Lifting The Grey Curtain: Analyzing the Ecosystem of Android Scam Apps

Zhuo Chen, Lei Wu*†, Yubo Hu, Jing Cheng, Yufeng Hu, Yajin Zhou †, Zhushou Tang, Yexuan Chen, Jinku Li, and Kui Ren, *Fellow, IEEE*

Abstract—Mobile applications (apps) are extensively involved in online scams. Previous studies mainly target *malicious* apps that either compromise victims' devices (*e.g.*, malware and ransomware), or lead to privacy leakage and abuse (*e.g.*, creepware). Recently, an emerging kind of app *makes profits by providing scam services rather than compromising devices or abusing privacy*. We name these apps as **scamware** due to their deceptive behavior, which poses a new threat to (mobile) users. However, the characteristics and the ecosystem of scamware remain mysterious.

This paper takes the first step toward systematically studying scamware. In total, $1{,}262$ ground-truth scamware are collected from December 1, 2020, to May 1, 2022. Specifically, we first investigate the social tricks used by scamware, and then analyze the participants and their relationships to demystify the ecosystem behind scamware. Finally, we reveal the scamware development features to facilitate the detection of scamware. Our study also gives some interesting findings, *e.g.*, 1) the crowd-sourcing strategy is adopted to develop scamware, *i.e.*, the *scammers* are the core members, while other participants are hired as peripherals; and 2) the online app generators have been abused to facilitate development; and 3) the money mule based payment is prevalent, and the case study shows the money flow is around $\$2{,}593{,}346$ per day. We believe that our findings will facilitate the community and law enforcement agencies to mitigate this threat, and we will release the source code of our tools to engage the community.

Index Terms—Cyber Crime, Online Scam, Android App, Empirical Study.

1 Introduction

Online scam is a pervasive and costly global issue. Over the past few years, the world has suffered from various online scams, such as phishing [10], romance dating scam [4], credit card fraud [1], [48], [75], and others. The losses caused by online scams are increasing every year [8]. In 2020, the reported loss of global scam crimes exceeded $4.1 billion. According to the Federal Trade Commission (FTC) [11], the mediums that facilitate online scams include emails, websites, advertisements, *etc*. Previous works have paid attention to these mediums and studied how to combat online scams [29], [61], [65], [77].

In recent years, mobile applications (apps) have been extensively involved in online scams. In addition to some benign apps that are exploited by scammers to commit fraud, i.e., spreading phishing messages via Twitter. There are emerging apps created with the intent to facilitate fraudulent activities. These apps play an essential role in reported attacks [33], [67], [78] that profit from victims by *providing scam services, rather than compromising devices or abusing privacy*. Due to their scam behaviors, in this paper, we call them *scam apps*, or **scamware** for short. Figure 1 gives two examples of scamware. The first one is a gambling scam app that provides a seemingly fair gambling platform, but the withdraw function does not respond as promised. After victims spend money participating in the game, they cannot withdraw money. The second one is a financial fraud app that disguises to be a legitimate cryptocurrency exchange for victims' deposits.

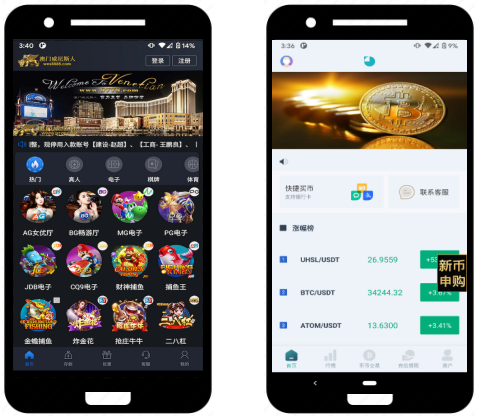


Fig. 1: (a) A *gambling* scam. (b) A *financial fraud* scam.

The proliferation of scamware has caused substantial financial losses. For example, the COVID-19 pandemic has led to an increase in online shopping, which boosts low-quality online shopping fraud. FTC estimated that this fraud had caused more than $\$245$ million loss in 2020 [20]. Another example is the romance scam that lures victims to invest in a dishonest investment app [7]. Such scams

- *Zhuo Chen, Lei Wu, Jing Cheng, Yufeng Hu, Yajin Zhou and Kui Ren are with the Department of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China. E-mail: {hypothesiser.hypo, lei_wu, chengjing, yufenghu, yajin_zhou, kuiren}@zju.edu.cn*
- *Zhushou Tang and Yexuan Chen are with QI-ANXIN Technology Group Inc. E-mail: {ellison.tang, greysign}@gmail.com*
- *Yubo Hu and Jinku Li are with the Department of Computer Science and Technology, Xidian University. E-mail: {yuboHu, jkli}@xidian.edu.cn*

* *Lei Wu is the corresponding author.*

† *These authors are also affiliated at Key Laboratory of Blockchain and Cyberspace Governance of Zhejiang Province.*

have caused $304 million loss in 2020 [18]. As such, it is urgent for the community to understand the characteristics of scamware and their ecosystem.

Previous studies [9], [45], [53], [66], [73] mainly target *malicious apps* (*i.e.*, malware, such as *backdoor* and *trojan*) that compromise/damage victims' devices. Some other studies [71], [74] focus on apps that may lead to privacy leakage and abuse (*e.g.*, *creepware*) for interpersonal attacks. However, due to the different features/behaviors of scamware, previous works aiming to detect/analyze malware cannot be used to cope with this new threat effectively. Recently, several ad-hoc studies proposed focusing on specific domains of scamware, *i.e.*, gambling scam app [35], [50] or dating scam app [51], [52], [80]. Although these studies provide interesting findings for particular types of scamware, the characteristics of scamware as a whole, including the social tricks used by scamware and the underlying ecosystem, remain mysterious. Under such a circumstance, it is difficult to support further investigations.

Our Work. In this paper, we aim to answer the following three research questions comprehensively: 1) *what are the social tricks used by scamware*? 2) *what is the underlying ecosystem to support scamware*? 3) *what are the development features of scamware*?

To support our study, we continuously collected the ground-truth scamware involved in real cases, spanning from December 1, 2020, to May 1, 2022, provided by an Anonymous Authority [1]. In total, we established the scamware dataset with 1,262 Android apps.

Based on the ground-truth datasets, we perform the following studies, ranging from scamware social trick analysis, scamware ecosystem analysis, and scamware feature analysis. First, we inspect the scamware dataset and incident reports. We establish the categorization of social tricks and conclude their scam kill chain (multi-stage chain of events culminating to attack [84]) in Section 7. Second, to demystify the underlying ecosystem, we investigate the system structure and characterize the stages, including development, delivery, scamming, and profit transfer (from Section 6.1 to Section 6.4). Finally, we reveal the development features that can be used to detect scamware in Section 7.

Contributions. To the best of our knowledge, our study is the first systematic work to study scamware and their ecosystem at scale. Our study provides the following contributions:

- *We quantitatively demonstrate the categorization of scam social tricks.* Specifically, we manually investigate 1,262 scamware samples with incident reports and build a categorization (including 5 top-categories, 19 sub-categories, and 9 profit tricks) to classify the scam social tricks. By doing so, our analysis suggests that the *Financial* (41.28%) scams and *Gambling* (22.50%) scams are the most prevalent ones.
- *We comprehensively demystify the scamware ecosystem.* Our analysis shows that multiple participants are involved, *i.e.*, *the crowd-sourcing developer* for development, *the agent* for distribution, and *the money mule* for money laundering. Specifically, the scammer is the *core* member, and the other participants are *periphery*, and have formed a *core-periphery* structure.
- *We reveal the development features of scamware.* Our study points out some unique features, *e.g.*, scamware developers are more inclined to develop hybrid apps (82.88% in scamware vs. 51.19% in benign apps), and abuse app generators (23.93% in scamware vs. 10.08% in benign apps) to facilitate the development. In addition, to accommodate the emerging generator-based apps, we build an analysis framework [2] to automatically identify generators, decrypt, and separate template code.

Findings. Our study also provides some interesting findings, and the following are prominent:

- *The crowd-sourcing strategy is adopted to develop scamware.* A number of small individual developers are hired to develop scamware. However, each one of them contributes only a few apps (57.48% are less than 4 apps). Meanwhile, the developers are distributed in multiple countries and become a transnational collaboration.
- *Scamware leverage covert distribution channels and use "access code" widely.* None of scamware is propagated through app markets, and most of them (67.2%) are propagated through social media. Meanwhile, the "access code" is widely used (28.40%) to make the propagation stealthy, especially in Porn (55.26%).
- *Scamware extensively uses money mule based payment services to transfer profits.* Most payment services (94.9%) adopted by scamware are money mule based payments [5]. Although not feasible to reveal the whole picture, we successfully identify the money flow for some scamware with flawed implementation. The transaction amount of one payment service is up to $2,593,346 per day.

2 BACKGROUND

In this section, we introduce some app development concepts, such as development paradigms and assistant tools (*i.e.*, app generator, online payment service).

2.1 App Development Paradigms

There are three development paradigms, as follows:

- **Native apps:** are developed in a platform-specific programming language (*e.g.*, Android apps are developed in Java/Kotlin). They are cross-platform incompatible with better running performance.
- **Web apps:** are developed in web techniques (*e.g.*, HTML, CSS, and JavaScript), which can be loaded in browsers (*e.g.*, Chrome, Safari) or embedded WebViews. They are not standalone and must be a vessel of other apps.
- **Hybrid apps:** are the combination of native and web apps. Hybrid apps are standalone like native ones but internally are built using web techniques [28] (*e.g.*, WebView). Hybrid apps can be separated into two parts: local clients and remote services. Local clients bridge users' requests to remote services, while remote services provide service per users' requests.

In this study, we only focus on the native and hybrid apps in the Android platform.

1. Anonymous for the double-blind review.

2. To engage the community, the analysis framework is released in the project: **https://github.com/HypoopyH/Scamware_tools**

2.2 App Generators

The thriving *app generators* [69] ease the app development process. App generators lower the bar of technical skills required for app development by integrating the user-supplied code snippet, media resource files, or website addresses with the predefined template codes (namely *boilerplates* in HTML5 [43]) to build an app. Moreover, app generators try to simplify the pipeline of app development, distribution, and maintenance [68], *e.g.*, most app generators offer encryption methods (*e.g.*, RC4, TEA) to protect the entire APK file, while some online app generators (OAG) provide services to maintain app version upgradation. All these features of app generators allow the creation of apps in batches.

