.g., ringing, off-hook, and idle states.), Android OS, the phone manufacturer, and phone model information from a victim's device. These apps support 20 to 80 different models, mainly from Samsung and LG.

When the call state is off-hook (`CALL_STATE_OFFHOOK`), the fake call screen shows a dialing screen with an attacker-chosen number paired with the corresponding recipient name in the victim's contact list. However, the actual dialing number is different from this number displayed by the attacker. When the call connection is established, the fake screen shows a call screen mockup that contains the current call time. Finally, when the call state switches to idle, the fake call screen is closed. These fake call screens are similar to the default call screens for each device model, as shown in Figure 2.

For incoming calls, fraudsters selectively display their fake call screens with specific attacker-chosen numbers belonging to actual organizations. The fraudsters thus deceive victims into believing that they are receiving calls from the individual whom the fraudsters impersonate.

3.4 Fake Call Voice

Fake call voice refers to a functionality that plays a voice recording when call redirection occurs. We observed that 521 apps played voice recordings that sounded like customer service messages. These messages were recorded in mp3 files, and the phishing apps played them when outgoing call redirection occurred.

The purpose of fake call voice is to make it seem as if the victim's redirected outgoing call is connected to a bank's or financial company's phone line. When the victim calls the financial institution's actual public number, a phishing app usually plays a greeting message and then connects to the agent. Attackers play a recorded voice message that corresponds to the outgoing call number, deceiving victims into believing that they are connected to a real financial institution. For instance, such a recorded voice message is "Thank you for calling. This is your lifetime financial partner, anonymous insurance. Please wait until there is an available customer service consultant."

Note that fraudsters cover a large number of authentic phone numbers for call redirection, all of which are redirected to a small number of phone numbers under their control. Therefore, when

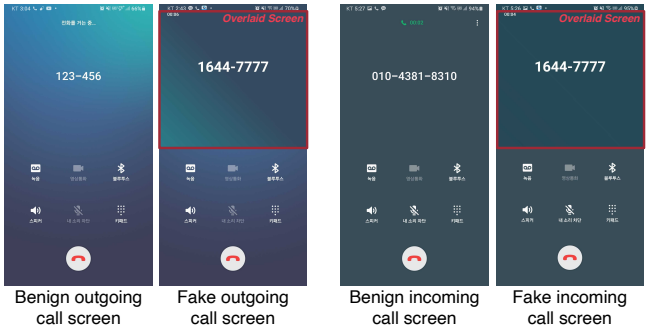

Figure 2: Examples of fake outgoing/incoming call screens.

phishing a victim via phone calls, they need to impersonate the receiver of their victims' calls. This functionality enables them to enmesh their victim in a scamming scenario corresponding to this victim's outgoing call by playing an appropriate greeting message before the victim's call is connected.

3.5 Dynamic DEX Loading

44 phishing apps dynamically loaded DEX files fetched from encrypted local resources and their command and control servers. We observed that these DEX files were encrypted using Bangcle [20], thereby hindering manual and automatic inspection of their semantics. Phishing apps used the `dalvik.system.DexClassLoader` API to dynamically load DEX files. Furthermore, they used the Java reflection API to load the dynamic DEX loading API. Phishing apps called the DEX loading API using reflection by combining the API name and class with `StringBuffer`. It is a classic evasion technique that malware has used to hide its API usage from static detection [76, 100].

3.6 Phishing App Groups

This section describes the clustering analysis results on the 1,017 phishing apps. To identify app groups that implement similar functionalities, we clustered the collected phishing apps based on the required permissions and declared Intent filters in their manifest files; therefore, each group has an identical set of requested permissions and Intent filters.

Among 40 groups, we observed five groups with more than 50 phishing apps, covering 742 apps (72.9%). We then analyzed randomly sampled apps from each group. Specifically, we examined how they implemented call direction by manually examining the Broadcast receivers that called upon an `android.intent.action.NEW_OUTGOING_CALL` Intent for each sampled app.

Group Name	# of apps	Sampled apps	# of different bank apps
Group 1	407	82	42
Group 2	131	27	17
Group 3	75	15	8
Group 4	75	15	7
Group 5	54	11	7

Table 1: Manual Analysis Results of grouping phishing aps using the same set of permissions and Intent filters.

Table 1 shows the statistics for sampled apps from the five app groups. In each group, we observed that the sample apps share

a similar app layout and source code decompiled via JADX [101]. However, they have different obfuscated source code and logo image resources used for showing the identities of banking apps. For example, 80 apps in Group 1 share the almost identical decompiled source code in their Broadcast receivers but are disguised as 42 different banking apps. We believe that fraudsters generate various app variants by changing image resources and obfuscating variables, classes, and package names of their original phishing app. The existence of various apps in each group also implies that fraudsters have been changing the app signatures that anti-virus products may use for detection, which we discuss in Section 4.

4 CURRENT DEFENSES AGAINST VOICE PHISHING APPS

We investigate existing defense apps and past AOSP patches aimed at preventing voice phishing. We then present their shortcomings that next-generation voice phishing defenses should overcome.

4.1 Existing Defense Apps

To investigate current app-level phishing mitigation available in Google Play, we collected four phishing defense apps that are specifically designed to detect voice phishing apps: whowho [110], Phishing Eyes [40], Anti Scam [93], and Anti Spy [102]. We used the search keywords "voice phishing defense" and "voice phishing prevention" to compile the list.

Phishing app detection. All the apps leverage blacklists of voice phishing apps. Since the Android system allows a mobile app to extract all package names in the same device, they check for the presence of known package names of voice phishing apps. However, this blacklist-based approach suffers from false negatives when encountering new phishing apps. The Anti Scam app [93] implements an additional approach to check for dangerous permissions that enable outgoing call redirection, reading/writing call logs, reading phone call states, and reading/writing contacts. However, this approach also yields a large number of false positives because the permissions that voice phishing apps leverage are widely used. 28,430 benign apps use at least one permission among the permissions noted above.

Call redirection detection. Only one app, whowho, supports the detection of call redirection. It detects any changes to the user's outgoing call number and identifies the app responsible for this call number change.

This app detects call redirection by comparing the outgoing call number with the phone number in the most recent call log. Specifically, the app creates an Intent receiver that the Android system calls upon receiving `android.intent.action.NEW_OUTGOING_CALL` to identify an original phone number. It is paramount for this app to know this original number before any other apps change the outgoing number in the Intent; therefore, this defense app specifies the priority of the Intent receiver to the high number of 2,147,483,647, preventing other app receivers from being invoked before this app. After identifying the original number, the app collects the outgoing number from the latest call log. When the two differ, the app then checks for the presence of any phishing package names in the currently installed device. When all the aforementioned conditions are satisfied, the app warns users.

This approach has several problems. (1) When the Android receiver of a phishing app has the same highest priority and is invoked before whowho's receiver, the defense app cannot obtain the original outgoing call number. (2) Several phishing apps modify the latest call log to the original phone number after call redirection occurs. If a phishing app changes the latest call log before whowho reads it, this defense app is unable to detect whether call redirection has occurred. In other words, this defense app is unable to retrieve the original outgoing call number due to its limited access to internal Android system information.

Limitations of app-level defenses We argue that these detection apps fail to establish a sufficient level of security protection for mobile users. This failure stems from two limitations: (1) blacklist-based approaches are trivially bypassable, and (2) only limited information is accessible to defense apps.

Three apps have leveraged blacklists of suspicious package names to warn users about the presence of suspicious apps. However, the attackers are able to develop an APK with their choice of package name, trivially bypassing the blacklists of known phishing package names. Considering that voice phishing apps are often distributed by SMS messages, not by official app markets, fraudsters have no restrictions in choosing their package names.

