Journal of Computer Virology and Hacking Techniques
https://doi.org/10.1007/s11416-023-00477-y

ORIGINAL PAPER

# iOS mobile malware analysis: a state-of-the-art

**Madihah Mohd Saudi[1] · Muhammad Afif Husainiamer[1] · Azuan Ahmad[1] · Mohd Yamani Idna Idris[2]**

Received: 14 November 2022 / Accepted: 3 April 2023


**Abstract**
In earlier years, most malware attacks were against Android smartphones. Unfortunately, for the past few years, the trend has shifted towards attacks against the Apple iOS smartphone. Consequently, an in-depth analysis of the malware and iOS architecture is important to identify the best mitigation solution against malware exploitation. Hence, this paper presents a state-of-the-art deep analysis of malware against iOS smartphones. This includes comprehensive studies of malware architecture involving payload, propagation, operating algorithm, infection, and activation with underlying integration with a phylogenetic concept. Phylogenetic, borrowed from the biology field, can identify any evolution of the origin of the malware involved. To support this deep analysis of malware, a preliminary study was conducted using 12 malware samples, by focusing on social media and online banking. This took place in a controlled laboratory using hybrid analysis. The result showed that there is a way to identify the evolution of malware and as a result, a model has been developed. Based on the evaluation, 4% of mobile applications matched the patterns developed in this model. This proves that the model developed in this paper can detect any possible security exploitation related to social media and online banking for iOS mobile applications. This work can be used as guidance for other researchers working on similar interests in the future.

**Keywords** Hybrid analysis · iOS exploitation · Malware classification · Mobile malware · Online banking · Phylogenetic · Social media

## 1 Introduction

### 1.1 Background

The growing use of mobile applications has significantly changed communication and access to information such as personal data, emails, bills, and bank account details. The increasing number of smartphone users raises security concerns, such as those contributing to the growth of mobile malware on both iOS and Android platforms [1]. Malware can lead to harm, or damage data and devices, especially via online banking and social media exploitation. A report by Kaspersky discovered 676,190 malicious installation packages in the third quarter (Q3) of 2021, decreasing by 209,915 from the previous quarter and 445,128 from Q3 2020 [2]. For the first quarter (Q1) of 2021, a McAfee report stated that there were 2.34 million mobile malware cases, of which 389 were on iOS [3]. The McAfee report also stated that the number of new mobile malware incidences rose by 71% (1.35 million) in Q1 2020, while new iOS malware grew by over 50% (3,249), which is one of the reasons for the development of this paper [4]. Furthermore, in 2019, high-risk vulnerabilities were discovered in 38% of iOS mobile applications, compared to Android mobile applications with 43% [5]. In 2017, 40% of iOS malware attacks targeted financial services [6]. This raises the question of how iOS malware classification can be developed to mitigate security exploitation by attackers.

In this paper, social media exploitation is referred to as an act that causes victims to lose productivity and in which devices can be monitored and remotely controlled by the attackers. While, online banking exploitation is an act whereby victims lose financially, wherein their devices are exploited for banking information such as user ID and password.

Work by [7] reported different iOS zero-day malware in 2019, such as watering hole spyware. This type of attack

✉ Madihah Mohd Saudi
madihah@usim.edu.my

[1] Cyber Security and Systems (CSS) Research Unit, Faculty of Science and Technology, Universiti Sains Islam Malaysia, 71800 Nilai, Malaysia

[2] Department of Computer System and Technology, Faculty of Computer Science and Information Technology, Universiti Malaya, 50603 Kuala Lumpur, Malaysia

Published online: 18 May 2023

is able to access confidential information such as iMessage, photos, and global positioning system (GPS) locations. While FinSpy was able to steal detailed personal information including SMS/MMS messages, telephone call records, emails, contacts, videos, files, and GPS locations [8], whereas Exodus could exfiltrate information including contacts, audio recordings, images, videos, GPS location, and device information [9]. In March 2021, Facebook closed down a hacking attempt by Evil Eye that used the social media platform to escalate Insomnia malware, a malicious program, to track Uyghur Muslims in China's Xinjiang region [10].

