Analyzing Ground-Truth Data of Mobile Gambling Scams

Geng Hong*, Zhemin Yang*, Sen Yang*, Xiaojing Liao†, Xiaolin Du*, Min Yang*, Haixin Duan‡
*Fudan University, †Indiana University Bloomington, ‡Tsinghua University
*{ghong17, yangzhemin, syang15, xldu20, m_yang}@fudan.edu.cn, †xliao@indiana.edu, ‡duanhx@tsinghua.edu.cn

***Abstract*—With the growth of mobile computing techniques, mobile gambling scams have seen a rampant increase in the recent past. In mobile gambling scams, miscreants deliver scamming messages via mobile instant messaging, host scam gambling platforms on mobile apps, and adopt mobile payment channels. To date, there is little quantitative knowledge about how this trending cybercrime operates, despite causing *daily* fraud losses estimated at more than $522,262 USD.**

This paper presents the first empirical study based on ground-truth data of mobile gambling scams, associated with 1,461 scam incident reports and 1,487 gambling scam apps, spanning from January 1, 2020 to December 31, 2020. The qualitative and quantitative analysis of this ground-truth data allows us to characterize the operational pipeline and full fraud kill chain of mobile gambling scams. In particular, we study the social engineering tricks used by scammers and reveal their effectiveness. Our work provides a systematic analysis of 1,068 confirmed Android and 419 iOS scam apps, including their development frameworks, declared permissions, compatibility, and backend network infrastructure. Perhaps surprisingly, our study unveils that public online app generators have been abused to develop gambling scam apps. Our analysis reveals several payment channels (ab)used by gambling scam app and uncovers a new type of money mule-based payment channel with the average daily gambling deposit of $400,000 USD. Our findings enable a better understanding of the mobile gambling scam ecosystem, and suggest potential avenues to disrupt these scam activities.

I. Introduction

Gambling scams are a prevalent form of online fraud in which a scammer claims to offer an advantage in gambling activities (e.g., casino, poker, sports betting) to trick players out of money. Gambling scams have already caused huge financial losses to individuals and businesses. As reported in [1], the Canadian Lottery scam netted more than $5 billion from U.S. victims and was making around £500,000 a month in the U.K. In China, online gambling scams are with the highest per-capita financial loss among all kinds of scams [2].

With the fast growth and popularity of mobile markets today, gambling scams are extending their reach to mobile computing. In mobile gambling scams, miscreants deliver scamming messages via instant messaging (IM) and host scam gambling platforms on mobile apps. Those apps prevent victims from cashing out their winnings [3]. Compared with web-based scams recently studied [4], [5], [6], [7], mobile gambling scams are characterized by using mobile-side social engineering attacks (e.g., scamming messages via instant messaging apps) to distribute gambling scam apps, taking advantage of mobile payment channels, and utilizing mobile computing techniques to bypass traditional web-based fraud detection [8], [9], [10]. Prior works on web-based scams mainly focus on detecting scam gateway sites [5], identifying the scam campaigns [4], or scam websites' lifetime [7]. A key limitation of these works was that they were based on external measurements with limited visibility into the kill chain of gambling scams, e.g., how does a scammer launch social engineering attacks to lure victims; what is the modus operandi of the scam platform; and what is the financial profitability of such scam. So far, little has been done to analyze such mobile scam apps, not to mention any effort to understand the underground ecosystem behind them.

This paper presents the first systematic study of mobile gambling scams based on a ground-truth dataset. The data pertains to 1,461 scam incident reports associated with 1,068 Android apps and 419 iOS apps, spanning from January 1, 2020 to December 31, 2020, provided by an Anonymous Authority. Given the ground-truth data, we conduct a qualitative analysis of the incident reports and develop a suite of measurement and dedicated reverse-engineering tools which enable us to perform a large-scale study to unearth a mobile-based kill chain of gambling scams. More specifically, we aim to answer the following questions: What social engineering techniques are used by miscreants in mobile gambling scams? How do they operate gambling scam apps? How do they provide evasive payment channels under rigorous financial censorship?

Looking into the ecosystem of mobile gambling scams, we are surprised to find that this new threat is trending with a great impact on today's mobile ecosystem. More specifically, through a qualitative analysis on these scam incident reports, we study social engineering techniques used by scammers to trick victims into participating in gambling scams. We observe that scammers established connections with victims via mobile apps including IM applications (55.4%) and social network apps (28.1%), such as online Q&A, job hunting, or dating apps, and faked their profiles based on the information of the victim's social networks. Interestingly, we observed that victims are proactively requesting scam gambling apps after being baited. Compared with cases where scammers persuade victims to download scam apps, this yields significantly higher scam losses ($47K *vs.* $30K on average per case). Through the investigation of 1,487 gambling scam apps, we observed two public online app generators, i.e., DCloud and APICloud, being abused to develop scam apps. During gambling scam app development, the certificates provided by these app generators have been used to sign 137 scam apps. Importantly, we discover that gambling scam apps tend to declare lower requirements for minimum OS versions being supported (4.4

for Android gambling scam apps vs. 5.0 for reputable apps in Google Play, see Section VI-D) to ensure compatibility between versions and, thus, can cover more victims, even though their develop SDK versions are not outdated.

Furthermore, we analyze the payment channels employed by mobile scam apps and discover eight payment channels used by scammers, including traditional bank payment channels (e.g., direct debits), online payment services (e.g., Alipay and WeChat Pay), cryptocurrency (e.g., Tether and CGPay) and money mule-based payments (e.g., Alipay and Idlefish). We also bring to light about new techniques for anonymous transactions, i.e., Idlefish Money Mule. To support this payment channel, scammers recruit money mules, who are sellers of a flea market app called the Idlefish, to transfer gambling deposits. Our study uncovers 110 Idlefish stores involved in money laundering, which we have reported to the platform. Our study further investigates the revenue under this payment channel by traversing 17,144 payment links, which show that the average daily revenue is up to $400,000 USD. We reported our findings to the affected parties, including Apple, HUAWEI, Xiaomi, Getui, DCloud and Idlefish, who are serious about these risks. Some of them expressed gratitude for our help via bounty programs.

Contributions. We summarize the contributions as follows:

- We conduct the first in-depth empirical study of mobile gambling scams based on ground-truth data. Our study investigates the kill chain of mobile gambling scams and the actors involved.
- We reveal social engineering techniques used by miscreants via a qualitative analysis on 1,461 mobile gambling scam incident reports.
- We characterize both Android and iOS gambling scam apps, including their development frameworks, declared permissions, compatibility, and backend network infrastructure. Our study reveals that public online app generators have been abused to develop gambling scam apps.
- We study the payment channels (ab)used by gambling scam app. We also uncover a new type of money mule-based payment channel and measure its revenue.

II. Background

As mentioned earlier, a gambling scam is a type of cybercrime fraud that tricks victims out of money in gambling activities, such as poker, casinos, and sports betting. Compared to traditional gambling scams, which simply build impossible-to-win systems, modern gambling scam activities tend to prevent victims from cashing out. For example, when the victims want to cash out their winnings, they may be told that their accounts are frozen or ridiculously over-charged [3]. Note that in our study, we focus on gambling scam activities. The legitimacy of gambling is out of the scope of this study.

Moving from the Internet to mobile devices [11], today, gambling scams increasingly happen through mobile apps deployed to victims' smartphones than victims browsing websites. Here, we introduce a typical mobile gambling scam example from a real-world scam incident report, provided by Anonymous Authority, to show how such scam activities operate and how each entity interacts with each other.

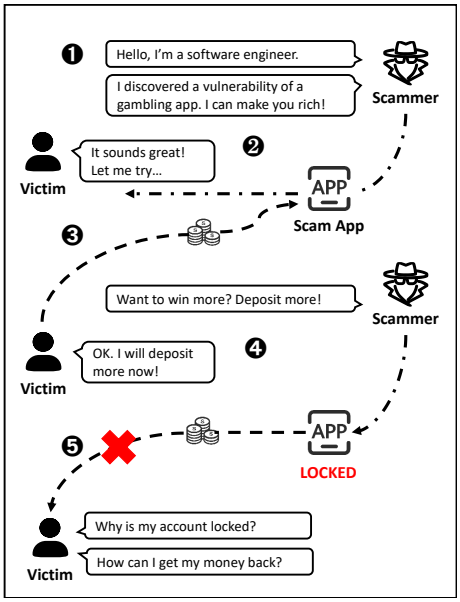

Figure 1. The operational pipeline of mobile gambling scam.

Table I
THE BREAKDOWN OF THE GROUND-TRUTH DATASET.

Type	#
Scam Incident Reports	1,461
Android Gambling Scam Apps	1,068
iOS Gambling Scam Apps	419

Figure 1 illustrates the operational pipeline of mobile gambling scams, which consists of four stages: connection establishment, app delivery, gambling deposit, and scamming. First, a scammer bootstraps the attack by establishing the connection with a victim (❶). To build and earn trust, a scammer usually creates a fake profile and contacts the targets through popular social media apps, e.g., dating apps or job hunting apps. The scammer then usually lures the victim to download a scam gambling app (❷) by offering a "too-good-to-be-true" bonus, which can exploit a vulnerability of a gambling app to earn money. Once the victim trusts the scammer and downloads the mobile gambling scam app, he will fund his account (❸) to start gambling via multiple types of payment channels (e.g., bank, third-party payment, cryptocurrency, money mule) embedded in the gambling apps. Meanwhile, the scammer will bait the victim to continually deposit money (❹) by offering more lucrative returns. The scammer then will lock the victim's account (❺), preventing the victim from withdrawing his/her gambling income. In this study, we focus on cases where gambling scam activities are *operated on mobile apps*. To the best of our knowledge, it is the first study of this kind.

III. Groundtruth Data

We have collaborated closely with Anonymous Authority throughout this research effort. During this time, we obtained

a comprehensive and detailed dataset of 1,461 scam incident reports, 1,068 gambling scam Android apps and 419 iOS apps, spanning from January 1, 2020, to December 31, 2020. Note that, in this dataset, either scam victims lived in China, or scam activities happened in China. We summarize the high-level statistics of our dataset in Table I. Among them, 456 scam apps have been taken down, while others remain active at the time of submission due to investigative interest by Anonymous Authority.

For each online gambling scam app we investigate, we have some or all of the following information, which we analyze in more detail in Section V and VI, as elaborated below.

• **Scam app samples**. This dataset consists of 1,487 online gambling scam app samples (1,068 Android samples and 419 iOS samples). The file size of samples ranges from 23 KB to 93.9MB. The smallest one only contains five Java classes and implements most of its functions via WebView [12]. Meanwhile, the largest one consists of multiple SDKs (e.g., push service SDK, third-party payment SDK) to support various functionalities. As an instance, we observe that one scam app integrates a live show SDK for gambling activity live streaming to lure victims.

• **Scam incident reports**. The scam incident reports consist of detailed information of the mobile gambling scam incident from victims, including the time of the incident, amount of loss, and incident summary (including how scammers establish a connection with victims, how scammers build and earn the trust of victims, and how scammers deliver scam apps to victims, etc.). The length of incident summaries ranges from 171 words to 1,931 words, with an average of 606.66 words and an standard deviation of 190.36 words.

IV. Ethics

Our data is similar to that used in prior cybercrime studies [13], [14], [15]. It originates from law enforcement procedures to seize and record scam activities. Employing such data might raise ethical issues. Therefore, we carefully manage our research activities to ensure they stay within legal and ethical boundaries.