Moreover, these defense apps also have a limited capability to access information pertaining to other apps. By the Android system design, the defense apps are able to obtain other apps' manifest files, which include permissions, Intent filters, and components. However, they are unable to observe the dynamic execution behaviors of other apps. Thus, app-level detection has focused on leveraging a classifier based on limited features, such as package names, APK signatures, and permissions, which allow fraudsters to easily manipulate.

Existing anti-virus solutions. Existing anti-virus solutions also use their own methodology to statically detect mobile phishing apps. However, note that their designed goal is to detect malicious mobile apps, not mobile phishing apps. They have focused on identifying apps exfiltrating personal information, hijacking passwords, or tracking victims' physical locations [48, 92, 98, 103, 115]. We checked 12 live phishing apps obtained from one anti-virus corporation by uploading those to VirusTotal. Only 16.4% of the existing solutions classified them as malicious apps. These results demonstrate that existing anti-virus solutions also suffer from false negatives due to their detection strategy of leveraging known malware signatures.

High privacy cost. We observed that one app (i.e., Phishing Eyes) sends users' incoming/outgoing call numbers, SMS texts, voice calls, or installed app lists to its server, each of which is privacy-sensitive information [21, 72, 73]. We believe that the existing approach of sending privacy-sensitive information demands a high privacy cost, rendering this approach impractical.

4.2 Patching Android APIs

Google has proposed several updates to the ways of using Android APIs for implementing call redirection and overlaying call screens [9, 31, 89, 111]. These updates usually restrict access to these APIs and require user consent. For instance, the recent changes of requiring the "Display over other apps" permission to overlay a

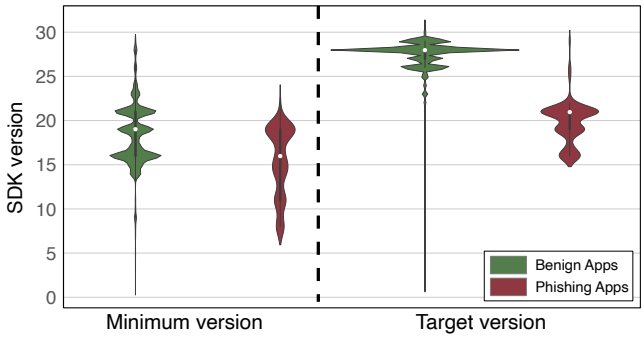

Figure 3: SDK version distribution: the width of each curve represents the frequency of SDK versions; the white dot shows the median; and the thick center line represents the interquartile range.

UI over other apps and the usage of `CallRedirectionService` to make outgoing calls may mitigate voice phishing threats.

These behavior changes of Android APIs have been released in a new version of Android SDK. To apply these updates, an Android app should be compiled with an updated Android SDK, instructing the Android system to enforce the updated policy according to its target Android SDK version.

We believe that patching Android APIs requiring a new SDK version is ineffective in mitigating voice phishing threats. We observe that fraudsters simply use low target versions of the Android API level to develop their phishing apps. For example, `PROCESS_OUTGOING_CALLS` is deprecated in Android 10 (API level 29). However, when a fraudster uses a target SDK version lower than 29 in developing their phishing app, they are still able to abuse `PROCESS_OUTGOING_CALLS` in any Android device. Furthermore, if the attacker lowers their target SDK version below Android 6 (API level 23), Android devices use the installation time permission model even for dangerous permissions [95], such as `PROCESS_OUTGOING_CALLS`, instead of enforcing the runtime permission model [12]. It is well acknowledged that users tend to blindly accept installation time permissions [18, 41, 70]. This is due to the backward compatibility of Android since the Android system still needs to support Android apps built with old versions of Android SDK.

Fraudsters have been abusing this backward compatibility to make their phishing apps not exhibit the updated behaviors introduced by new Android SDKs. Therefore, fraudsters are even motivated to use old SDK versions to bypass any mitigation deployed in the later SDK versions. For apps in Google Play, Google has not allowed uploading APKs that do not meet the minimum target API level [11]. By contrast, phishing apps have no restrictions in using a low target API level since they have been distributed over SMS messages.

Figure 3 shows the minimum and target SDK versions between phishing and benign app groups; the phishing and benign app groups consist of 1,017 and 59,869 apps, respectively. We collected these 59,869 apps in Google Play [15, 16, 17, 35] from April 2019 to March 2021 using two methods. (1) We collected 10,024 apps by periodically crawling 100 highly ranked apps in each of the 35 app categories. (2) To cover less popular apps, we randomly crawled 49,845 apps from Google Play APK mirror sites [15, 16, 17].

As the figure shows, the median value of the target SDK versions of voice phishing apps is 21, while the median value of those benign apps is 28. This means that voice phishing apps intentionally use lower target SDK versions than do benign apps. Therefore, patching Android APIs is ineffective in preventing these phishing apps from abusing the APIs.

5 OVERVIEW

We propose an Android system-level defense, HearMeOut, which is designed to monitor voice phishing behaviors. We enumerate the security and privacy requirements that our new defense system addresses (§5.1). We then describe the overall architecture of HearMeOut (§5.2).

5.1 System Requirements

We enlist security and privacy requirements that a next-generation voice phishing defense should address to overcome the limitations of the existing methods (§4).

Runtime detection. We argue that static detection methods, including blacklist-based approaches, are ineffective in identifying new phishing apps. Fraudsters can obfuscate their phishing apps, which makes it difficult to extract unchanging and distinguishable features, including method signatures and distinctive code patterns. Furthermore, they are able to keep mutating their phishing apps until they pass a target system of static detection [90, 104]. They also unload their core functionality of voice phishing via dynamic DEX files and leverage their command and control servers to update their DEX files (§3.5). Therefore, the static detection of using known signatures prolongs the arms race between fraudsters and detection systems.

To bring positive changes to this arms race, a new defense system should support the dynamic detection of phishing behaviors when they actually occur. The goal of voice phishing apps, including new ones that bypass static detection, is to perform abusive behaviors in runtime. Thus, focusing on the identification of these runtime behaviors avoids compiling and managing a vast volume of known detection signatures.

System-level detection. The app-level detection of phishing apps fails to provide a sufficient level of protection for users. By design, an Android app has limited access to other apps' information; an Android app can only access the manifest files of other apps and their APK binaries [32, 33]. The Android framework does not allow any system and user apps to monitor other apps' runtime behaviors. Therefore, monitoring runtime behaviors requires support from the Android framework.

Client-side defense. Detecting voice phishing apps in app markets is not directly effective in protecting phishing victims. We note that voice phishing apps are distributed via SMS texts, and phishing victims install these apps on their devices even though they are not from legitimate app markets (§2). Thus, enhancing the vetting process of these apps in app markets has limited impact in preventing victims from being abused.

Less privacy-intrusive. A defense method should be less privacy intrusive such that users have no needs to provide their privacy-sensitive information to another party. Also, it should not leak privacy-sensitive information to other mobile apps.

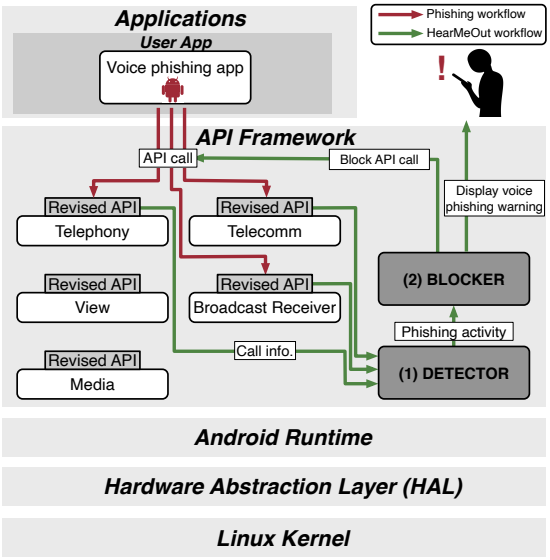

Figure 4: HearMeOut architecture.

5.2 Overall Architecture

We propose HearMeOut, an Android system that detects voice phishing activities in runtime and warns users of the detected activities via system-level notifications.