In June 2020, the FBI released an official statement claiming that cybercriminals were taking advantage of the high usage of mobile banking applications. Since 2020, there has been a 50% rise in mobile banking. The FBI also warned people to be vigilant when installing applications for smartphones and tablets since some of them may be harmful. Banking trojans, which are harmful applications that masquerade as other applications such as games or utilities, are being used by cybercriminals to steal banking information [11]. This malware can steal confidential information, yet currently, there have been few studies examining the detection of iOS malware. The studies of [12–16] discussed iOS exploitation implications in general. Based on the cases above, the question arises as to what features are involved in developing an iOS malware detection model for social media and online banking exploitation.

This paper bridges this research gap by developing a new iOS mobile malware detection model. It consists of 30 new correlation patterns between malware behaviour and iOS architecture. Malware behaviour involves infection, activation, payload, operating algorithm, and propagation, whereas iOS architecture focuses on Cocoa Touch, media layer, core services, and core operating system (OS). The primary concern for iOS architecture is possible exploitation in its layers.

Many bio-inspired algorithms have been applied in different areas of cybersecurity especially in malware detection, such as Genetic Algorithm (GA), Fuzzy Logic (FL), Negative Selection Algorithm (NSA), and Danger Theory (DT) by [17–20]. In comparison with a bio-inspired algorithm, the phylogenetic method will be able to detect the root origins of malware evolution. It concerns the history of its evolution and is connected to a tree diagram with various organisms and taxonomic groups [21]. The phylogenetic concept underpins the development of this paper's mobile malware classification and was used as the basis for producing a new iOS mobile malware detection model. The strength of this concept is its capability to identify the root of its ancestry, and a prediction can be made for any evolution of iOS malware. As a result, possible social media and online banking exploitation can be detected via this proposed model.

The main challenges for this paper are the rise of malware exploitation on social media and online banking, and a lack of analysis and mitigation solutions for iOS platforms, including jailbreak and iOS architecture exploitation. Few studies have been conducted in relation to iOS in comparison to Android, whose studies include [22–28]. The techniques used for previous studies were based on audio signal processing, artificial neural networks, feature selection, app similarity graph, feature weighting, combination API, and permission and formal methods. However, these techniques were only applied to the Android platform, and few studies were conducted on iOS exploitation. In addition, the most common method of iOS exploitation is using jailbreaks such as work by [29–32]. A jailbreak is used to eliminate AppStore restrictions, achieve full authority, add features, view the iOS device log, and bypass GSM providers and network consumption restrictions. Therefore, this paper has developed a model to detect malware on the iOS platform.

Previous studies, such as [12–16], only focused on static or dynamic analysis. Work by [12] used the static approach, which entails a grayscale histogram and reverse engineering of the application code to obtain the function from the source code, while [13] introduced five conditions the application must meet to determine a potentially exploited channel. By combining dynamic (low-level debugger and debug server) and static (reverse analysis tool IDA) analysis, the authors traced the data object's origins back to the first triggered method for intercepting communication data. Work by [14] proposed a system that provides DNS and URL filtering using several backlists on mobile browser of iOS and Android, but it used manual evaluation to decide on the blacklist. Work by [15] introduced a threat model to test and analyse applications in Android and iOS, using a man-in-the-middle attack (MiTM) but only focusing on mobile banking applications. The authors used static and dynamic analysis for the testing process.

Work by [16] focused on static analysis only, using feature vector extraction to count the number of occurrences of a specific group of opcodes and a machine-learning algorithm to detect iOS mobile malware. Work by [33] suggested using Image Similarity-based Statistical Parameters (ISSP) and a visualization technique in which the disassembled malware code is turned into grayscale images. A Faster Region Proposals Convolution Neural Network (F-RCNN) classifier is used to train a vector made up of grayscale images with statistical characteristics. The proposed technique has an overall average classification accuracy of 98.12%. To identify and categorize Android malware, work by [34] provided a hybrid technique that extracts various features using static and dynamic malware analysis. These researchers also produced two dataset for detecting Android malware (dataset 1) and classifying its families (dataset 2). According to the experimental results, hybrid classification outperforms static and dynamic data in terms of detection and classification.

Static and dynamic analyses focus only on certain features' limitations and can miss a significant part of the detection. Therefore, this paper has implemented hybrid analysis with phylogenetic as the underlying concept for iOS malware detection to increase the accuracy rate.

