Received December 6, 2020, accepted December 31, 2020, date of publication January 8, 2021, date of current version January 26, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3049819

Detection, Traceability, and Propagation of Mobile Malware Threats

LONG CHEN[1,2,3], CHUNHE XIA[1,4], SHENGWEI LEI[1], (Graduate Student Member, IEEE), AND TIANBO WANG[1,5], (Member, IEEE)

[1]Beijing Key Laboratory of Network Technology, Beihang University, Beijing 100191, China
[2]Innovation Technology Research Institute, Beijing Topsec Network Security Technology Company Ltd., Beijing 100085, China
[3]Home Internet Operation Center, China United Network Communications Ltd., Beijing 100032, China
[4]School of Computer Science and Information Technology, Guangxi Normal University, Guilin 541004, China
[5]School of Cyber Science and Technology, Beihang University, Beijing 100191, China

Corresponding author: Tianbo Wang (wangtb@buaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61862008, Grant U1636208, and Grant 61902013; in part by the Shijiazhuang Science and Technology Research and Development Plan under Grant 201130351A; and in part by the Beihang Youth Top Talent Support Program under Grant YWF-20-BJ-J-1038.

ABSTRACT In recent years, the application of smartphones, Android operating systems and mobile applications have become more prevalent worldwide. To study the traceability, propagation, and detection of the threats, we perform research on all aspects of the end-to-end environment. With machine learning based on the mobile malware detection algorithms that integrate the dynamic and static research of the identification algorithm, application software samples are collected to study sentences. Through knowledge labeling and knowledge construction, the association relationship of knowledge is extracted to realize the research of knowledge map construction. Flooding is closely correlated with the complexity of the Android mobile version of the kernel and malicious programs. A static dynamic analysis of the mobile malicious program is carried out, and the social network social diagram is constructed to model the propagation of the mobile malicious program. We extended the approach of deriving common malware behavior through graph clustering. On this basis, Android behavior analysis is performed through our virtual machine execution engine. We extend the family characteristics to the concept of DNA race genes. By studying SMS/MMS, Bluetooth, 5G base station networks, metropolitan area networks, social networks, homogeneous communities, telecommunication networks, and application market ecosystem propagation scenarios, we discovered the law of propagation. In addition, we studied the construction of the mobile Internet big data knowledge graph. Quantitative data for the main family chronology of mobile malware are obtained. We conducted detailed research and comprehensive analysis of Android application package (APK) details and behavior, relationship, resource-centric, and syntactic aspects. Furthermore, we summarized the architecture of mobile malware security analysis. We also discuss encryption of malware traffic discrimination. These precise modeling and quantified research results constitute the architecture of mobile malware analysis.

INDEX TERMS Android mobile malware, threat traceability, family chronology, propagation models, detection analysis, infected system environment, knowledge map construction, architecture of mobile malware security analysis.

I. INTRODUCTION

In recent years, with the popularity of smartphones and the rapid development of mobile applications around the world [1], mobile programs have become the main entrance to the Internet, becoming an important part of the storage and end-to-end transmission of massive data. Android-based mobile terminals have quickly occupied the mainstream market because of their openness, completeness, creativity and hardware compatibility. According to the Operating System Market Share Worldwide, the Android operating system has occupied first place in the mobile operating system share in recent years [2].

With the popularity of 4G technology and the development of 5G worldwide, the mobile Internet has been greatly developed [3]. Particularly with the popularity of smartphone

The associate editor coordinating the review of this manuscript and approving it for publication was Debashis De.

terminals and the improvement of their performance [4], various mobile Internet services and applications have been widely accepted by Internet users. However, since mobile smart devices carry a large amount of user information, the applications of mobile smart devices, while convenient for users, are easily abused by malicious programs [5]. Android allows applications from other, unofficial markets to be downloaded and installed. This gives malware developers the opportunity to place repackaged malicious applications in third-party app stores or sites and attack Android devices [1]. There are great security risks [6]–[10]. Due to the open nature of Android devices, mobile device manufacturers are rapidly producing various Android versions worldwide. Android mobile malware is enriched in the cumbersome app store download and installation authorization scenarios [11].

On the other hand, the rapid deployment of cloud-based mobile app stores and websites and the real-time iterative updating of complex versions of mobile malware seem to promote a complex situation in which mobile malware spreads.

Mobile malware threats and their manner of spreading are different from other traditional viruses, zombies, Trojans, worms, and infections, and the forms of spreading, distributing, transmitting, and controlling of massive terminals for malicious conduct activities in the mobile Internet-based mobile malware continue iterating in cloud data centers.

The Android system is not only a mobile phone but is also widely used in Internet of Things (IoT) devices. With the rapid development of the mobile Internet, the iteration speed of mobile application updates is extremely fast, and it is extremely important to study the security of the Android system.

1. The number of Android-based terminals is massive. The total installed base of IoT-connected devices is projected to amount to 75.44 billion worldwide by 2025 [12].

2. The mobile Internet is widely distributed and has been extended to IoT device networks, including infrastructure, automobiles, and smart watches.

3. Compared with the Internet, the mobile Internet has stronger ability to interact with people and computing environments, devices and cyberspace.

4. Even the latest Android security solutions cannot fundamentally solve the security problems of the Android system, and mobile operating system security issues are increasing day by day, posing a threat.

5. Mobile application iteration is fast, and it continues to accelerate iterations according to business needs, forming a complex and dynamic mobile Internet big data network space.

Application software supply chain attack [13] have occurred frequently. These mobile Internet malicious programs can cause serious harm to users, operators, society, and the economy [14]–[17].

Hazards to users: it will hijack a large of user mobile phones, send a large number of advertising short messages or multimedia messages, dial a large number of calls, maliciously order services and illegally connect to the Internet to generate a large amount of network traffic, etc., which causes economic losses to users. Additionally, it will steal user information, such as address books, call records, message content, location information, and account information, resulting in the leakage of user privacy information [11]. In addition, it will destroy all or part of the user's mobile phone system function, maliciously delete user software and completely drain the battery, which interferes with the normal function of mobile phones.

Hazards to operators: it can force infected mobile phones to continuously send spam to the network or dial a specific phone, blocking mobile communication services [18]. In addition, it can give users a feeling of an "accounting error", promote users' distrust in operators, affect business development, result in poor user network perception and affect corporate network brand images.

Harm to the economy and society: it can cause property loss of global users. Moreover, the formation and embarrassment of the black interest chain brings potential instability factors on a global scale and causes leakage of important information and trade secrets held by various countries.

The rapid iteration and spread of mobile malware in various application stores and third-party markets have enriched the global mobile Internet ecosystem with a large number of threats and risks [5]. By protecting the source of threats and means by which threats may occur, studies developed a complete set of mobile Internet application security detection and prevention system. We need to conduct research in all links of the source of the threat, the route of transmission, and the environment of the infection.

In order to explore the theory of mobile malware on detection, traceability, and propagation, the scenarios include SMS/MMS, Bluetooth, 5G base station networks, metropolitan area networks, social networks, homogeneous communities, telecommunication networks, terminal, kernel environments and application market ecosystem scenarios.

II. RESEARCH FRAMEWORK ON MALWARE SECURITY

A. MOBILE APPLICATION SECURITY RESEARCH FRAMEWORK

Application security research includes application source security, application propagation security, application terminal security, and application content security as shown in Fig. 1.

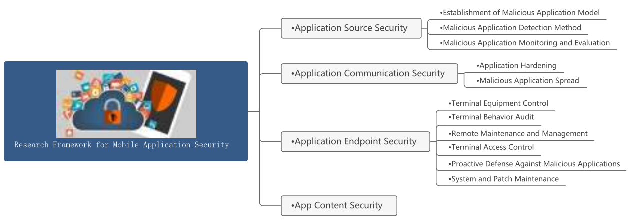

Research Framework for Mobile Application Security
•Application Source Security
•Establishment of Malicious Application Model
•Malicious Application Detection Method
•Malicious Application Monitoring and Evaluation
•Application Communication Security
•Application Hardening
•Malicious Application Spread
•Application Endpoint Security
•Terminal Equipment Control
•Terminal Behavior Audit
•Remote Maintenance and Management
•Terminal Access Control
•Proactive Defense Against Malicious Applications
•System and Patch Maintenance
•App Content Security

FIGURE 1. **Research framework for mobile application security.**

B. DEFINITION AND CLASSIFICATION OF MALWARE

A malicious program is usually a piece of a program written with an intent to attack. These threats can be divided into two categories: threats that require host programs and threats that are independent of each other. The former is basically a program fragment that cannot be independent of an actual application, utility, or system program, and the latter is a self-contained program that can be scheduled and run by the operating system. Fig. 2 shows that malware consists of system architecture of android [19].

In this article, 178,155 real mobile malicious program data were detected under the environment of China Unicom large network, and the mobile Android program data files were decompiled by using a traffic probe. We collected the information about a malicious program and categorized the malicious applications into 14 categories of 48 malicious behavior attributes as shown in Fig. 3. Its classification and definition mainly include the following: information theft, system tampering, system destruction, information destruction, function blocking, malicious deduction, tariff consumption, deception fraud, malicious resource consumption, plug-in bundles, malicious virus/Trojan bundles, malicious promotion, malicious monitoring and remote control. Moreover, according to the behavioral attributes of malicious applications, a fundamental requirement for all behaviors to be determined to be malicious is that the behavior exists without the user's authorization or knowledge.

The purpose of this link is to identify and classify network attack data. The main models are a classification model based on the K-nearest neighbor algorithm (KNN), a classification model based on a decision tree algorithm, a classification model based on a neural network, and a classification model based on the support vector machine (SVM).

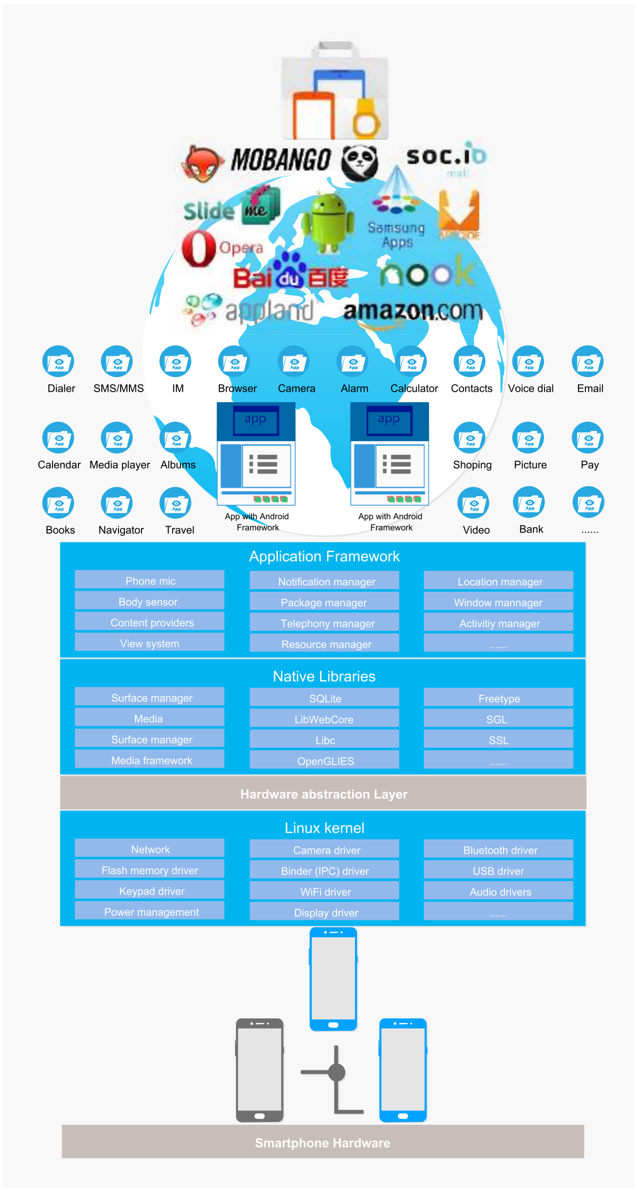


FIGURE 2. Android architecture.

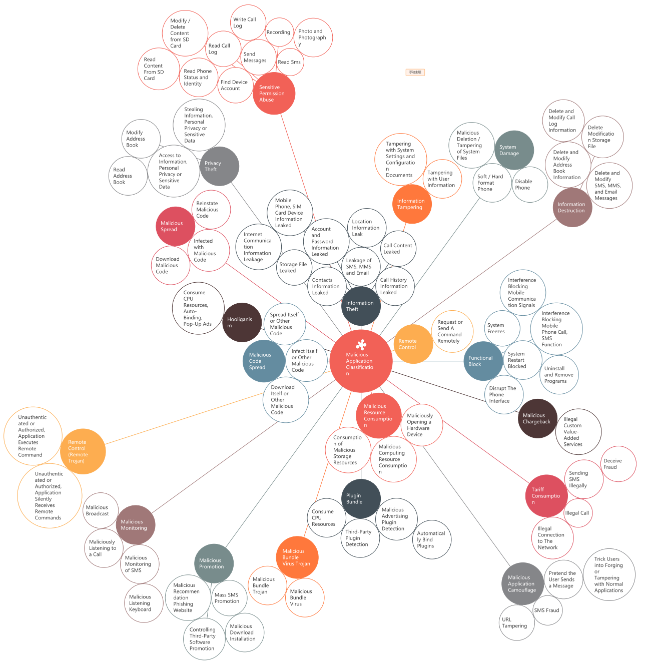

Malicious Application Classification
Sensitive Permission Abuse
Modify / Delete Content from SD Card
Write Call Log
Recording
Photo and Photography
Read Call Log
Send Messages
Read Sms
Read Content From SD Card
Read Phone Status and Identity
Find Device Account
浮动主题
Privacy Theft
Stealing Information, Personal Privacy or Sensitive Data
Modify Address Book
Access to Information, Personal Privacy or Sensitive Data
Read Address Book
Malicious Spread
Reinstate Malicious Code
Infected with Malicious Code
Download Malicious Code
Hooliganism
Consume CPU Resources, Auto-Binding, Pop-Up Ads
Malicious Code Spread
Spread Itself or Other Malicious Code
Infect Itself or Other Malicious Code
Download Itself or Other Malicious Code
Information Theft
Mobile Phone, SIM Card Device Information Leaked
Internet Communication Information Leakage
Account and Password Information Leaked
Location Information Leak
Storage File Leaked
Leakage of SMS, MMS and Email
Call Content Leaked
Contacts Information Leaked
Call History Information Leaked
Information Tampering
Tampering with System Settings and Configuration Documents
Tampering with User Information
System Damage
Malicious Deletion / Tampering of System Files
Soft / Hard Format Phone
Disable Phone
Information Destruction
Delete and Modify Call Log Information
Delete Modification Storage File
Delete and Modify Address Book Information
Delete and Modify SMS, MMS, and Email Messages
Remote Control
Request or Send A Command Remotely
Functional Block
Interference Blocking Mobile Communication Signals
System Freezes
Interference Blocking Mobile Phone Call, SMS Function
System Restart Blocked
Uninstall and Remove Programs
Disrupt The Phone Interface
Malicious Chargeback
Illegal Custom Value-Added Services
Malicious Resource Consumption
Maliciously Opening a Hardware Device
Consumption of Malicious Storage Resources
Malicious Computing Resource Consumption
Tariff Consumption
Deceive Fraud
Sending SMS Illegally
Illegal Call
Illegal Connection to The Network
Plugin Bundle
Consume CPU Resources
Malicious Advertising Plugin Detection
Third-Party Plugin Detection
Automatically Bind Plugins
Malicious Application Camouflage
Pretend the User Sends a Message
Trick Users into Forging or Tampering with Normal Applications
SMS Fraud
URL Tampering
Remote Control (Remote Trojan)
Unauthenticated or Authorized, Application Executes Remote Command
Unauthenticated or Authorized, Application Silently Receives Remote Commands
Malicious Monitoring
Malicious Broadcast
Maliciously Listening to a Call
Malicious Monitoring of SMS
Malicious Listening Keyboard
Malicious Promotion
Malicious Recommendation Phishing Website
Mass SMS Promotion
Controlling Third-Party Software Promotion
Malicious Download Installation
Malicious Bundle Virus Trojan
Malicious Bundle Trojan
Malicious Bundle Virus

FIGURE 3. Malicious application classification.