This research has been approved by our institution's IRB. The approval process is similar to the exempt review in the U.S. because this study is considered as "minimal risk" when we consulted with the IRB staffs. Note that our research only uses the previously collected data (collected by collaborated Anonymous Authority), and any possible Personal Identifiable Information (PII) in scam incident reports was removed before the reports were shared with us.

Apart from acquiring the approval of our institutions, we also comply with the principles identified in the Menlo Report [16]. More specifically, to evaluate "balancing the risks and benefits" based on the Menlo Report, we carefully designed our experiment and ensured that our research did not contribute any financial profits to the criminal. For example, our deposit testing (Section VII) neither sent any money to benefit the criminals nor attempted to communicate with scammers. In our research, we repeatedly placed deposit requests and then withdrew those requests after collecting transaction information (e.g., cryptocurrency addresses). Over the course of this paper, we did not interact with either the victims or the scammers.

In addition, this research aims to profile mobile gambling scam activities to the research community, enable law enforcement and policymakers to better understand and provide insight into these new types of scam activities. We are sure that the benefits to the general public far exceed any knowledge that the criminals might obtain from the high-level details presented in our paper.

V. Analysis of Social Engineering Tricks

To investigate social engineering attacks [17] used in mobile gambling scams, we conduct a qualitative study of scam incident reports. Each report is associated with an independent victim. [1] Their ages range from 14 to 79 (2.1% of $\leq$20, 70.6% of 21-40, 22.9% of 41-56, 4.4% of $\geq$57), and their education levels vary from elementary school to doctoral degree (2.8% of elementary school, 21.5% of middle school, 23.3% of high school, 52.4% of college). In particular, this study seeks to answer the following research questions:

RQ1 How does a scammer establish connections with a victim to build trust?

RQ2 How does a scammer deliver a scam app to a victim?

RQ3 How does a scammer lure victims to deposit money continually?

RQ4 What is the logic of mobile gambling scams?

A. Data Coding

We used the qualitative open-coding technique [18] on 1,461 scam incident reports to study the aforementioned questions. Specifically, two cybersecurity professionals independently reviewed a random set of 75 incident reports (about 5% of the total) and resolved inconsistencies while generating initial codebooks. They then independently coded the remaining 1,311 reports and compared their coded results by Krippendorff's alpha coefficient, a widely used statistical measure of the agreement achieved when coding a set of units of analysis. Krippendorff's alpha of this study is 0.85, higher than the reliability threshold in social science [19]. Afterwards, they resolved all disagreements in coding phrases for each incident to generate the final codes. The ultimate codebook developed provides labels for the footprints of a scammer exploiting human psychology to establish a connection, deliver an app, and lure a victim to deposit money continually. Table II shows the codebook. In total, it took two human laborers around two weeks to complete the procedure. It is worth noting that the coding book has a limited coverage due to the vantage point of our study. Some incident reports do not mention the corresponding social engineering tricks.

[1] We use incident report IDs to refer to anonymized victims.

Table II
NUMBER OF INCIDENT REPORTS FOR EACH SOCIAL ENGINEERING TRICKS.

Type	Sub-type	Method	# (%)	Our Work	Previous Works
Connection Establishment	Role Play	Professional (e.g., gambling tutor)	331 (22.7)	✔	[20], [6]
		Intimate relationship (e.g., boy/girl friend)	204 (14.0)	✔	[21], [22], [9]
		Authority (e.g., law enforcement officers)	10 (0.7)	✔	[20], [23]
		Others (e.g., customers, cyber pals)	22 (1.5)	✔	–
	Total	-	567 (38.9)*		
App Delivery	Scammer Driven	Invite victims to participate gambling	748 (51.2)	✔	–
		Ask victims to gamble as agents	81 (5.5)	✔	–
	Victim Driven	Share winning experience in a chat channel	457 (31.3)	✔	–
		Show off profit gains on social media	145 (9.9)	✔	–
	Total	-	1431 (97.9)*		
Deposits	Incentive	Require victims winning reach a threshold to payout	405 (27.7)	✔	–
		Require additional margin to payout	131 (9.0)	✔	–
		Pay tax before payout	55 (3.8)	✔	[24]
		Require deposit reach a threshold to get bonus	55 (3.8)	✔	[24]
	Pressure	Freeze victims' account	64 (4.4)	✔	[24]
		Blame victims for operating illegally	60 (4.1)	✔	
		Blame victims for providing invalid bank card	36 (2.5)	✔	–
	Total	-	806 (55.2)*		
Scamming Logic	Disable App Functionality	Victims cannot withdraw balance	1154 (79.0)	✔	–
		Victims cannot login apps	246 (16.8)	✔	–
	Modify Account	Balance wiped out	22 (1.5)	✔	–
	Total	–	1422 (97.3)*		

* The percentage of *Total* is less than 100% since some incident reports do not mention corresponding social engineering tricks.

Table III
VICTIM RECRUITMENT CHANNEL.

Category	Sub-Category	# (%)
Traditional	Telephone	136 (9.3)
	SMS	13 (0.9)
	Email	2 (0.1)
	Total	151 (10.3)
Instant Message	Wechat	682 (46.7)
	QQ	120 (8.2)
	Others (e.g., WhatsApp)	8 (0.5)
	Total	810 (55.4)
Online Social Network	Dating	253 (17.3)
	Short-video sharing	80 (5.5)
	Online Q&A	21 (1.4)
	Local forum	16 (1.1)
	Job hunting	13 (0.9)
	Others	28 (1.9)
	Total	411 (28.1)
No Mention	-	89 (6.1)
Total	-	1461 (100.0)

B. Connection Establishment

Firstly, we analyze how a scammer bootstraps the attack by establishing a connection with the victim, coded as *Connection Establishment*, as shown in Table II. We code a strategy as *Role Play* when a scammer plays a role to establish a connection and build trust. For instance, ID-1380 described how a scammer built a connection with him:

> *" I published the house renting information on a local information sharing forum, and then a man contacted me to ask if the house was rented out. I replied that the house had not been rented. After that, he told me his business focused on furniture trading and frequently chatted with me. I put down my guard gradually. Until someday, he recommended me a gambling app... "*

Further, the roles played by scammers are organized into three main methods based on our codebook: *Professional*, *Intimate relationship* and *Authority*. When coding as *Professional* (22.7%), scammers act as the technical staff or gambling tutor who can hack into the gambling system or master gambling secrets to help victims profit. The code of *Intimate relationship* (14.0%) describes that scammers establish romantic or close relationships with victims (e.g., boy/girl friend) to build trust. Apart from that, the code *Authority* (0.7%) means that the scammer chooses a role with a higher social status and prestige, e.g., law enforcement officers, to contact victims.

As shown in Table III, in our study, we observe that scammers seek victims on online social networks (OSN), such as online Q&A, short-video sharing, job-hunting, and dating platforms, and then set up role play based on the information a victim left on the OSN. This approach is different from traditional scams [25], [26], which usually make telephone calls or send emails to establish connections. *Scammers also tend to set up role-play with higher social statuses, professional techniques, or intimate relationships when establishing connections with victims.* For instance, ID-627 trusted the scammer because the scammer showed his "professional" in information technology:

> *" He (the scammer) is the leader of a software company. One day, he told me that the system of XX[anonymized] has a vulnerability, and he was leading a technical team to solve that issue. Two days later, he said that while fixing*

the issue, he found a vulnerability of a gambling app that can be exploited to earn money... "

We also observe many cases (14.0%) where scammers strike up a relationship with victims to build their trust before delivering gambling apps. For example, as ID-844 described:

" I happily chatted with him (the scammer), and soon he became my boyfriend. He informed me that there was a quick way to make money in mid-May... "

Finding I: *Scammers tend to palm themselves off as holding roles with high social statuses, professional techniques, or intimate relationships when establishing connections with victims.*

Finding II: *Scammers prefer to seek victims and establish connections via online social networks (OSN) (e.g., online Q&A, job-hunting).*

C. App Delivery

A critical step in gambling scam activities is delivering apps to victims. To achieve this goal, scammers leverage several psychological strategies, as shown in Table II. We code a strategy as *Scammer Driven* (56.7%) when scammers persuade the victims to download gambling apps. The code of *Victim Driven* (41.2%) means that a victim proactively requests a gambling app after being baited, e.g., scammers continuously discuss and show gambling profit gains in group channels or posting their gambling income on social media. As an instance, ID-192 said:

" He (the scammer) invited me into a group channel. In that channel, I observed how the gambling tutor taught us gambling, how group members bet and won. They claimed they all made a successful payout. Afterward, the scammer said that winning in a new lottery app is very easy. I witnessed that the group members expressed their intention to participate. So I could not wait anymore and proactively contacted the scammer to download the gambling scam app. "

Interestingly, when comparing the scam losses between the cases coded as *Scammer Driven* and *Victim Driven*, we found that scam losses of *Victim Driven* are significantly higher than *Scammer Driven*, with the average scam losses of *Scammer Driven* of $30,648.8 (the median of $11,938.0) and the average of *Victim Driven* of $47,302.7 (the median of $20,045.9). This difference indicates that social engineering tricks, which lure victims to request scam gambling apps proactively, are more effective at tricking victims out of money.

Finding III: *Social engineering tricks, which lure victims to request scam gambling apps proactively, are more effective to trick victims out of money.*

D. Deposits

After delivering scam gambling apps, scammers start to enforce victims to deposit money continually (coded as *Enforce Deposit*, as shown in Table II). This step can be categorized into two sub-types based on our codebook: the "Incentive" and the "Pressure". In *"Incentive" strategy* (44.2%), a scammer will show incentives for continually depositing money, such as winning bonuses. Meanwhile, 11.0% of scammers put psychological pressure on the victims if they do not make deposits, coded as *"Pressure" strategy*. For instance, ID-544 depicted that s/he was blamed for providing an incorrect payback bank card number:

" ...the customer service said that my bank card number was incorrect...Only with one more deposit could I change the card number. "

Although scammers prefer to use incentives to lure victims (44.2% *vs.* 11.0%), we found that these two strategies pose similar success rates. Here, we define the success rate as the number of victims who made deposits following social engineering tricks (instead of cheat awareness) over the total cases with the code of the "Incentive" or the "Pressure". We observe that these two strategies have similar success rates: almost half of the victims were encouraged to make deposits through social engineering tricks. For *"Incentive" strategy*, its success rate is 47.1%. The *"Pressure" strategy* features a 48.1% success rate.

Finding IV: *To ensure victims make deposits, scammers tend to leverage incentive strategies instead of imposing psychological pressures, even though similar success rates are achieved with either strategy.*

E. Scamming Logic

All incident reports in our study are associated with the cash out scam, i.e., scammers prevent victims from cashing out gambling incomes. As shown in Table II, we categorize their scamming logic into two sub-types: *Disable App Functionality* and *Modify Account*. Most scammers deploy their scamming logic by disabling scam apps' functionalities, which means that victims' accounts are frozen, or the cash out function is disabled. Meanwhile, 22 scammers directly wiped out the victims' balance, which was coded as *Modify Account*.