Figure 4 illustrates the overall architecture of HearMeOut, which consists of two components: Detector and Blocker. These components operate as Android system services. At a high level, these components work together to perform two steps: (1) Detector detects suspicious voice phishing activities; and (2) Blocker blocks detected suspicious voice phishing activities.

Note that the Android framework has several software layers [34]: it consists of the Linux kernel, hardware abstraction layer (HAL) that manages Linux kernel drivers, Android runtime, the API framework, and system/user apps. As Figure 4 shows, voice phishing apps perform phishing behavior by invoking Android APIs provided by five service managers at the API framework layer (§3). The context information that pertains to invoking each of these APIs, such as the invocation time, actual arguments, and the app name of calling the API, provides rich information to distinguish phishing behaviors from benign ones. We revised these APIs to pass context information to Detector.

Detector. Detector detects voice phishing behaviors that phishing apps conduct. It detects call redirection, call screen overlay, and fake call voice. Detector leverages the context information when invoking the APIs of Telephony, Telecomm, Broadcast Receiver, View, and Media modules.

Blocker. Blocker is designed to block phishing behaviors that Detector identifies. When a voice phishing app exhibits call redirection, call screen overlay, or fake call voice, Blocker blocks any of these phishing behaviors. In particular, when a call redirection attempt occurs, Blocker displays a warning dialog, as shown in Figure 5, along with an alarm sound and vibration simultaneously. The warning dialog is designed to leave an active warning that hinders users from ignoring the warning message. For this, we

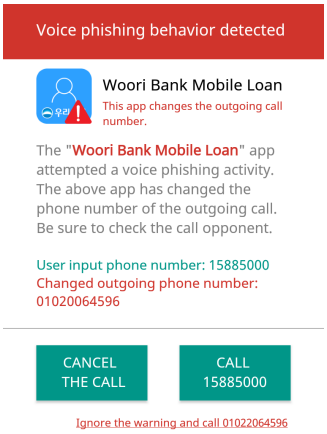

Figure 5: Phishing redirection call detection warning message in HearMeOut.

referenced previous active warning studies for web browsers and Android [5, 6, 39, 42, 74].

The warning dialog provides a description that explains that call redirection occurred due to a specific Android app while providing three options: (1) make an outgoing call to the original phone number that the user provided, (2) make an outgoing call to the redirected phone number, or (3) terminate the outgoing call. We intentionally make it difficult for users to choose the second option by not providing a clickable button.

6 IMPLEMENTATION

We implemented HearMeOut in Android 8.1 [30] and revised all phishing-relevant APIs at the Android framework layer to implement the detection of phishing activities. We now describe the detection method for each type of phishing behavior, along with criteria to distinguish between phishing and normal activities. To identify the characteristics of benign functionalities, we used the 59,869 benign apps crawled from Google Play (§4.2). For each phishing functionality, we sampled 200 benign apps that have permissions or Intent filters required to implement the respective functionality. That is, this preliminary analysis guided us in devising a detection policy for each phishing functionality.

Call redirection. To detect a phishing behavior that commits call redirection, Detector compares the original call number the victim enters to the final outgoing call number. When call redirection occurs, Detector starts by identifying the original and final call numbers. It then checks whether the final number includes the original number. If the initial call number is not a substring of the final number, Detector regards this recall redirection attempt as a phishing activity.

Specifically, we revised `broadcastIntent()`, which initiates call redirection [94] to obtain the initial call number. We also modified `setResultData()`, which changes the outgoing call number [36] and extracts a package name that changes the number via calling this API. Detector checked these two numbers in `onCreateOutgoingConnection()`, which is a callback that occurs upon every outgoing call event [8].

We note that the phishing apps' pattern of changing outgoing call numbers is highly distinctive from that of benign apps. We analyzed 1,045 apps that use the PROCESS_OUTGOING_CALLS permission and the ACTION_NEW_OUTGOING_CALL Intent filter, which are required conditions for call redirection. We manually analyzed the decompiled code of their Intent receivers for ACTION_NEW_OUTGOING_CALL; we observed that most apps were designed to add local prefix numbers or to simply acknowledge outgoing call events. We found only three exceptions of benign apps that automatically appended a specific number (i.e., *281) to the original call number to support the "number plus" service [29]. On the other hand, voice phishing apps attempt to replace an original call number with a completely different call number that belongs to the attacker. Therefore, leveraging the inclusion relationships between the initial and final call numbers enables detecting phishing activities without false positives and negatives.

When Detector detects call redirection, Blocker cancels the outgoing call at `onCreateOutgoingConnection()` and displays the warning dialog shown in Figure 5 using `WindowManagerImpl.addView()`.

Note that we directly changed the function body of each API, instead of demanding Android SDK updates to enforce the security policies that depend on a target SDK version (§4.2). HearMeOut is thus able to monitor abusive usage patterns of APIs regardless of the target SDK versions that phishing apps use.

Call screen overlay. Detector checks whether the call screen overlay (1) appears immediately after an incoming or outgoing call occurs and (2) covers the area where the original phone number appears. We devised the first condition based on our observation that all call screen overlays work in tandem with call redirection (§3.3). The second condition captures phishing behaviors that hide original call numbers.

Specifically, Detector first checks whether `WindowManagerImpl.addView()`, which dynamically adds the overlay to the layout [47], is called after `onCreateIncomingConnection()` or `onCreateOutgoingConnection()` is called. Detector checks whether the time difference between two API invocations is within one second. Detector then computes the position and size of the created overlay via `WindowManagerImpl.addView()` and checks whether it covers the calling number of the call screen in a Pixel 2 device. Detector computes whether the overlay covers the entire width of the screen where the call number is displayed.

We analyzed 200 randomly sampled apps from 59,869 apps that declared `android.permission.SYSTEM_ALERT_WINDOW`. No apps displayed an overlay on the call screen nor hid the authentic phone number with this overlay. Only phishing apps attempted to cover the display portion where the phone number appears (§3.3).

When Detector detects the call screen overlay, Blocker removes the call screen overlay in the `WindowManagerImpl.removeView()`.

Fake call voice. Detector detects any attempts to play a fake call voice by checking whether a target app plays any voice recording immediately after the outgoing call occurs. Specifically, Detector checks if `MediaPlayer.start()`, which plays an audio/video resource [37], is called after `onCreateoutgoingConnection()` is invoked within a fixed time interval.

Activity	# of Phishing Apps	# of Benign Apps	Accuracy
Call redirection	12	6	
Screen overlay	6	15	100%
Call voice	6	20	
Total	12	33	100%

Table 2: Accuracy of each phishing detection method.

When Detector detects the fake call voice, Blocker blocks playback of the voice file in `MediaPlayer.start()`.

7 EVALUATION

We evaluate the efficacy of HearMeOut in detecting voice phishing behaviors (§7.2) and measure the performance overhead of HearMeOut in identifying such behaviors (§7.3).

7.1 Experimental Setup

We conducted experiments with a Pixel 2 smartphone running the revised AOSP 8.1 with HearMeOut [30]. The device has a 2.35 Hz 2-core CPU, 64 GB storage, 4 GB RAM, and a 2,700 mAh battery.

Benchmarks. We ran a series of experiments on 12 live voice phishing apps collected from the anti-virus corporation (§3). Note that it is difficult to obtain live phishing apps because phishing apps usually have a short time window for their active phishing campaigns. After this short time window, they terminate their command and control servers, rendering phishing apps no longer functional. We also prepared 53 benign apps sampled from our benchmarks of 59,869 apps in the Play Store [15, 16, 17, 35].