III. DETECTION OF MOBILE MALICIOUS PROGRAMS

The detection mechanisms are comprehensive methods to comprehensively detect the increasingly rampant mobile malicious programs based on multiple mechanisms.

A. ANDROID KIT

The APK file is an archive file and usually contains the following files and directories:

Assets: Contains application assets that can be retrieved by the AssetManager. It is also called the original resource file (it will not be compiled and no id will be generated). It can be used to store some large resource files of projects, such as pictures, music, fonts, etc.

Lib directory: Contains the compiled code specific to the processor software layer, where almost all shared library (.so) files are stored.

META-INF directory: Contains information about signatures, manifest files, application certificates and resource lists.

Res directory: Contains resources that have not been compiled as .arsc (this resource will not be compiled, but an id will be generated). The main resource files in projects can be stored, such as pictures (*.png, *.jpg), text, etc.

AndroidManifest.xml: Consists of the entry file of the Android application. It describes the components (activities, services, etc.) exposed in the package, their respective implementation classes, various data that can be processed, access permissions, and referenced library files. In addition to declaring Activities, Content Providers, Services, and Intent Receivers in the program, permissions and instrumentation (security control and testing) can also be specified.

class.dex: Android on Dalvik virtual machines running .dex. Thus, the dex file contains all the app code, and the decompiler tool to can be used to obtain the java code.

Resources.arsc: Compiled Android-generated application resource. For a precompiled binary translation resource file, it stores value types of resources as well as other nonasset types of resource-related information.

B. STATIC AND DYNAMIC ANALYSIS

1) STATIC ANALYSIS

Static analysis consists of analyzing the source code or checking the file permissions of the application [20]. Static analysis is aimed at program samples. Without actual program execution, the characteristics, behaviors or defects of the samples are analyzed. The object of static analysis is generally the program source code or object code, such as assembly code or bytecode, etc. [21]. The most useful and commonly used components of APK static analysis contain applications and classes.dex from AndroidManifest.xml, which reports the compiled classes executed by the Dalvik virtual machine.

a: ANDROID STATIC ANALYSIS BASED ON API CALL

The constant update of the Android platform enables the API to call a large number of system hardware and services to interact with the bottom layer of the system. The API includes a wealth of useful semantic values. Decompiled source code can be used to study Android API calls through Natural Language Processing (NLP) and compare them with the system permission list [22]. Static data analysis is performed to extract features from the APK. This entails extracting API calls and their parameters from bytecode, filtering API calls based on their relative frequency of use between benign and malicious applications, and comparing them with permission characteristics [23].

With the development of deep learning, API calls for mobile malware detection and family classification are modeled. Each API method call model [24] can resist the evolution of API calls and orders over time. The high-level semantics of API call graphs require attackers to make more efforts to avoid detection [25]. The API call graph can also represent all possible execution paths that a mobile malicious program can execute at runtime [26].

The main categories of features obtained by static analysis are as follow: binary files, API calls, permissions, commands, resource calls, URLs, regional geographic origin, code size, and rule combinations [27]. Other features are presented through the use of static and dynamic analysis features (such as API calls, permissions, system calls, function calls, privacy usage and leak information graphs). Permission and intent have static characteristics, and API calls have dynamic characteristics. Binary classification of mobile malicious programs based on static features, category classification and family classification of mobile malicious programs based on dynamic features can be realized [28]. Analysis of semantic value, resource characteristics and source code through Natural Language Processing (NLP) can also be used to discover malicious programs in programs [29].

Android is composed of the most basic packages, classes, and functions. Therefore, mobile malicious programs can be classified as API calls from the source code [22]. Sentence scanning can be performed for package-level and class-level features and for function-level tokens in source files. Sentences can be further scanned by a neural network [24].

b: ANDROID STATIC ANALYSIS BASED ON PERMISSIONS

Android applications need certain permissions when calling system APIs for data services and information interaction. The Android system controls the API calls that affect system privacy and security through the permission grant mechanism during installation. However, due to the open source nature of the Android system, the lack of mechanism control, and the blindness and excessiveness of programmers when applying for permissions, applications often overuse permissions, increasing the impact of program vulnerabilities [30]. According to the application permissions in the Android manifest file, malicious applications are more likely to require only one permission, whereas benign applications usually require two to three permissions [31]. Distinguishing between critical permissions and less critical permissions [32] can assess the risk of Android mobile malicious programs [33]. Like API calls, Natural Language Processing(NLP) can identify the need for a given permission [34].

Currently, most popular applications are overprivileged, and with overauthorization of Android applications [35], the growth of the Android platform permission set provides access to new permissions [36]. The required Android permissions are centrally defined in the AndroidManifest.xml file of the Android platform. There are nearly hundreds of valid Android permissions, such as writing text messages, obtaining accounts, reading logs, etc. [29].

The permissions required by the application are used as a common function for APK malicious code detection [31], [32], [37]. The required Android permission set is defined in the AndroidManifest.xml file of the Android platform. Currently there are nearly a thousand valid Android permissions, such as writing short messaging service (SMS), obtaining account, reading log, etc. The static analysis tool is used for Android feature extraction. Permission can be represented by a Boolean vector, which is represented in the AndroidManifest.xml file. Boolean vector, which is represented in the AndroidManifest.xml file.

2) DYNAMIC ANALYSIS

Dynamic analysis is an analysis method based on the actual execution of the program. It is a process of verifying or discovering the nature of the software by running a specific program and obtaining information such as the output or internal state of the program [38]. This technology studies the behavior of apps installed on smartphones, and then checks whether these apps are benign or mobile malicious. Dynamic analysis includes anomaly-based or behavior-based detection technologies that can check the behavior of smartphones after installing the application [20]. After analyzing the behavior of smartphones, these technologies evaluate the status of installed applications as malicious or benign. Together with other mechanisms, behavior analysis has been used in the literature to develop mobile malware detection schemes, including sandboxes [39], behavior detection of abnormal mobile malware based on Internet communication characteristics [40] and analysis of network traffic patterns through machine learning methods [17].

Malicious code dynamic analysis is usually performed in a real/virtual test environment using various conditions to run and activate the malicious code sample. By monitoring all behavior patterns generated by the sample during the running process, the changes are observed in its execution process and data. When all the data in the process of running the program are obtained to make a judgment on whether its behavior is legal [41]–[43], the analysis environment is called a sandbox environment [44]. The sandbox environment is sometimes simply called a sandbox. It is an environment for testing programs whose sources are code that is untrustworthy, destructive, or whose intent cannot be determined. All changes in the sandbox will not cause any loss to the operating system. This technology is widely used by computer technicians. The sandbox is an important environment for observing the running characteristics of samples. Dynamic analysis based on a sandbox environment is usually also used as part of host intrusion detection.

3) HYBRID ANALYSIS

Several authors have combined static and dynamic technologies to overcome the weaknesses of these technologies and design reliable technologies for mobile malware detection on smartphones. Articles using this technique only cover a few topics, such as enhancing the performance of traditional detection software [45], analyzing Android applications, and focusing on automatic static and dynamic mobile malware analysis [46] machine learning technology is used in application classification and helps to assess the maliciousness of previously unknown Android applications [47] and behavioral signature classification technology is also used [48].

C. MALWARE RESEARCH SENTENCED MECHANISM

1) MALWARE SAMPLE COLLECTION RESEARCH

By controlling the spread of the malware network, sample collection, research and plugging sentences, a malware prevention system and key scientific issues are researched in order to protect the safety of the mobile Internet. The download source, download channels and terminal runtime environment of an application are monitored through the network side mobile technology, and the network traffic is monitored to analyze and identify the malware with the propagation network. Through the flow control algorithm and the domain name service (DNS) intercept parsing, our research blocks the spread of malware, as shown in Fig. 4.

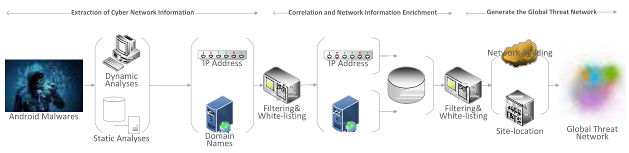


FIGURE 4. **Malware research sentenced mechanism approach overview.**

System call extractions, anomaly detection [4], control flow graph structure, static taint analysis [49], static code analysis, anomaly detection [50] and other analytical techniques are used to classify the mobile malware program.

The program will report suspicious samples, monitor log management, and monitor and dispose of the strategy by the malicious program feature library, such that suspected malicious programs and features are identified by filtering using rule-based maintenance and management. Malware analysis research of sentences, the CNCERT subsystem interfaces, region interfaces, the overall system management and other architectures are included, as shown in Fig. 5.

We deployed monitoring programs on the gateway to monitor the download and transmission behavior of the Android platform applications transmitted on the network. Management and control platform malware were deployed in the network side and centralized to collect, judge and block the malicious software samples on the Internet as shown in Fig. 6.

The malware was decompiled by static analysis tools, and the malicious behavior points were output according to sensitive APIs and sensitive strings. The dynamic analysis tool can dynamically load and run suspected program software through the sandbox and monitor its operation. Static and dynamic analyses are combined with manual analysis to compare and determine the suspected malicious program and the known malicious program samples.

By splitting the gateway network traffic monitoring, client reporting and user application store applications, we obtained

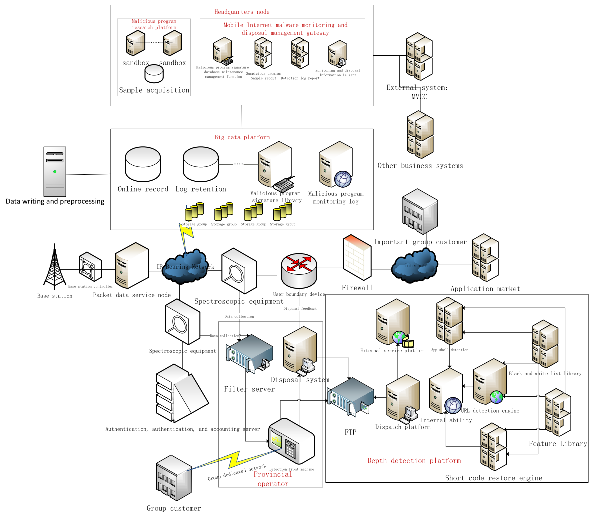


FIGURE 5. Network topology design.

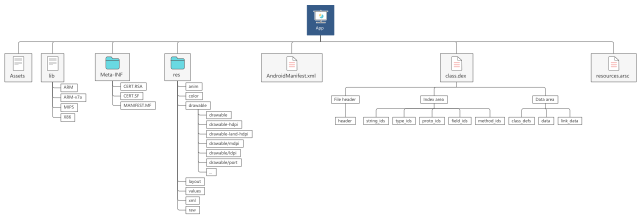


FIGURE 6. Code excerpts from a fake inst malware sample: the complex and Heavy use of reflection can thwart static code- based feature extraction.

software samples and use static and dynamic detection technology to research sentences. Through the centralized control system, the main control URL, the domain name and the IP address of the malicious program are blocked on the gateway and the DNS, and the download channel of the malicious program is blocked.

By analyzing the user's online logs, complaint information, and network traffic [51], the malware network behavior rules, the whole network malware download, the master URL and the malware MD5 signature were obtained. The feature library is used to scan the terminal to detect maliciousness on mobile phones. The malware network centralized control platform utilizes the users' access logs for accurate judgment.

2) RESEARCH JUDGMENT

Through the integration of dynamic and static research identification algorithm, application software samples were collected to study sentences. By scanning behaviors of applications, such as license application, system monitor API or short/multimedia messaging service (MMS) messaging, remote connection URL and other acts, the results of the research sentence are given. The network logs are combined to determine whether the user's mobile phone has accessed the malware download URL or the main control URL and whether there is any keyword information that is given away, such as the IMSI/IMEI/terminal model of the user to determine whether the mobile terminal is poisoned.

Data analysis is performed by feature extraction, analysis and large-depth data associated with the mobile machine learning to detect malware malicious content detection, risk detection, piracy detection, detection shell, and reduction of the shortcode that detects a malicious URL.

3) BLOCKING RESEARCH

The malware's blocking flow control system and DNS prevent users from accessing the malware download URL and the master URL to protect the network.

The malicious URL is sent to the front end of the flow control system to form a blacklist. After the deep packet inspection (DPI) device resolves the intranet and Internet traffic and obtains the URL or domain name request from the user, if the blacklist is hit, the preserver sends a reset packet or a DNS reset page packet to the user. Returning results or normal DNS resolution results are slower [51] and will be discarded by the user host, blocking access to malicious URLs [52]. The malware IP address blacklist is sent to the firewall for blocking.

D. FAMILY CHRONOLOGY AND ARCHITECTURE OF MOBILE MALWARE

1) FAMILY CHRONOLOGY OF MOBILE MALWARE

According to the research sentence, we obtained the main Family Chronology of mobile malware (Fig. 7). Floatgame, Backstage and GenericBludger are active (Table 1) and can provide guidance for the study of the species and the number of active mobile malicious programs.

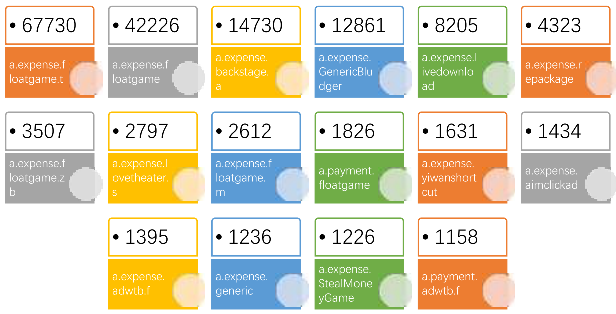


FIGURE 7. The main family chronology for mobile malware.

2) ARCHITECTURE OF MOBILE MALWARE SECURITY ANALYSIS

Through long-term research on our real mobile malware data of 178,155 instances, we have conducted detailed research and comprehensive analysis from APK details, behavior, relationship, resource centric, and syntactic aspects.