2.3 Online Payment Services

The rise of mobile shopping and consumption has stimulated the emergence of online payment services, including online bank transactions, third-party payment services (*e.g.*, Paypal, Alipay), and emerging Cryptocurrencies (*e.g.*, Bitcoin, Ethereum). These payment services provide the ability to accept payments online. Generally, merchants need to register a merchant account and then bind their merchant information (or Cryptocurrency addresses) to the interface of these payment services. So that consumers can see the basic information of the payee when they initiate a transfer request. In addition, to escape from regulation, the money mule based payment services [32], [72] are widespread in the underground. Note that a money mule [32] is a person who receives money from a third party (any legal payment service) and transfers it to another one, obtaining a commission for it.

3 Motivation

Mobile scam/fraud incidents have been widely observed in the wild in recent years. Typically, these incidents are revealed by the victims. For example, the victim of a loan app provided the following report:

> *"I was in urgent need of money recently. At this time, someone in the chat group recommended me a loan app. He (the scammer) said the app could offer up to* 30,000 *Chinese yuan and sent me screenshots of his successful loan. I trusted him and paid the registration fee of about* 500 *yuan. But in the end, the loan didn't get on my card at all! When I tried to get in touch with him, he had disappeared! "*

Accordingly, we conducted an investigation to understand the malware behavior of this loan app and reveal the underlying ecosystem to support it. Malware and its ecosystem have been widely studied for more than one decade, and there are a number of tools [24], [44], [63] and approaches [31], [87] could be used to perform the analysis.

Intuitively, there may exist some malicious behavior in this loan app. To figure it out, we scanned the app with state-of-art malware detection tools (*e.g.*, VirusTotal [24]). Surprisingly, no malicious code could be identified. After conducting a careful investigation, we found that *the malicious behavior of the app comes from the social tricks rather than the malicious code*. Unfortunately, tools relying on the detection of malicious code are not capable of analyzing such apps.

What's more, our investigation also suggests that *the underlying ecosystem of such apps is thoroughly different from the traditional malware ecosystem* which consists of four stages [37], [58], [84], [85], including *development*, *delivery*, *malicious code running* and *clean-up*. For instance, most malware is delivered through the app markets [60], while this loan app is propagated by social media. Besides, this loan app focuses on transferring benefits rather than cleaning up the attack traces in the last stage.

Obviously, traditional malware analysis tools and approaches are ineffective in analyzing scamware. New insights and approaches are required to serve the need. To this end, this paper takes the first step toward systematically studying scamware. Specifically, our research is driven by answering the following three research questions:

RQ1 What are the social tricks used by scamware?
RQ2 What is the underlying ecosystem to support scamware?
RQ3 what are the development features of scamware?

In the following sections, we first introduce the data collection in Section 4. Based on the ground-truth datasets, we perform the following studies, ranging from scamware social trick analysis Section 5 for **RQ1**, scamware ecosystem analysis in Section 6 for **RQ2**, and scamware feature analysis in Section 7 for **RQ3**, respectively.

4 Data Collection & Ethics

4.1 Data Collection

Although scamware are prevalent, building a credible scamware dataset remains a huge challenge due to the lack of an effective way to perform the ground-truth validation. Under such a circumstance, users' feedback is a relevant reliable way to identify malicious behaviors of scamware. Hence we collaborated with an Anonymous Authority to establish a ground-truth scamware dataset.

Scamware Dataset. The *scamware* dataset is collected from an Anonymous Authority, spanning from December 1, 2020, to May 1, 2022. The dataset consists of two parts: scamware samples and their incident reports. Specifically, when a scam incident is reported by the victim, the Authority experts will manually contact the victim, download the app sample, and investigate the scam activity to build the corresponding incident report. A typical incident report includes the reporting time, economic losses, and details about the scam behaviors (*i.e.*, how scammers deliver apps, seduce victims to pay, and finish scam activities). To ensure the accuracy and credibility of our dataset, we have implemented stringent criteria. Specifically, we only focus on cases that involve clear records of scam processes. This selective approach helps us maintain a high-quality dataset with reliable report information. In addition to incident reports, we also analyze the manifests of the respective apps. By parsing the apps' manifests, we gather developer information and filter out benign apps that have legitimate developer information but are being exploited by scammers. This step is crucial in distinguishing between apps created by scammers and those being exploited for fraudulent activities. Our rigorous

process of collecting and cross-referencing data ensures that our dataset primarily consists of scamware, thereby providing a reliable resource for further analysis and research.

In total, we establish the scamware dataset with 1,262 Android samples and the corresponding incident reports. The file size of samples ranges from 16 KB to 164 MB. The smallest one only has one activity, which wraps a WebView [13]. The largest one has many porn pictures and multiple SDKs (*e.g.*, live show SDK and payment SDK).

Meanwhile, to identify the features of scamware, we establish another two datasets for comparison, *i.e.*, benign dataset and traditional malware dataset, as follows:

- **Benign Dataset.** Apps in the *benign* dataset are randomly collected from reliable Android app markets (*e.g.*, Google Play [21] and XIAOMI app store [22]). There are 60,611 apps in the benign dataset, and they have been validated by well-known detection tools (*e.g.*, VirusTotal [24]).
- **Traditional Malware Dataset.** Apps in the *traditional malware* dataset are collected from Virusshare [23]. We choose to use the latest released version (2019), which contains 66,727 android malware samples, including Trojan, Backdoor, and others [9], [45], [53], [66], [73].

We will release part of our dataset to engage the community after the paper is accepted.

4.2 Ethics

Our study aims to profile the scamware, enable law enforcement to understand scamware, and propose efficient mitigation strategies. It follows the similar approach of some prior cybercrime studies [48], [50], [75], which also cooperate with Authorities and obtain ground-truth datasets.

In the study, we strictly managed our activities to ensure our behaviors stayed within the ethical boundary. For the data collection, the data we obtained was delivered from the victims on their own initiative. To protect the victims' privacy, all related personal information has been anonymous following the GDPR [2] by the Authority. We did not interact with the victims throughout the whole study.

As to the data analysis, we ensure that our experiments do not contribute any benefit to the criminals. For example, in the delivery analysis (Section 6.2), we actively generated sharing links and obtained information (*e.g.*, the invitation reward). To avoid distributing scamware, we only generated sharing links but did not send them to others. In the money transfer analysis (Section 6.4), we repeatedly requested the payment interface to collect the payment information (*e.g.*, the payee information and the address of cryptocurrency). To avoid benefiting the criminals, we withdrew the payment after the collection.

5 Social Trick Analysis

Unlike traditional malware makes profits by exploiting vulnerabilities and compromising devices [36], scamware employs scam social tricks to make profits. To answer **RQ1**, we conduct a qualitative study of scamware and incident reports to profile the social tricks. First, we categorize apps and summarize their social tricks. After that, we analyze the social tricks in-depth and conclude scamware kill chain (multi-stage chain of events culminating to attack [84]).

5.1 App Categorization

The scamware we collected are of different types, which need to be categorized to support further analysis.

5.1.1 Categorization Process

By following the data coding approaches [49], [55], [79], we perform a three-step method to categorize scamware and demystify the social tricks. Three security experts spend two weeks working together, as a group, to implement the method accordingly.

- **Step I: Pre-processing Data.** First, 190 apps (about 15% of the total dataset) and their incident reports are randomly selected as the initial candidate set. On one side, each group member works independently to manually collect app-running information (*e.g.*, advertisement banners and screenshots). On the other side, each group member needs to analyze the incident reports and draw conclusions from the scam behaviors. After that, the collected raw data are divided into pictures, keywords, and scam descriptions.
- **Step II: Building Preliminary Categorization.** In this stage, the consensus of the preliminary categorization, including the top-category and the sub-category, needs to be formed. By identifying the pictures and the keywords, the group first establishes the top-category (*e.g.*, *Porn*, *Gambling* and *Financial*). After that, the group determines the scam descriptions for all samples in the same top-category. Accordingly, the top-category can be further divided into multiple sub-categories (*e.g.*, *Financial - Loan & Credit Platform*, *Financial - Cryptocurrency Trading*) according to their different scam scenarios. Meanwhile, each sub-category is encoded with an appropriate category name and corresponding social tricks.
- **Step III: Finalizing Categorization.** Based on the preliminary categorization, the remaining 1,072 scamware samples are further encoded. Note that there may exist corner cases that cannot be covered, which would lead to deviations and affect the subsequent encoding. To address the deviation problem, we apply a strategy based on Krippendorff's *alpha coefficient* [62]. Specifically, for a small deviation ($\alpha >= 0.85$) within the threshold of social science, the existing categorization will be slightly adjusted/refined. For an excessive deviation ($\alpha < 0.85$), a new sub-category will be created to complete the categorization. Finally, the remaining unclear cases which cannot be categorized, will be labeled as "miscellany" (e.g., *Porn - Porn Miscellany*), if no detailed information could be identified from their incident reports.

To better illustrate the process, we present a concrete example in the following. In step I, our group members inspect the initial samples and record their information independently. An example of the record is as follows:

> *"This app showed advertisements ("Shocking! Within 48 hours, bitcoin increased by 540%") in homepage and offered a cryptocurrency trading service. . . . From the incident report, the trading service is fake and the app could not be opened anymore on January 15, 2021."*

In step II, we first assign its top-category as *Financial* since it provides financial-related pictures and investment keywords. Then according to the scam description, this app

TABLE 1: The Categorization of Scamware.