In some cases, we observe scammers make up excuses to delay being aware. The ID-138 expressed his experience:

" ... On someday, the gambling tutor sent me a message: 'the company's manager has been arrested for illegally opening an enterprise account, and his communication account has been taken down. Please do not contact the manager or tutor in the short term. Your gambling account balance will be refunded in one month. Don't contact us, or you will be at your own risk.' As a result, at the end of the month, I found that I didn't receive any payout, and couldn't open the gambling app either. Only then did I realize that I was cheated and turned to the police. "

Finding V: *Unlike traditional online gambling frauds, which trick victims by charging additional fees or selling bet cheating software, the scamming logic of the online gambling scams in our study is mainly to disable scam apps' functionalities.*

F. Discussion

Some of the social engineering tricks used to establish connections have also been discussed in other fraud research, e.g., connect victims as an authority or intimate in the "419 scams" [20] and the online dating fraud [21], as shown in

Table II. In our research, we present the first systematic study based on a unique and valuable scam incident report dataset, which are associated with a larger size of victims and a longer time-span, compared with previous work. Our study reports that mobile gambling scammers contact victims for more diverse reasons (e.g., house renting and job hunting) through various approaches (e.g., Local forums or Online Q&A platforms). In addition, for the first time, our research revealed the social engineering tricks used to deliver scam gambling apps, enforce deposits, and show scamming logic. Note that some high-level psychological tricks, such as showing incentive or pressure, have been mentioned in previous fraud research [22], [23], but those studies did not focus on gambling-specific tricks.

VI. Gambling Scam App Analysis

In this section, we study app behaviors of scam gambling apps, including their development frameworks, permissions, the compatibility, and the network infrastructure.

Genuine gambling apps. We collect a set of genuine gambling apps for behavior comparison. The genuine gambling apps refer to the reputable and legitimate gambling apps. Specifically, we first fetch popular gambling websites from SimilarWeb [27], then filter top-ranked gambling websites based on Tranco Top-1M Ranking List [28]. Afterwards, we validate these websites and associate apps through certificates (e.g., removing any certificate with nonsense string) and VirusTotal labels. To further validate the legitimacy of apps, we also check whether the app is published on Google Play or App Store. We confirm 156 websites and download apps associated with these sites. Finally, we successfully collected 182 genuine Android gambling apps and 134 genuine iOS gambling apps through the above methods.

A. Public and Non-public Apps

We identify how many of the scam apps in our study are available to the public. This analysis took place on January 4, 2021, and we used all 1,487 scam apps for app matching. First, as for iOS apps, we extract each scam app's name and unique bundle identifier by inspecting the `CFBundleDisplayName` and `CFBundleIdentifier` filed in the *Info.plist* file. Then, we search app names through utilizing the iTunes Search API [29]. For each app in the search results, we extract the `bundleId` from its meta-information for app matching. Our result shows that none of the iOS scam apps in our dataset are indexed in the App Store.

As for Android apps, we first leverage Androguard [30] to extract the package name from each APK file in our dataset. We then use the package name for app searching in *Google Play* [31], *Androzoo* [32], and other third-party app markets, such as *MyApp* [33], *Xiaomi App Store* [34], *iuuu9* [35], and *anxz* [36]. After retrieving apps in the search result, we use apktool [37] to find the signing key of each app for app matching. However, none of the package names matched successfully in *Google Play*, *Androzoo*, *MyApp*, or *Xiaomi App Store*; only five scam apps with package name exactly the same were found in *iuuu9* and *anxz*. Interestingly, those apps were updated with innocent-looking app descriptions and screenshots. When launching those apps, users will be redirected to a scam gateway to download scam apps.

Table IV
Examples of app development frameworks fingerprints.

Framework		Namespace / Header	File
React Native	Android	com.facebook.react.*	assets/index.android.bundle
	iOS	ReactNativeHelper.h	main.jsbundle
DCloud	Android	io.dcloud.*	io/dcloud/all.js
	iOS	-	PandoraApi.bundle/all.js
Cocos2dx	Android	org.cocos2dx.*	lib/*/libcocos2djs.so
	iOS	cocos2d.h	-
Unity3D	Android	com.unity3d.player.*	lib/*/libunity.so
	iOS	UnityFramework.h	*/mscorlib.dll-resources.dat
Cordova	Android	org.apache.cordova.*	assets/www/cordova.js
	iOS	CDV.h	www/cordova.js
APICloud	Android	com.uzmap.pkg.*	assets/uzmap/module.json
	iOS	UZModule.h	uz/module.json

Apps sideloading. We find that none of the gambling scam apps in our dataset reside in reputable App markets. However, sideloading apps on Android and iOS requires configuration changes in the respective system settings. We try our best to figure out how typical, non-technical victims manage to carry out a series of actions to sideload apps. On the one hand, by visiting scam app download links recorded in the reports, we learned that scammers typically provide step-by-step tutorials for sideloading apps on app download pages. Victims only need about 4-5 clicks to sideload an app following these tutorials. An example of the tutorial can be found in Appendix A. On the other hand, although the incident reports do not provide the details about whether or how scammers instruct victims to sideload scam apps, the Anonymous Authority has confirmed that some scammers did provide detailed technical support for doing so.

Note that scam apps leverage the Apple Developer Enterprise Program, originally designed to create and distribute proprietary enterprise iOS apps for internal uses, to distribute gambling scam apps on iOS [38], [39]. In this way, there is no need for victims to have a jailbroken phone to install iOS scam apps. We have reported our findings to the owners of those enterprise certificates, but we have not received any response yet. In addition, we had already reported these abused enterprise certificates to Apple, and they acknowledged our report.

Finding VI: *None of the gambling scam apps in our dataset can be found through reputable App markets. Instead, scammers usually launch attacks with the sideloaded app through step-by-step instructions and technical support.*

B. Development Framework Identification

In our study, we investigate the development framework used by scammers for generating scam apps. To fingerprint the development frameworks of scam apps, we first determine

Table V
APP DEVELOPMENT FRAMEWORKS.

Framework	Genuine Gambling		Gambling Scam	
	# Android App	# iOS App	# Android App	# iOS App
React Native [40]	8	4	70	17
DCloud [41]	0	0	59	7
Cocos2dx [42]	19	11	9	0
Unity3D [43]	24	27	8	0
Cordova [44]	19	5	7	0
APICloud [45]	0	0	4	0
Others	12	8	4	1

35 popular mobile app development frameworks (e.g., React Native, Cocos2dx, Cordova, see the full list in Table VII in Appendix B) and then extract unique features from them, such as namespace in Java and Kotlin, the Objective-C header file, and the dynamic link library (*.so*), image, or script. Table IV shows examples of fingerprints adopted in our study.

We identified eight development frameworks used by scam apps. The results are listed in Table V. Interestingly, we observe scammers abused public app generators (OAGs) [46], i.e., DCloud and APICloud (Row 2 and Row 6 of Table V), for the scam app generation. Particularly, 63 Android apps and seven iOS apps in our study were generated by DCloud and APICloud, which account for 4.7% of scam gambling apps in our study. Note that this percentage is slightly lower than that of OAG-generated *unwanted apps* as reported in [47] (4.7% *vs.* 12.1% [2]). Surprisingly, when investigating app certificates of OAG-generated scam apps, we also found that 137 scam apps were signed by Digital Heaven, the company of the DCloud framework. When generating apps, the DCloud framework will automatically sign OAG-generated apps using its own certificate. Even worse, DCloud's certificate has been abused by scammers to sign gambling scam apps. In our study, we observe that 71 scam apps are signed by DCloud but not generated by DCloud. We have reported this issue to DCloud.

Finding VII: *Scammers abuse Online App Generator (OAG) to automatically generate scam apps, as well as signing those apps.*

Scam app clustering. As apps with similar file structures and high code similarity suggest a high possibility of belonging to the same gambling scam campaign [9], we conduct a clustering analysis of gambling apps leveraging their file structure similarity and code similarity. For file similarity, given two APKs/IPAs, we calculate their Jaccard distance of resource files. More specifically, we first calculate the SHA256 hash of each resource file, then combine the hash values into a set for each APK/IPA. Afterwards, we compare two hash sets regarding their Jaccard distance. For code similarity, we calculate the similarity of dex files and so files, or executable files and dylib files, respectively, on Android and iOS. Then, we apply DBSCAN [48], a density-based cluster algorithm, on the similarity matrix to cluster the apps. Note that iOS certificates are generally abused, and thus, we do not utilize this feature to cluster iOS scam apps. We only merge Android gambling apps signed with the same developer signature, as they belong to the same gambling scam campaign. Moreover, if there are overlapping domains of apps' backend servers in any clusters, we also merge them as they share the same infrastructure.

To this end, 1,068 Android gambling apps and 419 iOS gambling apps were clustered into 134 and 11 groups, respectively. Note that for Android apps, 49 groups have their sizes larger than 1 while the number for iOS apps is six. Table VIII and Table IX in Appendix C show the top-5 clusters of Android and iOS gambling apps. We can find that most apps in the same cluster share the same prefix of package names and bundle IDs. For example, for the largest cluster of Android and iOS gambling apps, more than 81.4% of the Android apps and all iOS apps share the common package name prefix com.yibo.*. In addition, 86.8% of Android gambling scam apps in this cluster are issued by the same certificate with the Distinct Name of *yibo*.

Finding VIII: *The most popular scam app family of Android and iOS (in our dataset) are Yibo, which consists of 555 Android apps and 191 iOS apps, spreading over 733 backend domains.*

C. Permission Analysis

Both Android and iOS implement permission models to control apps' access to sensitive data and system resources. By default, apps are not allowed to perform any sensitive operations without corresponding permissions. We investigate the permissions declared and requested by scam apps.

Android scam apps. As for Android, we leverage the *get_permissions()* API of Androguard to extract and study the usage of permissions by scam apps. In this way, we successfully identify 636 unique permissions, including 119 AOSP–defined permissions and 517 custom permissions defined by the app vendor or other third-party services from Android scam apps. As for the custom permissions, we observe that 446 (86.3%) of them are used for push service, such as *"com.hgapp.bet365.permission.JPUSH_MESSAGE"* for JPush SDK, which helps the scammers keep users' "stickiness" by sending push messages, and 6.19% are permissions used for reading or writing settings (e.g., create a shortcut on home screen) defined by each vendor.

Especially, we pay attention to the usage of dangerous Android permissions [49] since they are closely related to user privacy. The scam apps request 4.6 dangerous AOSP permissions on average, while some apps even declared 15 dangerous permissions including *READ_CONTACTS* and *READ_SMS*. Figure 2 shows the distribution of dangerous AOSP permissions requested across gambling scams and genuine gambling apps. Unlike genuine gambling apps, scam apps integrated Instant Messaging SDKs to construct built-in IM channels, which provide a steady and uncensored communication approach to launch social engineering attacks on victims. Therefore, they request the corresponding permissions, such as

[2]Here we divided the sum of OAG unwanted packages (rank 2, 5, 10 of Table V = 40,693) by the sum of all unwanted packages (335,471) in [47] to get this result.

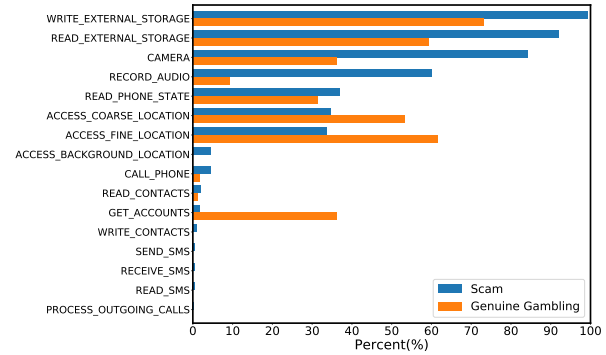

Figure 2. Dangerous permissions requested by Android scam apps and genuine gambling apps.