7.2 Detection Accuracy

We measured the accuracy of HearMeOut in detecting phishing activities in runtime. Table 2 shows our experimental results. Each row corresponds to a phishing activity that we tested for HearMeOut. The second and third columns show the numbers of phishing and benign apps, respectively, that implement the corresponding activity. We counted these numbers by checking whether the respective apps implement the corresponding functionality by manually analyzing the decompiled code and manifest files of 53 benign and 12 phishing apps. The benign apps we collected for accuracy testing include the following: "number plus" service apps that use call redirection APIs, spam blocking apps that use overlay, and blocking APIs, music playback apps that use voice play APIs, and personal information management apps that access SMS messages and contact lists.

Triggering phishing activities. We manually conducted the following tests for each app to execute APIs that HearMeOut monitors.

- Call redirection: We made an outgoing call to the specific call number that the app under testing shows in its main activity. We observed that the main windows of all six phishing apps displayed the actual phone numbers of the organization that they disguised themselves as. For example, to trigger call redirection from the six phishing apps, we made an outgoing call to 1644-7777, which is an authentic Shinhan savings bank's phone number that the app shows in its main activity window. For benign apps, we followed the guideline of each app to turn on the call redirection function (e.g., the "number plus" service [29]).

Activity	w/o HearMeOut		w/ HearMeOut	
	Phishing Apps	Benign Apps	Phishing Apps (ΔTime)	Benign Apps (ΔTime)
Call redirection	155.03	154.25	155.81 (+0.78)	154.42 (+0.17)
Screen overlay	10.86	10.86	10.97 (+0.11)	10.93 (+0.07)
Call voice	12.33	12.30	12.52 (+0.19)	12.50 (+0.20)

Table 3: Average execution time (ms) of each benign and phishing activity with/without HearMeOut.

- Screen overlay and call voice (media play): For every phishing app, we triggered call redirection to invoke screen overlay or fake voice call functionalities. Remind that these functionalities work together with call direction (§3). For each benign app, we played music or triggered a functionality that shows overlaid alert messages with phone number information for scamming calls.

As the table shows, HearMeOut accurately detected phishing activities without false positives and negatives. These results show that the detection methods of Detector, which leverage the abusive patterns of using Android APIs, are effective in detecting phishing activities.

Note that we did not use these live phishing apps to devise detection policies for diverse phishing behaviors (§3). Nevertheless, HearMeOut still shows high detection accuracy, demonstrating that HearMeOut captures common phishing patterns of abusing APIs. We also observed that the existing blacklist- or signature-based methods [40, 93, 102, 110] did not detect any of these live phishing apps, demonstrating their limitations. They suffer from false negatives when encountering new apps for which they do not have matching signatures (§4.1).

7.3 Performance Overhead

We measured the execution time and power consumption of HearMeOut in identifying phishing activities for 53 benign and 12 phishing apps.

Execution time. We ran each phishing activity 10 times using the activity-triggering method mentioned in Section 7.2 and measured the average execution time for each activity. In particular, we measured the execution time from when an activity started to when the activity was completed. For each activity, the execution time was measured according to the following criteria.

- Call redirection: execution time from the occurrence of an outgoing call Intent to the connection of the outgoing call.
- Call screen overlay: execution time of the `addView()` method.
- Fake call voice: execution time from the `MediaPlayer` instance creation to the `MediaPlayer.start()` method completion time.

To measure the impact of HearMeOut on the performance of the entire Android system, we compared its performance with and without HearMeOut. Table 3 shows the average execution time for each activity with and without HearMeOut. We observed that the execution overhead of HearMeOut was negligible; the additional average latency in detecting a phishing activity was 0.36 ms. We note that this additional latency includes the execution times of the monitoring logic in revised APIs, the detection process of Detector.

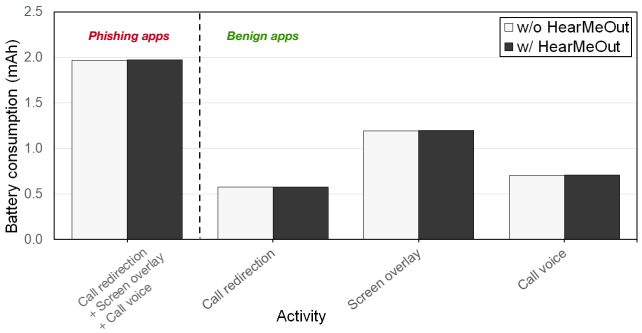

Figure 6: Average battery consumption for each scenario with/without HearMeOut.

For each functionality, we plotted the empirical cumulative distribution function (CDF) of observed latencies with and without HearMeOut. We noticed that the latency differences were negligible. HearMeOut required up to 0.65 ms of additional latency in median latency (50% of the CDF) and 5.8 ms of additional latency in the 90% of the CDF that call redirection causes.

Power consumption. We conducted each activity triggering scenario 10 times and measured the average power consumption of HearMeOut for 53 benign and 12 phishing apps. To measure the amount of consumed power, we used the Battery Historian profiler [7], which is a battery information measurement module developed by Google. This tool estimates the battery consumption of an Android device caused by the Android system, running apps, Wi-Fi, CPU calculations, etc. Among these factors, we focus only on analyzing the power consumption of apps under testing and the Android system, which are directly related to phishing activity.

Figure 6 shows the average battery consumption for each activity with/without HearMeOut. We observed that the additional battery consumption of HearMeOut is negligible; the additional average battery consumption of the activity detection process was 0.00068 mAh (0.09% increased) compared to without HearMeOut.

8 USER STUDY

To measure the efficacy of HearMeOut in mitigating voice phishing threats involving call redirection, we recruited 45 participants and investigated their responses to phishing behaviors. We describe user study designs (§8.1) and experimental results (§8.2).

8.1 Experimental Design

Phishing app. We implemented a mock phishing app that conducts call redirection, call screen overlay, and fake call voice. We designed the phishing app to disguise itself as the Woori mobile bank app, one of the largest banks in South Korea. We implemented call redirection to redirect any outgoing calls to a Woori Bank official service center to our actor's mobile phone number.

Actor. We hired an actor to play the role of a voice phishing fraudster. The actor had prior experience of working as a service representative at a bank call center for three years. The actor pretended to be a Woori Bank call service agent. We instructed the actor to impersonate a Woori Bank agent and ask for personal information from participants who made calls to the actor. Specifically, the

actor asked for five types of personal information: name, occupation, residential address, bank account number, and social security number.

Participants. From November to December 2021, we recruited a total of 45 participants, consisting of 26 males and 19 females (ages 20 to 63). All participants were Android mobile phone users and had called a Woori Bank official service center within the past month. In the recruiting process, we advertised the user study as a call interaction observation study.

Each participant was offered $10 to participate in their own phishing scenario, which took approximately 30 to 40 minutes. For each participant, we prepared a Pixel 2 mobile phone pre-installed with our phishing app. We then asked them to call the Woori Bank's official service center. The participants were also asked to inquire about applicable loan products along with loan interest rates, loan repayment periods, and loan limits. Participants' calls to the Woori Bank service center were redirected by our phishing apps to our actor. When the participants asked about applicable loan products, the actor requested their name, occupation, bank account number, and social security numbers to look up the loan information.

From the 45 participants, we randomly assigned 23 participants to a group not using HearMeOut (i.e., the control group) and the remaining 22 to a group using HearMeOut (i.e., the experimental group). Accordingly, the 22 participants in the experimental group were intended to observe HearMeOut displaying phishing redirection warning messages (see Figure 5).

At the end of the user study, each participant was asked to complete a survey asking about their awareness of the ongoing voice phishing campaign and the meaning of the warning dialog. The questionnaires are described as follows:

- Did you think the phishing app you used was a real banking app? Why, or why not.
- Did you think the person on the phone was a Woori Bank service representative? Why, or why not.
- Did you understand the message in the HearMeOut warning dialog?