Top-category	Sub-category	Num & Percentage	User Seduction			Purchase/Deposit			Follow-up			Profit Trick[1]
			U1	U2	U3	D1	D2	D3	F1	F2	F3	
Porn	**Sex Trafficking**	70 (%=5.55)	●				○				○	Romance Fraud
	Porn Live	57 (%=4.52)			●	●	○				●	Romance Fraud
	Pornography Purchase	28 (%=2.22)	●		○	○	●		●			Low quality Delivery
	Porn Miscellany	34 (%=2.69)										-
Gambling	**Gambling Games**	161 (%=12.76)	○	○	●			●		●	○	Black-Box Operation
	Lotteries	77 (%=6.10)	○	●			●		●	○		Phishing
	Sports & E-sports Betting	38 (%=3.01)	●					●		●	○	Black-Box Operation
	Gambling Miscellany	8 (%=0.63)										-
Financial	**Loan & Credit**	231 (%=18.30)	●	○	○	●			●			Advanced Fee
	Financial Investment	125 (%=9.90)	●	○			●			○	●	Investment Scam
	Cryptocurrency Trading	76 (%=6.02)	●			○	●			○	●	Investment Scam
	Insurance Products	39 (%=3.09)		●			●		●			Credit Card Fraud
	Financial Miscellany	50 (%=3.96)										-
Platform	**Chatting Platform**	121 (%=9.59)	●		○			●			○	Rumor Spread
	Crowd-sourcing Platform	67 (%=5.31)	○			●			○	○		Advanced Fee
	E-commerce Platform	30 (%=2.38)	●	●			●		●			Low quality Delivery
	Platform Miscellany	20 (%=1.58)										-
Auxiliary Tool	**Sharing Service**	18 (%=1.43)			●	○					○	Accessory Act
	Advertising Service	12 (%=0.95)			●							Accessory Act

[1] Due to the page limit, The profit social tricks are detailed in the appendix repo.
[2] Explanation of different manifestations, all definitions of manifestations are detailed in Section 5.2
U1: Fake/exaggerate advertising. U2: Disguised. U3: Free Trial.
D1: Activation Fee. D2: Product/Service Purchase. D3: Token Recharge.
F1: None or Low-quality Delivery. F2: Disabled Functionality. F3: Server Interrupted.
The two symbols in the table show whether the given manifest is of major (●) or minor (○) adopted in the given app category.

offers a fake cryptocurrency service. After discussion, our group established the consensus to assign it as *Financial - Cryptocurrency Trading* and assign its social trick as *Investment Scam: Apps seduce investors into buying fake investment products by offering false/exaggerated information that offers huge returns with little risk.* In the same way, we analyze all samples and establish the preliminary classification.

In step III, we solve the deviations in the preliminary categorization. Typically, we show an excessive revision as an example.

> *"This app is disguised as an official insurance app, offering accident insurance, life insurance, etc. It required detailed personal information (*e.g.*, telephone, credit card) to register. . . . From the incident report, the victim entered his credit card password, then discovered his card had been stolen."*

This app also belongs to the *Financial* top-category. However, unlike the previous ones, it does not trick users into paying, but steals credit cards. To tackle the excessive deviation, we create a new sub-category as *Financial - Insurance Product* and mark the social trick as *Credit Card Fraud: Apps ask victims to pay with a credit card and record their payment card PIN to steal funds*. After the revisions, we establish the final categorization.

5.1.2 Categorization Result

In total, we identify 5 top-categories (*i.e.*, Porn, gambling, financial, service, and auxiliary tool) with 19 sub-categories in Table 1. The first column shows the top-categories, the second column shows the sub-categories, and the distribution of categories is depicted in the third column. The *Financial* scamware account for the largest portion, which makes up 41.28% of all samples. The second largest is *Gambling* which occupies 22.50%, followed by *Platform* (18.86%), *Porn* (14.98%), and *Auxiliary Tool* (2.38%).

We also conclude with 9 social tricks (*e.g.*, *investment scam, advanced fee*) in the last column of Table 1, as follows:

1) **Romance Fraud:** Apps provide live streams or dating rooms for victims. Scammers induce money from the victims by pretending to keep a romantic relationship with the victims.
2) **Black-Box Operation:** Apps offer a variety of entertainment activities/games and lure victims to deposit. Due to the rigged design/operations, the victims will lose all cash.
3) **Low quality Delivery:** Apps falsify apocryphal products that do not exist or describe products with fake statements that don't match the fact. Victims make the payment but the quality or quantity of the goods or services received is much less than described.
4) **Advanced Fee:** Apps notify victims that they are eligible for a large financial fund, but must first pay a tax or fee to get the fund. The victims paid the advance but never not received the money promised.
5) **Investment Scam:** Apps seduce investors into buying investment products by offering false/exaggerated information and claiming to offer great returns with little risk. But these investment products are all Ponzi schemes.
6) **Credit Card Fraud:** Apps ask the victims to pay with a credit card, surreptitiously record their payment card PIN, and ultimately steal the funds from the victim's credit card.
7) **Rumor Spread:** Apps get commissions from certain organizations or individuals, and deliberately spread fake information (*e.g.*, volatility of stock).
8) **Phishing:** Apps disguise themselves through icons, names, GUI, etc., converting the victims to trust they are formal and get profit.
9) **Accessory Act:** Apps are not engaged in any scam behaviors directly but promote other scamware distribution or

maintenance to get a commission from their revenue.

It is interesting that the profit process is entirely voluntary, *i.e.*, the victims are willing to (even actively) pay for the scam or provide sensitive information during that period.

Finding I: The categories and social tricks are diverse in scamware. Among these categories, the Financial (41.28%) and Gambling (22.50%) account for most.

5.2 Scamware Kill Chain Analysis

After concluding the social tricks, we analyze the scamware kill chain, and summarize the following common stages:

- **User Seduction.** Unlike traditional malware that actively compromises devices, the subsequent processes of scamware can only be carried on after the victims fall into the scam. Therefore, the first stage of the kill chain is to entice the victims by obtaining their interest or trust. Among all social tricks, we conclude three different manifestations: (U1) fake/exaggerated advertising: leverage fake/exaggerated advertising, (U2) disguised as official: disguise as authoritative apps or formal companies' products, and (U3) free trial: offer the opportunity to try capabilities for free in a limited period.
- **Purchase/Deposit.** After obtaining the victims' interest/trust, the scamware focus on making revenue. It is worth noting that they do not take aggressive behavior. Instead, to reduce the victims' vigilance, they masquerade as normal transactions to lure money.
  Thus, the second stage is to seduce the victims into purchasing. There are three manifestations: (D1) activation fee: require users to pay activation for advanced capabilities, (D2) product/service purchase: offer products/services for users to purchase, and (D3) token recharge: exchange token designed for special capabilities.
- **Follow-up.** Finally, the scamware snatch the money. After the follow-up is over, the scam ends and the scammer runs away with the profits. It is only at this moment that the victims realize that they have suffered losses.
  According to the description from the incident reports, we present three manifestations of follow-up operations: (F1) none or low-quality delivery: deliver the goods/services which were of measurably lesser quality or quantity than described, (F2) disabled functionality: freeze users' accounts or disable withdrawing function, and (F3) server interrupted: shut down remote servers, and loss connect.

To better illustrate the scamware kill chain, we take the *Financial - Cryptocurrency Trading* sample in the previous subsection 5.1 as an example. First, in the user seduction stage, it leverages exaggerated advertising (*i.e.*, exaggerated currency value increase advertising) to attract victims. This activity can be classified as *fake/exaggerated advertising*. If the victims fall into the trap, it further offers a fake currency transaction service. This activity can be determined as *product & service sell*. Finally, in the *follow-up* stage, it shuts down the remote server, resulting in *server interrupted*. To sum up, the corresponding kill chain can be concluded as fake/exaggerated advertising $\rightarrow$ product & service sell $\rightarrow$ server interrupted.

Similarly, we analyze all scamware and finally complete categorization with the kill chain. In Table 1, the middle columns show the kill chain and their corresponding manifestations based on the proportion, *i.e.*, *major* ($>= 50\%$) or *minor* ($< 50\%$). The result suggests that the manifestations and tricks used in different scenarios vary greatly.

Finding II: Although the manifestations of scamware vary greatly in different scenarios, they share the common kill chain and behave innocuously before the last stage.

6 Scamware Ecosystem Analysis

As discussed in Section 3, the underlying ecosystem of scamware is thoroughly different from the traditional malware ecosystem which consists of four stages [37], [58], [84], [85]. In this section, we aim to demystify the scamware ecosystem to answer **RQ2**, which can be further divided into the following two sub-questions:

- **RQ2-1** What are the stages of the scamware ecosystem? What are the corresponding participants?
- **RQ2-2** What position do these participants have in the ecosystem? What are their features?

Based on our observation, the scamware ecosystem can be profiled in Figure 2. Specifically, there are four stages: *(i)* development, *(ii)* delivery, *(iii)* scamming, and *(iv)* profit transfer. Besides the scammers, we find three entities that participate in the ecosystem: `crowd-sourcing developer`, `agent`, and `money mule`. There are also some public services that have been abused, *e.g.*, OAGs (online app generators), cloud servers, and domain registrars.

What's more, there is a clear hierarchical relationship, *i.e.*, a *core-periphery* structure, between these participants of the scamware ecosystem. Specifically, the scammers are the core members, while the others are the peripherals hired by the scammers. These participants facilitate the scams and become an integral part of the ecosystem. In the following, for each stage, we will delve into the details to reveal the corresponding participants and their features.

Finding III: The scamware ecosystem consists of four stages: development, delivery, scamming, and profit transfer, and the corresponding participants form a core-periphery structure.

6.1 Scamwawre Development

In this stage, we focus on the developers of scamware. To this end, we first collect data used to distinguish developers of scamware (*i.e.*, developer signatures, OAG user IDs, GUI snapshots). Based on that, we then conduct an association analysis to demystify the developer's constitution. Finally, we verify the identities of OAG developers and summarize the way they participate in the ecosystem.