The summary of the architecture of mobile malware security analysis is shown in Fig. 8. The contents of malware were decomposed one by one, and the results obtained were as comprehensive as possible, achieving independent and

TABLE 1. **Malware family chronology.**

MD5	Malware Family Chronology	Amount
ef5ec93603c7aa8e0f8805dece983035	a.expense.floatgame.t	67730
ef5ec93603c7aa8e0f8805dece983035	a.expense.floatgame	42226
469b3ccf9c2b69cdeb5d73f1dba0dc48	a.expense.backstage.a	14730
c40929f049a4adbe7dd7d55102a0fe68	a.expense.GenericBludger	12861
f3a35b526e3f5928daa3b1f69bace60e	a.expense.livedownload	8205
d738437b52df1d2f7e616826963a5439	a.expense.repackage	4323
a43c6bcc8d0d97d30d23a1636007c8ff	a.expense.floatgame.zb	3507
ce1b9442a47161c3aca7ea5b6d30a1ce	a.expense.lovetheater.s	2797
21b930cbb78e9db16a985595a9477281	a.expense.floatgame.m	2612
4c8a1443ad279144c3319472b48cf08a	a.payment.floatgame	1826
bc8d226ee8ef0ddbad3b53e15dbd8780	a.expense.yiwanshortcut	1631
f345a353e4628937394965ca27590bb9	a.expense.aimclickad	1434
f52f0ec36d9bbda47b67da458f1bc517	a.expense.adwtb.f	1395
99a3a04effb69f4576a0d4bbbf476a85	a.expense.generic	1236
1647e1b291695331ef90716e788c62ed	a.expense.StealMoneyGame	1226
444ad7aca27deb1734025017b5b708dc	a.payment.adwtb.f	1158
. . . .	. . . .	. . . .
Total		178155

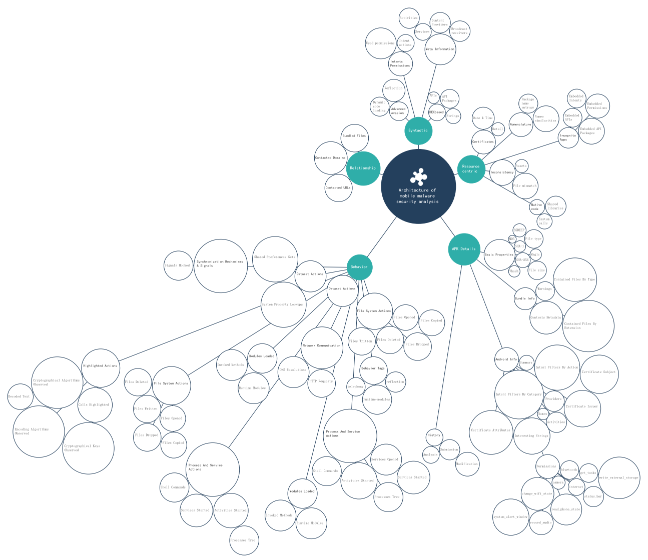


FIGURE 8. **Architecture of mobile malware security analysis.**

complete subitems that complemented each other. These precise modeling and quantified research results constitute the architecture of mobile malware security analysis.

E. ANDROID MALWARE BEHAVIOR DETECTION THROUGH DERIVING COMMON GRAPH CLUSTERING

Younghee Park proposed deriving common malware behavior through graph clustering [53]. We expanded this to the area of Android mobile malware detection in the Linux kernel. To realize the detection of mobile malicious programs, a mobile malicious program data set is first selected to obtain a kernel Android behavioral object diagram representing its behavior. Then, the same mobile malware family is clustered into a graph (weighted universal Android behavior GRAPH). This shows the behavior of all members of the family. For a particular classification family of mobile malware, this approach reliably generates single graphics. Finally, the cluster single plot is achieved at low expense to detect the movement of the malicious programs' unknown sample. Through the construction of a ''virtual sandbox'' and Android behavior analysis, we conducted research on the feature detection of mobile malware families.

1) KERNEL OBJECT ANDROID BEHAVIORAL GRAPH GENERATION

The graphic representing the *behavior of the* Android malicious program instance is the *kernel object Android behavior graphic*. The Android kernel object is a memory block in the Android kernel. The storage block is a data structure whose members maintain information about the object. Android's Linux kernel control includes hardware management, process management, a driver model, a network stack, security and so on.

The kernel object Android behavioral graph is constructed based on the information collected from the running mobile malware. The mobile malware is executed in a virtual environment. The virtual environment is used to capture the parameters of each system call executed. Based on the intercepted system call trace, this method identifies the relationship between Android kernel objects.

The kernel object Android behavioral graph (KOABG) is a weighted directed graph described by $g = (V, E, \mu, \lambda)$.

V is a set of vertices (vertices v), and different vertices represent corresponding Android kernel objects. The attribute of the vertex indicates the specific name of the Android kernel object.

$\mathrm{E} \subseteq \mathrm{V} \times \mathrm{V}$ is a set of edges e, which represents the dependency between two Android kernel objects. The dependency is represented by the handle type and the handle value. The object handle indicates an identifier to indicate system resources inside the Android kernel, and the identifier is allocated by the application.

μ: $\mathrm{E} \rightarrow \mathrm{N}^{+}$ is a function that assigns positive weights to edges (initially, $\mu(\mathrm{e}) = 1\ \forall \mathrm{e} \in \mathrm{E}$)).

λ: V→N^+ is a function that assigns positive weights to vertices(initially, λ(v) = 1 ∀v ∈ V).

Android kernel objects are data structures that represent system resources (such as files or threads).

Regarding the Android program initialization time, Android will start running a supported component Linux process and a main thread. The main thread is responsible mainly for processing UI related events and distributing related events to corresponding components for processing. Therefore, the main thread is usually called the UI thread.

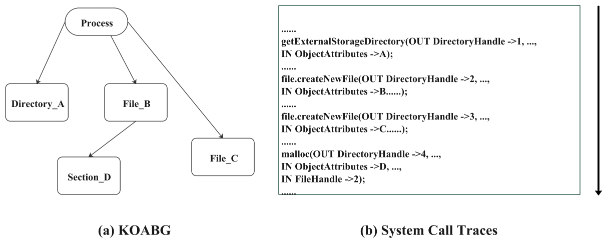
The permissions on the Linux file system represent the access permissions of the corresponding users or user groups and others to this file and are completely irrelevant to the permissions that this file has when it runs. For example, the system user has the read, write and execute permissions for this file, the system group users have the read and execute permissions for this file, and others have only the execute permission for this file. The Android default application does not have any permissions to operate other applications or system-related features, and applications need to explicitly apply for corresponding permissions when performing certain operations. Ordinary .apk programs are run at the non-root, nonsystem level.

When accessing an Android kernel object through a system call, the handle and object attributes of the object are the parameters of the call. The object name is identified by the s*tructattributes* parameter in the intercepted system call.

Every Android kernel object has a creator function and a destructor. For example, *CreateNewFile* creates a new file or directory, or opens an existing file, directory, device, or volume, and *CloseHandle* terminates the Android kernel object with the handle created by *CreateNewFile*. In addition, *Process.start* will create a new process and its main thread. The object name is extracted by reading the structattributes parameter value. When *CloseHandle* or *Process.kill* is called with this handle, the Android kernel object can no longer be accessed.

After the Android kernel object is created through the system call, its handle parameter is passed to other system calls. Other Android kernel objects associated with the system call will access the Android kernel object. The dependency between the process and thread objects is represented by the edges between corresponding vertices in KOABG. The weight of all edges in KOABG is 1.

System call tracks kernel objects in Android behavior diagrams. KOABG starts with an Android kernel object Process. Each vertex in the figure represents an Android kernel object created during program execution. The object name is displayed in the vertex. The generated object is implicitly related to the initial process. For example, the *Section_D* object is created from the *File_B* object by calling the CreateNewFile system call, thereby adding the side from *File_B* to *Section_D*. Since multiple Android kernel handle values can indicate the same Android kernel object at runtime, the handle value is not used to identify vertices (Fig. 9).



```
......
getExternalStorageDirectory(OUT DirectoryHandle ->1, ...,
IN ObjectAttributes ->A);
......
file.createNewFile(OUT DirectoryHandle ->2, ...,
IN ObjectAttributes ->B......);
......
file.createNewFile(OUT DirectoryHandle ->3, ...,
IN ObjectAttributes ->C......);
......
malloc(OUT DirectoryHandle ->4, ...,
IN ObjectAttributes ->D, ...,
IN FileHandle ->2);
......
```

FIGURE 9. The obtained system call tracks the kernel object Android behavior graph.(Numbers represent handle values, and letters represent Android kernel object names.).

2) "VIRTUAL SANDBOX" CONSTRUCTION

By constructing a "virtual sandbox" to simulate the real system environment, real-time monitoring and dynamic analysis, malicious program behavior can be implemented to accurately detect and kill unknown deformed viruses.

Through the virtual sandbox design system environment simulation, a large number of API calls and executions are simulated, covering most of the quantitative core mechanisms of kernel object behavior, including but not limited to:

(1) File system: Core system files are stored and files created and modified during the execution of processes in the virtual sandbox can be tracked. When the antivirus engine scans concurrently, different virtual processes have isolated file systems.

(2) Process, thread, synchronization object, scheduling, clock: The virtual sandbox simulates complete kernel object behavior, and schedules threads according to the virtual clock or scheduling logic.

(3) Control system: The virtual sandbox also implements a complex and massive control system as well as a variety of controls. The family obfuscator uses the kernel object behavior and various messages and window characteristics of the control as a means of an antivirtual machine.

(4) High virtual execution efficiency: The antivirus engine can complete the scan of the object to be scanned in a relatively short time (millisecond level).

(5) We have designed a complete operating system environment simulation, including the file system, process, thread, scheduling logic, clock, and synchronization object.

(6) Capturing and recording the behavior of the program during virtual execution: The virtual sandbox can capture and record the kernel object behavior generated when the program is virtually executed in the sandbox. Such recording can be at the system call level or at a higher abstraction.

We perform research to achieve precise detection and killing of unknown viruses, deformed viruses and other viruses through the leading "virtual sandbox" and behavior analysis, real-time monitoring and dynamic analysis of viruses by simulating the real system environment. Even when the network is not connected to achieve virus detection and killing capabilities, its deployment can achieve rapid, lightweight, and accurate killing.

3) ANDROID BEHAVIOR ANALYSIS

Examples of the mobile malware cluster group operation of KOABG aggregated into a single behavioral GRAPH achieve mobile malware identification. Thus, $\mathbf{G} = \{g_1, \ldots, g_m\}$ is a set of *m*th Android collection kernel objects of the Android behavior GRAPH according to the *m*th construction GRAPH call tracking system behavior collected during execution of binary files, where binary files are those classified as the same mobile malware family.

Definition of G $= \{g_1, g_2, \ldots, g_m\}$ is a set of weighted directed graphs, and the **G** weighted common supergraph is the **G** weighted directed graph $g = (V, E, \alpha, \beta, \lambda, \omega)$. Therefore, there is *a* subgraph isomorphism from g_i to $g \forall i \in \{1, \ldots, m\}$.

Assuming that any other weighted common mother graph **G** has fewer nodes than g, *g* is the weighted lowest common mother graph **G** of a set of graphs, and represents *WMinCS* (**G**).

Let $f_i : V_i \rightarrow V$ be a subgraph *G* of the isomorphism between g_i and *g*. Let *(u, v)* be a pair of vertices of V. Then each side right weight e = (u, v) is *k* / *m*, assuming accurate presence of the *k*-th different isomorphic $f_j(e_{il}) = (f_j(u_i), f_j(v_l))$, Make $f_j(e_{il}) = e$. The weight of the vertex only needs to be set to 1.

Behavior graphs can be clustered [54]. A *Graph isomorphism* is the bijective mapping *f* between the vertices *G* and *H of the* two graphs so that the *G* of the vertices *u* and *v* are adjacent to *G* if and only if *f(u)* and *f*(*v*) are adjacent to *H*, *the subgraph isomorphism between two graphs G and H* is an isomorphism between the *G* and a subgraph *H* [55].

The weighted lowest common mother map (WMinCS) is based on the clustering method [56]. Two graphs *G* and *H* of the *maximum common subgraph (mcs)* are expressed as *mcs*(*G*, *H*), a *G* sub GRAPH. The weighted largest common subgraph(*wmcs*) of a group of graphs **G** is a *subgraph* of the largest size, and there is an isomorphic subgraph in all graphs.

The *minimum nominal superscript* (*MinCS*) of *G* and *H* is defined as Equation (1).

$$MCS(G,H) = msc(G,H) \cup (G - msc(G,H)) \cup (H - msc(G,H)) \quad (1)$$

*Both G-mcs(G,H)*and *H-mcs*(*G*,*H*) are graphs obtained by removing the *mcs of G* and *H* from *G* and *H* respectively.

The weighted smallest common subgraph can cluster a group of kernel android behavioral object graphs denoted as **G.** Assume that $g = WMinCS(\mathbf{G})$ has been constructed for a group of mobile malware instances. According to the parameter Θ, a *weighted general behavior graph* (*WCABG*) of **G** is constructed. $WCABG_\Theta$(G) is a subgraph *g that* contains all edges *g that* have a weight greater than Θ, and the vertices connected by these edges. $WCABG_\Theta$(G) is simply called *WCABG*.

WCABG summarizes the important behavioral attributes of all mobile malware instances in the same family. A weighted general behavior graph (WCABG) can be constructed from a set of KOABGs. WMinCS computing is an NP-complete problem due to the complexity of the problem. Thus, an algorithm to approximate a set of graphical **G** of WMinCS can be used; WCABG generates an approximation algorithm (an Approximate algorithm for WCABG Generation).

The weighted maximum common subgraph(*wmcs*) of a pair of weighted graphs can be derived to approximate the WMinCS algorithm [57], [58]. GRAPH behavior between two weighting WMinCS by Equation (2) is generated.

$$WMinCS(g_i, g_j) = (g_i - wmcs(g_i, g_j)) \cup (g_j - wmcs(g_i, g_j)) \cup wmcs(g_i, g_j) \quad (2)$$

The approximate WMinCS equation between two graphs is extended to a set of graphs G $= \{g_1, g_2, \ldots, g_m\}$. In generating a set of random behavior GRAPH 's rear, *WMinCS*$(g, g_i)(2 \le i \le m)$ is iterated *m*-1 times for calculating approximation *WMinCS*(**G**). The weights of *WMinCS*(g, g_j) are calculated repeatedly for the edges defined in the Equation (2). After clustering all graphs, **G**'s *wmcs* is shared by all important behavior graphs, and the subgraph is called *HotPath* (in WCABG there is one called HotPath critical path, which appears in all members of the family. The path has a maximum weight of 1, the edge composition).

Finally, the approximation derived WMinCS for the set is simplified into a weighted general behavior graph ($WCABG_\Theta$(G)), at a predefined threshold (Θ). The result is that the weighted common Android behavioral graph (WCABG) is generated as a representative graph of each mobile malware family. WCABG summarizes all the attributes shared by most mobile malicious program instances and the edges or dependencies that appear in specific parts of the mobile malicious program binary files, and only summarizes the *HotPath* that represents the behavior displayed by the execution of all mobile malicious programs.

The kernel object Android behavior is a research result based on the malware family. The data set used is a wide range of binary files, even malicious samples that cannot be classified. At the same time these graphs are derived using different methods such as system calls, user input and pollution analysis. Behavior graphs can be widely used in various malware research to overcome the limitations of signature-based malware detection.