Ethics. We designed a user deception study to simulate a real-world voice phishing scenario. We obtained IRB approval with the following restrictions. (1) We were only to collect personal information pertaining to the user study. (2) We were not to tamper with the WooriBank daily operations. (3) We were to obtain two consent forms from each participant: one for the informed study (i.e., call interaction observation) at the beginning of the user study and the other for the intended study (i.e., voice phishing) at the end of the user study. (4) We were to destroy collected personal information belonging to each individual, except for aggregated statistics for publication.

Furthermore, we explicitly informed the participants that they did not need to provide unwanted personal information during the experiment and had the option to stop the user study at any time. In addition, after the study was finished, each participant was informed of the original purpose of the study and given the option to have all experimental data discarded. No participants requested the disposal of their user study results.

	Control	Experimental
Total # of Participants	23	22
Call redirection occurs	23 (100%)	0 (0%)
Trusts callee as a bank agent	20 (87%)	0 (0%)
Gives personal information	16† (69.6%)	0 (0%)

† We exclude four participants who were reluctant to provide personal information in the user study.

Table 4: Experimental results for participants with and without HearMeOut.

8.2 Experimental Results

Table 4 summarizes the experimental results of the user study. Overall, we observed that the participants in the control group without HearMeOut were highly susceptible to voice phishing. In particular, 20 participants (87%) believed that their outgoing calls were connected to a Woori Bank service representative, and 16 participants (69.6%) provided their personal information. On the other hand, none of the 22 participants in the experimental group with HearMeOut called the redirected phone number. These results demonstrate that the displaying warning message of HearMeOut in Figure 5 is effective in mitigating voice phishing threats involving outgoing call redirection.

Control group without HearMeOut. We noticed that 20 participants had called the Woori Bank's service center within the past month. However, they did not raise their doubt on possible voice phishing campaigns. Table 5 shows reasons that contributed to the victims being deceived by voice phishing. All the deceived participants trusted the person they called as an authentic service representative because they had directly entered the phone number themselves. They did not think that the outgoing call number could be changed by a phishing app. Also, 10 of 20 participants were deceived because the recipient's call number on the outgoing call screen was the authentic number with which the phishing app had covered the redirected number using a fake call screen overlay. In addition, six participants believed that they were connected to Woori Bank because of the Woori Bank's automatic greeting voice that the phishing app played. We emphasize the critical threat that voice phishing apps pose especially when outgoing call redirection, call screen overlay, and fake call voice operate together.

Among the 20 deceived participants, 10 participants had prior experiences of receiving voice phishing calls several times. They were already aware of the threat that voice phishing renders. However, they did not account for the possibility of getting their outgoing calls changed. These results reveal that the general public are less familiar to the unique threat that voice phishing apps pose than does traditional voice phishing.

Interestingly, three participants noticed that the recipient of the outgoing call was not a Woori Bank service representative. Two of them answered that they suspected the recipient because they were not first connected to Woori Bank's automatic response system (ARS) but were instead directly connected to the service representative. Another participant answered that he noticed that the actor's questions were too simple compared to a real agent's questions based on his prior experience of requesting a loan product.

We noticed that the participants' suspicion of the authenticity of the phishing app affected the success rate of the voice phishing campaign. All 12 participants who believed the phishing app as

Reasons	# of participants
Just entered the call number to Woori Bank	20 (100%)
Callee's reliable tone and words	12 (60%)
Correct number displayed in the call screen	10 (50%)
Woori Bank's automatic greeting voice played	6 (30%)

Table 5: Reasons for believing that the recipient was a Woori Bank agent. Participants were able to select multiple reasons.

an authentic Woori Bank app were deceived by voice phishing. By contrast, 10 participants raised doubt on the authenticity of the phishing apps. Three participants of them noticed that the recipient of their calls was not a Woori Bank service representative. However, 70% were still deceived due to call redirection.

Experimental group with HearMeOut. None of the 22 participants in the experimental group called the redirected phone number, indicating that the HearMeOut warning prevented the participants from calling the redirected number.

Among the 22 participants, 12 were aware of all three clickable options in Figure 5. They said that there was no reason to call the redirected number. Eight participants only recognized two buttons and not the button to call the redirected number. They did not read the warning message carefully. However, in the end, the participants did not call the redirected number and called the original phone number, which fulfills the intended goal of preventing phishing. The eight participants pressed the call button that connected to the phone number that they entered. Another participant made a call after deleting the phishing app. After reading the warning, she thought it would be a good idea to delete the phishing app because it changed the outgoing call number. The other one stopped the experiment before any outgoing call occurred; he thought that the phone might be connected to a phishing scammer.

9 LIMITATIONS AND DISCUSSION

HearMeOut deployability. HearMeOut has a limitation in that it requires the Android system changes to implement Detector and Blocker. This means that HearMeOut requires support from key AOSP maintainers and phone manufacturers [65], such as Google, Samsung, and Motorola. Note that Android phone manufacturers have been providing security updates every one to three months [10]. We believe that HearMeOut can be shipped to end users via these security updates of Android frameworks.

We hope that these key players become more active in identifying phishing activities in runtime. As we mentioned, finding more phishing apps and updating Android APIs have a limited impact on preventing phishing apps from abusing victims (§5.1). These phishing victims download their phishing apps via URLs and give full consent to them, and fraudsters use low Android SDK versions to bypass the restrictions in using APIs. We argue that AOSP changes for HearMeOut are small but effective in preventing mobile phishing victims from being abused.

Other voice phishing methods. Note that there are other types of phishing mitigation, such as phishing voice classification and warning of suspicious SMSs. These methods focus on identifying primitive phishing attempts. However, HearMeOut focuses on identifying phishing call redirection activities that mobile phishing apps conduct to deceive victims, such as call redirection and call screen overlay. Therefore, HearMeOut is a complementary tool to the existing approaches, but is designed to identify advanced phishing activities.

Comparison with incoming call voice phishing. Voice phishing using outgoing calls could be more effective in deceiving victims than voice phishing methods using incoming calls. Our experimental results show that 87% of participants were tricked by phishing campaigns involving call redirection and that 69.6% disclosed their personal information. The success rate of this outgoing call voice phishing campaign was two to six times higher than for incoming call voice phishing campaigns. Aburrous *et al.* [1] measured the success rate when a female co-worker contacted employees by phone and asked for their Internet banking credentials. They were able to deceive 32% of the employees into giving out their e-banking credentials. In more recent work, Tu *et al.* [109] executed an impersonation telephone phishing scam and found that the spoofed Caller ID had a significant effect in tricking victims. They achieved a 10.33% deception rate in convincing recipients to divulge the last four digits of their social security numbers. A distinct advantage of outgoing call voice phishing over incoming voice phishing is that victims trust the outgoing call number because they enter it themselves. In our experiment, all 20 deceived participants answered that they trusted the callee as a Woori Bank agent because they had entered the Woori Bank call number themselves.

Android countermeasures. Android has been restricting app installation from unknown sources by explicitly asking a user's consent when the user attempts to install an app by clicking the URL in a phishing SMS message. However, numerous legitimate apps from corporations and organizations have been using the same installation channel, educating the general public more susceptible to app installation from unknown sources. Also, Google Protect has been warning users when the users attempt to install known malicious apps. However, we observed that 12 live phishing apps in Section 7.1 were not triggered by Google Protect until their phishing campaigns were over, rendering them alive for up to three weeks. We believe that this is due to (1) the short-lived nature of mobile phishing apps that keep changing their app signatures and (2) the distribution channel of using SMS messages, not exploiting official app markets, including Play.

Call redirection in iOS. iOS does not provide APIs to perform call redirection in iOS [59], which renders the attackers unable to change a recipient call number. Also, iOS does not allow creating an overlay on top of other apps. Only an app in the foreground is able to display an overlay [60, 63], thus making the attackers unable to create a fake call overlay to replace the outgoing call number displayed on the screen. By contrast, iOS apps can play media files in the background [62], which the attacker may abuse to play fake call voices. In addition, fraudsters have restrictions in distributing iOS phishing apps; iOS provides limited ways of installing apps besides App Store. Thus, victims may install phishing apps only through iTunes or Xcode on their devices [61, 64]. However, these installation channels require physical access to iOS devices, thus rendering fraudsters difficult to install phishing apps on victims' iOS devices. We believe that these exclusive policies not allowing to install apps from third-party sources are not applicable to Android ecosystem. Android is based on an open-source ecosystem that

involves diverse device manufacturers and other legitimate app markets.