6.1.1 Developer-related Data Collection

For all samples in the scamware dataset, we collect the following data:

- **Developer Signature.** We obtain the developer signatures by using *"get_signature()"* API provided by Androguard [44].Notably, we observe that some app generators

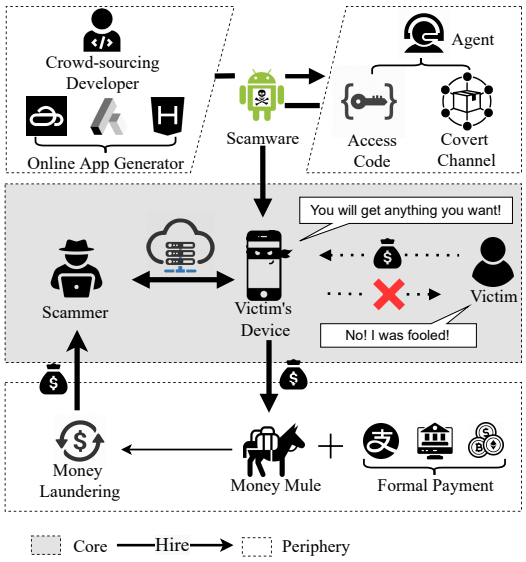


Fig. 2: The structure of scamware ecosystem.

(*i.e.*, Dcloud and Apkeditor) will pack the apps with their signatures, which (18.5% of all signatures) are excluded from the developer signatures.

- **OAG User ID.** A developer needs to register a user ID to use the OAG for development (see Section 2.2), and the ID will be packed by the OAG into the APK files. For example, APICloud locates the ID in *"smali/complile/Properties.smali"*. Such patterns can be used to collect OAG IDs. By doing so, we collect 297 OAG user IDs in total.
- **GUI Snapshot.** The GUI design is an essential developer feature [70]. We built a tool to automatically install and launch the apps on physical devices (*i.e.*, two Nexus 4s and two Nexus 6p) to take GUI snapshots. Besides, we customize a firmware image by modifying the source code of AOSP [14] to grant the snapshot permission by default. For a given app, we capture snapshots every 15 seconds for one minute after its startup.

6.1.2 Developer Association Analysis

Different apps may belong to the same entity. To demystify the constitution of developers, we perform the developer association analysis. Specifically, the association rules are defined as follows:

- *Signature association*: referring to the previous study [76], we associate the signatures with the same hash.
- *User ID association*: as mentioned earlier, we collect OAG user IDs, and associate the ones with the same user ID.
- *Snapshot association*: we leverage the SIFT algorithm [64] to extract key points and calculate Euclidean distance as the similarity between snapshots.

By performing the analysis, we determine 317 different developer groups. The top 10 groups (ranked by the group size, which means the number of apps developed by the developers of the group) are summarized in Table 2. Surprisingly, the majority of scamware are developed by the small groups rather than the large ones, *i.e.*, the average group size is 3.98, while groups whose sizes are less than 10 contain 64.8% of all apps.

TABLE 2: Top 10 Group of the Developer Association Result.

Rank	Apps (%)	Categories Composition			
		Porn	Gambling	Financial	Platform
1	72 (5.71%)	35 (48.6%)	0	37 (51.4%)	0
2	31 (2.46%)	4 (12.9%)	6 (19.4%)	21 (67.7%)	0
3	22 (1.74%)	3 (13.6%)	0	18 (86.4%)	0
4	15 (1.19%)	0	0	15 (100%)	0
5	15 (1.19%)	0	15 (100%)	0	0
6	13 (1.03%)	0	1 (7.7%)	9 (69.2%)	3 (23.1%)
7	12 (0.95%)	11 (91.7%)	0	1 (8.3%)	0
8	12 (0.95%)	1 (83.3%)	3 (25.0%)	4 (33.3%)	4 (33.3%)
9	12 (0.95%)	0	0	9 (75.0%)	3 (25.0%)
10	12 (0.95%)	0	0	12 (100.0%)	0

Since the *Auxiliary tool* does not appear in the top 10, we remove this category from the table.

Obviously, the result does not obey the well-known *80/20 rule* (aka, *Pareto principle* [3]). For instance, the largest group merely contains 72 apps (5.71%). Besides, the majority of apps of a group belong to the same category. As a matter of fact, for a group whose size is less than 5, all apps belong to the same category.

Finding IV: The constitution of scamware developers does not obey the 80/20 rules. The sizes of most developer groups are small (3.98 on average). The majority of apps of a group belong to the same scamware category.

6.1.3 Developer Identity Analysis

To demystify the real identities of the developers and the way they take part in the ecosystem are challenging goals. Fortunately, the abuse of OAGs exposes some clues.

Generally, the OAGs require developers' real identities to register developer accounts. We leverage this feature as the breakthrough to obtain real identities. To avoid raising ethical issues, we hand over these IDs to the Authority without connecting to the OAGs directly. With the assistance of the Authority, we obtained the statistical results to profile the developers, from the following three aspects:

- **Registration Identity.** The OAGs support two kinds of registration methods, one for individual developers and the other for organizations. From the records, only one ID is registered by the organization, and the rest 296 IDs (99.6%) are registered by individual developers. Furthermore, the majority (57.48%) of individual developers develop less than 4 apps. The average number of apps per developer is 3.23. The result shows that most scamware developers are small individuals rather than large organizations, which is consistent with the related findings in Section 6.1.2.
- **Ecosystem Relationship.** To figure out the way individual developers participate in the scamware ecosystem, the Authority contacts 15 developers developing the largest number of apps. Surprisingly, all of the contacted developers are hired for part-time online jobs, *e.g.*,

  " I lost my job and desperately needed some financial income. I saw a demand for developing porn apps on an online part-time site, with a 250 Chinese yuan reward. It told me to use the DCloud to package a URL . . . I don't know who posted the message because the site is anonymous."

  It suggests that the developers are temporarily hired by the scammers and not be involved in scam behaviors.

TABLE 3: The Developer Geographical Distribution.

Area	Country	Number	Percentage
Asia	China	255	85.86%
	Vietnam	8	2.70%
	Thailand	3	1.01%
	Japan	1	0.34%
	Total	**267**	**89.90%**
North America	United State	16	5.39%
	Canada	6	2.02%
	Total	**18**	**10.10%**
Total	-	297	100%

- **Geographic Location.** We further list the geographic location of all developers in Table 3. Though the geographic distribution of developers might be affected by the bias of the scamware dataset, it is clear that the developers are located in multiple countries. Namely, the scamware gangs are cross-border with transnational collaborations.

As such, we can conclude that the scammers adopt the crowd-sourcing strategy to hire a large number of small individual developers. By doing so, scammers may escape from the supervision.

Finding V: Most OAG developers (99.6%) are small individuals hired by scammers through the crowd-sourcing strategy, as the periphery members, to hide their identities.

Finding VI: The scamware gangs are cross-border with transnational collaborations.

6.2 Scamware Delivery

In this stage, we focus on the following three perspectives, *i.e.*, the distribution channels, the invitation strategy, and the registration process, to investigate the scamware delivery.

6.2.1 The Distribution Channels

It is not easy to determine the distribution channels of scamware. Fortunately, the Authority helps us verify the process of app acquisition of 250 victims. Again, we only obtain the statistical result without detailed personal information to avoid any ethical issues.

In summary, we find that the majority of scamware samples (67.2%) are propagated through social media, while part of them (14.0%) are downloaded through the advertisements of the online networks (see Table 4). Besides, there are also a few samples (13.2%) distributed through traditional distribution methods (*e.g.*, SMS and Email with download links or APKs). Note that there still exist some cases (5.6%) that cannot be located. Interestingly, *no scamware samples are obtained from the official app markets (*e.g., *GooglePlay [21])*. Compared with malware which is mainly obtained from the app stores [60], the distribution channels of scamware are non-public and covert.

Finding VII: Unlike malware which is mainly obtained from the app markets, scamware is spread through non-public and covert channels, such as social media.

TABLE 4: The Scamware Distribution Channel.

Category	Sub-Category	Number(%)
Social Media	Wechat	92 (36.8%)
	QQ	41 (16.4%)
	Tiktok	14 (5.6%)
	Alipay	7 (2.8%)
	Others(*e.g.*, Dingtalk, Telegram)	14 (5.6%)
	Total	**168 (67.2%)**
Online Network	Job Hunting	14 (5.6%)
	Game	5 (2.0%)
	Others (*e.g.*, forum)	16 (6.4%)
	Total	**35 (14.00%)**
Traditional	Telephone	20 (8.0%)
	SMS	12 (4.8%)
	Email	1 (0.4%)
	Total	**33 (13.2%)**
Not Found	-	14 (5.6%)
Total	-	250 (100.0%)

6.2.2 The Invitation Strategy

The participants that are responsible for scamware delivery are called "*agents*" by the victims. The incident reports suggest that these agents often proactively invite the victims.

To understand the invitation strategy, we try to contact the agents and ask them the following two questions: *(i)* What benefits can I get from an invitation? *(ii)* How can I become an agent? Although most agents maintain a high level of vigilance and refuse to respond, we successfully interact with a few of them (two from the chatting group, and one from the telephone) to collect useful information. Specifically, users can participate in the delivery stage in the following two ways:

- *Sharing "access code"*. The access code is not only the prerequisite for activating a new account but also an ID that records the initiator's identity. After successfully inviting new users, the initiator will get some rewards.
- *Joining the ecosystem as an agent*. The users can be hired as agents to distribute scamware in one (small) area. Specifically, the agent is responsible for handling user's registration and inquiries. By doing so, the agent gets a steady salary in return from the scammers.

In brief, the agents are converted from the users. As the periphery members, they are responsible for scamware delivery (to make a profit) but not performing the scam behaviors.

Finding VIII: Agents are the periphery members responsible for scamware delivery, which can be converted from the users through the invitation strategy.

6.2.3 The Registration Process

Before using, victims should register an account to activate the app. Specifically, in the registration process, we focus on the "access code" requirement, which is an important feature of the invitation strategy.

We randomly selected 250 scamware samples and another 250 benign samples for comparison. We manually register accounts of each sample and summarize the result

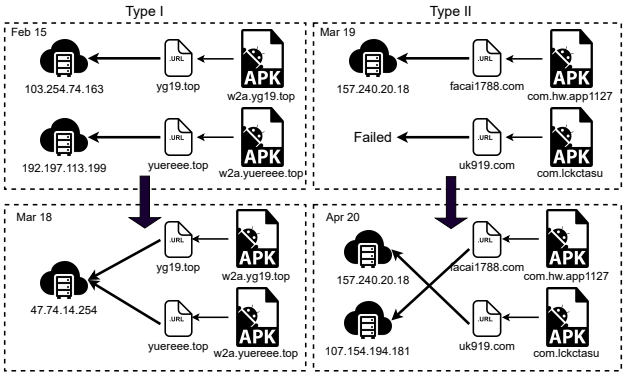


Fig. 3: Typical IP sharing strategy.

in Table 7 in the appendix repo [3]. And many scamware samples (28.40%) require an "access code" as a prerequisite for registration, while no benign apps have the same requirement. Specifically, the "access code" rate is high (55.26%) in the *Porn* category. The result suggests that the "access code" is widely used to reduce the risk of exposure to supervision.