On this basis, Android behavior analysis is performed. The virtual sandbox behavior analysis is realized through the virtual machine execution engine, which can track and record the behavior of the program running in it. The antivirus engine records the object behavior through the object behavior and can evaluate the maliciousness of the program through the heuristic analysis algorithm.

The virtualized execution engine is responsible for implementing the instruction set simulation, and the virtual sandbox implements two execution engines.

a: VIRTUALIZATION EXECUTION ENGINE

Through virtualization technology, an independent address space is divided for the target code, and private time slices are allocated to the object behavior by taking over interrupts and exceptions, so that the target code can be executed in a controlled manner. The execution efficiency can be almost equal to that of the real machine, and only the execution environment switching brings a small amount of overhead.

b: DYNAMIC TRANSLATION EXECUTION ENGINE

The technology that analyzes the target code and controllably translates the execution of the local object behavior is called dynamic translation. The virtual sandbox realizes fine-grained control of the object behavior through the dynamic translation execution engine.

Preliminary evaluation of object behavior functions and library files is performed through heuristic analysis algorithms. For functions and libraries that are not harmful to the system, shallow execution is performed through the virtualized execution engine, which is fast and efficient. Dynamic translation the object behavior function and library files, fine-grained analysis, and tracking the sequence of behaviors determine whether it is malicious to the system.

4) MOBILE MALWARE FAMILY BEHAVIOR DETECTION

Based on building the kernel objects Android behavior diagram (or KOABG), a suspicious program can be intercepted in the virtual environment mobile malware in the system call parameters and execution. Then, the KOABG of the suspicious program is compared with the WCABG of the previously identified mobile malware family.

Let the KOABG of the suspicious program instance be denoted as g_{new}. Suppose this KOABG is compared with WCABG for a family of mobile malware. Equation (3) in $\delta(g_{new}, WCABG)$, is calculated as g_{new} and the WCABG similarity between, where | g | represented in GRAPH g is the number of sides. $W(e_j, E_i)$ means that the WCABG of the edge E_i and the g_{new} of the edge e_j are matched by the weight w.

$$\delta\left(g_{new}, WCBG\right) = \frac{\sum\limits_{<e_j,E_i>\in w} W\left(e_j,E_i\right)}{\min(|g_{new}|,|WCBG|)} \tag{3}$$

The weights of all these matching edges add up. Then, divide the sum by the minimum value of the edges in the WCABG or KOABG.

As shown in Equation (4), make a judgment D about mobile malicious programs.

$$D = \left(\delta\left(g_{new}, WCBG\right) \geq \gamma\right) \wedge \left(HotPath \subseteq g_{new}\right) \tag{4}$$

To determine that the suspicious instance is mobile malware, the similarity between KOABG and WCABG must exceed the predefined threshold γ, and KOABG must include the HotPath of the generated WCABG [53]. In summary, given the modular design of the Android program, the calls of the API authority rules are relatively close, so that the behavior of the method is determined based on GRAPH. It can be effectively used in the determination of the family of mobile malware.

Based on the idea of deriving common Android malware behavior through graph clustering, we have extended the study of common behaviors (also known as ‘‘genes’’) of virus families.

In summary, the modular design of the Android program, call the API authority rules and of relatively closed, so that the behavior of the method is determined based on GRAPH. It can be effectively used in the determination of the family of mobile malware.

Based on the idea of Deriving common Android malware behavior through graph clustering, we have extended the study of common behaviors (also known as ‘‘genes’’) of virus families.

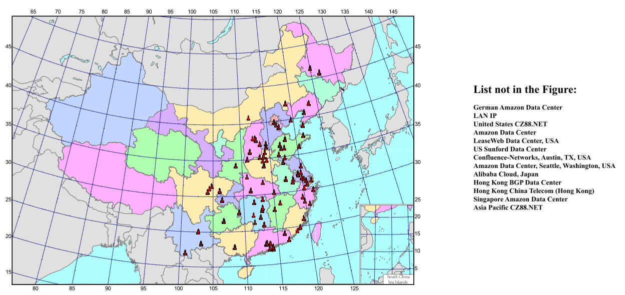
At the same time, through KOABG, the essential characteristics of malicious code can be deeply analyzed, and the threat information of the static code and the kernel object Android behavioral graph generation of dynamic code can be perceived in real time. Through technologies such as WCABG, high feature multiplexing, malicious code DNA identification, and malicious code DNA fragment recombination, it has strong application value for family malware of the same behavior type. The above method can reduce redundant data in the feature library, greatly reduce the occupation of terminal resources, effectively save terminal performance consumption, and does not affect the normal process.

We extend the family characteristics to the concept of DNA race genes.

Traditionally, the detailed feature information of viruses is recorded. One feature corresponds to one virus, and there is a one-to-one relationship. Viruses that are not in the feature database cannot be checked and killed. Virus races have common behavioral information. We can extract core behaviors as race genes to form a gene recognition library. First, the feature library is smaller, which reduces terminal resource usage. At the same time, because of the behavior of racial genes, new types of variants can be identified and killed to solve the problem of variants.

The antimalware technology we developed adopts the concept of family gene recognition to reduce redundant data in the signature database and accurately identifies and kills various viruses such as Trojan horses and worms. It identifies different threats by optimizing the signature database and accurately identifies and kills all known virus variants. As a behavior-driven engine, it is based on the core gene behavior of the virus race, which is different from the behavior matching method of traditional engines, through the system call parameters and execution of core gene behavior information, high multiplexing, malicious code DNA identification, and malicious code DNA fragment reorganization and other technologies to efficiently detect malicious threats.

In summary, the developed the gene recognition technology occupies less terminal resources and solves the problem of variants. Through the construction of a ‘‘virtual sandbox’’



List not in the Figure:

German Amazon Data Center
LAN IP
United States CZ88.NET
Amazon Data Center
LeaseWeb Data Center, USA
US Sunford Data Center
Confluence-Networks, Austin, TX, USA
Amazon Data Center, Seattle, Washington, USA
Alibaba Cloud, Japan
Hong Kong BGP Data Center
Hong Kong China Telecom (Hong Kong)
Singapore Amazon Data Center
Asia Pacific CZ88.NET

FIGURE 10. Geographic breakdown of malware source: CHIUNICOM.

and Android behavior analysis, we conducted the research on the behavior detection of mobile malware families.

F. MACHINE LEARNING BASED ON MOBILE MALWARE DETECTION ALGORITHM

Malicious programs in the sample space of the Bayesian classification may be classified:

$$P\left(C_m|a_1,a_2,\ldots,a_n\right)=\frac{P\left(C_m\right)*P\left(a_1,a_2,\ldots,a_n|C_m\right)}{P\left(a_1,a_2,\ldots,a_n\right)} \tag{5}$$

C_m represents target attribute m. The value of the molecule, which is the largest, according to the maximal posterior hypothesis.

$$\begin{aligned} C_{map} &= \underset{C_m\in C}{arg\ max} P\left(C_m\mid a_1,a_2,\ldots,a_n\right) \\ &= \underset{C_m\in C}{arg\ max} P(a_1,a_2,\ldots,a_n\ |C_m)*P(C_m) \end{aligned} \tag{6}$$

$P(C_m)$ is the frequency of each category in the target sample.

When the value of n is large, it needs to be implemented in a very large sample space. We make a simple hypothesis in Naïve Bayes, which is independent of each other. Naive Bayes assumes conditional independence of conditional probability distribution, that is, it ignores the relationship between features and makes each feature a separate assumption.

$$\begin{aligned} C_{NB} &= \underset{C_m\in C}{arg\ max}\ P\left(a_1,a_2,\ldots,a_n|C_m\right) \\ &= \underset{C_m\in C}{arg\ max} \prod_{i=1}^{n} P\left(a_i|C_j\right) \end{aligned} \tag{7}$$

Estimate the related $P\left(a_i|C_j\right)$ attribute probabilities, using this method to calculate probabilistic detection of mobile malicious programs.

Herein, by China Unicom core router and an outlet port optical device deployment points, by con verging the shunt flow probe detects the movement program.

IV. TRACEABILITY OF MOBILE MALICIOUS PROGRAMS

This study detected 178,155 real mobile malicious programs in China Unicom large network environment. Through the flow probe, we decompiled Android data files of China Mobile and obtained detailed Guiyang (China ''Data Center Capital'') infected mobile malware data. Along with the IDC (Internet Data Center) for rapid deployment on a global scale and comprehensive coverage of triple play, mobile malware sources (Fig. 10) are located in the cloud based IDC data centers and emerging wide network server (Table 2). In particular, the new radio and television networks are low cost and grant approval of the record mechanism so that the mobile malware is concentrated in them and frequently downloaded on the large number of terminals. The star-shaped topology network of the massive users is the propagation network of mobile malicious programs, and it has the characteristics of real-time rapid iterative changes with factors such as usage habits, product contact push and program popularity.

TABLE 2. Top Android malware related to the distribution.

Network Physical Location	Hits
Mianyang City, Sichuan Province	116001
Guiyang City, Guizhou Province Unicom	14026
Xintong City, Henan Province	7511
Dongguan City, Guangdong Province	4191
Ningbo Unicom, Zhejiang Province	3729
Fuzhou Unicom, Fujian Province	3589
Alibaba Cloud, Japan	2267
Yueyang City, Hunan Province Unicom	1967
Tianjin Unicom	1891
Xiangtan City, Hunan Province Unicom	1805
LAN IP	1405
Tongtong, Tongren District, Guizhou Province	1360
Shenzhen Unicom, Guangdong Province	1286
Xinyu City, Jiangxi Province	1237
Sichuan Province	1167
Unicom, Suzhou, Jiangsu Province	955
Chongqing Unicom	955
Daqing City, Heilongjiang Province,NetScience Technology Unicom CDN node	786
Harbin City, Heilongjiang Province	686
Zhengzhou City, Henan Province	645
Tai'an City, Shandong Province	555
Qingdao, Shandong	543
....	......
Total	178155

We deployed the probe to decompile the traffic through the metropolitan area network exit aggregation device. It was detected through the static, dynamic and machine learning mechanisms, and then we traced the source according to the traffic detailed list, DNS detailed list, big data analysis and knowledge map.

Flooding is closely correlated with the complexity of the Android mobile version of the kernel and malicious programs. Through our research, 49.21% (87,674/178,155) of mobile devices are running suspected infections in older versions of Android (Android 7.0 or less) in its application ecosystem of widespread vulnerabilities. From Table 3, we found that the Redmi base band version of Android 6.0 and 7.0 is infected the most with mobile malicious programs.

Global malware threats traceability, process, and pipeline models are given through our research as shown in Fig. 11 and explained below:

TABLE 3. **Mobile malware and android version and kernel relevance.**

System Version	Small Version Code	Hits
Android6.0.1	Linux; U; Android 6.0.1; Redmi 4A BuildMMB29M	19213
Android7.1.2	Linux; U; Android 7.1.2; Redmi 4X BuildN2G47H	6641
Android6.0.1	Linux; U; Android 6.0.1; Redmi 3S BuildMMB29M	6592
Android7.1.2	Linux; U; Android 7.1.2; Redmi Note 5A BuildN2G47H	5799
Android7.0	Linux; U; Android 7.0; Redmi Note 4X BuildNRD90M	5654
Android6.0	Linux; U; Android 6.0; Redmi Note 4 BuildMRA58K	5357
Android6.0.1	Linux; U; Android 6.0.1; Redmi 4X BuildMMB29M	5003
Android7.1.2	Linux; U; Android 7.1.2; MI 5X BuildN2G47H	4672
Android4.4.4	Linux; U; Android 4.4.4; HM NOTE 1S BuildKTU84P	4658
Android6.0.1	Linux; U; Android 6.0.1; MI 4LTE BuildMMB29M	4477
Android4.4.4	Linux; U; Android 4.4.4; HM 2A BuildKTU84Q	4442
Android6.0.1	Linux; U; Android 6.0.1; Redmi 4 BuildMMB29M	4153
Android7.1.1	Linux; U; Android 7.1.1; MI MAX 2 BuildNMF26F	3375
Android4.4.4	Linux; U; Android 4.4.4; 2014811 BuildKTU84P	3268
Android5.0.2	Linux; U; Android 5.0.2; Redmi Note 2 BuildLRX22G	3060
Android6.0.1	Linux; U; Android 6.0.1; Redmi Note 3 BuildMMB29M	2992
Android5.0.2	Linux; U; Android 5.0.2; Redmi Note 3 BuildLRX22G	2853
Android5.1.1	Linux; U; Android 5.1.1; Redmi 3 BuildLMY47V	2814
Android7.0	Linux; U; Android 7.0; MI 5 BuildNRD90M	2791
Android4.4.4	Linux; U; Android 4.4.4; HM NOTE 1LTE BuildKTU84P	2773
Android6.0	Linux; U; Android 6.0; Redmi Note 4X BuildMRA58K	2732
Android7.1.2	Linux; U; Android 7.1.2; Redmi 5 Plus BuildN2G47H	2437
Android7.0	Linux; U; Android 7.0; Mi-4c BuildNRD90M	2134
Android5.1.1	Linux; U; Android 5.1.1; 2014813 BuildLMY47V	2084
Android7.1.1	Linux; U; Android 7.1.1; MI 6 BuildNMF26X	1761
Android7.1.2	Linux; U; Android 7.1.2; Redmi 5A BuildN2G47H	1724
Android5.1.1	Linux; U; Android 5.1.1; Redmi Note 3 BuildLMY47V	1670
Android6.0.1	Linux; U; Android 6.0.1; MI NOTE LTE BuildMMB29M	1627
Android4.4.2	Linux; U; Android 4.4.2; 2014501 BuildKOT49H	1457
Android6.0.1	Linux; U; Android 6.0.1; Redmi Note 4X BuildMMB29M	1451
Android4.4.4	Linux; U; Android 4.4.4; HM 1S BuildKTU84P	1407
Android6.0.1	Linux; U; Android 6.0.1; MI MAX BuildMMB29M	1383
Android6.0.1	Linux; U; Android 6.0.1; MI 4W BuildMMB29M	1139
Android4.4.4	Linux; U; Android 4.4.4; 2014813 BuildKTU84P	1069
....	....	....
Total		178155

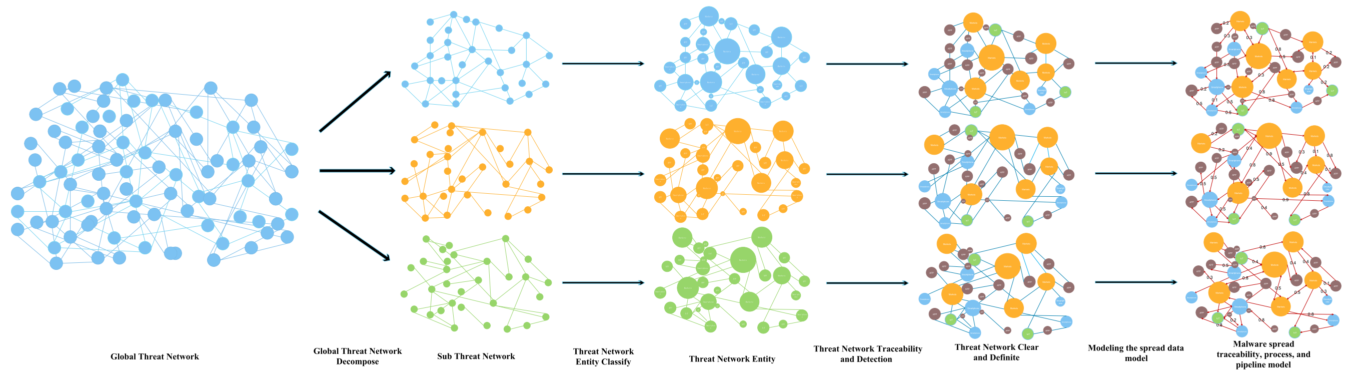

Global Threat Network
Global Threat Network Decompose
Sub Threat Network
Threat Network Entity Classify
Threat Network Entity
Threat Network Traceability and Detection
Threat Network Clear and Definite
Modeling the spread data model
Malware spread traceability, process, and pipeline model

FIGURE 11. **Global malware threats traceability, process, and pipeline model.**

1. First-level stage:

The global malware network first distinguishes small networks through geopolitics, language and cultural regions, and the mobile Internet ecosystem (mobile Internet ecological stores, applications, global networks, and terminal computing environments).