10 RELATED WORKS

Phishing. Previous research has proposed numerous techniques for analyzing and identifying email phishing attacks [54, 55, 56, 87]. The most common way to detect phishing attacks has been content filtering in an anti-spam manner, including URLs, messages, and attachments [27, 38, 54, 78, 105, 106, 114]. IdentityMailer [4] detects email phishing attacks based on the trained behavior models of senders. The authors extracted header information from an email, timing patterns, and stylometric features for each user. Ho *et al.* [54, 55] investigated lateral phishing whereby attackers take over a compromised legitimate account and then send phishing emails.

Another recent line of research focuses on the identification of phishing webs. Several approaches rely on blacklisting by leveraging crowdsourcing [43, 75, 83, 88] and reputation systems [77] to improve detection accuracy. Han *et al.* [51] analyzed the entire life cycle of phishing websites by using a honeypot system. In particular, they measured the impact of blacklisting services of the reported or discovered sites by security companies from the time they were first installed. Unfortunately, blacklisting services have their own limitations since phishing attacks rarely last more than a few days [99]. Phishing blacklists also suffer from incomplete coverage [14], behavior-based JavaScript evasion [86], and cloaking [58, 85].

Telephone scams. There has been a surge in research in the telephone domain for telephone support scams [53, 82], voice spam [19, 66, 108, 113], and voice phishing [22, 23, 49, 50, 79, 109].

Maggi [79] studied the modus operandi of incoming call voice phishing attackers by analyzing the reports of voice phishing victims. In particular, the authors investigated several attributes, including Caller ID, transcribed conversations, subject, and language, and revealed that the phishers relied on automated responders to make it easy to conduct and maintain their phishing campaigns. Tu *et al.* [109] conducted a large-scale phishing study, showing that incoming call telephone scams were unexpectedly successful. This work has uncovered that the incoming call telephone scam can be made more convincing by spoofing the caller ID. Chang and Lee [22] proposed an incoming call voice phishing detection system incorporating the codec parameters of the transmitted speech information. Several data collection systems have been studied, such as emplacing phone bots [80] or deploying honeypots [50].

None of these techniques can fully characterize the voice phishing threats using the victim's outgoing call from voice phishing apps and, therefore, are incapable of defending against attacks from these apps. Unlike the existing voice phishing studies, however, we first measured the effect of voice phishing using outgoing calls, analyzed the apps used for voice phishing, and designed HearMeOut, a mitigation method of voice phishing.

11 CONCLUSION

We conducted the first investigative study that manifests the key functionalities of voice phishing apps, which have been largely understudied. Our analysis results reveal (1) key voice phishing functionalities, including call redirection, and (2) the limitations of currently available mitigation methods by demonstrating how they fail to protect victims. We propose HearMeOut, which addresses the shortcomings of existing approaches, and suggest changes to the Android system for monitoring suspicious runtime behaviors involving voice phishing. Our user study demonstrates that HearMeOut is capable of preventing users from calling fraudsters by displaying active warnings. Our findings facilitate a better understanding of Android voice phishing apps. We conclude by calling for changes to the Android system in order to expand protection to users from voice phishing apps that are not even from legitimate app markets.

ACKNOWLEDGMENTS

We thank the anonymous reviewers and our shepherd Ben Greenstein for their concrete feedback. We also appreciate Ahnlab and Financial Security Institute for providing voice phishing apps. This work was supported by Samsung Research and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT), No.2020-0-00209.

REFERENCES

[1] M. Aburrous, M. A. Hossain, K. Dahal, and F. Thabtah. Experimental case studies for investigating e-banking phishing techniques and attack strategies. *Cognitive Computation*, 2(3):242–253, 2010.
[2] N. P. Agency. National police agency - voice phishing state 2016 to 2020. https://www.data.go.kr/data/15063815/fileData.do, 2021.
[3] AhnLab. Ahnlab - leader in cyber threat analysis and response. https://www.ahnlab.com/, 2022.
[4] T. ain't you: Blocking spearphishing through behavioral modelling. Stringhini, gianluca and thonnard, olivier. In *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 78–97, 2015.
[5] D. Akhawe and A. P. Felt. Alice in warningland: A large-scale field study of browser security warning effectiveness. In *Proceedings of the USENIX Security Symposium*, pages 257–272, 2013.
[6] H. Almuhimedi, F. Schaub, N. Sadeh, I. Adjerid, A. Acquisti, J. Gluck, L. F. Cranor, and Y. Agarwal. Your location has been shared 5,398 times! a field study on mobile app privacy nudging. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 787–796, 2015.
[7] Android. Android developers - setup battery historian. https://developer.android.com/topic/performance/power/setup-battery-historian, 2021.
[8] Android. Manifest permissions - process outgoing calls. https://developer.android.com/reference/android/Manifest.permission#PROCESS_OUTGOING_CALLS, 2021.
[9] Android. Manifest permissions - system alert window. https://developer.android.com/reference/android/Manifest.permission#SYSTEM_ALERT_WINDOW, 2021.
[10] Android. Android developers - android security bulletins. https://source.android.com/security/bulletin?hl=ko, 2022.
[11] Android. Android developers - google play target api level requirement. https://developer.android.com/distribute/best-practices/develop/target-sdk, 2022.
[12] Android. Android developers - request app permissions. https://developer.android.com/training/permissions/requesting, 2022.
[13] Android. Android developers - unknown sources. https://developer.android.com/studio/publish#unknown-sources, 2022.
[14] S. Aonzo, A. Merlo, G. Tavella, and Y. Fratantonio. Phishing attacks on modern android. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1788–1801, 2018.
[15] APKMirror. Apkmirror - free and safe android apk downloads. https://www.apkmirror.com/, 2020.
[16] APKMonk. Apkmonk - download android app apks free. https://www.apkmonk.com/, 2020.
[17] APKPure. Apkpure - download apk free online downloader. https://apkpure.com/, 2020.
[18] 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, 2012.
[19] Y. Bai, X. Su, and B. Bhargava. Detection and filtering spam over internet telephony-a user-behavior-aware intermediate-network-based approach. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 726–729, 2009.

[20] Bangcle. Android hardening protection, load the encrypted dex file from memory dynamically. https://www.bangcle.com/, 2021.

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

[22] J.-H. Chang and K.-H. Lee. Voice phishing detection technique based on minimum classification error method incorporating codec parameters. *IET signal processing*, 4(5):502–509, 2010.

[23] K. Choi, J.-l. Lee, and Y.-t. Chun. Voice phishing fraud and its modus operandi. *Security Journal*, 30(2):454–466, 2017.

[24] I. Chosun. Voice phishing damage over the past 5 years is 1.7 trillion won. 170,000 victims occurred. https://it.chosun.com/site/data/html_dir/2020/09/28/2020092802480.html, 2021.

[25] ChosunBiz. "create a fraudulent banking app and intercept the call to the customer center"... voice phishing is evolving. https://biz.chosun.com/site/data/html_dir/2021/03/16/2021031602609.html, 2021.

[26] ChosunBiz. Police arrest two voice phishing criminals... only 20 victims were tempted by'return loan'. https://biz.chosun.com/site/data/html_dir/2021/02/17/2021021700316.html, 2021.

[27] A. Cidon, L. Gavish, I. Bleier, N. Korshun, M. Schweighauser, and A. Tsitkin. High precision detection of business email compromise. In *Proceedings of the USENIX Security Symposium*, pages 1291–1307, 2019.