In comparison, other works, such as [35–41], have used process mining, a fuzzy clustering algorithm, a persistent phylogeny tree model, a discrete-time Markov chain (DTMC), a Bayesian network algorithm, and an extension of the graphical lasso. These techniques were applied only on the Android and Windows platforms. Thus, in this paper, the phylogenetic concept has been implemented to predict the future evolution of iOS malware by developing a classification used as input for the iOS mobile malware detection model. Based on the previous studies mentioned above, this paper has overcome the existing challenges by focusing on the iOS platform, using hybrid analysis and the phylogenetic concept.

### 1.2 Contributions

The contributions of this paper can be summarized as follows.

- It presents insight into works related to social media exploitation, online banking exploitation, iOS, and phylogenetics.
- It provides an iOS malware classification, which is developed using a combination of functions extracted from malware. The classification is then mapped into phylogenetic. This consists of 30 new correlation patterns between malware behaviour and iOS architecture. From 30 malware classifications, 22 can be used against social media and online banking exploitation.
- It also provides an iOS malware model detecting social media and online banking exploitation. During the evaluation, this model successfully detected seven out of 150 mobile applications with possible exploitation vulnerabilities related to social media and online banking.

### 1.3 Organization

The remainder of this paper is organized as follows. Section 2 discusses the relevant works used in this paper. This consists of social media exploitation, online banking exploitation, iOS architecture, iOS mobile malware exploitation, iOS mobile malware detection, phylogenetics, malware behaviour, iOS version, surveillance features, and accuracy. Section 3 presents the methodology, while Section 4 consists of the findings and malware detection modelling. Section 5 presents the evaluation of the proposed model, while Section 6 presents the discussion, limitations, and improvements. Section 7 presents the conclusion.

## 2 Related works

This section discusses the literature on social media exploitation, online banking exploitation, and iOS, focusing on the platform's architecture and mobile malware exploitation and detection. Then, phylogenetic theory and its approach towards iOS are discussed. The chapter also discusses malware behaviour, the iOS version, surveillance features, and accuracy.

### 2.1 Social media exploitation

As of March 2021, Facebook had closed down a hacking operation using the social media site to spread Insomnia malware, a malicious program used to monitor Uyghur Muslims in China's Xinjiang region. A group of hackers, collectively known as Evil Eye and linked to Chinese government entities, distributed iOS and Android malware on various websites to monitor activists, journalists, and protestors. The spyware known as Insomnia works on any web browser operating any version of iOS 10 and 11, as well as iOS 12.0, 12.1, 12.3, 12.3.1, and iOS 12.3.2, according to research published by the security firm Volexity. Once installed, this spyware gains access to users' contacts, location, message data, and information from third-party applications [10].

Work by [42] discussed the common cybercrimes, mitigation techniques, and protection related to social media. They provided recommendations and techniques to prevent cybercrime regarding research studies. They declared that common types of cybercrime are spamming, hacking, malware, DoS attacks, phishing and social engineering, online identity theft, and cyberstalking. Work by [43] presented a detailed review of the most recent and relevant research papers on social media security and privacy, as well as the types of threats and attacks that affect users, released between 2018 and 2020. This work contributed to a firmer grasp of important factors in social media influencing security and privacy.

Work by [44] detailed several situations involving online social network threats and their remedies, including using various models, frameworks, and encryption approaches to protect social network users from multiple attacks. They presented various solutions and conducted a comparative analysis of different studies to understand the survey better. Work by [45] describes how cyberattacks are criminal attempts to infiltrate a user's or an organization's network to steal confidential or personal data for personal benefit and how social media users and organizations are exposed to cybercrime. The result suggests preventative actions that can slow or minimize the number of cybercrimes perpetrated against individuals and businesses. Work by [27] developed mobile malware classifications based on API and permission-based call logs, audio, and GPS that can be used for social media malware detection. This classification is beneficial to

application development. The test results show that 16% of the mobile applications were classified as vulnerable to call log exploitation, 13% to audio exploitation, and 9% to GPS exploitation.

In conclusion, the works above discussed social media in terms of security and privacy but did not detail the OS of the social media application, either iOS or An-droid, except for [27], which focused on the Android environment.