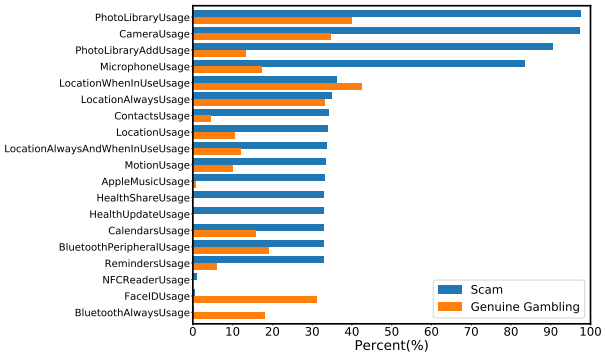

Figure 3. Permissions requested by iOS scam and genuine gambling apps.

RECORD_AUDIO and *READ_CONTACTS*, for this purpose. On the contrary, genuine gambling apps rarely request these permissions. As for *GET_ACCOUNTS*, most genuine apps request this permission for Google Mobile Services (GMS) SDK integration, while most gambling scam apps do not integrate such functionality.

iOS scam apps. As for iOS, to access users' private data, developers are required to include specific key-value pairs in their app's *Info.plist* file which indicates requesting permissions and corresponding purposes. To analyze permission usage of iOS apps, we first unzip each *.ipa* file to extract the *Info.plist* file in our dataset. We leverage plistlib [50] to parse it and investigate the declared permission and purpose.

Altogether, we identify 419 iOS apps declaring 19 unique permissions. The number of permissions declared varies from 0 to 17, with an average of 7.8 and a standard deviation of 5.9. Similar to Android scam apps, the most frequently used permissions are reading and writing PhotoLibrary, Camera, and Microphone. We also observe 138 (32.9%) apps declaring the same 16 permissions, with the required description string *"APP requires your consent to access your microphone, camera, etc."*. We find that such purpose strings are generally used by online tutorials [51], and developers are expected to replace them with their true purposes. However, it is obvious that the scam apps neither decrease the claims of permissions as needed nor state their intention of using these permissions.

Compared to gambling scam apps, we also find Android and iOS genuine gambling apps request location permissions more often than scam gambling apps, which may be because

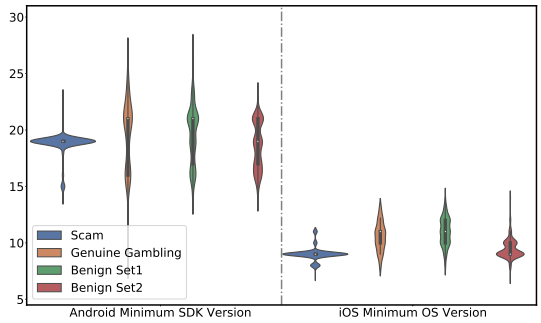

Figure 4. Development SDK version distribution. Benign Set1 and Set2 refer to apps in Google Play and MyApp for Android, App Store US and App Store China for iOS, respectively. Gambling scam apps tend to declare lower requirements for minimum OS versions being supported (4.4 for Android gambling scam apps vs. 5.0 for reputable apps in Google Play).

legal gambling usually has regional restrictions. In addition, we observe that the genuine gambling iOS apps usually request permissions *FaceIDUsage* for Face ID login, which is rarely used by scam gambling apps.

> **Finding IX**: *Compared with genuine gambling apps, gambling scam app request more dangerous permissions to support social engineering tricks (e.g., keep users' "stickiness" via sending push messages or constructing built-in IM channels).*

D. Compatibility Analysis

To inspect the compatibility of scam apps, we analyze the development SDK version of these apps. More specifically, we identify the `MinSdkVersion`, which specifies the minimum platform API level required for the app to run, and the `TargetSdkVersion`, which designates the platform API level that the app targets. As for Android apps, we leverage Androguard to extract the `minSdkVersion` and `targetSdkVersion` attributes from the *AndroidManifest.xml* file. Considering iOS apps, we first extract the *Info.plist* file and then inspect the `MinimumOSVersion` and `DTPlatformVersion` field from it, which are similar to `MinSdkVersion` and `TargetSdkVersion` in Android. To compare app compatibility between scam apps and genuine apps, we additionally crawl the top-500 apps from Google Play and a third-party App Store, MyApp [33], and the top-200 free apps from Apple App Store in the U.S. and China on December 10, 2020, to identify their `MinSdkVersion` and `TargetSdkVersion`.

Figure 4 illustrates the comparison results for Android and iOS apps. We find gambling scam apps tend to declare lower requirements for minimum OS versions to ensure compatibility across versions and, thus, can cover more victims, even though their develop SDK versions are not outdated. In particular, almost all scam apps declare their min SDK version to be 19 (Android version 4.4) or lower, only 48.8%, 48.6%, and 67.8% for genuine gambling apps, reputable apps in Google Play, and MyApp, respectively. What's more, we observe 80 scam apps that even support Android 4.0.4, which was released in 2012, with target SDK 28 and 29 (Android 9 and 10). Similar findings can be observed in iOS apps: most

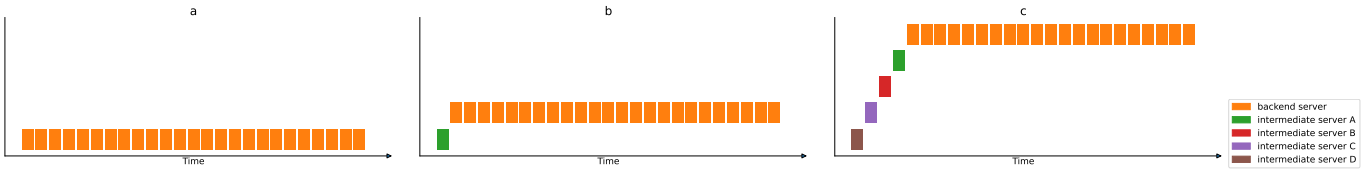

Figure 5. Typical traffic flow sequence pattern. The x-axis represents the elapsed time since the app was launched, the y-axis indicates the different remote server apps connects, while the number of rectangular boxes stands for the traffic volume.

iOS scam apps declare their minimum OS version to be 9.0, which was released in 2015, while the mode of minimum OS version of genuine gambling app is 11.0, which was released in 2017. In addition, the median number of min SDK version and target SDK version in genuine gambling apps are 21 and 29, respectively, which indicates the compatibility of genuine gambling apps is consistent with benign apps but not with gambling scam apps.

Finding X: *Gambling scam apps tend to declare lower requirements for minimum OS versions to ensure compatibility across versions and, thus, can cover more victims, even though their develop SDK versions are not outdated.*

E. Network Infrastructure Analysis

Backend server identification. Identifying the backend infrastructure enables us to understand how mobile scam apps operate. In particular, we (1) design and implement a static analyzer to extract backend server URLs from scam apps automatically and then (2) use mitmproxy [52] to profile their network traffic. Our approach is elaborated on below.

Backend URL identification. We observe that scam apps either leverage network APIs to fetch data from the backend servers or use the WebView component to display the content of specific domains. To extract backend URLs for Android apps, we first collect a list of network request APIs and WebView APIs, such as *WebView.loadUrl(String url)*, from official documents and previous works [53], [12], [54]. Then, we use Androguard to traverse the target app and locate the callers of these APIs.

After locating the position of API calls, we then perform a backward taint analysis for the API's parameter to find the corresponding value of the URL. To be more specific, we first build an inter-procedure control flow graph (ICFG) for the app and mark the local variable of candidate parameters of the called API *WebView.loadUrl* as taint source. We then perform a backward taint analysis along the ICFG to collect all tainted instructions and variables. Specifically, we forward traverse these instructions to reconstruct the string-related operations such as initialization and concatenation of *StringBuilder* and *StringBuffer*. Furthermore, in the case of calling Android-specific APIs, which read the string from asset files, we extract corresponding resources files from the decoded APK file to resolve the string value. Meanwhile, we also record extra semantic information for follow-up filtering work. We extract and save the field name of tainted fields, and, for the used resource strings, we extract their string id and string name from the */res/values/strings.xml* file. In our study, we filter out domains requested by third-party libraries. Particularly, we first collect the top-10 SDKs in each category from AppBrain [55] and the corresponding package namespace, then we filter out the web requests initiated by SDKs rather than apps.

As for iOS apps, we use the keys obtained from Android backend server analysis, locating whether they exist in iOS static resource files (e.g., Info.plist). If they exist, we extract the corresponding value as backend URL candidates. We also extract the strings embedded in the Mach-O file and filter out invalid URLs and domains by Python *validators* library [56]. Inspired by prior work [57], since iOS libraries and their counterparts on Android will share common features, we also filter out the domains used by third-party libraries, which were collected in Android backend analysis.

Traffic profiling. We first write 403 lines of Python scripts to automatically spawn an app in physical devices and capture its traffic for 10 minutes. To reduce interference between different apps, we uninstall the previous app before installing the new one. To this end, we install and capture traffic of 1,068 Android apps and 19 iOS apps and successfully collect 21.28 GB of raw traffic log files in total. Note that some iOS apps failed to install due to revoked enterprise certificates. To reduce irrelevant traffic, we filter out the CDN static resources and third-party service traffic through the list of CDN domain [58] and third-party domains acquired from the previous step.

Given those traffic log files, we build the traffic flow sequence by each request's timestamp and map the target host of requests to a number corresponding to the order they first appear in the app's traffic. Besides, inspired by related works [59], [60], which show the first few seconds of a packet sequence indicating the most useful features for website fingerprint, we focus on the first few packages (in our study, we set the threshold to be 100 packages) after the backend server appeared in network traffic. By calculating the sequence similarity (via Euclidean distance [61]) between each traffic, we successfully group them into three clusters.

As shown in Figure 5, apart from directly fetching scam content from backend servers, shown in Figure 5-a, we notice that some apps will first connect to one or more intermediate servers, and these servers will return with HTTP 200 or 302 (for redirection) status code. More concretely, if scam apps receive responses with encrypted backend servers, they will decrypt them first. As shown in Figure 5-b or Figure 5-c, after interacting with the intermediate server(s), apps will fetch scam content from the actual backend server.

Limitation. Our backend URL identification is implemented by a static analysis approach, which cannot cover the code loaded dynamically in nature. Besides, the native *.so* file was

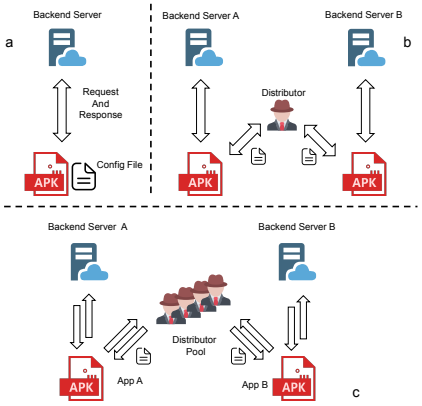

Figure 6. Typical backend network architecture.

not covered in our implementation. In taint analysis, we use the APIs collected from related work and documents as our source. Even though we try our best to complete this list, the scammers may choose rarely used API to bypass this identification. In traffic profiling, we install our proxy certificate both in iOS and Android systems to capture SSL traffic. Even though this approach could cover common cases, we could not guarantee that all gambling scam apps are implemented without SSL pinning.