2. Second-level stage:

The mobile Internet exits are detected through our mobile malicious program system, as well as DNS service systems, detailed large data flow bills, and terminal application ecosystems, to identify threat network entities and to identify node entities (IP address or domain name) of similar entities to conduct systematic research. We perform data mining for massive heterogeneous data on data services such as data cleaning, transformation, formatting, and deduplication of structured, semistructured, and unstructured big data as well as computing services such as retrieval, association, classification, and statistics.

3. The third stage:

The end-to-end research of mobile malicious program propagation is carried out to define threat networks.

Data and calculations are processed through big data analysis. Develop a data engine to store and manage various types of data resources related to cyberspace, including communication relationships, communication logs metadata, raw stream data, disposal information management, and sample data management. Computational analysis mining provides calculations such as element correlation, back analysis, communication analysis, feature retrieval, and statistical analysis to carry out threat analysis. In the end, the IP threat network, domain threat network, and network information threat network are clearly defined and defined.

4. Fourth level stage:

Trace the source, process and traffic channel model of the mobile malicious prestige network for data modeling and study its model characteristics and propagation rules.

Collision analysis and correlation analysis is performed between heterogeneous systems of the global mobile Internet, and an audit-based model is devised to iteratively mine the mining results. At the same time, it conducts deep mining of data based on business scenarios, automated association analysis, and traceability analysis.

V. PROPAGATION MODEL

Mobile malicious programs are mainly spread between stores and store users, and the propagation models are mostly described by partial differential equations, similar to the general epidemic mathematical propagation model.

In all kinds of networks and environments, we summarize the complex network theory, social network theory, machine learning, artificial intelligence, stochastic processes, graph theory and other disciplines and make a summary based on the propagation model, as shown in Fig. 12.

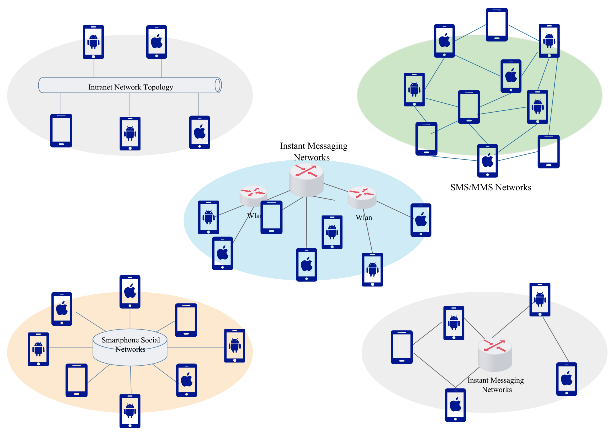

Intranet Network Topology
Instant Messaging Networks
SMS/MMS Networks
Wlan
Wlan
Smartphone Social Networks
Instant Messaging Networks

FIGURE 12. Networks and environments propagation model.

A. NETWORK-LEVEL PROPAGATION

1) SMS/MMS, BLUETOOTH MODEL

Earlier, Cheng's model (analysis model of hybrid malware) [59] proposed an analysis model to analyze the speed and severity of the propagation of hybrid malware, such as the Commwarrior. The model by Mickens is for MMS and Bluetooth (probabilistic queuing framework [60]) via propagation on short-range wireless interfaces. With regard to the SEIR model, (Ramachandran) Ramachandran and Sikdar [61] proposed an analysis model to explore the impact of various propagation mechanisms, such as downloading from the Internet or P2P networks, via Bluetooth, WLAN and infrared interfaces and via MMS or SMS messages The impact of transmission on the dynamics of malware propagation was studied.

a: SMS/MMS

The probabilistic queuing framework [60] models the propagation of mobile viruses over short-range wireless interfaces. The pseudo-base station SMS car is a typical representative of this model.

Thus, the standard homogeneous infection dynamics in each queue are simulated by:

In the epidemic model, the assumption is that infected individuals can recover and will become susceptible again after recovering [62]. Therefore, the differential equations describing the dynamics of an epidemic model are described as follows:

Inclusion of 3 statuses: susceptible (S), infectious (I), or recovered (R).

λ represents the ratio of the number of infections through the number of deaths within a certain period due to infection.

N represents the total number of individuals susceptible to infection and recovery. We assume that the birth rate is equal to the death rate, and the total population is a constant. Thus $S(t)+I(t)+R(t)=N$.

$S(t)$ represents the number of individuals infected by malware at time t, that is, the number of individuals susceptible to malware infection.

$I(t)$ represents the number of individuals who have been infected with malware, and indicates the ability to spread malware to vulnerable individuals.

$R(t)$ represents those individuals who are infected and then recover from the malware. Such terminals must not be infected or infect others.

α represents the average recovery rate.

β represents that average amount of exposure to infected individuals is sufficient. This is called the exposure rate.(infection rate).

δ represents the rate at which the average immunity is lost and the rate at which the individual becomes sensitive again.

$$\frac{dS(t)}{dt} = -\frac{\beta S(t)I(t)}{N} + \lambda(I(t) + R(t)) + \delta R(t) \tag{8}$$

$$\frac{dI(t)}{dt} = \frac{\beta S(t)I(t)}{N} - (\alpha + \lambda)I(t) \tag{9}$$

$$\frac{dR(t)}{dt} = \alpha I(t) - \lambda R(t) - \delta R(t) \tag{10}$$

$$N = S(t) + I(t) + R(t) \tag{11}$$

b: BLUETOOTH

Bluetooth spreads mobile malicious programs [59]. Through the terminal-centered geometric self-diffusion model, $R_C$is the distance radius between the terminals. Unlike the infectious disease model, the BT model has no intermediate host/immune isolation model [62]. We assume that a single infection circle is generated from a point source infected by BT at time r, and s is the unit of duration. Then the spatial infection increment at $r + s$ is

$$G'(r,s) \triangleq \frac{dG(r,s)}{ds} = \beta_{BT}\frac{S(r+s)\cdot\frac{1}{2}\rho\pi R_c^2}{N}c\sqrt{G(r,s)} \tag{12}$$

If the terminals are evenly distributed, c is a proportional constant, and the incremental space at time t infects all infection circles as

$$\frac{dI_{BT}(t)}{dt} = \int_0^t I'_{MMS}(\tau)G'(\tau, t-\tau)d\tau \tag{13}$$

$$c = 2R_C\sqrt{\rho\pi} \tag{14}$$

The growing number of short-range device identification and interaction scenarios provides a scenario for Bluetooth security. For example, the inquiry, identification, interaction, and operation of extensive bicycle sharing all require the participation of Bluetooth. The short-distance transmission of data and instructions between different terminals through Bluetooth constitutes a new security scenario for the IoT.

2) 5G NETWORK BASE STATION GEOGRAPHIC NETWORK PROPAGATION(TWO-LAYER PROPAGATION MODEL)

Gao and Liu [63] [64] once proposed a two-layer model to simulate the propagation process of viruses based on Bluetooth and SMS in a geographic network composed of cellular towers and a logical connection network composed of mobile phones [62]. The lower layer is a cellular tower network based on geographic information.

Our research shows that, based on cloudized and virtualized 5G networks, mobile malware detection and interception are deployed at the core network level to cope with low latency and high concurrency IoT networks, and new application security forms based on artificial intelligence, Internet of Things and cloud computing, such as self-driving cars, industrial Internet, etc.

3) METROPOLITAN AREA NETWORK

Szongott *et al.* [65] proposed a model to show how mobile malware is prevalent among infectious media between devices and infect almost the entire metropolitan area within a few hours. This architecture differs between two different infection environments: road and location. The probability of infection of devices located in these locations is defined as follows:

$$A_i = \beta \times \tfrac{l\times a}{i} \tag{15}$$

In the metropolitan area network, Internet mobile applications (which constitute the second space of human life) for urban geospatial information, a wide range of physical ‘‘scanning applications, location-based applications, travel, electronic wallets, smart home’’ and other applications and metropolitan area network infrastructure constitute an end-to-end mobile Internet ecological environment. It poses a threat to the protection and security of personal information within the metropolitan area.

B. TELECOM NETWORK SECURITY

The core network and access network of telecom operators have become important infrastructures in the Internet era. The rapid deployment and high-cost maintenance of the world’s new generation of 5G telecommunications equipment requires cost-effective solutions to protect the telecommunications core network, access network and terminals from mobile malicious programs. It is necessary to protect smart phone devices or telecom operators from mobile malicious programs by using certain functions (such as wireless interfaces in communication (SMS/MMS/IMS, Bluetooth, 4G and 5G)).

SMS/MMS/IMS can seamlessly spread malicious code to a large number of users across the network, relying on methods including zombies, Trojan horses and worms to attack [66], [67].

In view of the vulnerabilities of the mobile phone, malicious injection of the operator’s network traffic caused the smartphone to be hijacked and injected with mobile malicious programs. A malicious detection mechanism can be deployed to the core of the mobile Internet to detect and prevent signaling-based attacks. These attacks may be related to the hijacking of mobile phones (mobile botnets). Hijacking of mobile phones (mobile botnets) will then generate signaling traffic, which is sent to the mobile core network by the mobile phone before performing denial of service attacks. The root and root directories under the user control authority of the smart phone disable the protection mechanism of the operating system, and then allow the user to install any application that can change the structure and function of the system environment on the smart phone. These applications extend access rights and use them for malicious purposes with harmful effects [68].

In terms of terminal protection, the development of numerous antivirus software programs has improved the defense mechanism against the spread of malicious mobile programs on smartphones. However, for the rapidly iterating mobile terminal hardware and operating system, these antivirus software programs have not been widely downloaded. Moreover, because part of the operating system has a back-to-sleep mechanism, it has large resource consumption, cannot discover unknown threats and is generally poor and unable to meet the full protection of each customized smartphone. Moreover, this solution only provides terminal protection and cannot guarantee the security of the network link, and the success rate of catching viruses is very low.

Therefore, the security solutions used to build the core network must be reconsidered. Although based on this base station, a detective and protective antivirus software system for smartphones has been implemented to cut off the spread of smartphone viruses, the low flexibility and portability of the base station still have great limitations [67]. Moreover, most of the existing mobile malicious program countermeasures, such as those developed in response to SMS/MMS mobile malicious programs, use centralized methods implemented in the operator's network. However, these centralized defense mechanisms are only effective for mobile malicious programs spread through a centralized infrastructure [69]. At the same time, there are also the high cost and complex deployment of network resource equipment related to the core network.

The spread of mobile malicious programs in a new generation of the mobile Internet environment dominated by high-speed 5G networks poses a major risk. The deployment of future 6G, content center networks and SDN networks, and new networks of popular core applications may all become the main targets of new mobile malicious programs. Unknown types of mobile malicious programs spread and harm terminals in a distributed manner without the operator's network prevention and control mechanism. Many types of mobile malicious programs use the proximity of devices to replicate themselves in a distributed manner, making them difficult to detect. Due to the lack of a suitable network provider and highly dynamic prevention and control topology (hindering possible defense lines), it is also challenging to protect terminals and core nodes from these mobile malicious programs [69]. Most existing mobile malicious program detection applications rely on mobile malicious program databases. In addition, mobile terminals, including their limited processing power, storage space and battery power, all of these attributes constitute an obstacle to the timely distribution of mobile malicious program signature files between mobile devices [20], [70].

This article combines the deployment of the cyber threat situational awareness system to carry out a core network with deployment of a mobile malicious program protection system for convergent devices, and due to the low egress bandwidth of the mobile network, it can be monitored almost 100% by the mobile core network flow, so proceeding to the large-scale deployment has achieved remarkable results.

C. SOCIAL NETWORK COMMUNICATION

Social networking is the basis for malware transmission. Combining complex interpersonal relationships with the ever-changing personal behaviors results in a complex social network. Therefore, the formation mechanism and evolution of social networks are the core issues of malware propagation dynamics.

A country, a language region, an urban area, industry distribution, and regional distribution all constitute elements of social network communication.

Based on the inference relationship between user portraits and user acquisition, mobile malicious programs and user behaviors in social graphs, a user recognition model was constructed to characterize the user's cognitive process, and mobile malicious programs and user behaviors based on user portraits and accesses were studied. The user's cognitive model of the dependency graph (the structure of the belief network), and the convex inference function are approximated by convex optimization, and then the breakthrough is to learn the dependency graph and the conditional probability table from the original large-scale heterogeneous data set (the strength of dependence in the belief network).

Let $G = (V, E, X, Y)$ for social networks, V for user nodes, E for connections between users, X for user attributes, and Y for connection attributes. At the same time, there are multiple mobile malicious programs spreading together on the network, and there is competition or promotion between each other. The existing propagation model cannot determine the correlation of the propagation. Therefore, the introduction of the content correlation matrix $I_{L\times L}^{CR}$represents the correlation between multiple programs. The contextual export of $I_{L\times L}^{CR}$ by the program in the knowledge graph. The effect of program propagation is affected by the network topology at the current moment, and the network topology changes due to the propagation of the program, which in turn affects the propagation of the program at the next moment.

Models for spreading changes: Social contact networks are the basis of malware spreading. Combining complex interpersonal relationships with changing personal behaviors leads to complex social networks. Therefore, the formation mechanism and evolution of social networks are the core issues of malware propagation dynamics.

$$G^t = f_{info\_topo}\left(G^{t-1}, I_{L\times L}^{CR}\right) \tag{16}$$

Suppose the network topology at time t is G^t, with a total of M users, then the expected spread of the program whose content at time t is l $ex(t)$ is:

$$ex(t) = \sum_{i=1}^{M} P\left\{f_{topo\_info}\left[G^t, N\right], N\right\} \tag{17}$$

$$N = I_{L\times L}^{CR} \tag{18}$$

where P represents a single user's acceptance model of the program, accumulating the influence exerted by the neighbors of the user in Equation (17).

f_{topo_info} is a function that represents the influence of the network topology on the propagation of the program at the current moment, and the final output is the range of guided text propagation. In addition, the f_{info_topo} function can acquire the network topology state diagram during the process of program propagation. According to the topology map combined with the propagation scale expectation, the impact range of the current program propagation and the main target community impact degree can be fully understood.