## 2.2 Online banking exploitation

In June 2020, the FBI released an official statement declaring that cybercriminals are taking advantage of the high usage of mobile banking applications. Since the beginning of 2020, there has been a 50% rise in mobile banking. The FBI also warned people to be vigilant when installing applications for smartphones and tablets since some of them may be harmful. Banking trojans, which are harmful applications masquerading as other applications, such as games or utilities, are being used by cybercriminals to steal banking information. They also build fraudulent applications that imitate major financial institutions' official applications to trick users into entering their login information [11].

Work by [46] proposed an Intrusion Response System (IRS) based on a network's graphical network security model and a game-theoretic model of cyberattacks. The infection techniques, displayed behaviour, and network communication patterns of two popular banking trojans from the previous decade, Zeus (along with its companion ZitMo) and Emotet, were presented to provide additional insight into the future of banking trojans and malware in general. The study focused on Windows OS and Android. Work by [47] provided a summary of several studies published between 2009 and 2020 in 26 developed and developing countries from more than 24 different sources, with an average sample size of 460 users, to comprehend the adoption of mobile banking from the view of consumers. In addition, there have been a few other research publications on mobile banking, but there has been no quantitative survey or qualitative study, including in-depth interviews with professionals or users. This study did not include details on the OS of mobile banking, either iOS or Android.

Furthermore, work by [48] presented a model for mobile banking application selection that was proposed based on a combined fuzzy best-worst method (fuzzy-BWM) and fuzzy technique for order of preference by similarity to the ideal solution. The model can help financial institutions and customers overcome the difficulties of selecting an effective mobile banking application. Their approach has limitations in that expert decisions during pair-wise comparisons may be prejudiced; the existing BWM might be expanded to a new context to overcome the suggested model, and it does not entail details of the OS of mobile banking, either iOS or Android.

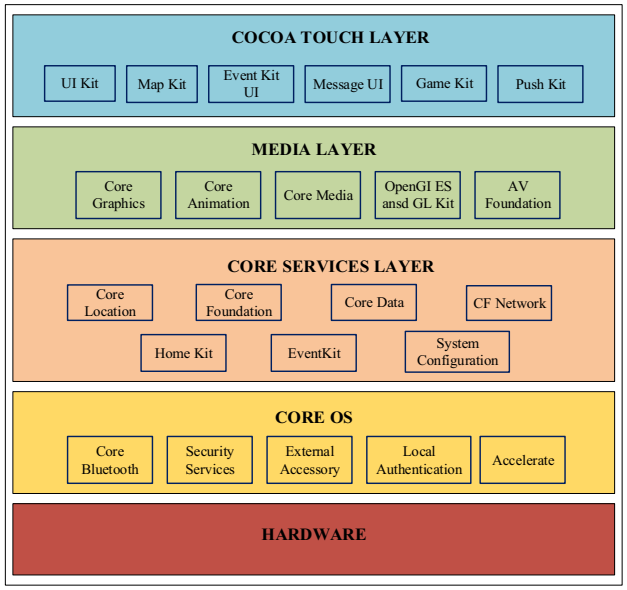


**Fig. 1** General Components Architecture of iOS

Work by [49] discussed several significant aspects of mobile banking in terms of threats, security requirements, and security solutions regarding limitations and improvements. This work did not include details of the OS of mobile banking, as being either iOS or Android. Work by [28] proposed applying a formal methods-based approach to detect banking malware in the Android environment. It reached precision and recall equal to 1 when evaluated with real-world Android applications. This work focused on the Android environment only.

In conclusion, several works are related to online banking but lack focus on the iOS environment.

## 2.3 iOS architecture

On 6 March 2008, Apple launched the first beta and a new operating system name: iPhone OS. Apple rebranded iPhone OS as 'iOS' in June 2010. The basic architecture for iOS is divided into four layers, which are Cocoa Touch, media, core services, and core OS [50]. Every layer has its own functions. The lower layers, such as the core services and core OS, manage basic services in iOS, while the upper layers, such as the media and Cocoa Touch, handle the user interface and advanced graphics [51]. Figure 1 presents the general components of the iOS architecture.