We filter out the CDN and the third-party libraries based on the existing CDN domain list and the top-10 SDK list from AppBrain; it may lead to false positives if the scammer uses a CDN or SDK not in the list. In addition, different from applying the static taint analysis in Android, we use *strings* util, which may lead to some false negatives due to the lack of modelling string operation such as initialization and concatenation.

Infrastructure profiling. Through the aforementioned approach, we collect 1,161 backend URLs associated with 1,277 scam apps. When analyzing their network infrastructure, besides the traditional infrastructure as illustrated in Figure 6-a, we observe 223 scam apps leveraging servers to update backend servers dynamically, and we named them *distributor*. To avoid being taken down, scammers use distributors to dynamically deliver domains of backend servers while backend servers provide core gambling services, such as account management, betting, and drawing, as shown in Figure 6-b and Figure 6-c.

According to our observation, a gambling scam app immediately communicates with a distributor after initialization, and the payload between the distributor and the client is relatively small. After the app loads the address of the backend server, it launches diverse gambling functions with much higher traffic volume. Thus, we differentiate distributors and backend servers with their activation order and network traffic volume.

After diving into these apps' distributor and backend domains' life cycle, we find more than 55.1% of backend servers were registered less than two years ago — and 12.2% less than one month — while the distributor domains were registered five years ago. The details of top distributors by lifetime can be found in Table XI in Appendix D. This difference may imply that the distributor is much more stable than the backend domain. More interestingly, as shown in Figure 6-c, apart from using dynamic backend server fetching, we observe that 108 scam apps leverage distributor pools to fetch their backend domains. If any of the distributors in the pool are available, scam apps can communicate with their backend servers successfully. We find that this infrastructure design makes scam content delivery more evasive, i.e., the scammer can easily migrate the backend service from one domain to another without distributing new scam apps.

Besides, we cluster the backend servers based on extracted domains. Then, we use passive DNS to query the IP address of each backend server. To further study the cloud services behind it, we use the ip2location dataset [62] to query the AS corresponding to each IP address. Table XII shows the popular ASes and the number of IP addresses of each AS. Although the backend services are distributed on 102 different ASes, most are concentrated on a few cloud services. For example, the top-five cloud services provide services that were used by 79.5% of the gambling scam backends. Most cloud services are located in or around China, which is aligned with the regional nature of the scam targets.

Finding XI: *Gambling scam apps leverage distributors (even distributor pools) to distribute backend servers, which significantly improves scam apps' robustness.*

VII. Payment Channels

Next, we analyze the various payment channels of mobile scam apps to understand their gambling deposit operations and estimate the revenue of online gambling scam activities. Since scam apps usually embed with multiple-factor authentication methods (e.g., CAPTCHA and referral code) before displaying payment details, it is challenging to automatically extract payment channels on a large-scale.

In our study, we manually investigate 100 scam apps [3] to determine their payment channels. More specifically, we install these apps on our five physical devices and manually investigate the payment channels. Considering gambling scam apps may hide malicious/improper behaviors or payment channels when detected in abnormal running environments, e.g., virtual machines, we conduct this study on physical devices. Particularly, once we identify a payment channel, we place an initial test deposit to determine the authenticity. If genuine, we determine whether we should repeatedly perform deposits and if this payment channel is associated with dynamic recipients. It is worth noting that our deposit testing neither sent money to benefit the criminals nor attempted to communicate with the scammers. In our research, we only repeatedly place deposit *requests* and then withdraw those requests after collecting transaction information (e.g., cryptocurrency address). For those payment channels with dynamic recipients, we conducted repeated deposits on a five-minute basis beginning

[3] 19 of the evaluated apps are iOS, which are in the same set of the iOS apps in Section VI.E. Similarly, the rest of 81 Android apps are randomly selected from the runnable Android apps.

Table VI
THE DISTRIBUTION OF PAYMENT CHANNELS.

Category	Sub Category	#
Bank Transfer	Direct Debit	84
Online Payment	Alipay	25
	WeChat Pay	14
	UnionPay	8
Cryptocurrency	Tether-ERC20	9
	Tether-TRC20	10
	CGPay	15
Money Mule Based Payment	Alipay	29
	Idlefish	5
Total		100

from 14:00 to 14:30 on December 17, 2020. We placed 120 deposit requests in total related to 101 different recipients. Through this method, we successfully registered accounts on 92 gambling scam apps. Eight apps required a mandatory referral code, and we observed 197 different payment channels accepted by those scam apps.

Overview. Table VI shows the number of scam apps associated with each payment channel. On average, each app consists of 2.14 payment options. In addition to traditional bank payment channels (e.g., direct debits) and online payment services to fixed recipients (e.g., WeChat Pay and Alipay, which is similar to Paypal; It allows users to send and receive money online through their own accounts), we observe scammers heavily adopt anonymous payment methods (e.g., cryptocurrency, including Tether-ERC20 [63], Tether-TRC20 [64], CGPay [65]) or use money mule-based payment methods (e.g., Idlefish Money Mule (Section VII-A) and Alipay Money Mule). Scammers recruit money mules with Idlefish stores or Alipay accounts to transfer gambling deposits. We observe that scam apps sometimes disable bank payment channels and only enable cryptocurrency and money mules, which may aim to evade banks' financial censorship. Interestingly, compared to gambling scam apps, we find that most genuine gambling apps deploy only traditional payment channels, such as credit cards and wire transfers.

Compared to previous fraud research, our research introduces the payment channels used in mobile gambling scams and showcases new and stealthy channels for anonymous transactions, i.e., Idlefish Money Mules, which supplements the traditional payment approach mentioned by previous fraud activities (e.g., credit card by [6], [14], [66], bank transfer by [67], [68], online payment by [9], [66], [69] and cryptocurrency by [70], [71]).

Below, we characterize two typical payment channels of mobile gambling apps — Idlefish Money Mule and Cryptocurrency — and estimate the scamming revenue on these two channels.

A. Idlefish Money Mule

In our study, we unveil a stealthy and previously-unknown payment channel Idlefish Money Mule, which is adopted by five scam apps in 100 investigated scam apps. Here we summarize the operational pipeline of this payment channel discovered in our research.

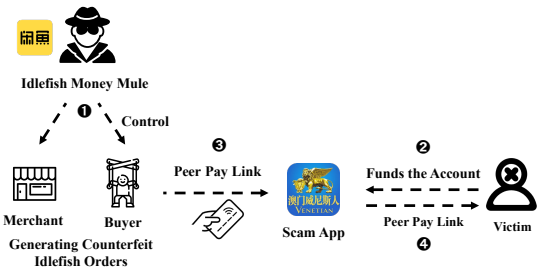

Figure 7. The operational pipeline of Idlefish Money Mule.

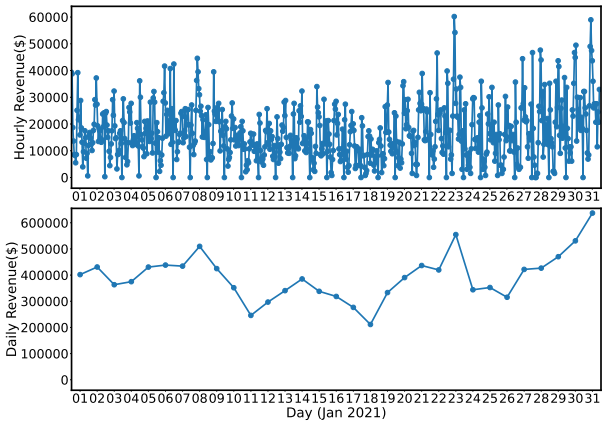

Figure 8. The transaction flow of Idlefish Money Mule.

Operational pipeline. Figure 7 shows that in the payment channel of Idlefish Money Mule, scammers recruit money mules, who control sellers and buyers of a flea market app called Idlefish [72], to transfer gambling deposits (❶). More specifically, when a victim funds the account to gamble (❷), a scam app will trigger a task, which asks a money mule to generate an Idlefish order of the account (❸) and then return the Peer Pay Link, which is a kind of link generated to allow the payment by other parties (i.e., peer). The Peer Pay Link is associated with the counterfeit Idlefish order to the victim (❹). After that, scam apps usually ask the victim to scan the QR Code (which redirects the victim to the Peer Pay Link webpage) to deposit money. The Peer Pay Link webpage looks exactly the same as the normal online payment webpage (see Figure 11), making the payment innocent. During the payment procedure, victims do not need to register in IdleFish. Instead, they pay via the Peer Pay Link in the Online Payment Apps (e.g., AliPay). When finishing the purchase on Idlefish, the money mule will transfer the income and obtain commissions.

Paralleling the Idlefish Money Mule with a traditional money mule [13] in the underground marketplaces, the Idlefish money mule do not require goods to be shipped. The money mules only provide fake product purchasing orders (both the buyer and seller of the transaction are the money mules themselves) and ask the victims to pay for this order.

Revenue analysis. As mentioned earlier, we repeatedly placed deposit requests on scam apps with Idlefish Money Mule

for one month, from January 1, 2021, to January 31, 2021. From analyzing network requests of the scam app, we observe that each Idlefish payment request is associated with a unique payment link (e.g., ppage.jgrkjo.cn/zfb/pc/10648**), which records the Idlefish product, its price and a timestamp. Interestingly, the last seven-digit number of the payment link is uniquely and continuously indexed with each payment request. After a short period (about five minutes), the payment status will be changed to "paid" or "expired", and the page shows either "this order has been paid" or "this order has been expired." By traversing these auto-increment payment links, we are able to collect all the *paid* payment links of our measurement duration (26,731 in total).

Since deposits via this payment channel do not necessarily come from the gambling scam apps, we refer to the daily revenue as an "upper bound" estimation. Inspired by [71], we filter potentially unrelated transactions to victims by known deposit payment patterns. For example, some scam apps only allow users to deposit ¥2,000, ¥5,000 or ¥10,000 CNY (about $307, $769, $1538 USD). Thus, transactions differing from those amounts will be eliminated. This method enables us to estimate the revenue under the payment channel of Idlefish Money Mule. To this end, we study 17,144 payment records, where yields a revenue of $400,000 USD per day with 553.03 transactions. The daily revenue from the Idlefish is about 60x larger than the cryptocurrency payment channel (as discussed in Section VII-B). Its average transaction amount is 2.7x, while the number of transactions is 22.3x. We observe that the peak often occurs between 10:00 and 13:00 and the valley from 4:00 - 8:00 (UTC+8). There are no transactions between 23:00 - 24:00, which may be their server downtime for maintenance. Here we acknowledge the limitation of our revenue analysis: while we have tried our best to eliminate transactions unrelated to victims, the transactions used for revenue estimation could still include non-gambling scam deposits. Hence, we consider the results as the "upper bound" estimation.

Payment request characteristics. Given those payment links, we extract 1,500 Idlefish product information entries in the categories of furniture (47.4%), clothes (14.6%), jewels (8.1%), handbags (6.13%), etc. Interestingly, we observe the prices of these products are always in full thousands and largely deviate from the normal prices (e.g., a pencil sharpener for $1,538 USD).