1) CONSTRUCTION OF SOCIAL RELATIONSHIP GRAPH

Based on the SMS, MMS, the IMS (the instant messaging service) malicious behavior continues to spread and danger gradually increase [71], such that the SMS, MMS and the IMS and other forms of the distribution and dissemination for the entire network have become an attack vector for mobile smart terminals. For example, the SMS, the MMS and the IMS may be used to deliver content and maintain a communication with the malicious attacker. For SMS, the attacker uses SMS to send a URL link and then tricks the user into using this URL to open a browser window. For MMS, the message itself it may be a malicious file. The victim will most likely open and download the message. Moreover, with several years of rapid development of the IMS and the spread of social software based on the IMS, it has become a new model based on instant messaging, groups, and social circles of friends to complete the transmission and spread of malicious samples of many types. Therefore, taking worms as an example, an effective SMS/MMS/IMS-based malicious propagation method should take into account the social relationship graph between mobile devices in the mobile Internet.

The mobile number arrangement based on major countries has strong regularity and regional division, and the information based on the country-based authentication system as the identification of mobile phones in the Internet ecosystem has become relatively public key information, and has formed an end-to-end network communication channel. If Alice and Bob have a social connection, Alice is more likely to open a message from Bob. However, if Alice 's smartphone has been infected by an SMS/MMS/IMS- based worm and this malicious message is sent to Bob, Bob 's smartphone is likely to be infected by the worm. The fact is that if two smartphone users have never sent messages to each other, even though Alice 's smartphone has been infected, Bob 's phone will not be infected. The detailed list of SMS / MMS/IMS communication information between two smartphones can predict whether they will participate in the propagation path of mobile malicious programs.

Telecom operators of details of a single large data (including SMS, telephone, traffic and large portraits of data) can build social graph. The data for big data character portraits also provides a foundation and verification environment for our research. The social relationship graph is represented by an undirected weighted graph $G = (V, E)$, where the vertex of V corresponds to the smartphone in the mobile Internet network, and the number of E edges corresponds to the communication exchanged between any two smartphones detailed order quantity, i and j. Vertex degree i, expressed as d_i, is, the number of smartphones (the number of links on behalf or on behalf of smartphone owners d_i friends). The number of message records initiated from i to j *is* represented by C_{ij}.

$f(i)$ maps each vertex $i \in v$ and $f(i,j)$ maps each edge $(i, j) \in E$. Therefore, $f(i)$ and $f(i, j)$ can be used to determine the weights of vertices and edges, respectively. The mapping function of the weights of vertices and edges is described as follows.

$$f(i) = d_i \tag{19}$$

$$f(i, j) = C_{ij} + C_{ji} \tag{20}$$

Right vertices and edges of weights as important indicators represent the possibility of being infected with mobile malicious programs. From Equation (19) it can be seen, depending on the vertex weights d_i. Since Android and other open source mechanisms and systems do not support continuous updates for the first time, the vulnerability cannot be completely avoided. For mobile malicious programs based on SMS/MMS/IMS, a smartphone with a higher degree of intrusion means that it is more likely to be infected and more likely to infect other smartphones. Therefore, smartphones with high degrees should be assigned higher vertex weights.

Any two smartphones can be interacted by Equation (20). If the two-smartphone machine has passed the SMS, MMS or IMS communicates, and they are more likely to open a message and activate each other's worm infections. This social relationship diagram shows how smartphones connect to each other and how worms use this relationship to spread.

Table 4 shows an example of a social relationship graph. We use the number of exchanges between two smartphone messages, recording i and j as the weights C_{ji}. The number of mobile phone communications per unit time can measure the social relationship between any two smart phones [72].

TABLE 4. Messages between two smartphones record traffic.

Between Smartphones	WAT
A and B	8
A and D	24
A and E	6
B and E	40
B and C	12
B and F	8
C and F	32
C and G	18
D and E	12
D and H	8
E and F	12
E and H	28
F and G	24
F and I	24
G and I	16
G and J	20
H and I	16
H and J	12

In Table 4, each vertex is abstracted as a smartphone, and by dividing the maximum within one week, WAT is normalized between any two smartphone WATs and can be obtained as the relationship shown in Fig. 13.

An accurate social relationship graph can be built by analyzing the message records. The graph may reflect the propagation path of the worm, and the worm by using the host terminal infected the communication book for social network communication.

2) USE SOCIAL NETWORK GRAPHS TO MODEL MOBILE MALWARE PROPAGATION

An accurate social relationship graph can be built by analyzing the message records. The graph may reflect the propa-

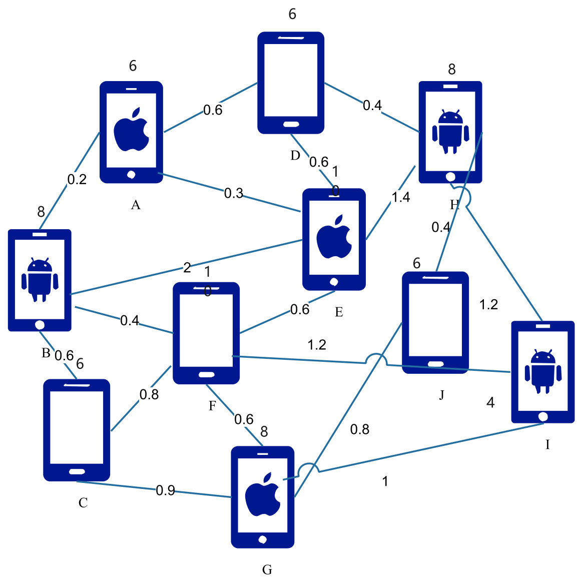


FIGURE 13. SMS/MMS social relationship diagram.

gation path of the worm, and the worm by using the host terminal infected the communication book for social network communication.

a: STATE TRANSITION RELATIONSHIP

According to the propagation characteristics of SMS/MMS/IMS- based worms, three types of epidemic status are considered: susceptible (S), infected (I) and recovered (R). We assume that susceptible occurs at t, and the infected and recovered nodes are represented by S(t), $I(t)$ and R(t) *respectively.* The conversion process of the worm propagation state is shown in Fig. 14.

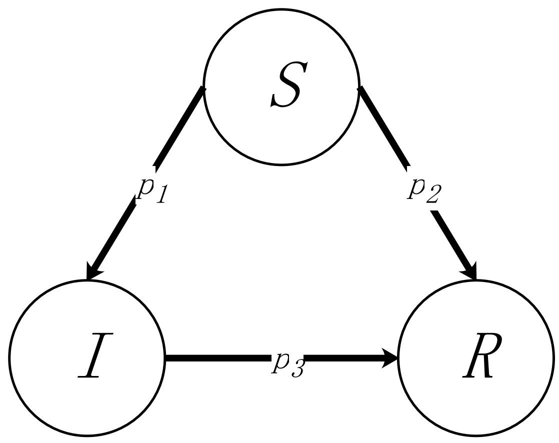


FIGURE 14. Smartphone based on SMS / MMS state transition worm propagation of the relationship.

Some terms of the model are explained as follows:

p_1 represents the probability that node state S becomes the node status I.

p_2 represents the probability that node state S becomes the node state R.

p_3 represents the probability that node state I becomes the node status R.

b: STATE TRANSITION ALGORITHM

To reasonably describe the infection ability of mobile malicious programs in smartphones based on social network graphs, we introduce the node's infection degree i, expressed as ID_i , which is used to measure the danger degree from infected smartphones to vulnerable smartphones.

Two important factors to characterize individual differences that are based on SMS / MMS / IMS affect transmission dynamics of worms.

One is the infection factor. IF_{ji} is used to indicate the degree of infection from node j to node i ($0 \leq IF_{ji} \leq 1$). If IF_{ji} is equal to 0, it means that node j *is* not infected with node i. If RF_{ij} is equal to 1, it means that node j has a strong infection i on the node.

The other is the resistance factor. $RF_{ij\ to}$ is used to indicate that node i is infected by some kind of worm ($0< RF_{ij} \leq 1$). If RF_{ij} is equal to 1, it means that the node i has a strong ability to resist nodule infection j. Let T represents a state transition from a wireless node transmitted with an elapsed threshold to its state. Let N_i denote the number of infectious friends of the node I, and ID_i, $RF_{ij,}$ and IF_{ji} can be described as follows.

$$ID_i = \frac{1}{N_i}\sum_{j=1}^{j=N_i}\left[\left(\omega_1\frac{C_{ji}}{C_{max}} + \omega_2\frac{C_{ji}'}{C_{max}}\right) \times \frac{IF_{ji}}{RF_{ij}}\right] \tag{21}$$

$$RF_{ij} = \omega_3\frac{1-\lambda_2 e^{-\beta t}}{1+\lambda_2 e^{-\beta t}} + \omega_4\frac{1-\lambda_2 e^{-\beta C_{ji}}}{1+\lambda_2 e^{-\beta C_{ji}}} \tag{22}$$

$$IF_{ji} = \omega_5\frac{\lambda_1 e^{\alpha t}-1}{\lambda_1 e^{\alpha t}+1} + \omega_6\frac{\lambda_1 e^{\alpha C_{ji}}-1}{\lambda_1 e^{\alpha C_{ji}}+1} \tag{23}$$

Here C_{ji} from j to i transmitted within a week S the MS number, C_{ji}' is from j to i transmitted within a week MMS number of messages. C_{max} is the maximum number of messages sent between any two smartphones in a week. λ_1 and $\lambda_2$are constants to be determined according to actual needs. α and β are the adjusted factors IF_{ji} and RF_{ij} *respectively*. ω_1, $\omega_2, \omega_3, \omega_4, \omega_5$ and ω_6 represent weighting factors, $\omega_1+\omega_2 = 1$, $\omega_3+\omega_4 = 1$, $\omega_5 + \omega_6 = 1$.

The formula for the state transition process is as follows.

1. The network initialization. All nodes use SMS/MMS/IMS to communicate with each other.

2. The node initialize the state. Randomly select a node and set its state to *the* State I, and the status of the other nodes is set to State S.

3. It can be used to collect information by analyzing their friends' message records.

4. Node I can access T. When node I has communicated with his / her friends Δt through SMS / MMS in a certain period of time.

Case 1: Regarding node i, if its status is I, it can visit its friend node. If his friend's node J state is, whereas if the ID section Node J is not less than T, Node J its state from *to* I is likely top_1 th. Otherwise, node j keeps its previous state. If

IF_{ij} is equal to 0 or RF_{ji} is equal to 1, node j changes its state from *to* R corresponding to probabilit$_y p_2$. Meanwhile, node I to its state fr*om* I *to* R useful corresponds probabilit$_y$ p_3.

Case 2: Repeat the beginning of step 4 until all nodes in the network are visited.

5. T is equal to t plus Δt. This completes the algorithm.

D. HOMOGENEOUS COMMUNITY COMMUNICATION

A homogeneous community is a community composed of network groups with common cultural structure, life and ideology. The commonality in a homogenous community is called a homogenous attribute. According to social psychology theory, individuals have many aspects of education, occupation, wealth, etc. The homogeneous nature is the basis for reaching consensus and forming group behavior convergence. The behavior-driven time series model of social network structure studies the impact of information sharing and information interaction on the network structure, analyzes the network evolution rules under the influence of user behavior factors and information dissemination factors, and is a dynamic discovery of homogeneous communities.

We propose that there is an enrichment of mobile malicious programs in homogeneous communities, which has a socialized propagation mechanism for common network groups. A homogenous community of malware has resulted in a large number of organized malicious applications.

The top-30 firmware version community mobile malware covers more than 67.2% of all malware samples, which means that a small number of communities will contribute a large number of malware samples. The concentration of its community also provides a basis for the probability distribution for the monitoring of the mobile malware distribution.

E. APPLICATION OF MARKET-LEVEL ECOSYSTEM COMMUNICATION

The rapid global popularity of smartphones has promoted the rapid growth of the application market and has increased the interest of mobile malicious programs to exponential growth. Some cracking methods can be modified to crack paid applications and spread as free downloads and are separated from the operating mechanism of the mobile application store and is not subject to official and third-party management. Android's multiple heterogeneous markets and the open source development and direct installation mechanism of APK programs make it easy for malicious programs to penetrate the market [73]–[76]. Malicious applications in the Android market are constantly increasing and iterating. Driven by the ecological interest chain of the application market, malicious users or hackers can easily submit infected applications to the Android market without any mechanism restrictions. However, the Android market does not check, test, or filter out [77].

Since third-party application developers are benefiting the ecological chain, they hope to increase the download volume, stickiness, and activity of applications. The smartphone platform provides a friendly environment for developers, such as program libraries, interfaces, and various emulators. They can also be used to develop or secondary develop applications that can effectively endanger the privacy and security of smartphone users. Malicious code also It can be repackaged into normal applications and rooting modules, thereby making a security hole into the smartphone operating system [76], [78], [79]. The heterogeneity, security mechanism and management mode of the application store enable mobile malicious programs to survive. Therefore, mobile malicious programs can leak private information, abuse smartphone functions and even gain ROOT privileges [15], [80], [81].

The profit-driven mobile Internet ecology makes the mobile phone market (such as the Google Play store) increasingly drive the download and dissemination of mobile programs through program ratings, without adequate security mechanisms. Android program of review, credibility and assessment should be a reasonable distribution of statistics users and security research personnel written release. However, plug-ins, order- swiping and agency models and even official agreements have caused the proliferation of false reviews, affecting the ranking and dissemination of apps [82], [83]. The security model of the Android platform through the use of signature-based scanning does not require a trusted security agency to sign developer certificates, thereby allowing the use of self-signed digital certificates to sign many applications. In this case, the certificate has not been verified by a trusted third party to allow anonymous developers to publish a large number of programs on the market, resulting in poor source origin and integrity protection [81], [83].

Once an application is installed on the system, the user will lose some control over the operation of the application. Usually without user authorization, the installed application can control several smart phone permissions, including call, SMS, camera, GPS, Bluetooth, WLAN wireless network and 4G/5G access [83], [84]. Different smart phone platforms implement mandatory access control to restrict arbitrary access to system resources (including the Internet, location or cellular services). The permission list authorization module is forcibly implanted in the installation process. The system will display the required permission list to the user, but cannot selectively accept or reject the access permission. Therefore, many users accept these license requests without considering their over-authorization and software vulnerabilities [68]. The application can also upload information to a remote server without notifying the user. In this case, the data will be transmitted to the attacker via a remote server without the user's knowledge [20], [81].

We found that publishers often submit multiple malware samples to the market in a short time. According to our research data on mobile malware, Android malware spreads through the mobile store market [85].

Malware authors write/rewrite mobile malicious programs, upload them to websites and application stores, wait for downloads, install them on mobile phones, communicate with remote servers, and obtain information and profit.

According to our research data, malicious program providers are more inclined to deploy malicious program download sources in the hosted cloud storage host. The privacy of the cloud platform and the high cost of money per unit of time provide convenience for the spread of mobile malicious programs.

Many popular applications with high download numbers are still malicious, and the popularity of the applications is not an effective indicator of their quality.

The spatial and temporal distribution is based on third-party markets and websites, and its performance is highly iterative.