Finding IX: The "access code" is widely used (28.4% of all) as a prerequisite of using scamware, especially for those in the *Porn* top-category (55.26%).

6.3 Scamware Scamming

The social tricks performed by the scammers have been studied in Section 5. Here we aim to reveal the approach used by scamware to maintain the scam services while escaping from supervision.

To this end, we build a tool to automatically identify and collect the addresses of the backend servers of the scam services. Specifically, we first summarize the commonly used network classes and frameworks in Table 8 (see the appendix repo). From the framework documents and Google developer documents [25], [26], we build up a network request API list, such as *android.webkit.WebView: void loadUrl(java.lang.String)*. The value of backend URL is always in the format as "java.lang.String" in API argument. Then, we leverage SOOT [63] to load the APK file and search network request APIs in the list. If the arguments of APIs are *immediateBox* type, we can pinpoint the backend URLs; otherwise, the arguments might be handled through some string conversion methods (*e.g.*, add(), replace()), so the backend URL cannot be obtained directly.

To restore values of the arguments, we first perform backward taint analysis, and record all string-related methods (*e.g.*, "java.lang.String"). After that, we leverage JSA [42] to infer the value of those arguments. Finally, we manually analyze all URLs and filter the benign ones (*e.g.*, Facebook). In total, we collect 1,873 URL addresses with 1,264 valid domain addresses. Besides, we also collect `DNS` and `WHOIS` information about the 1,264 domains per day from January 1, 2021, to September 1, 2021. We observe that there are three strategies leveraged by those backend servers, as follows:

3. Appendix repo is located in https://github.com/HypoopyH/Scamwaretools/blob/main/appendix.pdf

- **Strategy I: Flexible IP binding.** The binding relationship between domains and IP addresses represents the mapping between the registered domains and the physical servers. Though the domains are hard-coded in the apps, the bound IP addresses can be changed. We conduct a domain-IP binding analysis to determine the strategy of the flexible binding.
  Specifically, 307 domains (24.29%) have changed their IP addresses at least once. To these domains, the average duration time of the binding is only 11.65 days. Interestingly, among the 300 domains with the most extended validity period, 221 domains have changed IP more than once. It indicates that a flexible domain strategy is effective for maintaining the remote service for stable running.
- **Strategy II: IP Sharing.** Multiple domains may share the same IP address. There exist two types: *(i) Type-I*: multiple domains share the same IP address at the same time; and *(ii) Type-II*: domain A uses an IP address that is previously used by domain B.
  To provide a better illustration, we present examples in Figure 3. The left part gives an example of Type-I sharing. The domain `yg19.top` and `yuereee.top` both associate with the IP `47.74.14.254` on March 18, 2021. The right part gives an example of Type-II sharing. The domain `uk919.com` associates to IP `157.240.20.18` on April 20, 2021. But on March 19, 2021, this IP was associated with `facai1788.com`. Among the 307 flexible IP binding domains, 43 (14.00%) use Type-I IP sharing, and 121 (39.42%) use Type-II. In total, 163 domains leverage the strategy of IP sharing. The reason behind this needs more experiments to demystify, but we propose a hypothesis: The servers (IP addresses) are limited so the maintainer has to choose an effective server from the server pools to maintain the domain.
- **Strategy III: Abused Public Services.** To find out the owner of the domains, we leverage the `WHOIS` records to analyze the registrars. In total, we collect 164 different registrars. The top 10 registrars are listed in Table 9 (see the appendix repo). The total number of domains in the table is 828, occupying 65.5% of all domains. Clearly, the Alibaba Cloud has been abused by scamware, which occupies more than 26% of all domains. The result suggests that cloud services lack sufficient supervision. Apart from that, the scammers widely leverage anonymity services (*e.g.*, GoDaddy) to hide their true identities.

Finding X: To circumvent the supervision, the scammers manipulate the domains-IP relationships to enhance the availability of the services. Besides, some public services (*e.g.*, GoDaddy, Alibaba Cloud) are abused as well.

6.4 Scamware Profit Transfer

In the final stage, we investigate the process of scamware transferring the illicit profits to the scammers. Specifically, we first identify the payment services leveraged by scamware and then measure the money flow if possible.

6.4.1 Identify the Payment Services

As mentioned in the background 2.3, apps utilize payment services to obtain profit. To identify the payment services

TABLE 5: The Scamware Payment Services Analysis Result.

Category	Payment Interface	Number(%)
Money Mule Based	AliPay	26 (44.1%)
	WeChat Pay	12 (20.3%)
	Bank Transaction	11 (18.6%)
	Cryptocurrency	4 (6.8%)
	Zalo	2 (3.4%)
	PayPal	1 (1.7%)
	Total	**56 (94.9%)**
Formal	AliPay	2 (3.4%)
	WeChat Pay	1 (1.7%)
	Total	**3 (5.1%)**

leveraged by scamware, we randomly select 50 scamware samples to perform the analysis.

Specifically, we use the physical devices [4] to accommodate all these samples. First, we install the apps, and register new accounts. Then, we initiate the payment request for each payment entrance and capture the corresponding network package using WinPcap [6]. It is worth noting that the payment recipients deserve more attention. For example, some payments may rely on the money mule (see Section 2.3), which can be revealed by observing the recipients, i.e., for a given service, if two payments initiated by the same user have two different recipients, then the service may use money mule to perform the payment. Hence, if the request gets a valid response, we keep making requests per hour for the next five hours to guarantee enough time window for the backend servers to distribute the requests to different recipients (if any).

Our security experts, again, work independently to identify the payment services by initiating the payment requests but never actually transferring money to the scammers. Namely, we immediately withdraw the payment requests after receiving the valid response packages. Note that this method does not work for scamware that require the access codes. As such, we successfully identify 59 different payment services in 38 samples with 295 payment responses.

Based on the response packages, we can identify the payment types by applying the following two-step method:

- **Identify Money Mule.** We observe the recipient information returned by multiple payment requests. If the recipient information of one payment entrance is different (including those initiated by the same user at different times, or initiated by different users), we consider this recipient as a money mule. Also, the payment service is money mule based.
- **Identify Payment Interface.** Via search engines (*e.g.*, Tianyancha [12]), we build a list of licensed payment services, which can be identified through their license (such as Alipay). For services not in this list, we manually analyze the payment UIs (*e.g.*, bitcoin icon) to determine the payment interfaces.

The result is listed in Table 5. In short, we observe that most payment services (94.9%) adopted by scamware are money mule based payment services rather than formal payment services. Among these money mule based payments, 41 (73.2%) payment services based on third-party payment interfaces (especially the AliPay and WeChat Pay), 11 (19.6%) services based on bank transactions, and 4 (7.1%) services based on cryptocurrency.

Interestingly, most money mule based payments do not directly invoke formal payment interfaces. Instead, they employ indirect payment methods, such as generating QR codes, or letting users copy the payee accounts, which are shown in Figure 7 (see in the appendix repo). Among the 59 money mule based payments, only 4 payments directly invoke the Alipay SDK, and the rest 55 payments employ indirect payment methods. Conversely, benign apps will not take indirect methods because it would complicate the payments.

Finding XI: The money mule based payment services are widely adopted by scamware (94.9% of all) to transfer profits. Besides, the majority (93.2%) of them employ indirect payment methods, such as generating QR codes, or letting users copy the payee accounts.

6.4.2 Measure the Money Flow

The money flow of payment services can reflect the scam's profitability. However, it is a great challenge to get the ledger due to the black-box design. Fortunately, we successfully discovered two types of money mule based payments that can be used to analyze money flow.

- **Flawed Digital ID Payment.** Generally speaking, formal payments have strict identity authentication and time limits to prevent malicious snooping. However, we find a kind of money mule based payment has implementation flaws. After initiating the requests, these payments jump to a URL, which consists of a domain name, API interface, and a unique digital ID (*e.g., x8.134.69.241/api/payPage.html?id=56****). Unlike formal transactions, these payments neither have time logout nor access control, which means that we can monitor all ongoing transactions. The transaction digital ID is generated randomly, but the return message contains a debug field. Specifically, when we submit an error ID, we will get the "already expired" or "not created yet" return codes, which can be used to quickly locate the correct ID through the binary search.
  To ensure an uninterrupted utilization of the system's cookies, we employ actual mobile devices to transmit requests. Our approach entails simulating transaction requests at two-minute intervals by altering the ID within the request to acquire order content. Subsequently, we use a local *Wireshark* agent to capture the returned packet body and proceed with parsing the data to evaluate the status of funds and ascertain the success of the transaction. This approach helps us to accurately monitor the malicious order in the system.
  Accordingly, we collect 368,874 transactions from a flawed payment service within 30 days, spanning from Nov 14, 2021 to Jan 13, 2022. Surprisingly, the average daily flow is $2,593,346, indicating the huge amount of money laundering. Figure 4 gives the average transaction amount in one day. It shows that the payment transactions are most active during 9:30 - 17:40 (UTC+8) and inactive

4. Two Nexus 4s and two Nexus 6p, with NDK versions from 18-26.

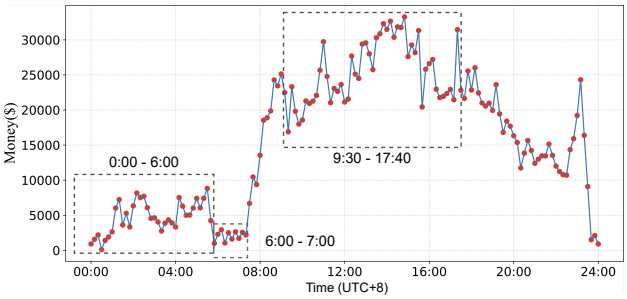


Fig. 4: The average money flow of digital ID payment (per 10 minutes).