We determine 631 products with significant price deviation (i.e., the price is 10 times more than the normal price; Here the normal price is calculated from the average price of top-20 search results for the same product in Idlefish) and search for them on Idlefish to identify the potential Idlefish stores of these products for manual validation. In this way, we find 110 Idlefish stores uniquely associated with these products. When we try to bypass the gambling platform to purchase these products, these stores all claim products have been sold out and refuse to sell. Surprisingly, Idlefish label all of those stores as highly reputable. We have reported our findings to Idlefish, which has taken down those illicit stores.

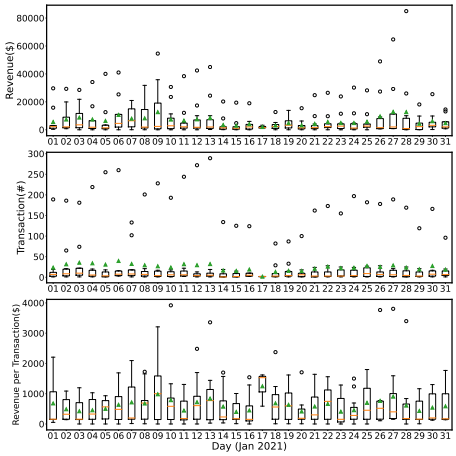

Figure 9. The transaction flow of Cryptocurrency payment.

Finding XII: *We unveil a stealthy and previously unknown payment channel, Idlefish Money Mule, and observe up to $400,000 USD daily revenue in this channel.*

B. Cryptocurrency Payment

In our study, we collect 19 Tether (USDT) addresses related to cryptocurrency payments. Due to the transaction traceability of Tether, we can estimate the *upper bound* of total gambling deposits under these addresses. Specifically, we gather all transactions of these 19 addresses via `List Token Transfer Events` Restful API provided by *etherscan.io* [73], *api.trongrid.io* [74] for Tether-ERC20 and Tether-TRC20, respectively. Similar to the "upper bound" analysis of Idlefish Money Mule, we filter transactions that are potentially unrelated to victims by known deposit payment patterns. Some scam apps only allow users to deposit money at least ¥100 CNY (about $15 USD). Thus transactions less than this threshold are eliminated. This enables us to estimate the revenue under the payment channel of Tether. To this end, we collected 7,900 payment records from 19 Tether addresses.

Figure 9 illustrates gambling deposit volumes over time. We observe that the average daily revenue of each Tether address ranges from $185.05 to $25,296.07 USD, with an average of $6,620.57 USD. The average number of transactions per day is 24.78, with $267.13 USD per transaction. The list of cryptocurrency addresses can be found in Table XVII.

Finding XIII: *Gambling scam apps are in favor of Tether (USDT) and CGPay as the Cryptocurrency payment channel.*

VIII. Discussion

Our study analyzes a ground-truth dataset of mobile gambling scams with 1,487 scam apps and 1,461 incident reports, which is the largest confirmed scam apps and ground-truth incident records dataset ever reported. We found evidence through our extensive analyses of the kill chain of mobile gambling scams ranging from social engineering tricks to scam app generation and distribution, which fuels the international scam activities. When interpreted by professionals, our initial results demonstrate useful findings and may be

used downstream by law enforcement and public policymakers for impactful structural interventions to the mobile gambling scam. In particular, we suggest a suite of mitigation approaches below.

A fundamental prerequisite to mitigate such security issues is the effective detection of gambling scam apps on a large scale. In our study, we profile scam apps to point out potential features that can be used for detection. Specifically, considering characteristics of scam apps, instead of operating maliciously (e.g., stealing personal information), we find the design of scam apps aim at (1) increasing users' "stickiness," e.g., integrating push service to send users notifications (Section VI-B); (2) guaranteeing app reachability, e.g., using the sideloading technique (Section VI-A) with low minimum platform API level requirement (Section VI-D); (3) equipped with stealthy payment channels, especially those anonymous payment methods such as cryptocurrency and money mule (Section VII). Different from web-based fraudulent activities, we observe scam apps usually distributed via one-on-one conversations on IM and social apps (e.g., dating apps) (Section V-B). We suggest deploying a chatbot to gather threat intelligence (i.e., app download link) related to such scamming activities, further empowering the scam app detection. In addition, we provide the details of abusive apps in Appendix E (Table XIII, XIV, XV, XVI and X), which can also help the community to understand this threat.

Perhaps importantly, our study sheds light on two relatively under-explored stakeholders in mobile scam activities, i.e., online app generators, which was abused for scam app generation, and the flea market app (e.g., Idlefish), where money mules host illicit stores for money laundering. We suggest these two parties contribute to the disruption of the mobile gambling scam, e.g., flea market apps should deploy techniques to detect stores involved in money laundering (such as selling products with abnormal prices or suspicious transactions).

In addition, educating mobile users is still one of the most effective ways to prevent individuals from being scammed [6], [75]. We recommend passing on knowledge about social engineering tricks (Section V) as scam awareness training materials, including how scammers establish connections with victims, how scammers promote scam apps, how scammers lure victims to continually deposit money and explaining the gambling scam logic. Our investigation (Section VI-A) also reveals that few scamming apps are indexed in famous application markets like Apple App Store or Google Play. Avoiding downloading apps from unauthorized channels is recommended to battle scam apps.

Responsible Disclosure. We reported the abuse of online app generators, push services, app signing services, and money mules payment channels to the affected parties and received responses and acknowledges from Apple, HUAWEI, Xiaomi, Getui, DCloud and Idlefish. APICloud and JPush expressed gratitude for our help and mentioned that they would cooperate with the law enforcement departments for further action. Unfortunately, we did not receive any response from Meizu after contacting them several times until paper submission. The detail of responsible disclosure can be found at [76].

IX. Related Works

Studies on phishing and scam activities. Numerous studies have looked into phishing [77], [78], [79], [80] and scam activity [5], [6], [4] profiling and detection. For example, Kharraz *et al.* [5] automatically identified the survey scam ecosystem using learning techniques. Miramirkhani *et al.* [6] performed a systematic study of technical support scams and identified their infrastructure and campaign. Dam *et al.* [4] performed large-scale analysis of pop-up scams on typosquatting URLs and present characteristics of such web-based scam campaigns. Hao *et al.* [13] gave insights into the underground economy behind reshipping scams, which launder money through acquiring expensive goods via stolen credit cards. Park *et al.* [25] presented a data-driven empirical analysis of targeted Nigerian scams observed on Craigslist based on a scam dataset collected by experiment. Apart from the web-based scam, a mobile-based scam shows quite different technical details and luring strategies. FraudDroid [8] dynamically analyzes the app's UI state transition and network traffic to detect ad fraud in mobile Android apps. Hu *et al.* [9] performed a study to detect and comprehend the characteristics of fraudulent dating apps. Unlike previous research, which is mainly based on external measurement without the validation of scamming behaviors, with the help of ground-truth data of mobile gambling scam apps, we are capable of analysing this new trending scam with the kill chain of gambling scams.

The closest to our study are [81] and [82], which discuss the ecosystem of online gambling and their third-party online payment and network infrastructures. However, both papers focus on online gambling instead of the gambling scam.

Malicious/Illicit mobile apps detection and analysis. Malicious/illicit mobile apps detection has been studied for long. On the Android platform, Zhou *et al.* [83] systematized or characterized existing Android malware with more than 1,200 samples. Arp *et al.* [84] combined static analysis and linear Support Vector Machines (SVM) technique to detect Android malware Applications with explanations. Zhang *et al.* [85] classified Android malware via dependency graphs by extracting a weighted contextual API dependency graph as program semantics to construct feature sets. More recently, Mclaughlin *et al.* [86] proposed a new detection method for Android malware that uses a deep Convolutional Neural Network (CNN). The neural network automatically learns malware features from the raw opcode series, thereby eliminating the need for hand-engineered malware features. Zhang *et al.* [87] proposed to incorporate domain knowledge into machine learning models to better detect fast-evolving malware variants.

Different from numerous malware analysis works on Android, little has been done on iOS applications. PiOS [88] leverages static analysis to detect privacy leakage in iOS apps. iRiS [89] also proposes a dynamic analysis approach

by porting Valgrind to iOS to find the abuse of private APIs. Damopoulos *et al.* [90] dynamically analyzed iOS software in terms of method invocation to detect malicious code. Garcia *et al.* [91] investigated the features belonging to iOS malware and classified samples of 36 iOS malware families discovered between 2009 and 2015. Cimitile *et al.* [92] designed a malware detector aimed at malicious iOS samples exploiting machine learning and an opcode-based feature set. Lee *et al.* [93] reported the first measurement study on iOS apps with hidden crowdturfing UIs.

In contrast to the apps handled by previous works, gambling scam apps usually do not use private APIs or present patterns of malicious behaviors, which the aforementioned detection tools cannot detect. Instead, our study analyzes a ground-truth dataset to profile scam apps to point out potential features that can be used for detection. Considering the findings of OAG abuse, [46] reported the security issues/vulnerabilities of OAG itself, which had a different study scope, compared with our work. Recently, [47] mainly focused on unwanted distribution channels vectors on Android devices. They discovered that some OAGs were abused to generate and publish unwanted apps (some of them were used to promote fraud ads). In this paper, we take a step closer to real-world scam gambling activities and report that the OAGs have been used in real-world gambling scams. Besides, our study shows that some OAGs' certificates have been abused to sign other gambling scam apps (but not generated), which advances the understanding of the OAG abuse.

X. Conclusion

With the rapid growth and popularity of mobile markets today, mobile gambling scams have caused tremendous financial damage to individuals and corporations. This paper provides the first empirical study based on ground-truth data of the mobile gambling scam consisting of 1,461 scam incident reports associated with 1,068 Android apps and 419 iOS apps provided by an Anonymous Authority. In particular, we reveal social engineering techniques used by miscreants via a qualitative analysis of these mobile gambling scam incident reports. Such knowledge can be used as materials for scam awareness training. In addition, we develop a suite of measurement and dedicated reverse-engineering tools which enabled us to characterize both Android and iOS gambling scam apps, including their development frameworks, declared permissions, the compatibility, and the backend network infrastructure. Our research reported two abused public online app generators, i.e., DCloud and APICloud, used by miscreants to create and sign scam apps. Moving forward, we also examine the payment channels used by mobile scam apps and estimate the revenue of the mobile gambling scam under the payment channel of cryptocurrency and Idlefish Money Mule. Our findings bring new insight into the mobile gambling scam ecosystem. Such understanding and artifacts will help better defend against mobile gambling scam activities.

Acknowledgements

We would like to thank the anonymous reviewers and the shepherd for their insightful comments that helped improve the quality of the paper. This work was supported in part by the National Natural Science Foundation of China (U1836210, U1736208, U1836213, 62172104, 62172105, 61972099, 61902374, 62102093, 62102091), Natural Science Foundation of Shanghai (19ZR1404800), and partially sponsored by a grant from Qi An Xin Group Corporation. Min Yang is the corresponding author, and a faculty of Shanghai Institute of Intelligent Electronics & Systems, Shanghai Institute for Advanced Communication and Data Science, and Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, China.

References

[1] M. Griffiths, "Hi-tech gambling scams," *The Criminal Lawyer*, pp. 4–5, 2004.

[2] Q. 360. (2019) Report on internet fraud trends in 2019. https://zt.360.cn/1101061855.php?dtid=1101062366&did=610412125.

[3] scam detector. (2020) Online casino scam. https://www.scam-detector.com/article/online-casino-scams/.