As shown in Fig. 1, the hardware layer is composed of the physical chips fused with the iOS circuit. The core OS layer is the lowest layer that interacts with the hardware directly. It has an operating system that is above the other layers. This layer handles memory management (allocation

and de-allocation, once the application has finished), file management, and network management. The core service layer serves as the foundation on which the other layers are built. It contains several features, including data protection, SQLite database, file-sharing support, iCloud storage, XML support, and in-app purchases. Media layers manage graphics, audio, and video capabilities. Graphic framework, audio framework, and video framework are the three frameworks that form the media layer. These frameworks help access photos and videos stored on the device and manipulate images using filters and 2D sketching. The Cocoa Touch layer defines the look and feel of iOS applications by providing key frameworks. This layer oversees multi-tasking, touch-based input, push notifications, and many high-level system services [1]. The main concern related to iOS architecture is possible exploitation in the layers. This exploitation is also related to social media and online banking. Hence, these exploitations are the main focus of this paper and the reason for introducing the malware classification and malware detection for the iOS platform.

### 2.4 iOS mobile malware exploitation

There are many previous studies related to mobile malware exploitation for Android platforms. More researchers have focused on Android due to its open source policy, and there is a lack of work on a security solution for iOS platforms. Currently, most research is focused on attacks on jailbroken devices. Table 1 shows previous studies associated with iOS mobile malware exploitation.

Based on Table 1, it can be concluded that most attacks focused on jailbroken de-vices. Jailbreak is a technique to penetrate Apple products' iOS operating system. Users may use this approach to achieve full authority, view the iOS device log, and bypass restrictions of GSM providers and network consumption [53]. However, some attacks exploit non-jailbroken devices, where the attackers utilize iOS and private API vulnerabilities to attain control functions. When jailbreak was used, the iPhone's surveillance features were indirectly affected. This paper has focused on the surveillance features, which are SMS, call log, GPS, audio, and camera and related them to iOS mobile malware exploitation. From this, a mobile malware classification has been developed.

### 2.5 iOS mobile malware detection

There has been a demand for mitigation solutions for iOS malware detection for the past few years. Table 2 shows previous studies on iOS mobile malware detection.

Based on the Table 2, this paper has overcome the challenges of existing iOS mobile malware detection by using hybrid analysis and phylogenetic concepts to predict the future evolution of iOS malware.

### 2.6 Phylogenetic

Phylogenetic is beneficial in identifying the sources of evolving malware genes. It deals with the history of evolution, depicted in a tree diagram of various organisms and taxonomic groups [21]. Several types of phylogenetic tree models exist, including a minimum spanning tree, a persistent phylogenetic tree, and a dendrogram [38]. Figure 2 is an example of a phylogenetic diagram in general.

Figure 2 represents the ancestral population from which all other species descend is a root. A node represents an ancestral population branching point. The taxa or known as the terminal of the branch population (for example taxa A, B, C, D or E) represents the population that is used to designate the terminals located at the top of each branch.

Apparently, there are several works by [17–20] that used the bio-inspired concept to produce cybersecurity solutions. Table 3 shows a summary of the comparison of the different bio-inspired algorithms.

Work by [35] proposed adopting the mining method to identify a malicious Android app's belonging family. The authors used the process mining technique to automate the process and obtain accuracy ranging between 0.882 and 0.987 for the family recognition task, examining 12,604 Android application datasets. Another study that used process mining is that of [36]. The authors suggested a complex malware detection and monitoring approach based on process mining techniques. This extracted a declarative model called SEF from malware traces representing a fingerprint of their dynamic behaviour. The method was tested for malware identification on over 1,200 compromised application dataset in ten malware families.

Work by [37] used the process mining technique to detect a malware phylogeny model by analysing the system call traces recorded during application execution. It was combined with a fuzzy clustering algorithm to determine the malware samples' derivation degree, where 4,000 infected applications across 39 malware families were analysed. The results showed that the method could cluster by comparing families with similar behaviour during malicious work. Work by [39] described a formalism inspired by phylogenetic methods to determine the similarity between trace data provided by malware families. In comparing the discrete-time Markov chain (DTMC) representation from the traces, the Kullback-Leibler Divergence (KLD) rate measures the proximity of an unknown malware trace to several malware families. The author showed the links between traces in the form of a network. Weights on the network edges measured the correlation between the nodes.