[28] CNN. A phone scammer posing as a detective called a police station in south korea. it didn't end well. https://edition.cnn.com/2019/11/26/asia/south-korea-police-phone-scam-intl-hnk-scli/index.html, 2021.

[29] L. COMPANY. Auto two number. https://play.google.com/store/apps/details?id=kr.co.thecheat.autotwonumber, 2021.

[30] A. Developers. Android 8 release notes. https://developer.android.com/about/versions/oreo/android-8.1, 2021.

[31] A. Developers. Android documentation, CallRedirectionService. https://developer.android.com/reference/android/telecom/CallRedirectionService, 2021.

[32] A. Developers. Android documentation, getEntry. https://developer.android.com/reference/java/util/zip/ZipFile#getEntry(java.lang.String), 2021.

[33] A. Developers. Android documentation, getInstalledApplications. https://developer.android.com/reference/android/content/pm/PackageManager#getInstalledApplications(int), 2021.

[34] A. Developers. Android documentation, platform architecture. https://developer.android.com/guide/topics/manifest/intent-filter-element, 2021.

[35] dflower. Google play python api. https://github.com/dflower/google-play-crawler, 2014.

[36] A. Documentation. BroadcastReceiver - setResultData. https://developer.android.com/reference/android/content/BroadcastReceiver#setResultData(java.lang.String), 2021.

[37] A. Documentation. start. https://developer.android.com/reference/android/media/MediaPlayer#start(), 2021.

[38] S. Duman, K. Kalkan-Cakmakci, M. Egele, W. Robertson, and E. Kirda. EmailProfiler: Spearphishing filtering with header and stylometric features of emails. In *Proceedings of the IEEE Annual Computer Software and Applications Conference*, pages 408–416, 2016.

[39] S. Egelman, L. F. Cranor, and J. Hong. You've been warned: an empirical study of the effectiveness of web browser phishing warnings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1065–1074, 2008.

[40] P. Eyes. Hi there, we are phishing eyes. https://www.phishingeyes.com/, 2021.

[41] A. P. Felt, S. Egelman, and D. Wagner. I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*, pages 33–44, 2012.

[42] 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 ACM Symposium on Usable Privacy and Security*, pages 1–14, 2012.

[43] E. Fink, M. Sharifi, and J. G. Carbonell. Application of machine learning and crowdsourcing to detection of cybersecurity threats. In *Proceedings of the US Department of Homeland Security Science Conference*, 2011.

[44] E. Fletcher. That's not your neighbor calling. https://www.consumer.ftc.gov/blog/2018/01/thatsnot-your-neighbor-calling, 2018.

[45] E. Fletcher. Government imposter scams top the list of reported frauds. https://www.ftc.gov/news-events/blogs/data-spotlight/2019/07/government-imposter-scams-top-list-reported-frauds, 2019.

[46] FSI. Financial security institute. https://www.fsec.or.kr/fsec/index.do, 2015.

[47] G. Git. WindownManagerImpl.java. https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/view/WindowManagerImpl.java, 2021.

[48] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang. Riskranker: scalable and accurate zero-day android malware detection. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services*, pages 281–294, 2012.

[49] S. E. Griffin and C. C. Rackley. Vishing. In *Proceedings of the Annual Conference on Information Security Curriculum Development*, pages 33–35, 2008.

[50] P. Gupta, B. Srinivasan, V. Balasubramaniyan, and M. Ahamad. Phoneypot: Data-driven understanding of telephony threats. In *Proceedings of the Network and Distributed System Security Symposium*, 2015.

[51] X. Han, N. Kheir, and D. Balzarotti. PhishEye: Live monitoring of sandboxed phishing kits. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1402–1413, 2016.

[52] HanKyung. If it takes once, get out 100 million... fear of the voice phishing app. https://www.hankyung.com/it/article/2020101200371, 2020.

[53] D. Harley, M. Grooten, S. Burn, and C. Johnston. My pc has 32,539 errors: how telephone support scams really work. *Virus Bulletin*, 2012.

[54] G. Ho, A. Cidon, L. Gavish, M. Schweighauser, V. Paxson, S. Savage, G. M. Voelker, and D. Wagner. Detecting and characterizing lateral phishing at scale. In *Proceedings of the USENIX Security Symposium*, pages 1273–1290, 2019.

[55] G. Ho, A. Sharma, M. Javed, V. Paxson, and D. Wagner. Detecting credential spearphishing in enterprise settings. In *Proceedings of the USENIX Security Symposium*, pages 469–485, 2017.

[56] C. Huang, S. Hao, L. Invernizzi, J. Liu, Y. Fang, C. Kruegel, and G. Vigna. Gossip: Automatically identifying malicious domains from mailing list discussions. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security*, pages 494–505, 2017.

[57] T. W. in Asia. Overseas chinese in japan warned on phone scams demanding bank transfers. https://www.scmp.com/week-asia/lifestyle-culture/article/3116336/overseas-chinese-japan-warned-phone-scams-demanding, 2021.

[58] L. Invernizzi, K. Thomas, A. Kapravelos, O. Comanescu, J.-M. Picod, and E. Bursztein. Cloak of visibility: Detecting when machines browse a different web. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 743–758, 2016.

[59] iOS. Apple developers - callkit. https://developer.apple.com/documentation/callkit, 2022.

[60] iOS. Apple developers - creating and combining views. https://developer.apple.com/tutorials/swiftui/creating-and-combining-views, 2022.

[61] iOS. Apple developers - developer forums. https://developer.apple.com/forums/thread/106125, 2022.

[62] iOS. Apple developers - enabling background audio. https://developer.apple.com/documentation/avfoundation/media_playback_and_selection/creating_a_basic_video_player_ios_and_tvos/enabling_background_audio, 2022.

[63] iOS. Apple developers - overlays. https://developer.apple.com/documentation/mapkitjs/mapkit/overlays, 2022.

[64] iOS. Apple developers - xcode. https://developer.apple.com/kr/xcode/, 2022.

[65] S. Jain, J. Lindqvist, et al. Should i protect you? understanding developers' behavior to privacy-preserving apis. In *Proceedings of the USENIX Workshop on Usable Security*, 2014.

[66] N. Jiang, Y. Jin, A. Skudlark, W.-L. Hsu, G. Jacobson, S. Prakasam, and Z.-L. Zhang. Isolating and analyzing fraud activities in a large cellular network via voice call graph analysis. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services*, pages 253–266, 2012.

[67] JoonGang. Give me my information and my caution disappears... 'intercepted app'. https://www.joongang.co.kr/article/25031605, 2021.

[68] JoonGang. "managing only the phone, earning 4 million won a month"... the trap of 'part-time honey' targeting housewives and students. https://www.joongang.co.kr/article/25031604, 2021.

[69] K. B. S. (KBS). 100 million per hour, 700 billion damage... vocie phishing is still active even in the face of covid. https://news.kbs.co.kr/news/view.do?ncd=5226186, 2021.

[70] P. G. Kelley, S. Consolvo, L. F. Cranor, J. Jung, N. Sadeh, and D. Wetherall. A conundrum of permissions: installing applications on an android smartphone. In *International conference on financial cryptography and data security*, pages 68–79, 2012.

[71] K. Ki-jeong. Police capture 93 members of a vishing ring that led to the death of a 20-something. https://english.khan.co.kr/khan_art_view.html?artid=202011051706177&code=710100, 2020.

[72] J. Kim, T. Gong, K. Han, J. Kim, J. Ko, and S.-J. Lee. Messaging beyond texts with real-time image suggestions. In *Proceedings of the International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–12, 2020.

[73] J. Kim, T. Gong, B. Kim, J. Park, W. Kim, E. Huang, K. Han, J. Kim, J. Ko, and S.-J. Lee. No more one liners: Bringing context into emoji recommendations. *ACM Transactions on Social Computing*, 3(2):1–25, 2020.