[4] T. Dam, L. D. Klausner, D. Buhov, and S. Schrittwieser, "Large-scale analysis of pop-up scam on typosquatting urls," in *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES)*, 2019, pp. 1–9.

[5] A. Kharraz, W. Robertson, and E. Kirda, "Surveylance: Automatically detecting online survey scams," in *Proceedings of the 39th IEEE Symposium on Security and Privacy (S&P)*, 2018, pp. 70–86.

[6] N. Miramirkhani, O. Starov, and N. Nikiforakis, "Dial One for Scam: A Large-Scale Analysis of Technical Support Scams," in *Proceedings of the 24th Network and Distributed System Security Symposium (NDSS)*, 2017.

[7] D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker, "Spamscatter: Characterizing internet scam hosting infrastructure," in *Proceedings of the 16th USENIX Security Symposium (USENIX Security)*, 2007.

[8] F. Dong, H. Wang, L. Li, Y. Guo, T. F. Bissyandé, T. Liu, G. Xu, and J. Klein, "Frauddroid: Automated ad fraud detection for android apps," in *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2018, pp. 257–268.

[9] Y. Hu, H. Wang, Y. Zhou, Y. Guo, L. Li, B. Luo, and F. Xu, "Dating with scambots: Understanding the ecosystem of fraudulent dating applications," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2019.

[10] T. Liu, H. Wang, L. Li, X. Luo, F. Dong, Y. Guo, L. Wang, T. Bissyandé, and J. Klein, "Maddroid: Characterizing and detecting devious ad contents for android apps," in *Proceedings of The Web Conference 2020 (WWW)*, 2020, pp. 1715–1726.

[11] Scamwatch. (2021) Scam statistics. https://www.scamwatch.gov.au/scam-statistics?scamid=25&date=2020.

[12] Google, *WevView*, 2021, https://developer.android.com/reference/android/webkit/WebView.

[13] S. Hao, K. Borgolte, N. Nikiforakis, G. Stringhini, M. Egele, M. Eubanks, B. Krebs, and G. Vigna, "Drops for stuff: An analysis of reshipping mule scams," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015, pp. 1081–1092.

[14] N. Scaife, C. Peeters, and P. Traynor, "Fear the reaper: Characterization and fast detection of card skimmers," in *Proceedings of the 27th USENIX Security Symposium (USENIX Security)*, 2018, pp. 1–14.

[15] A. Noroozian, J. Koenders, E. Van Veldhuizen, C. H. Ganan, S. Alrwais, D. McCoy, and M. Van Eeten, "Platforms in everything: analyzing ground-truth data on the anatomy and economics of bullet-proof hosting," in *Proceedings of the 28th USENIX Security Symposium (USENIX Security)*, 2019, pp. 1341–1356.

[16] D. Dittrich, E. Kenneally *et al.*, "The menlo report: Ethical principles guiding information and communication technology research," US Department of Homeland Security, Tech. Rep., 2012.

[17] K. Krombholz, H. Hobel, M. Huber, and E. Weippl, "Advanced social engineering attacks," *Journal of Information Security and applications*, vol. 22, pp. 113–122, 2015.

[18] A. Strauss and J. Corbin, *Basics of qualitative research*. Sage publications, 1990.

[19] A. F. Hayes and K. Krippendorff, "Answering the call for a standard reliability measure for coding data," *Communication methods and measures*, vol. 1, no. 1, pp. 77–89, 2007.

[20] M. A. Dyrud, "I brought you a good news: An analysis of nigerian 419 letters," in *Proceedings of the 2005 Association for Business Communication Annual Convention*, 2005, pp. 20–25.

[21] G. Suarez-Tangil, M. Edwards, C. Peersman, G. Stringhini, A. Rashid, and M. Whitty, "Automatically dismantling online dating fraud," *IEEE Transactions on Information Forensics and Security(TIFS)*, vol. 15, pp. 1128–1137, 2019.

[22] J. Huang, G. Stringhini, and P. Yong, "Quit playing games with my heart: Understanding online dating scams," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment(DIMVA)*. Springer, 2015, pp. 216–236.

[23] F. Stajano and P. Wilson, "Understanding scam victims: seven principles for systems security," *Communications of the ACM*, vol. 54, no. 3, pp. 70–75, 2011.

[24] M. D. Griffiths, "Griffiths, m. d. "crime and gambling: a brief overview of gambling fraud on the internet." *Internet journal of criminology*, 2010.

[25] Y. Park, J. Jones, D. McCoy, E. Shi, and M. Jakobsson, "Scambaiter: Understanding targeted nigerian scams on craigslist," in *Proceedings of the 21st Network and Distributed System Security Symposium (NDSS)*, 2014.

[26] S. Prasad, E. Bouma-Sims, A. K. Mylappan, and B. Reaves, "Who's calling? characterizing robocalls through audio and metadata analysis," in *Proceedings of the 29th USENIX Security Symposium (USENIX Security)*, 2020, pp. 397–414.

[27] Similarweb. (2021) Top gambling websites in the world — similarweb. Https://www.similarweb.com/top-websites/category/gambling/.

[28] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *Proceedings of the 24th Network and Distributed System Security Symposium (NDSS)*, 2019.

[29] Apple, *iTunes search API.*, 2020, https://affiliate.itunes.apple.com/resources/documentation/itunes-store-web-service-search-api/.

[30] Androguard. (2020) Androguard. https://github.com/androguard/androguard/.

[31] Google. (2020) Google play store. https://play.google.com/store/apps.

[32] U. du Luxembourg. (2020) Androzoo. https://androzoo.uni.lu/.

[33] Tencent. (2020) Myapp android app store. https://android.myapp.com/.

[34] Xiaomi. (2020) Xiaomi app store. https://app.mi.com/.

[35] iuuu9. (2020) iuuu9 app store. https://www.iuuu9.com/.

[36] Anxz. (2020) Anxz app store. https://www.anxz.com/.

[37] iBotPeaches. (2020) Apktool - a tool for reverse engineering android apk files. https://github.com/iBotPeaches/Apktool.

[38] Apple. (2021) Install custom enterprise apps on ios. https://support.apple.com/en-us/HT204460.

[39] Apple, *Apple Developer Enterprise Program*, 2021, https://developer.apple.com/programs/enterprise.

[40] Facebook, *React Native Document*, 2021, https://reactnative.dev/docs/getting-started.

[41] DCloud, *DCloud Document*, 2021, https://ask.dcloud.net.cn/docs/.

[42] Cocos2d, *Cocos2d-x Docs*, 2021, https://docs.cocos.com/cocos2d-x/manual/en/.

[43] Unity, *Unity User Manual*, 2021, https://docs.unity3d.com/Manual/index.html.

[44] T. A. S. Foundation, *Documentation - Apache Cordova*, 2021, https://cordova.apache.org/docs/en/latest/.

[45] APICloud, *APICloud Document*, 2021, https://docs.apicloud.com/.

[46] M. Oltrogge, E. Derr, C. Stransky, Y. Acar, S. Fahl, C. Rossow, G. Pellegrino, S. Bugiel, and M. Backes, "The rise of the citizen developer: Assessing the security impact of online app generators," in *Proceedings of the 39th IEEE Symposium on Security and Privacy (S&P)*, 2018, pp. 634–647.

[47] P. Kotzias, J. Caballero, and L. Bilge, "How did that get in my phone? unwanted app distribution on android devices," in *Proceedings of the 42th IEEE Symposium on Security and Privacy (S&P)*, 2021.

[48] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[49] Google, *Manifest.permission — Android Developers*, 2021, https://developer.android.com/reference/android/Manifest.permission.

[50] P. S. Foundation, *plistlib*, 2020, https://docs.python.org/3/library/plistlib.html.

[51] CSDN. (2020) ios permission configuration template. https://blog.csdn.net/zl18603543572/article/details/110914792.

[52] mitmproxy. (2021) mitmproxy is a free and open source interactive https proxy. Https://mitmproxy.org/.

[53] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel, "Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps," in *Proceedings of the 35th annual ACM SIGPLAN conference on Programming Language Design and Implementation (PLDI)*, 2014, pp. 259–269.

[54] Google, *URLConnection*, 2021, https://developer.android.com/reference/java/net/URLConnection.

[55] AppBrain. (2021) Android development tool statistics and market share. Https://www.appbrain.com/stats/libraries/development-tools.

[56] kvestKonsta Vesterinen. (2020) Python data validation for humans. Https://validators.readthedocs.io/en/latest/.

[57] K. Chen, X. Wang, Y. Chen, P. Wang, Y. Lee, X. Wang, B. Ma, A. Wang, Y. Zhang, and W. Zou, "Following devil's footprints: Cross-platform analysis of potentially harmful libraries on android and ios," in *Proceedings of the 37th IEEE Symposium on Security and Privacy (S&P)*, 2016, pp. 357–376.

[58] vysecurity. (2020) Domainfrontinglists. https://github.com/vysecurity/DomainFrontingLists.

[59] J. Hayes, G. Danezis, J. Hayes, and G. Danezis, "k-fingerprinting : A robust scalable website fingerprinting technique," in *Proceedings of the 25th USENIX Security Symposium (USENIX Security)*, 2016, pp. 1187–1203.

[60] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proceedings of the 23rd USENIX Security Symposium (USENIX Security)*, 2014, pp. 143–157.

[61] Wikipedia. (2021) Euclidean distance. https://en.wikipedia.org/wiki/Euclidean_distance.

[62] IP2Location. (2021) Ip2location lite. https://lite.ip2location.com/.

[63] Etherscan. (2021) Token tether usd. https://etherscan.io/token/0xdac17f958d2ee523a2206206994597c13d831ec7.

[64] T. Network. (2021) Usdt introduction. https://tron.network/usdt.

[65] CGPay. (2021) Linking the world cryptocurrency payment gateway. https://cgpay.cc/.

[66] R. G. Smith, "Consumer scams in australia: An overview." *Trends & Issues in Crime & Criminal Justice*, no. 331, 2007.

[67] 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)*. IEEE, 2017, pp. 235–250.

[68] M. Arafat, A. Qusef, and G. Sammour, "Detection of wangiri telecommunication fraud using ensemble learning," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*. IEEE, 2019, pp. 330–335.

[69] Y.-L. Zhang, J. Zhou, W. Zheng, J. Feng, L. Li, Z. Liu, M. Li, Z. Zhang, C. Chen, X. Li *et al.*, "Distributed deep forest and its application to automatic detection of cash-out fraud," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 5, pp. 1–19, 2019.

[70] M. Vasek and T. Moore, "There's no free lunch, even using bitcoin: Tracking the popularity and profits of virtual currency scams," in *International conference on financial cryptography and data security*. Springer, 2015, pp. 44–61.

[71] D. Y. Huang, M. M. Aliapoulios, V. G. Li, L. Invernizzi, E. Bursztein, K. McRoberts, J. Levin, K. Levchenko, A. C. Snoeren, and D. McCoy, "Tracking ransomware end-to-end," in *Proceedings of the 39th IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2018, pp. 618–631.

[72] Idlefish. (2020) Idlefish. https://2.taobao.com/.

[73] Etherscan, *Ethereum Developer APIs*, 2021, https://etherscan.io/apis.

[74] T. Network, *TRON Documentation*, 2021, https://tronprotocol.github.io/documentation-en/api/http.