Through our 178,155 server traces of real mobile malware data detected in the China Unicom network environment, mobile malware is found to be enriched in certain third-party store ecosystems (such as Redmi, MI and other mobile phones). The secondary development Android system of the producer and its supporting third-party application store constitute an end-to-end communication environment.

F. CONSTRUCTION OF MOBILE INTERNET BIG DATA KNOWLEDGE MAP

Through knowledge labeling and knowledge construction, the association relationship of knowledge is extracted to realize the research of knowledge map construction. The relationship types that can be extracted are shown in Fig. 15.

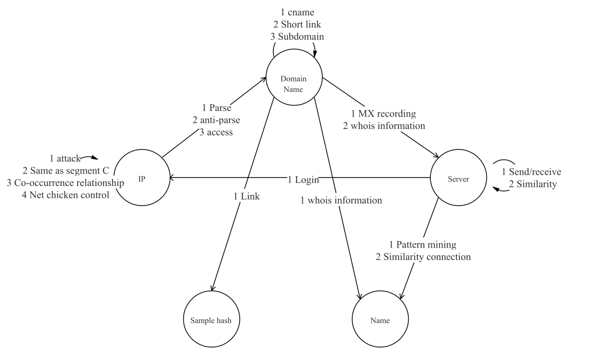


FIGURE 15. Graph relation extraction type.

From a technical perspective, relationship extraction can be divided into the following three forms:

Static extraction. That is, for some static features, such as whois information, the registration information corresponding to the domain name can be extracted.

The advanced big data storage architecture was selected to realize the complete data collection framework and normalization process, which could provide a variety of different data analysis methods, support real-time analysis, post analysis and other analysis modes, Comprehensive analysis and mining are mainly for all kinds of data resources stored in the file system or database, realizing statistics, retrieval, association, collision, and efficient processing of specific mining models and supporting the mining and analysis of unknown threats.

TABLE 5. Dynamic extraction.

No.	Dynamic Element 1	Dynamic element 2	Atlas relationship
1	Attack Source IP	Attacked IP attack	Relationship
2	Malicious Domain Name	IP	Resolution and anti-resolution relationship
3	IP	Domain	Access Relationship
4	Domain Name	Server login IP of the linked sample	Relationship
5	Domain Name	server MX record	Relationship
6	Domain Name	Sub domain	Relationship
7	Certificate	IP	Relationship
8	Certificate	Certificate server	Certificate similarity relationship

Dynamic extraction in Table 5 is the extraction of relationships that change dynamically over time, such as the attack relationship between the attack source IP and the attacked IP, the malicious domain name and IP resolution and antiresolution relationship, the IP and domain name access relationship, the domain name and the server login of the linked sample IP correspondence, the correspondence between the domain name and the server mail exchange (MX) record, the correspondence between the domain name and its subdomain names, the correspondence between the certificate and the IP, the similarity relations relationship between the certificate and the certificate server certificate, and so on.

In-depth analysis extraction, that is, the relationship that needs to be mined through in-depth analysis, such as the co-occurrence relationship, is to use the adjunct relationship of the attack to extract whether the control-end IP has launched an attack at the same time. There are also relationships generated by connecting information through similarity through model matching.

We used big data with connecting information through similarity model matching. Comprehensive analysis mining is mainly used for all kinds of data resources stored in a file system or database, realizing statistics, retrieval, association, collision, and efficient processing of specific mining models, and supporting the mining analysis of unknown threats. Through model matching, the relationship is generated through the similarity connection.

The extracted entities and relationships are used to connect all the entities that have relationships and build a fully connected relationship graph, which is a directed graph, as shown in Fig. 16.

The knowledge graph provides an intuitive display of all resource attributes and relationships in the secure resource pool in an interactive manner. Based on the knowledge graph, it supports rapid expansion analysis and investigation analysis capabilities and provides security analysis capabilities for graphs: All entities and relationships in all events of a query

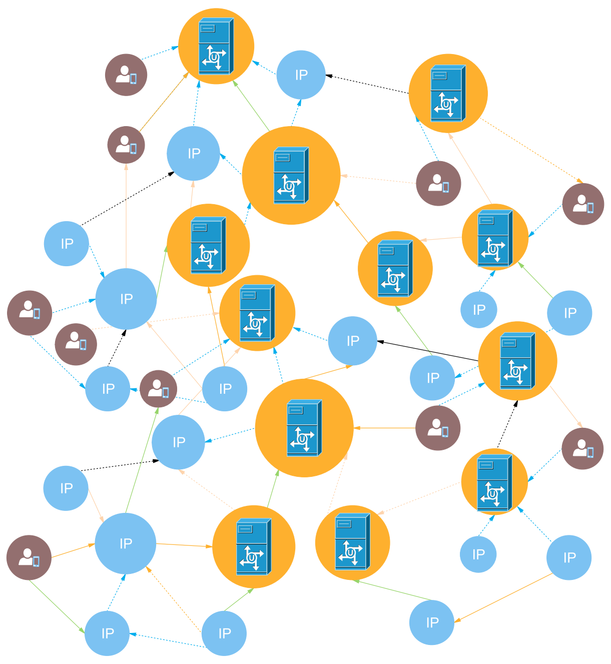


FIGURE 16. Fully connected relationship directed graph.

should be extracted to build a comprehensive relationship graph and automatically optimize the layout.

It can quickly perform line expansion analysis for an entity or relationship and supports rapid event drilling and one-click analysis of related security objects in an interactive form.

It can intuitively distinguish between the physical object and the strength of the relationship and the criticality of the entity in certain types of relationships.

A certain entity can be used as the center point to expand its related entities and relationships. It defaults to N degrees, and N can be set as: interaction (study entity attributes or relationships entity search analysis, locates the search entity) and relation search analysis(locate entities that match the relationship). Research on security analysis algorithms for knowledge graphs, including graph search, traversal, the shortest path, large-scale change of domain names, orphan pages, brute force cracking, port anomalies, abnormal logins, abnormal traffic and other security analysis algorithms.

Providing interactive analysis based on the knowledge map is the core research of knowledge generation and service. Through the relevant analysis of the facts and relationships of security incidents, combined with interactive analysis algorithms, fast and effective expansion analysis is performed.

VI. MACHINE LEARNING FOR ENCRYPTED MALWARE TRAFFIC PACKET ON SSL(HTTPS)

Nowadays, most of (mobile) web sites communicate with their users on SSL(HTTPS). There are two problems in the encrypted malware traffic packet. One is that it's difficult to detect the encrypted threat traffic resulting in the ineffective analysis and detection of attack characteristics of complex network such as botnet traffic. The other is that machine learning is featured by high false alarm rate, long training period and demanding data accuracy. For that, data packet was deeply analyzed through the data packet analysis method including the analysis of its header and content so as to realize the analysis of network traffic's characteristics. By using XGBoost, SVM, Random Forest and Logistic Regression and other machine learning algorithms to detect the encrypted malware traffic packet, we divided the identify characteristics. Finally, we found that XGBoost and Random Forest were are better for threatening dense flows. We obtained the SHAP evaluation [86] data of all important features on the classification of malware dense flow. In addition, we carried out the impact verification of the importance characteristics. The research we have carried out to help encrypt the traffic discrimination of malware is the improvement of our proposed model system.

VII. CONCLUSION

To study the traceability, propagation, and detection of the threats, we perform research on all aspects of the end-to-end environment. By controlling the spread of the malware network, sample collection, research and plugging sentences, the malware prevention system and key scientific issues are controlled in order to protect the safety of the mobile Internet. The network side technology monitors the download source, download channel and terminal running environment of the mobile application, while network traffic analysis identifies the malware transmitted in the network. With machine learning based on the mobile malware detection algorithms that integrate the dynamic and static research of the identification algorithm, application software samples are collected to study sentences. Through research, we obtained global malware threat detection, traceability, and propagation models. Through knowledge labeling and knowledge construction, the association relationship of knowledge is extracted to realize the research of knowledge map construction. This study aimed to perform detection on a large network in China Unicom mobile environment regarding 178,155 real malicious program data by using the data flow probe of Android mobile program data files to obtain detailed Guiyang (China "Data Center Capital") mobile malware infected program data. Flooding is closely correlated with the complexity of the Android mobile version of the kernel and malicious programs. A static dynamic analysis of the mobile malicious program is carried out, and the social network social diagram is constructed to model the propagation of the mobile malicious program. We extended the approach of deriving common malware behavior through graph clustering to the field of Android mobile malicious program detection in the Linux kernel. On this basis, Android behavior analysis is performed through our virtual machine execution engine to evaluate the maliciousness of the program through the heuristic analysis algorithm. We extend the family characteristics to the concept of DNA race genes. By studying SMS/MMS, Bluetooth, 5G base station networks, metropolitan area networks, social networks, homogeneous communities, telecommunication networks, and application market ecosystem propagation scenarios, we discovered the law of propagation. In addition, we studied the construction of the mobile Internet big data knowledge graph. Quantitative data for the main family chronology of mobile malware are obtained. We conducted detailed research and comprehensive analysis of Android application package (APK) details and behavior, relationship, resource-centric, and syntactic aspects. Furthermore, we summarized the architecture of mobile malware security analysis. We also discuss encryption of malware traffic discrimination. These precise modeling and quantified research results constitute the architecture of mobile malware analysis.

ACKNOWLEDGMENT

The authors would like to thank the two reviewers and editors for their professional advice, which has been of great help to our article adjustment and further research.

REFERENCES

[1] S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song, and H. Yu, "SAMADroid: A novel 3-Level hybrid malware detection model for Android operating system," *IEEE Access*, vol. 6, pp. 4321–4339, 2018.

[2] Statcounter. (2019). *Mobile Operating System Market Share Worldwide*. Mobile Operating System Market Share Worldwide. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide/2019

[3] G. Suarez-Tangil, J. E. Tapiador, P. Peris-Lopez, and A. Ribagorda, "Evolution, detection and analysis of malware for smart devices," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 961–987, 2nd Quart., 2014.

[4] A.-D. Schmidt, H.-G. Schmidt, L. Batyuk, J. H. Clausen, S. A. Camtepe, S. Albayrak, and C. Yildizli, "Smartphone malware evolution revisited: Android next target?" in *Proc. 4th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2009, pp. 1–7.

[5] N. K. Gyamfi and E. Owusu, "Survey of mobile malware analysis, detection techniques and tool," in *Proc. IEEE 9th Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Nov. 2018, pp. 1101–1107.

[6] A. I. Ali-Gombe, B. Saltaformaggio, J. Ramanujam, D. Xu, and G. G. Richard, "Toward a more dependable hybrid analysis of Android malware using aspect-oriented programming," *Comput. Secur.*, vol. 73, pp. 235–248, Mar. 2018.

[7] H. Cai, N. Meng, B. Ryder, and D. Yao, "DroidCat: Effective Android malware detection and categorization via app-level profiling," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 6, pp. 1455–1470, Jun. 2019.

[8] Z. Luoshi, N. Yan, W. Xiao, W. Zhaoguo, and X. Yibo, "A3: Automatic analysis of Android malware," in *Proc. 1st Int. Workshop Cloud Comput. Inf. Secur.*, 2013, pp. 1–5.

[9] Z. Ning and F. Zhang, "Hardware-assisted transparent tracing and debugging on ARM," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 6, pp. 1595–1609, Jun. 2019, doi: 10.1109/TIFS.2018.2883027.

[10] L. Onwuzurike, M. Almeida, E. Mariconti, J. Blackburn, G. Stringhini, and E. De Cristofaro, "A family of droids—Android malware detection via Behavioral modeling: Static vs dynamic analysis," 2018, *arXiv:1803.03448*. [Online]. Available: http://arxiv.org/abs/1803.03448

[11] P. Bhat and K. Dutta, "A survey on various threats and current state of security in Android platform," *ACM Comput. Surveys*, vol. 52, no. 1, pp. 1–35, Feb. 2019, doi: 10.1145/3301285.

[12] S. R. Department. (Nov. 2019). *Internet of Things (IoT)*. Number of connected devices worldwide from 2012 to 2020 (in billions). [Online]. Available: https://www.statista.com/statistics/471264/iot-numberof-connected-devices-worldwide

[13] D.-J. Wu, C.-H. Mao, T.-E. Wei, H.-M. Lee, and K.-P. Wu, "DroidMat: Android malware detection through manifest and API calls tracing," in *Proc. 7th Asia Joint Conf. Inf. Secur.*, Aug. 2012, pp. 62–69.

[14] L. Onwuzurike, M. Almeida, E. Mariconti, J. Blackburn, G. Stringhini, and E. De Cristofaro, "A family of droids-Android malware detection via behavioral modeling: Static vs dynamic analysis," in *Proc. 16th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2018, pp. 1–10.

[15] T. Isohara, K. Takemori, and A. Kubota, "Kernel-based behavior analysis for Android malware detection," in *Proc. 7th Int. Conf. Comput. Intell. Secur.*, Dec. 2011, pp. 1011–1015.

[16] V. Rastogi, Y. Chen, and W. Enck, "AppsPlayground: Automatic security analysis of smartphone applications," in *Proc. 3rd ACM Conf. Data Appl. Secur. Privacy - CODASPY*, 2013, pp. 209–220.

[17] A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici, "Mobile malware detection through analysis of deviations in application network behavior," *Comput. Secur.*, vol. 43, pp. 1–18, Jun. 2014.

[18] A. Firdaus, N. B. Anuar, A. Karim, and M. F. A. Razak, "Discovering optimal features using static analysis and a genetic search based method for Android malware detection," *Frontiers Inf. Technol. Electron. Eng.*, vol. 19, no. 6, pp. 712–736, Jun. 2018.

[19] Z. Wang, "Research on Android malware detection," Ph.D. dissertation, Harbin Inst. Technol., Harbin, China, 2017.

[20] M. Talal, A. A. Zaidan, B. B. Zaidan, O. S. Albahri, M. A. Alsalem, A. S. Albahri, A. H. Alamoodi, M. L. M. Kiah, F. M. Jumaah, and M. Alaa, "Comprehensive review and analysis of anti-malware apps for smartphones," *Telecommun. Syst.*, vol. 72, no. 2, pp. 285–337, Oct. 2019.

[21] H. Dong, "Research on the detection and protection technology of mobile applications," Ph.D. dissertation, Beijing Univ. Posts Telecommun., Beijing, China, 2014.

[22] L. Cen, C. S. Gates, L. Si, and N. Li, "A probabilistic discriminative model for Android malware detection with decompiled source code," *IEEE Trans. Depend. Sec. Comput.*, vol. 12, no. 4, pp. 400–412, Jul. 2015.

[23] Y. Aafer, W. Du, and H. Yin, "DroidAPIMiner: Mining API-level features for robust malware detection in Android," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.*, 2013, pp. 86–103.

[24] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "MalDozer: Automatic framework for Android malware detection using deep learning," *Digit. Invest.*, vol. 24, pp. S48–S59, Mar. 2018.

[25] S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu, "HinDroid: An intelligent Android malware detection system based on structured heterogeneous information network," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 1507–1515.

[26] A. Pekta and T. Acarman, "Deep learning for effective Android malware detection using API call graph embeddings," *Soft Comput.*, vol. 24, no. 2, pp. 1027–1043, Jan. 2020, doi: 10.1007/s00500-019-03940-5.