Work by [40] stated that malware phylogenetic can help researchers evaluate the nature of a new piece of malware code more quickly. The authors examined the use of expert knowledge of how the malware was developed within a

**Table 1** Previous Studies Associated with iOS Mobile Malware Exploitation

| Author | Strength | Weakness | Jailbreak/ Non-jailbreak |
|---|---|---|---|
| [29] | After jailbreak, the user may add features, parameters, and unfamiliar commands to the operating system<br>Enables users to create previously unrealized metadata relationships | Apple facilitates metadata development, allowing third-party application developers to use metadata and their respective analytical endeavours | Jailbreak |
| [30] | Voucher swap uses the semi-tethered jailbreaking approach to assess the impact of jailbreak on data integrity on iPhone | Change the system data by making a fake kernel task | Jailbreak |
| [31] | Heuristic model based on fingerprint and its web knowledge to identify the infected applications | Fingerprint hard to define app name | Jailbreak |
| [32] | Jailbreak can assist users in eliminating AppStore restrictions and openly downloading several applications | Jailbreak fails several data security policies, which makes it easier for users to steal their privacy and data using malicious software | Jailbreak |
| [52] | Component clustering with the same technique of code comparison (Centroid)<br>LibFinder automatically analyses the disassembled app | Precise mapping (one or several APIs for a specific set of activities) of a relationship is complicated and cannot be conducted easily with a dictionary | Non-jailbreak |

**Table 2** Previous Studies Associated with iOS Mobile Malware Detection

| Author | Strength | Challenge | Platform |
|---|---|---|---|
| [12] | A method aimed at differentiating between malicious and legitimate samples in the mobile environment and distinguishing the malware family and family variant (grayscale histogram) | Nonessential to reverse engineer the application code to obtain functions from the source code | Android & iOS |
| [13] | Determine whether a legal application is a potential channel or not based on five conditions<br>The application has network communication capability<br>People can understand the communication message of the application<br>The application process space can load the Cydia substrate framework<br>The application has inter-process communication, such as message queues and notifications<br>The application can access the file system | A limited number of dataset (iMessage only) | iOS |

**Table 2** (continued)

| Author | Strength | Challenge | Platform |
|---|---|---|---|
| [16] | A feature vector extracted by static analysis<br>Precision equal to 0.971 and a recall equal to 1 using the One Rule algorithm | Focuses on static analysis only | iOS |
| [54] | Review on security technologies of iOS and Android | Android technologies will allow further security efforts to continue precautionary research | Android & iOS |