[75] I. Kirlappos and M. A. Sasse, "Security education against phishing: A modest proposal for a major rethink," *IEEE Security & Privacy*, vol. 10, no. 2, pp. 24–32, 2011.
[76] Anonymous. (2021) Responsible disclosure. https://github.com/MobileGamblingScam/Reports.
[77] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," *Neural Computing and Applications*, vol. 25, no. 2, pp. 443–458, 2014.
[78] A. Oest, Y. Safei, A. Doupé, G.-J. Ahn, B. Wardman, and G. Warner, "Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis," in *2018 APWG Symposium on Electronic Crime Research (eCrime)*, 2018, pp. 1–12.
[79] 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 37th IEEE Symposium on Security and Privacy (S&P)*, 2016, pp. 743–758.
[80] 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 29th USENIX Security Symposium (USENIX Security)*, 2020, pp. 379–396.
[81] H. Yang, K. Du, Y. Zhang, S. Hao, Z. Li, M. Liu, H. Wang, H. Duan, Y. Shi, X. Su, G. Liu, Z. Geng, and J. Wu, "Casino royale: A deep exploration of illegal online gambling," in *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC)*, 2019, pp. 500–513.
[82] Y. Gao, H. Wang, L. Li, X. Luo, G. Xu, and X. Liu, "Demystifying illegal mobile gambling apps," in *Proceedings of the Web Conference 2021(WWW)*, 2021, pp. 1447–1458.
[83] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *Proceedings of the 33rd IEEE Symposium on Security and Privacy (S&P)*, 2012, pp. 95–109.
[84] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket," in *Proceedings of the 21st Network and Distributed System Security Symposium (NDSS)*, vol. 14, 2014, pp. 23–26.
[85] M. Zhang, Y. Duan, H. Yin, and Z. Zhao, "Semantics-aware android malware classification using weighted contextual api dependency graphs," in *Proceedings of the 21st ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2014, pp. 1105–1116.
[86] N. McLaughlin, J. M. Del Rincon, B. J. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doupe, and G. J. Ahn, "Deep android malware detection," in *Proceedings of the 7th ACM on Conference on Data and Application Security and Privacy (CODASPY)*, 2017, pp. 301–308.
[87] X. Zhang, Y. Zhang, M. Zhong, D. Ding, Y. Cao, Y. Zhang, M. Zhang, and M. Yang, "Enhancing state-of-the-art classifiers with api semantics to detect evolved android malware," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 757–770.
[88] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, "Pios: Detecting privacy leaks in ios applications." in *Proceedings of the 18th Network and Distributed System Security Symposium (NDSS)*, 2011, pp. 177–183.
[89] Z. Deng, B. Saltaformaggio, X. Zhang, and D. Xu, "iris: Vetting private api abuse in ios applications," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015, pp. 44–56.
[90] D. Damopoulos, G. Kambourakis, S. Gritzalis, and S. O. Park, "Exposing mobile malware from the inside (or what is your mobile app really doing?)," *Peer-to-Peer Networking and Applications*, vol. 7, no. 4, pp. 687–697, 2014.
[91] L. García and R. J. Rodríguez, "A peek under the hood of ios malware," in *Proceedings of the 11th International Conference on Availability, Reliability and Security (ARES)*, 2016, pp. 590–598.
[92] A. Cimitile, F. Martinelli, and F. Mercaldo, "Machine learning meets ios malware: Identifying malicious applications on apple environment," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP)*, 2017, pp. 487–492.
[93] Y. Lee, X. Wang, K. Lee, X. Liao, X. F. Wang, T. Li, and X. Mi, "Understanding iOS-based crowdturfing through Hidden UI Analysis," in *Proceedings of the 28th USENIX Security Symposium (USENIX Security)*, 2019, pp. 765–781.

APPENDIX A
iOS APP SIDELOADING OPERATIONAL PIPELINE

Specifically, as shown in Figure 10, victims can sideload a scam app through the following five steps: (1) Click the 'Install" from mobile browser; (2) Allow to download the configuration profile (a.k.a., enterprise certificate); (3) Install the configuration profile; (4) Enter a passcode; (5) App starts to install automatically.

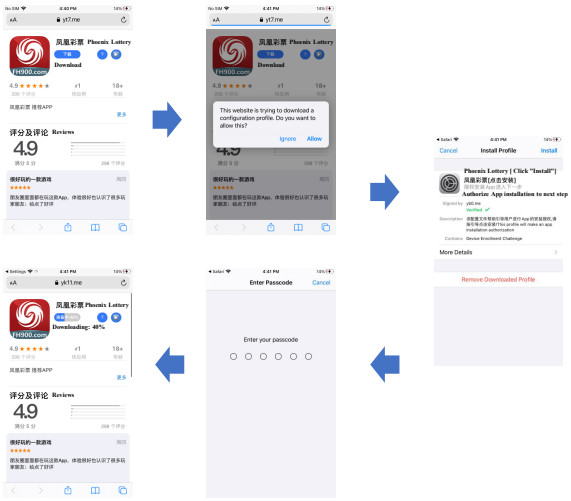

Figure 10. iOS Gambling Scam Apps Sideloading Operational Pipeline

APPENDIX B
POPULAR MOBILE APP DEVELOPMENT FRAMEWORK

Table VII
POPULAR MOBILE APP DEVELOPMENT FRAMEWORKS

APICloud	AppCan	Appcelerator	Appian
AppsBuilder	Bizagi	CocoaTouch	Cocos2dx
Cordova	Corona	DCloud	Ext JS
Flutter	Framework7	Intel App Framework	Intel XDK
Ionic Framework	jQuery Mobile	Kissflow	LungoJS
Mobile Angular UI	Mobincube	Monaca	Mono
Native Script	Onsen UI	PhoneGap	React Native
Sencha Touch	SproutCore	Swiftic	Unity3D
WeX5	Xamarin	Zoho Creator	

APPENDIX C
TOP GAMBLING SCAM APP CLUSTERS

Table VIII
TOP-5 ANDROID SCAM APP CLUSTERS

# Apps	# Certs	% Top-1 Cert	# Prefix	% Top-1 Prefix	# Backend URLs	# Backend Domains
555	2	86.8	4	81.4	547	543
56	2	67.9	55	3.6	55	55
44	2	93.2	34	25.0	42	42
43	43	2.3	43	2.3	172	60
36	2	86.1	6	66.7	26	26

Table IX
TOP-5 IOS SCAM APP CLUSTERS

# Apps	# Certs	% Top-1 Cert	# Prefix	% Top-1 Prefix	# Backend URLs	# Backend Domains
191	40	26.2	1	100.0	190	190
141	76	9.2	7	92.2	141	141
31	22	9.7	1	100.0	31	31
30	22	13.3	29	6.7	73	65
17	1	100.0	1	100.0	17	17

APPENDIX D
BACKEND SERVER INFORMATION OF GAMBLING SCAM APPS

Table X
TOP-5 DOMAIN CONTACTED BY THE SCAM APPS

Domain	Number
appapi.fhptcdn.com	16
www.lebogame.co	15
jfnchaba.hhmusgxoosbb.com	15
nsiugaba.jyvgpprtfmaa.com	15
isjnhaba.mefvjpodsypb.com	15

Table XI
TOP-5 OLDEST DISTRIBUTORS USED BY THE SCAM APPS

Domain	SLD	Registration Time
www.lebogame.co	lebogame.co	2015-04-09 23:03:22
webapi.bjsjdf.cn	bjsjdf.cn	2017-09-14 17:20:11
jkedappg.ctrldoremiwwr.com	ctrldoremiwwr.com	2017-10-24 09:50:13
fd5aappg.ctrldoremiabc.com	ctrldoremiabc.com	2017-10-26 05:50:04
appapi.fhptcdn.com	fhptcdn.com	2017-10-30 08:01:18

Table XII
TOP-5 AS WITH IP RELATED TO THE SCAM APPS

AS Number	AS	# IP
133199	SonderCloud Limited	2282
59371	Dimension Network & Communication Limited	1214
4837	China Unicom	369
4134	China Telecom	179
55720	Gigabit Hosting Sdn Bhd	100

APPENDIX E
DETAILED INFORMATION ABOUT GAMBLING SCAM APP

Table XVI
TOP-5 ISSUERS OF IOS SCAM APPS

Issuer	Number
Qingdao Silver Century Health Industry Group Co., Ltd	49
Petrochina Pipeline Company	29
Hangzhou Anve Technology Co., Ltd	20
AFFISHAUL WORLDWIDE, LLC	18
Yixinlian Mediacal Technology (Beijing) Co., Ltd	14

Table XIII
TOP-5 ANDROID PACKAGE NAME PREFIXES

Prefix	Number
com.yibo.*	452
com.qq11.*	100
io.fhpt.*	28
com.log.*	24
com.ying.*	18

Table XIV
TOP-5 IOS BUNDLE IDENTIFIER PREFIXES

Prefix	Number
com.yibo.*	222
com.mobile.*	130
io.fhpt.*	18
hb.com.*	5
com.ying.*	4

Table XV
TOP-5 ISSUERS OF ANDROID SCAM APPS

Issuer	SHA-1	Number
yibo	713e483e2023d79fa49e2ef72106796e5482e5d6	482
Digital Haven (Beijing) Network Technology Co., Ltd	baad093a82829fb432a7b28cb4ccf0e9f37dae58	137
test	ec544b1220066a8d20a752ff917a8a67379fcaf6	73
a	1029e04097cbc2f152dee7d6b8d3d70339afdc1c	31
Beijing Hurricane mobile Co.,Ltd	78be780e089d25f76e263ce6db0c29599fd86066	18

Table XVII
TETHER ADDRESS IDENTIFIED IN GAMBLING SCAM APPS

Network	Address
Ethereum	0x5313f2ffc31eeda6cb1fb6bcb972554f2d51d244
Ethereum	0x765efd79917896fa95d6fd9ae2c5d2447e9a5064
Ethereum	0x29abcea3c25f043a9aef7c5674e3d04f1821a487
Ethereum	0x9c26e05a466fff4ba5577adf6c4a897fe2e65933
Ethereum	0xa568c99c7e71dbc7a53973e9b1bb561741baf6a3
Ethereum	0xfb01c3c95b1d639b929312574b5576408f850f11
Ethereum	0x808627f44a63fe598eaf0f8b35643e171111323c
Ethereum	0x4b3fdffb57d6a87f26e6919befdf09b4829464ec
Ethereum	0x4dd6577e22c9eaaa3ad206583987ee3f147b5d6d
Tron	TCyUTagTJSodXZq12Pg91LrihevtuoxcTs
Tron	TRcNUA3afmDrVGLwinbC7VrrP7Tf8Yx994
Tron	TDHZ1jSdMx1JvzT9eFLd363vTqf9pKJf5D
Tron	TRdyfUskhp5CTZh4Hw2UhmkzeuQKicZn2U
Tron	TRHDFyJK8cMxR66sEkNEuM4HkaDQx4zPr4
Tron	TDaLVErcpma3j9hKKvvoR5hX2o9icZb9cX
Tron	TF3A6jeDBei3dwmzhC9imCvbPgxnaGdwuP
Tron	TWz3aeSjRrKi5eQgBEiNTpMYg5x3jDXP6r
Tron	TCpDZFQ4MQD1CgPMkwWCZnGin9hmjFN7Kb

APPENDIX F
PEER PAY LINK WEBPAGE SCREENSHOT

Figure 11. Peer Pay Link Webpage Screenshot