[27] A. Apvrille and T. Strazzere, "Reducing the window of opportunity for Android malware gotta catch 'em all," *J. Comput. Virology*, vol. 8, nos. 1–2, pp. 61–71, May 2012, doi: 10.1007/s11416-012-0162-3.

[28] L. Taheri, A. F. A. Kadir, and A. H. Lashkari, "Extensible Android malware detection and family classification using network-flows and API-calls," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2019, pp. 1–8.

[29] R. Mateless, D. Rejabek, O. Margalit, and R. Moskovitch, "Decompiled APK based malicious code classification," *Future Gener. Comput. Syst.*, vol. 110, pp. 135–147, Sep. 2020, doi: 10.1016/j.future.2020.03.052.

[30] B. Yang, Z.-S. Tang, H.-J. Zhu, and J.-C. Lin, "Method of Android applications permission detection based on static dataflow analysis," *Comput. Sci.*, vol. 39, pp. 16–20, 2012.

[31] B. Sanz, I. Santos, C. Laorden, X. Ugartepedrero, P. G. Bringas, and G. Á. Marañón, *PUMA: Permission Usage to Detect Malware in Android*. Berlin, Germany: Springer, 2013.

[32] B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, "Android permissions: A perspective combining risks and benefits," in *Proc. 17th ACM Symp. Access Control Models Technol.*, 2012, pp. 13–22, doi: 10.1145/2295136.2295141.

[33] H. Peng, C. Gates, B. Sarma, N. Li, Y. Qi, R. Potharaju, C. Nita-Rotaru, and I. Molloy, "Using probabilistic generative models for ranking risks of Android apps," in *Proc. ACM Conf. Comput. Commun. Secur. CCS*, 2012, pp. 241–252.

[34] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, "WHYPER: Towards automating risk assessment of mobile applications," in *Proc. 22nd USENIX Conf. Secur.*, 2013, pp. 527–542.

[35] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *Proc. 18th ACM Conf. Comput. Commun. Secur. CCS*, 2011, pp. 627–638.

[36] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Permission evolution in the Android ecosystem," in *Proc. 28th Annu. Comput. Secur. Appl. Conf.*, 2012, pp. 31–40, doi: 10.1145/2420950.2420956.

[37] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, "Pscout: Analyzing the Android permission specification," in *Proc. 2012 ACM Conf. Comput. Commun. Secur.*, 2012, pp. 217–228, doi: 10.1145/2382196.2382222.

[38] H. Mei, Q. Wang, L. Zhang, and J. Wang, "Software analysis: A road map," *Chin. J. Comput.*, vol. 32, no. 9, pp. 1697–1710, 2009.

[39] T.-E. Wei, C.-H. Mao, A. B. Jeng, H.-M. Lee, H.-T. Wang, and D.-J. Wu, ''Android malware detection via a latent network Behavior analysis,'' in *Proc. IEEE 11th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Jun. 2012, pp. 1251–1258.

[40] P. S. Chen, S.-C. Lin, and C.-H. Sun, ''Simple and effective method for detecting abnormal Internet Behaviors of mobile devices,'' *Inf. Sci.*, vol. 321, pp. 193–204, Nov. 2015.

[41] M. Egele, C. Kruegel, E. Kirda, H. Yin, and D. X. Song, ''Dynamic spyware analysis,'' in *Proc. USENIX Annu. Tech. Conf.*, Santa Clara, CA, USA, Jun. 2007, pp. 1–14.

[42] A. Vasudevan, ''Wildcat: An integrated stealth environment for dynamic malware analysis,'' Ph.D. dissertation, Univ. Texas Arlington, Arlington,TX, USA, 2007.

[43] C. Willems, T. Holz, and F. Freiling, ''Toward automated dynamic malware analysis using CWSandbox,'' *IEEE Secur. Privacy Mag.*, vol. 5, no. 2, pp. 32–39, Mar. 2007.

[44] I. Goldberg, D. Wagner, R. Thomas, and E. A. Brewer, ''A secure environment for untrusted helper applications confining the wily hacker,'' in *Proc. Conf. Usenix Secur. Symp.*, 1996, p. 1.

[45] X. Wang, Y. Yang, and Y. Zeng, ''Accurate mobile malware detection and classification in the cloud,'' *SpringerPlus*, vol. 4, no. 1, p. 583, Dec. 2015.

[46] T. Eder, M. Rodler, D. Vymazal, and M. Zeilinger, ''ANANAS—A framework for analyzing Android applications,'' in *Proc. Int. Conf. Availability, Rel. Secur.*, Sep. 2013, pp. 711–719.

[47] M. Lindorfer, M. Neugschwandtner, and C. Platzer, ''MARVIN: Efficient and comprehensive mobile app classification through static and dynamic analysis,'' in *Proc. IEEE 39th Annu. Comput. Softw. Appl. Conf.*, Jul. 2015, pp. 422–433.

[48] P. Wang and Y.-S. Wang, ''Malware behavioural detection and vaccine development by using a support vector model classifier,'' *J. Comput. Syst. Sci.*, vol. 81, no. 6, pp. 1012–1026, Sep. 2015.

[49] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, ''PiOS: Detecting privacy leaks in iOS applications,'' in *Proc. NDSS*, 2011, pp. 177–183.

[50] W. Enck, D. Octeau, P. D. McDaniel, and S. Chaudhuri, ''A study of Android application security,'' in *Proc. USENIX Secur. Symp.*, Aug. 2011, p. 2.

[51] N. Moustafa, J. Hu, and J. Slay, ''A holistic review of network anomaly detection systems: A comprehensive survey,'' *J. Netw. Comput. Appl.*, vol. 128, pp. 33–55, Feb. 2019.

[52] L. Invernizzi, S. Miskovic, R. Torres, S. Saha, S.-J. Lee, M. Mellia, C. Kruegel, and G. Vigna, ''Nazca: Detecting malware distribution in large-scale networks,'' in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2014, pp. 23–26.

[53] Y. Park, D. S. Reeves, and M. Stamp, ''Deriving common malware Behavior through graph clustering,'' *Comput. Secur.*, vol. 39, pp. 419–430, Nov. 2013, doi: 10.1016/j.cose.2013.09.006.

[54] D. J. Cook and L. B. Holder, *Mining Graph Data*. Hoboken, NJ, USA: Wiley, 2006.

[55] J. R. Ullmann, ''An algorithm for subgraph isomorphism,'' *J. ACM*, vol. 23, no. 1, pp. 31–42, Jan. 1976, doi: 10.1145/321921.321925.

[56] H. Bunke, P. Foggia, C. Guidobaldi, and M. Vento, ''Graph clustering using the weighted minimum common supergraph,'' in *Proc. Int. Workshop Graph-Based Represent. Pattern Recognit.* Springer, 2003, pp. 235–246, doi: 10.1007/3-540-45028-9_21.

[57] F. N. Abu-Khzam, N. F. Samatova, M. A. Rizk, and M. A. Langston, ''The maximum common subgraph problem: Faster solutions via vertex cover,'' in *Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl.*, May 2007, pp. 367–373.

[58] D. Conte, P. Foggia, and M. Vento, ''Challenging complexity of maximum common subgraph detection algorithms: A performance analysis of three algorithms on a wide database of graphs,'' *J. Graph Algorithms Appl.*, vol. 11, no. 1, pp. 99–143, 2007, doi: 10.7155/jgaa.00139.

[59] S.-M. Cheng, W. C. Ao, P.-Y. Chen, and K.-C. Chen, ''On modeling malware propagation in generalized social networks,'' *IEEE Commun. Lett.*, vol. 15, no. 1, pp. 25–27, Jan. 2011, doi: 10.1109/LCOMM.2010.01.100830.

[60] J. W. Mickens and B. D. Noble, ''Modeling epidemic spreading in mobile environments,'' in *Proc. 4th ACM Workshop Wireless Secur. WiSe*, 2005, pp. 77–86.

[61] K. Ramachandran and B. Sikdar, ''Modeling malware propagation in networks of smart cell phones with spatial dynamics,'' in *Proc. IEEE INFOCOM - 26th Int. Conf. Comput. Commun.*, May 2007, pp. 2516–2520.

[62] S. Peng, S. Yu, and A. Yang, ''Smartphone malware and its propagation modeling: A survey,'' *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 925–941, 2nd Quart., 2014.

[63] C. Gao and J. Liu, ''Modeling and predicting the dynamics of mobile virus spread affected by human behavior,'' in *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw.*, Jun. 2011, pp. 20–24.

[64] C. Gao and J. Liu, ''Modeling and restraining mobile virus propagation,'' *IEEE Trans. Mobile Comput.*, vol. 12, no. 3, pp. 529–541, Mar. 2013.

[65] C. Szongott, B. Henne, and M. Smith, ''Evaluating the threat of epidemic mobile malware,'' in *Proc. IEEE 8th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2012, pp. 443–450.

[66] M. Ghallali and B. E. Ouahidi, ''Security of mobile phones: Prevention methods for the spread of malware,'' in *Proc. 6th Int. Conf. Sci. Electron., Technol. Inf. Telecommun. (SETIT)*, Mar. 2012, pp. 648–651.

[67] Y. Song, X. Zhu, Y. Hong, H. Zhang, and H. Tan, ''A mobile communication honeypot observing system,'' in *Proc. 4th Int. Conf. Multimedia Inf. Netw. Secur.*, Nov. 2012, pp. 861–865.

[68] C. Mulliner, S. Liebergeld, M. Lange, and J.-P. Seifert, ''Taming mr hayes: Mitigating signaling based attacks on smartphones,'' in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2012, pp. 1–12.

[69] F. Li, Y. Yang, and J. Wu, ''CPMC: An efficient proximity malware coping scheme in smartphone-based mobile networks,'' in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[70] H. S. Chiang and W. J. Tsaur, ''Mobile malware Behavioral analysis and preventive strategy using ontology,'' in *Proc. IEEE 2nd Int. Conf. Social Comput.*, Aug. 2010, pp. 1080–1085, doi: 10.1109/SocialCom.2010.160.

[71] S. Peng, G. Wang, and S. Yu, ''Modeling malware propagation in smartphone social networks,'' in *Proc. 12th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Jul. 2013, pp. 196–201.

[72] A. G. Miklas, K. K. Gollu, K. K. Chan, S. Saroiu, K. P. Gummadi, and E. De Lara, ''Exploiting social interactions in mobile systems,'' in *Proc. Int. Conf. Ubiquitous Comput.* Springer, 2007, pp. 409–428, doi: 10.1007/978-3-540-74853-3_24.

[73] H. Qian and Q. Wen, ''A cloud-based system for enhancing security of Android devices,'' in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Intell. Syst.*, Oct. 2012, pp. 245–249.

[74] S.-H. Seo, D.-G. Lee, and K. Yim, ''Analysis on maliciousness for mobile applications,'' in *Proc. 6th Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput.*, Jul. 2012, pp. 126–129.

[75] L. Xueming, ''Access control research based on trusted computing Android smartphone,'' in *Proc. 3rd Int. Conf. Intell. Syst. Design Eng. Appl.*, Jan. 2013, pp. 213–215.

[76] Z. Qin, Q. Zhang, X. Zhang, and Z. Yang, ''An efficient method of detecting repackaged Android applications,'' in *Proc. Int. Conf. Cyberspace Technol. (CCT)*, 2014, pp. 1–4.

[77] M. Omar and M. Dawson, ''Research in progress–defending Android smartphones from malware attacks,'' in *Proc. 3rd Int. Conf. Adv. Comput. Commun. Technol. (ACCT)*, Apr. 2013, pp. 288–292.

[78] Y.-J. Ham, W.-B. Choi, H.-W. Lee, J. Lim, and J. N. Kim, ''Vulnerability monitoring mechanism in Android based smartphone with correlation analysis on event-driven activities,'' in *Proc. 2nd Int. Conf. Comput. Sci. Netw. Technol.*, Dec. 2012, pp. 371–375.

[79] B. Markelj and I. Bernik, ''Safe use of mobile devices arises from knowing the threats,'' *J. Inf. Secur. Appl.*, vol. 20, pp. 84–89, Feb. 2015, doi: 10.1016/j.jisa.2014.11.001.

[80] P. D. Meshram and R. C. Thool, ''A survey paper on vulnerabilities in Android OS and security of Android devices,'' in *Proc. IEEE Global Conf. Wireless Comput. Netw. (GCWCN)*, Dec. 2014, pp. 174–178.

[81] A. Mylonas, S. Dritsas, B. Tsoumas, and D. Gritzalis, ''On the feasibility of malware attacks in smartphone platforms,'' in *Commun. Comput. Inf. Sci.*, vol. 314, pp. 217–232, Jul. 2012, doi: 10.1007/978-3-642-35755-8_16.

[82] M. Kuehnhausen and V. S. Frost, ''Trusting smartphone apps? To install or not to install, that is the question,'' in *Proc. IEEE Int. Multi-Disciplinary Conf. Cognit. Methods Situation Awareness Decis. Support (CogSIMA)*, Feb. 2013, pp. 30–37.

[83] W. Te-En, A. B. Jeng, L. Hahn-Ming, C. Chih-How, and T. Chin-Wei, ''Android privacy,'' in *Proc. Int. Conf. Mach. Learn. Cybern.*, Jul. 2012, pp. 1830–1837.

[84] G. Delac, M. Silic, and J. Krolo, ''Emerging security threats for mobile platforms,'' in *Proc. MIPRO 34th Int. Conv. Inf. Commun. Technol., Electron. Microelectron.*, May 2011, pp. 1468–1473.

[85] C. Yang, J. Zhang, and G. Gu, ''Understanding the market-level and network-level Behaviors of the Android malware ecosystem,'' in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 2452–2457.

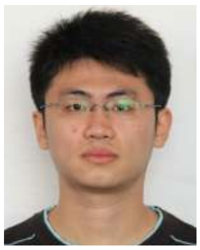
[86] S. M. Lundberg and S.-I. Lee, ''A unified approach to interpreting model predictions,'' in *Proc. 31st Conf. Neural Inf. Process. Syst. (NIPS)*, Red Hook, NY, USA: Curran Associates, Dec. 2017, pp. 4768–4777.

LONG CHEN is currently pursuing the Ph.D. degree with Beihang University under the supervision of Prof. C. Xia. He is currently the Researcher Fellow with Beijing Topsec Network Security Technology Company Ltd. His previous positions are the Director, the Deputy General Manager, the small CEO, and the Director of Innovation Studio in China Unicom. He has participated in several National Natural Science Foundations and other research projects as the Director and a Contributor. His research interests include network and information security, and intrusion detection technology.

CHUNHE XIA received the Ph.D. degree in computer application from Beihang University, Beijing, China, in 2003. He is currently a Supervisor and a Professor with Beihang University and the Director of the Beijing Key Laboratory of Network Technology. He has participated in different national major research projects and has published more than 70 research papers in important international conferences and journals. His current research interests include network and information security, information countermeasure, cloud security, and network measurement.

SHENGWEI LEI (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Beihang University, Beijing, China.

His research interests include network threat characterization and data mining.



TIANBO WANG (Member, IEEE) received the Ph.D. degree from Beihang University. He is currently a Lecturer with the School of Cyber Science and Technology, Beihang University. He has participated in several National Natural Science Foundations and other research projects as a Contributor. His research interests include network and information security, intrusion detection technology, and information countermeasure.