[74] K. Krol and S. Preibusch. Control versus effort in privacy warnings for webforms. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, pages 13–23, 2016.

[75] D. Lain, K. Kostiainen, and S. Capkun. Phishing in organizations: Findings from a large-scale and long-term study. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2022.

[76] L. Li, T. F. Bissyandé, D. Octeau, and J. Klein. Reflection-aware static analysis of android apps. In *Proceedings of the International Conference on Automated Software Engineering*, pages 756–761, 2016.

[77] G. Liu, G. Xiang, B. A. Pendleton, J. I. Hong, and W. Liu. Evaluating the wisdom of crowds in assessing phishing websites. In *Proceedings of the ACM Symposium on Usable Privacy and Security*, pages 1–13, 2011.

[78] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious urls: an application of large-scale online learning. In *Proceedings of the International Conference on Machine Learning*, pages 681–688, 2009.

[79] F. Maggi. Are the con artists back? a preliminary analysis of modern phone frauds. In *Proceedings of the IEEE International Conference on Computer and Information Technology*, pages 824–831, 2010.

[80] F. Maggi, A. Sisto, and S. Zanero. A social-engineering-centric data collection initiative to study phishing. In *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, pages 107–108, 2011.

[81] A. Marzuoli, H. A. Kingravi, D. Dewey, A. Dallas, T. Calhoun, T. Nelms, and R. Pienta. Call me: Gathering threat intelligence on telephony scams to detect fraud. *Black Hat*, 2016.

[82] N. Miramirkhani, O. Starov, and N. Nikiforakis. Dial one for scam: A large-scale analysis of technical support scams. In *Proceedings of the Network and Distributed System Security Symposium*, 2017.

[83] T. Moore and R. Clayton. Evaluating the wisdom of crowds in assessing phishing websites. In *Proceedings of the Internationalharuta:globecom:2016 Conference on Financial Cryptography and Data Security*, pages 16–30, 2008.

[84] D. S. news. Beware of the smishing and phishing text pretending to be a 'health checkup' guide. https://www.idsn.co.kr/news/articleView.html?idxno=32189, 2020.

[85] A. Oest, Y. Safaei, A. Doupé, G.-J. Ahn, B. Wardman, and K. Tyers. PhishFarm: A scalable framework for measuring the effectiveness of evasion techniques against browser phishing blacklists. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 1344–1361, 2019.

[86] A. Oest, Y. Safaei, P. Zhang, B. Wardman, K. Tyers, Y. Shoshitaishvili, and A. Doupé. Phishtime: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists. In *Proceedings of the USENIX Security Symposium*, pages 379–396, 2020.

[87] A. Oest, P. Zhang, B. Wardman, E. Nunes, J. Burgis, A. Zand, K. Thomas, A. Doupé, and G.-J. Ahn. Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale. In *Proceedings of the USENIX Security Symposium*, pages 361–377, 2020.

[88] PhishTank. PhishTank: Join the fight against phishing. https://www.phishtank.com/, 2016.

[89] A. Possemato, A. Lanzi, S. P. H. Chung, W. Lee, and Y. Fratantonio. Clickshield: Are you hiding something? towards eradicating clickjacking on android. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1120–1136, 2018.

[90] V. Rastogi, Y. Chen, and X. Jiang. Droidchameleon: evaluating android anti-malware against transformation attacks. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security*, pages 329–334, 2013.

[91] M. Sahin, A. Francillon, P. Gupta, and M. Ahamad. Sok: Fraud in telephony networks. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 235–250, 2017.

[92] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli. Madam: Effective and efficient behavior-based android malware detection and prevention. *IEEE Transactions on Dependable and Secure Computing*, 15(1):83–97, 2016.

[93] A. Scam. Smart phishing protect, anti scam. https://www.antiscam.co.kr/, 2021.

[94] A. C. Search. NewOutgoingCallIntentBroadcaster.broadcastIntent. https://cs.android.com/android/platform/superproject/+/master:packages/services/Telecomm/src/com/android/server/telecom/NewOutgoingCallIntentBroadcaster.java, 2021.

[95] A. C. Search. PermissionManagerService. https://cs.android.com/android/platform/superproject/+/master:frameworks/base/services/core/java/com/android/server/pm/permission/PermissionManagerService.java, 2022.

[96] F. S. Service. 2018 annual report. https://english.fss.or.kr/download.bbs?bbsid=1289364303986&fidx=1581467525296, 2019.

[97] F. S. Service. 2019 annual report. https://english.fss.or.kr/download.bbs?bbsid=1289364303986&fidx=1609131857837, 2020.

[98] F. Shen, J. Del Vecchio, A. Mohaisen, S. Y. Ko, and L. Ziarek. Android malware detection using complex-flows. *IEEE Transactions on Mobile Computing*, 18(6):1231–1245, 2018.

[99] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, and C. Zhang. An empirical analysis of phishing blacklists. 2009.

[100] L. Shi, J. Fu, Z. Guo, and J. Ming. " jekyll and hyde" is risky: Shared-everything threat mitigation in dual-instance apps. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services*, pages 222–235, 2019.

[101] skylot. JADX - dex to java decompiler. https://github.com/skylot/jadx, 2021.

[102] A. Spy. Cyber bureau, anti spy app. https://cyberbureau.police.go.kr/mobile/sub/sub_03_j.jsp, 2021.

[103] M. Sun, X. Li, J. C. Lui, R. T. Ma, and Z. Liang. Monet: a user-oriented behavior-based malware variants detection system for android. *IEEE Transactions on Information Forensics and Security*, 12(5):1103–1112, 2016.

[104] K. Tam, A. Feizollah, N. B. Anuar, R. Salleh, and L. Cavallaro. The evolution of android malware and android analysis techniques. *ACM Computing Surveys*, 49(4):1–41, 2017.

[105] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song. Design and evaluation of a real-time url spam filtering service. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 447–462, 2011.

[106] K. Thomas, F. Li, A. Zand, J. Barrett, J. Ranieri, L. Invernizzi, Y. Markov, O. Comanescu, V. Eranti, A. Moscicki, D. Margolis, V. Paxson, and E. Bursztein. Data breaches, phishing, or malware? understanding the risks of stolen credentials. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1421–1434, 2017.

[107] U. TODAY. Robocalls and scam calls persist during pandemic, so americans have stopped answering the phone. https://www.usatoday.com/story/tech/2021/02/12/robocalls-scammers-fraud-phone-calls-increase-fcc-ftc-efforts/6706727002/, 2021.

[108] H. Tu, A. Doupé, Z. Zhao, and G.-J. Ahn. SoK: Everyone hates robocalls: A survey of techniques against telephone spam. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 320–338, 2016.

[109] H. Tu, A. Doupé, Z. Zhao, and G.-J. Ahn. Users really do answer telephone scams. In *Proceedings of the USENIX Security Symposium*, pages 1327–1340, 2019.

[110] Whowho. Better when together, whowho. https://www.whowhocorp.com/ko/, 2021.

[111] Y. Yan, Z. Li, Q. A. Chen, C. Wilson, T. Xu, E. Zhai, Y. Li, and Y. Liu. Understanding and detecting overlay-based android malware at market scales. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services*, pages 168–179, 2019.

[112] E. O. Yeboah-Boateng and P. M. Amanor. Phishing, smishing & vishing: an assessment of threats against mobile devices. *Journal of Emerging Trends in Computing and Information Sciences*, 5(4):297–307, 2014.

[113] G. Zhang and S. Fischer-Hübner. Detecting near-duplicate spits in voice mailboxes using hashes. In *Proceedings of the International Conference on Information Security*, pages 152–167, 2011.

[114] M. Zhao, B. An, and C. Kiekintveld. Optimizing personalized email filtering thresholds to mitigate sequential spear phishing attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 658–664, 2016.

[115] Y. Zhou and X. Jiang. Dissecting android malware: Characterization and evolution. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 95–109. IEEE, 2012.