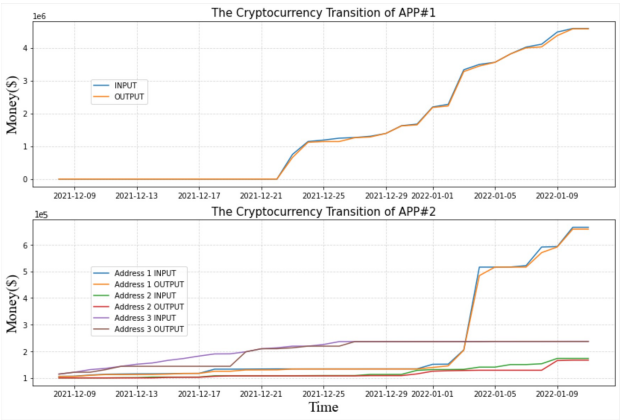


Fig. 5: The money flow of cryptocurrency payment.

during 0:00 - 6:00 (UTC+8), which suggests that *the money flow is relatively consistent with the local daily routine*. Furthermore, usually, no transaction can be found during 6:00 - 7:00 (UTC+8), which might be the maintenance time.

- **Cryptocurrency Payment.** Cryptocurrency addresses are necessary for the payment. There are two options to provide such an address, *i.e.*, reusing the old address, and creating a new address. Among the four cryptocurrency payments we observed, two temporarily created new addresses to transfer funds, while the remaining 2 payments reused the old addresses. Obviously, only the latter two can be used to trace the historical transactions. The two payments belong to two different apps, *i.e.*, *app#1* (one old address) and *app#2* (three old addresses), respectively.
  To obtain information about transactions related to these addresses and calculate the total funds, we utilize *Tronscan* [27] and the provided API to retrieve the necessary transaction data efficiently and accurately. In total, from Nov 8, 2021, to Jan 14, 2022, we collect 4 Tether addresses from these two apps with 546 *TRC20&TRC721* transfers (the accumulated amount is 5,362,609.7 USDT, an incredible huge number!). Figure 5 shows the money flow of these addresses for each app. The income and outcome amounts demonstrate that these addresses only serve as intermediate nodes, and the received funds will be quickly transferred to other addresses.

Finding XII: Some services with flawed implementation can be used to measure the money flow, *i.e.*, $2,593,346 per day on average. As to cryptocurrency payment, those payment addresses only served as the intermediate nodes to transfer funds.

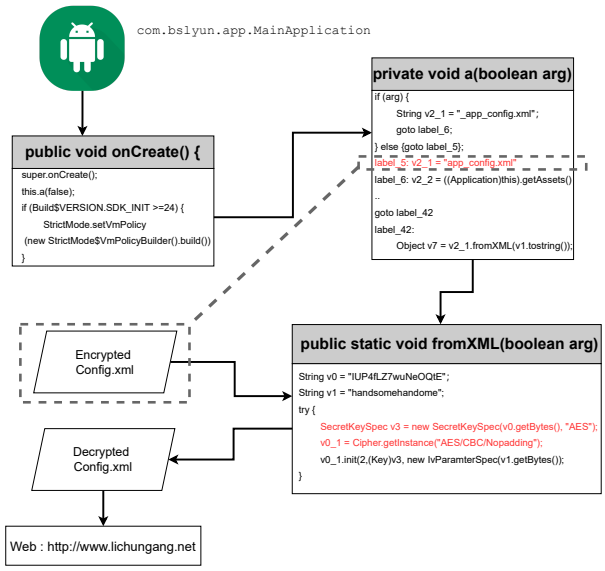


Fig. 6: The analysis process of a real framework-based app.

7 Scamware Feature Analysis

In this section, we perform app analysis and figure out the potential features to answer **RQ3**. To make the results more convincing, we compare the results with the *benign dataset* and the *traditional malware dataset*.

Depending on whether developed by the app-generator frameworks, apps can be classified into two types, *generator-based apps* and *non-generator-based apps*. On one side, for non-generator-based apps, we can rely on existing tools like Androguard [44] to extract information. On the other side, for generator-based apps, traditional tools have flaws: *(i)* generator-based apps are glued with template assets of app generators; *(ii)* some app generators use encryption to prevent reverse engineering.

It is necessary to decrypt APK files and separate user-provided assets from template code to analyze generator-based samples. Unfortunately, to the best of our knowledge, no effective tool can be used. Therefore, we investigate app generators and conclude the following rules:

- *(I)* There exist distinct features for each app generator, which can be used to distinguish generator-based apps. For example, for the apps generated by DCloud [19], the main activity name is set to `io.dcloud.PandoraEntry`.
- *(II)* The encryption method used by an app generator is fixed, which can be used to decrypt the encryption APK. For example, AppCan [16] uses native RC4 encryption, and Appmachine [17] uses TEA encryption.
- *(III)* The templates offered by app generators have fixed structures, which can be used to separate user-provided assets from template code. Therefore, it is possible to extract user-provided assets from the apps precisely.

These rules constitute the fingerprints of app generators, including the technique prerequisite, the main activity, the encryption method, and the configuration file location. The fingerprints uniquely determine the identity of a generator. We list the generator fingerprints in Table 6.

To better illustrate our analysis process, we present a real

TABLE 6: Examples of App Generators Fingerprints.

Generator	Prerequisite	Main Activity	Encryption	Configure File Location	Website
DCloud	-	io.dcloud.PandoraEntry	-	assets/data/dcloud_control.xml	dcloud.io
Yimen	-	com.lt.app	AES.CSC	assets/y.x	www.yimenapp.net
APICloud	-	com.uzmap.pkg.LauncherUI	RC4	config.xml	www.apicloud.com
React Native	React	-	-	com.feacebook.react.*	www.nativescript.org
Appmachine	-	app.Main	TEA	assets/resources.zip	www.appmachine.com
Ionic	AngularJS	io.ionic.starter.MainActivity	-	res/xml/config.xml	ionicframework.com
Appcan	-	org.*.engine.LoadingActivity	RC4	assets/widget/config.xml	www.appcan.cn

app based on the *BSL framework* [5] as an example (see Figure 6 in the appendix repo). First, we identify the main activity of the APK [6], leading us to categorize it as a framework-based app. After that, we pinpoint the app's entry point and discover the app configuration file located in assets/config.xml. However, we encounter an encrypted XML file, requiring decryption. Upon further investigation, we determine that the BSL framework employs AES encryption for assets, and we successfully locate the private key within the code. Utilizing this private key and the decryption algorithm, we ultimately decrypt the config.xml file, obtaining the real information, specifically the developer-provided URL link (www.lichungang.net).

We propose a generator analysis framework to enhance the efficiency of our investigation. This framework consists of three principal components, including generator recognition, resource stripping, and decryption. Specifically, during the generator recognition phase, the analysis framework first employs Apktool to decompile the APK file. It then examines the Manifest file to extract the main activity name and identifies the XML configuration file loaded during initialization. By considering the distinctive features of both the main activity and the XML configuration file, the framework distinguishes between different generators. Subsequently, in the resource stripping phase, the framework locates user-added content by referencing the XML file's configuration. This content may encompass images, code, and website URLs. The framework effectively isolates this encrypted user content from the pre-existing content within the app generators. Lastly, leveraging the encryption algorithm utilized by the app generator, we implement specific decryption algorithms with the aim of restoring meaningful user-generated content. In total, 47 kinds of app generators are successfully covered, occupying the vast majority of app generators all around the world (see Table 10 in the appendix). With the help of the framework, we study apps in-depth, including their development paradigms, generator frameworks, and permission.

7.1 Development Paradigm Analysis

As mentioned before, we only focus on the native apps and hybrid apps in this paper. (See in Section 2.1). The difference between these two development paradigms is whether to introduce the class of "WebView" [13]. Therefore, we reverse APK files by Androguard [44], and search "WebView" API calls (*i.e.*, android.webkit.WebView: void loadUrl).

Finally, we find $1{,}046$ scamware that leverage WebView API calls, which occupies 82.88% of all scamware samples. While for the benign and traditional malware datasets, the number (proportion) of hybrid apps are $31{,}024$ (51.19%) and $31{,}695$ (47.50%), respectively. The distinctive difference shows that scamware prefers the development paradigm of hybrid apps. In some extreme samples, the activities of scamware are merely implemented by the "WebView". As mentioned in Section 2.1, hybrid apps enhance cross-platform migration, this feature implies that scamware place a high value on cross-platform performance.

5. `com.bslyun.app.MainApplication`
6. `com.bslyun.app.activity.MainActivity`

7.2 App Generator Analysis

Through our generator analysis framework, we identify 7 different app generators used in 302 samples, occupying 23.93% of the scamware dataset. For the benign and traditional malware datasets, the proportion of the generator-based apps is only 10.08% and 1.86%, respectively.

Specifically, among these generators, the OAG (mentioned in Background 2.2) occupies the majority of generator-based samples(92.38%), especially the `DCloud` (69.21%) and `APICloud` (18.54%). In summary, we conclude that app generators have been extensively used to lower the technique barrier and facilitate development. What's more, the OAGs are abused and need additional supervision to prevent the proliferation of scamware.

7.3 Permission Analysis

Due to the security design of the Android system, apps need to be explicitly authorized by users to obtain dangerous permissions [15]. For each sample, we examine all permissions from the `AndroidManifest.xml` and identify dangerous permissions. Notably, we only identify the AOSP-defined dangerous permissions [15]. The average number of dangerous permissions required is 6.90 for scamware, while the numbers of benign apps and malware are 2.96 and 9.04, respectively. We also show the details of dangerous permission and the permission groups (see Figure 9 in the appendix repo). The benign dataset requires the least permissions while the traditional malware dataset requires the most. In summary, the permission requirement of scamware deviates from that of benign apps and traditional malware.

Finding XIII: The scamware have significant features: *(i)* scamware severely rely on the hybrid development paradigm (82.88%). *(ii)* the app generators (especially DCloud and APICloud) are widely used, occupying 23.93% of all scamware. *(iii)* scamware use more permissions than benign apps, while fewer than malware.

8 Discussion

Our study analyzes ground-truth android scamware, consisting of 1,262 android scamware and their incident reports. Through this paper, we uncover some valuable findings that can inspire Authority and public awareness.

At first, the most direct method to mitigate scamware is to improve the detection capability. We point out some features of scamware, which can be used for further automatic detection studies. *(i)* the scamware extremely rely on the hybrid development paradigm, and in extreme cases, *"WebView"* is the only valid view component. This special development preference indicates that scamware places a high value on cross-platform capabilities. *(ii)* the development frameworks are widely used in scam app development, especially the OAGs (*e.g.*, APICloud, DCloud). We propose a generator analysis framework that can mitigate these generator-based scamware. Currently, there are some works [41], [47] that have proposed detection methods utilizing our features.

In addition, we also shed light on several ecosystem features that facilitate supervision. *(i)* most developers are individual developers hired by the scammer (Section 6.1). This employment strategy indicates that the Authority should pay more attention to supervising job-hunting platforms, thereby increasing the cost of hiring developers. *(ii)* the covert distribution channel (Section 6.2). There are no scam samples obtained from official markets in our dataset, while most are distributed through social media (*e.g.*, WeChat, Tencent QQ). We recommend that social platforms strengthen the regulation to curb the spread of scamware. *(iii)* the money mules are abused to perform money laundering (Section 6.4).We recommend online payments (*e.g.*, AliPay, WeChat Pay) implement stricter supervision of abnormal transaction movements.

Recommendations Based on our observations and findings, there are some recommendations for users to bolster their security and protect their data during mobile app usage. *(i)* Exercise prudence regarding the absence of *access code*s and EULAs (Section 6.2). *(ii)* Acquaint oneself with prevalent scam tricks (Section 7) to enhance vigilance. Prompt awareness can considerably mitigate potential losses. *(iii)* Preferentially download apps from official app stores. Despite the openness of the Android ecosystem, the dependability of official app stores exceeds that of unregulated sources.

9 Related Work

9.1 Scam Analysis

Scam Scenarios Analysis. There have been several studies focused on different scam scenarios. Hu et al. [51] analyzed romantic dating apps and revealed their purpose. Hong et al. [50] performed a thorough gambling scam app analysis based on the ground-truth data with incident reports, and concluded some impressive findings. Stajano et al. [78] examined various scams and extracted the general principles of scam strategies. Blaszczynski et al. [34] characterized the relationship between gambling and crime, and concluded the behavior of illegal gambling.

Online Scam Analysis. Targeting online scams, Whitty et al. [82], [83] analyzed the romance online scam, and concluded that the romance scam has different stages. Konte et al. [59] studied the domain infrastructure by leveraging *whois* and *DNS* records. Garg et al. [46] pointed out that online scam was not random, but targeted towards specific communities. Kharraz et al. [56] provided a survey of online scams, which is helpful for online scam investigations. It is worth noting that Hong et al. [50] conducted an in-depth analysis of gambling apps based on a ground-truth dataset. In this work, we also utilized a ground-truth dataset for empirical research. However, while Hong's study focused on the scam tricks and features of gambling apps, our work extends to analyzing various types of scam apps, including gambling, pornography, and financial fraud. Furthermore, our work summarized the development characteristics of the apps, and built an analysis framework.

9.2 Security Analysis for Android Apps

Malware detection. Burguera et al. [36] proposed a dynamic analysis of application behavior based on earlier app approaches. Sun et al. [81] studied the permission requirements of Android apps and proved that this could be used to distinguish malware. Arora et al. [30] proposed a detection model by extracting the permission pairs from the manifest. Chen et al. [39] characterized existing Android ransomware and proposed a real-time detection system by monitoring the user interface and finger movements.

Vulnerability detection. Chan et al. [38] proposed a vulnerability-checking system to check vulnerabilities that may be leveraged by privilege attacks. Zhang et al. [86] implemented a prototype designed for detecting hijacking vulnerabilities. Joshi et al. [54] systematically elaborated on the vulnerabilities that exist in the Android system.

Similarity detection. Sebastian et al. [76] presented an approach based on ownership for identifying developer accounts. Chen et al. [40] presented graphical user interface similarity to detect application clones. Kim et al. [57] leverage multiple features and deep learning to detect malware.

10 Conclusion

Nowadays, mobile scams have caused severe financial losses. They are prevalent in the wild and proliferate rapidly. This paper takes the first step to systematically demystify the scamware and their ecosystem. As the prerequisite of our study, we cooperate with an Authority and build up a ground-truth scamware dataset consisting of 1,262 scamware and incident reports. We conduct our study by first analyzing the social tricks by inspecting all scam samples and incident reports. Based on the inspection, we establish a scam categorization and conclude the scamware kill chain. After that, we reveal the ecosystem of scamware and perform an in-depth investigation to demystify the characteristics of different participants. Finally, we leverage state-of-the-art reverse-engineering tools and propose a generator analysis framework to analyze scamware and point out some unique development features. Our research reveals many impressive findings that help the community and law enforcement defend against these emerging threats.

11 ACKNOWLEDGMENTS

We would like to thank all anonymous reviewers for their helpful suggestions and comments to improve the paper. This work is partially supported by the National Key R&D Program of China (No. 2022YFE0113200) and the National Natural Science Foundation of China (NSFC) under Grant No. 62172360, No. U21A20464, and No. U21A20467. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of funding agencies.

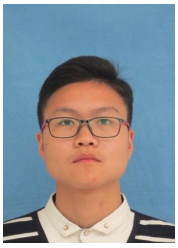
REFERENCES

[1] "credit card fraud," https://www.fbi.gov/scams-and-safety/common-scams-and-crimes/credit-card-fraud.
[2] "General data protection regulation," https://gdpr.eu/, accessed May 28, 2021.
[3] "Pareto principle," https://en.wikipedia.org/wiki/Pareto_principle, accessed June 6, 2021.
[4] "romance dating scam," https://www.fbi.gov/scams-and-safety/common-scams-and-crimes/romance-scams.
[5] "Why does fourth-party payment rise, and what is the charm of aggregate payment?" https://www.programmersought.com/article/52465472004/, accessed January 1, 2021.
[6] "Winpcap," https://www.winpcap.org/.
[7] "Romance scammer," https://www.consumer.ftc.gov/articles/what-you-need-know-about-romance-scams, June 2019, accessed January 1, 2021.
[8] "2020 internet crime report," https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf, May 2020, accessed January 1, 2021.
[9] "Malware categories," https://developers.google.com/android/play-protect/potentially-harmful-applications, October 2020, accessed January 1, 2021.
[10] "Phishing report," https://www.fbi.gov/scams-and-safety/common-scams-and-crimes/spoofing-and-phishing, May 2020, accessed January 1, 2021.
[11] "Privacy, identity & online security," https://www.consumer.ftc.gov/topics/privacy-identity-online-security, January 2020, accessed January 1, 2021.
[12] "Tianyancha," https://www.tianyancha.com/, January 2020, accessed January 1, 2021.
[13] "Webview api reference," https://developer.android.com/reference/android/webkit/WebView, May 2020, accessed January 1, 2021.
[14] "Android open source project," https://source.android.com/, January 2021, accessed January 1, 2021.
[15] "Android permission list," https://developer.android.com/guide/topics/permissions/overvie, January 2021, accessed January 1, 2021.
[16] "Appcan," http://www.appcan.cn/, January 2021, accessed January 1, 2021.
[17] "Appmachine," http://appmachine.com/, January 2021, accessed January 1, 2021.
[18] "Bbc report about romance scammer," https://edition.cnn.com/2021/02/21/us/losses-to-romance-scams-trnd/index.html, January 2021, accessed January 1, 2021.
[19] "Dcloud," https://www.dcloud.io/, January 2021, accessed January 1, 2021.
[20] "Ftc report of 2020," https://www.consumer.ftc.gov/blog/2021/02/top-frauds-2020, January 2021, accessed January 1, 2021.
[21] "googleplay," https://play.google.com/store, January 2021, accessed January 1, 2021.
[22] "Mistore," https://m.app.mi.com/, January 2021, accessed January 1, 2021.
[23] "Virusshare," https://virusshare.com/, January 2021, accessed January 1, 2021.
[24] "Virustotal," www.virustotal.com, January 2021, accessed January 1, 2021.
[25] "Google url developer document," https://developer.android.com/reference/java/net/URLConnection, January 2022.
[26] "Google webview developer document," https://developer.android.com/reference/android/webkit/WebView, January 2022.
[27] "Tron scan," https://tronscan.org, May 2022, accessed January 1, 2021.
[28] M. Ali and A. Mesbah, "Mining and characterizing hybrid apps," in *Proceedings of the International Workshop on App Market Analytics*, 2016, pp. 50–56.
[29] A. Almomani, B. B. Gupta, S. Atawneh, A. Meulenberg, and E. Almomani, "A survey of phishing email filtering techniques," *IEEE communications surveys & tutorials*, vol. 15, no. 4, pp. 2070–2090, 2013.
[30] A. Arora, S. K. Peddoju, and M. Conti, "Permpair: Android malware detection using permission pairs," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1968–1982, 2019.
[31] 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 *Ndss*, vol. 14, 2014, pp. 23–26.
[32] M. Aston, S. McCombie, B. Reardon, and P. Watters, "A preliminary profiling of internet money mules: an australian perspective," in *2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*. IEEE, 2009, pp. 482–487.
[33] B. Atkins, W. Huang *et al.*, "A study of social engineering in online frauds," *Open Journal of Social Sciences*, vol. 1, no. 03, p. 23, 2013.
[34] J. Banks, *Online gambling and crime: Causes, controls and controversies*. Routledge, 2016.
[35] P. Binde, U. Romild, and R. A. Volberg, "Forms of gambling, gambling involvement and problem gambling: evidence from a swedish population survey," *International Gambling Studies*, vol. 17, no. 3, pp. 490–507, 2017.
[36] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for android," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, 2011, pp. 15–26.
[37] J. Caballero, C. Grier, C. Kreibich, and V. Paxson, "Measuring {Pay-per-Install}: The commoditization of malware distribution," in *20th USENIX Security Symposium (USENIX Security 11)*, 2011.
[38] P. P. Chan, L. C. Hui, and S. Yiu, "A privilege escalation vulnerability checking system for android applications," in *2011 IEEE 13th International Conference on Communication Technology*. IEEE, 2011, pp. 681–686.
[39] J. Chen, C. Wang, Z. Zhao, K. Chen, R. Du, and G.-J. Ahn, "Uncovering the face of android ransomware: Characterization and real-time detection," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1286–1300, 2017.
[40] K. Chen, P. Liu, and Y. Zhang, "Achieving accuracy and scalability simultaneously in detecting application clones on android markets," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 175–186.
[41] Z. Chen, J. Liu, Y. Hu, L. Wu, Y. Zhou, Y. He, X. Liao, K. Wang, J. Li, and Z. Qin, "Deuedroid: Detecting underground economy apps based on utg similarity," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2023, pp. 223–235.
[42] A. S. Christensen, A. Møller, and M. I. Schwartzbach, "Precise analysis of string expressions," in *Proc. 10th International Static Analysis Symposium (SAS)*, ser. LNCS, vol. 2694. Springer-Verlag, June 2003, pp. 1–18, available from `http://www.brics.dk/JSA/`.
[43] R. Clark, O. Studholme, C. Murphy, and D. Manian, *Beginning HTML5 and CSS3*. Springer, 2012.
[44] A. Desnos *et al.*, "Androguard," 2011.
[45] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, 2011, pp. 3–14.
[46] V. Garg and S. Nilizadeh, "Craigslist scams and community composition: Investigating online fraud victimization," in *2013 IEEE Security and Privacy Workshops*. IEEE, 2013, pp. 123–126.
[47] Z. Gu, Z. Xu, H. Chen, J. Lan, C. Meng, and W. Wang, "Mobile user interface element detection via adaptively prompt tuning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 11 155–11 164.
[48] 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*, 2015, pp. 1081–1092.

[49] J. A. Holton, "The coding process and its challenges," *The Sage handbook of grounded theory*, vol. 3, pp. 265–289, 2007.
[50] G. Hong, Z. Yang, S. Yang, X. Liaoy, X. Du, M. Yang, and H. Duan, "Analyzing ground-truth data of mobile gambling scams," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 2176–2193.
[51] 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*, 2019.
[52] 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*. Springer, 2015, pp. 216–236.
[53] N. Idika and A. P. Mathur, "A survey of malware detection techniques," *Purdue University*, vol. 48, pp. 2007–2, 2007.
[54] J. Joshi and C. Parekh, "Android smartphone vulnerabilities: a survey," in *2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Spring)*. IEEE, 2016, pp. 1–5.
[55] S. H. Khandkar, "Open coding," *University of Calgary*, vol. 23, p. 2009, 2009.
[56] A. Kharraz, W. Robertson, and E. Kirda, "Surveylance: automatically detecting online survey scams," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 70–86.
[57] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for android malware detection using various features," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 773–788, 2018.
[58] D. Kiwia, A. Dehghantanha, K.-K. R. Choo, and J. Slaughter, "A cyber kill chain based taxonomy of banking trojans for evolutionary computational intelligence," *Journal of computational science*, vol. 27, pp. 394–409, 2018.
[59] M. Konte, N. Feamster, and J. Jung, "Dynamics of online scam hosting infrastructure," in *International conference on passive and active network measurement*. Springer, 2009, pp. 219–228.
[60] P. Kotzias, J. Caballero, and L. Bilge, "How did that get in my phone? unwanted app distribution on android devices," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 53–69.
[61] C. Kreibich, C. Kanich, K. Levchenko, B. Enright, G. M. Voelker, V. Paxson, and S. Savage, "Spamcraft: An inside look at spam campaign orchestration." in *LEET*, 2009.
[62] K. Krippendorff, "Computing krippendorff's alpha-reliability," 2011.
[63] P. Lam, E. Bodden, O. Lhoták, and L. Hendren, "The soot framework for java program analysis: a retrospective," in *Cetus Users and Compiler Infastructure Workshop (CETUS 2011)*, vol. 15, no. 35, 2011.
[64] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.
[65] S. Mansfield-Devine, "The dark side of advertising," *Computer Fraud & Security*, vol. 2014, no. 11, pp. 5–8, 2014.
[66] J. Milletary, "Citadel trojan malware analysis," *Luettavissa: http://botnetlegalnotice. com/citadel/files/Patel_Decl_Ex20. pdf. Luettu*, vol. 13, p. 2014, 2012.
[67] N. L. Muscanell, R. E. Guadagno, and S. Murphy, "Weapons of influence misused: A social influence analysis of why people fall prey to internet scams," *Social and Personality Psychology Compass*, vol. 8, no. 7, pp. 388–396, 2014.
[68] 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," pp. 634–647, 2018.
[69] 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 *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 634–647.
[70] R. Pears, Y. S. Koh, G. Dobbie, and W. Yeap, "Weighted association rule mining via a graph based connectivity model," *Information Sciences*, vol. 218, pp. 61–84, 2013.
[71] A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill, "Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem," 2018.
[72] J.-L. Richet, "Laundering money online a review of cybercriminals methods," *arXiv preprint arXiv:1310.2368*, 2013.
[73] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov, "Learning and classification of malware behavior," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2008, pp. 108–125.
[74] K. A. Roundy, P. B. Mendelberg, N. Dell, D. McCoy, D. Nissani, T. Ristenpart, and A. Tamersoy, "The many kinds of creepware used for interpersonal attacks," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 626–643.
[75] N. Scaife, C. Peeters, and P. Traynor, "Fear the reaper: Characterization and fast detection of card skimmers," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1–14.
[76] S. Sebastian and J. Caballero, "Towards attribution in mobile markets: Identifying developer account polymorphism," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 771–785.
[77] A. Srivastava, A. Kundu, S. Sural, and A. Majumdar, "Credit card fraud detection using hidden markov model," *IEEE Transactions on dependable and secure computing*, vol. 5, no. 1, pp. 37–48, 2008.
[78] 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.
[79] A. Strauss and J. Corbin, *Basics of qualitative research*. Sage publications, 1990.
[80] 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*, vol. 15, pp. 1128–1137, 2019.
[81] L. Sun, Z. Li, Q. Yan, W. Srisa-an, and Y. Pan, "Sigpid: significant permission identification for android malware detection," in *2016 11th international conference on malicious and unwanted software (MALWARE)*. IEEE, 2016, pp. 1–8.
[82] M. T. Whitty, "Anatomy of the online dating romance scam," *Security Journal*, vol. 28, no. 4, pp. 443–455, 2015.
[83] M. T. Whitty and T. Buchanan, "The online romance scam: A serious cybercrime," *CyberPsychology, Behavior, and Social Networking*, vol. 15, no. 3, pp. 181–183, 2012.
[84] T. Yadav and A. M. Rao, "Technical aspects of cyber kill chain," in *International Symposium on Security in Computing and Communication*. Springer, 2015, pp. 438–452.
[85] C. Yang, J. Zhang, and G. Gu, "Understanding the market-level and network-level behaviors of the android malware ecosystem," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 2452–2457.
[86] M. Zhang and H. Yin, "Appsealer: Automatic generation of vulnerability-specific patches for preventing component hijacking attacks in android applications." in *NDSS*. Citeseer, 2014.
[87] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *2012 IEEE symposium on security and privacy*. IEEE, 2012, pp. 95–109.

Zhuo Chen received the BE degree in information security from Huazhong University of Science and Technology in 2020. He is currently working toward the doctoral degree in cyberspace security at Zhejiang University. His current research interests lie in the underground economy, including unwanted app detection and anti money laundering.

Lei Wu is an Associate Professor with the School of Cyber Science and Technology, and the College of Computer Science and Technology, Zhejiang University, China. He obtained his Ph.D. degree from North Carolina State University in 2015. His research interest lies mainly in security areas, including system security and blockchain security.



Yubo Hu received the BE degree in information security from Xidian University, Xi'an, China, in 2021. He is currently working toward master's degree in the school of Cyber Engineering at Xidian University, Xi'an, China. His current research focuses on mobile security.

Jing Cheng received the BE degree in computer science and technology from Tongji University, in 2018. She is currently working toward master's degree in electronic information at Zhejiang University. Her research interests include mobile security and blockchian security.

Yufeng Hu received the Bachelor degree in mathematics and finance from Zhejiang University, in 2020. He is currently working towards a PhD degree in cyberspace Security at Zhejiang University. His research interests include blockchain security, smart contract security, anti-money laundering, and binary security.

Yajin Zhou received the Ph.D. degree in computer science from North Carolina State University, Raleigh, NC, USA. He is currently a ZJU 100 Young Professor with the School of Cyber Science and Technology, and the College of Computer Science and Technology, Zhejiang University, China. His research mainly focuses on smartphone and system security, such as identifying real-world threats and building practical solutions, mainly in the context of embedded systems (or IoT devices).

Zhushou Tang is currently a researcher of QI-ANXIN Technology Group Inc. He received his PhD degree at the School of Computer Science & Engineering of Shanghai Jiao Tong University in June 2020. His research interests include mobile security, program analysis, and blockchain security.

Yexuan Chen is currently a researcher of QI-ANXIN Technology Group Inc. His research interests include mobile security, web security, and blockchain security.

Jinku Li received the B.S., M.S., and Ph.D. degrees in computer science from Xi'an Jiaotong University, Xi'an, China, in 1998, 2001, and 2005, respectively. From March 2009 to February 2011, he was a research associate in the Department of Computer Science at North Carolina State University, Raleigh, NC, USA. He is currently a professor in the School of Cyber Engineering at Xidian University, Xi'an, China. His research focuses on system and mobile security.

Kui Ren is a Professor and Associate Dean of College of Computer Science and Technology at Zhejiang University, where he also directs the Institute of Cyber Science and Technology. Before that, he was SUNY Empire Innovation Professor at State University of New York at Buffalo. He received his PhD degree from Worcester Polytechnic Institute. He received IEEE CISTC Technical Recognition Award in 2017, SUNY Chancellor's Research Excellence Award in 2017, Sigma Xi/IIT Research Excellence Award in 2012, and NSF CAREER Award in 2011.