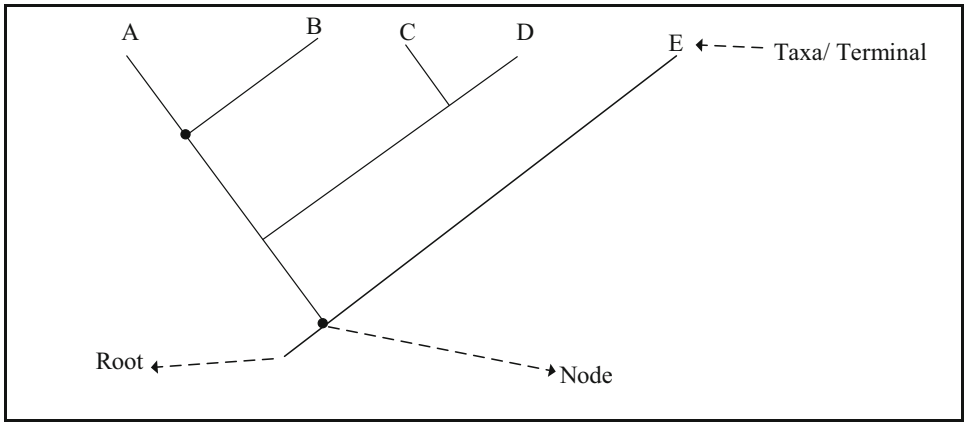


**Fig. 2** Example of General Phylogenetic Diagram

**Table 3** Comparison Summarization of The Different Bio-Inspired Algorithms

| Algorithm | Concept Description | Limitation |
|---|---|---|
| Genetic Algorithm (GA) | GA is an artificial intelligence and computer search heuristic process | GA finds continuous optimization of answer times difficult<br>Real-time GA implementations are constrained due to random solutions and convergence |
| Fuzzy Logic (FL) | FL is a standard generalization logic in which a concept can have a certain degree of truth | If the solution is not understood, it cannot be resolved via fuzzy logic<br>Fuzzy logic systems are expensive and require rigorous testing |
| Negative Selection Algorithm (NSA) | NSA is a non-self-pattern used to create a method for detecting anomalies | Scalability and coverage are significant barriers to the effectiveness of this algorithm as an efficient detection method |
| Danger Theory (DT) | DT states that the immune system distinguishes between dangerous and safe pathogens by detecting these pathogens or alarm signals from wounded or stressed cells and the tissues | It operates only with self-and non-self-systems yet is complex<br>Dangerous signals were difficult to identify |
| Phylogenetic | It deals with the history of evolution, often depicted in a tree diagram of various organisms and taxonomic groups | Not applicable |

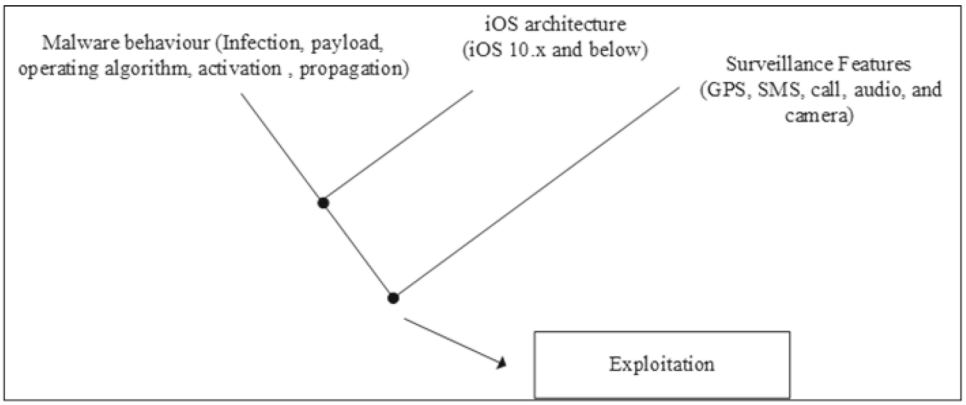


**Fig. 3** Mapping phylogenetic diagram to iOS malware classification

family. The authors previously provided this information in a Bayesian network algorithm as novel. The benefit of Bayesian network learning is the combination of expert knowledge and statistical data. Work by [41] introduced an extension of a graphical lasso, which finds a weighted combination of static and dynamic views to produce a phylogenetic graph for a family of programs. The findings showed that the authors could locate phylogenetic charts effectively and that combining several views to optimize the equation increases efficiency dramatically relative to any single view and many baselines, such as minimum spans.

Earlier in Fig. 2, the general diagram of the phylogenetic diagram works is presented. Whereas Fig. 3 shows the mapping of the phylogenetic diagram to iOS malware classification based on the phylogenetic concept in Fig. 2. In Fig. 3, the taxa of a malware classification consisting of malware behaviour, iOS architecture and surveillance features are illustrated. Then, this malware classification will be used to identify possible malware exploitation. Additionally, a further explanation of how the phylogenetic is mapped with iOS malware classification can be referred to in Table 4.

Then, based on the developed malware classification for iOS, further investigation is carried out to identify the related previous studies with phylogenetic for malware detection. As a result, Table 5 shows a summary of existing phylogenetic works.

Based on Table 5, it can be concluded that phylogenetic can be used as a detection solution in predicting future iOS mobile malware. Therefore, this paper has developed a new iOS mobile malware detection classification inspired by phylogenetic. Three (3) features are mapped to phylogenetic: malware behaviour, iOS version, and surveillance features. The details of these elements are explained in the following section.

**Table 4** Mapping phylogenetic to mobile malware classifications in iOS

| Phylogenetic | Mobile malware |
|---|---|
| Focuses on the mutative history and the relation between various organisms and taxonomic groups, depicted in a tree diagram | iOS malware genes evolution focuses on integrating malware behaviour, exploiting vulnerabilities, and surveillance features |
| | Malware behaviour consists of five (5) main elements: infection, activation, payload, operating algorithm, and propagation, while surveillance features consist of SMS, call logs, cameras, audio, and GPS |
| | The exploitation of vulnerabilities is based on the iOS version and exploitation chain. All elements are shown in a tree diagram |

## 2.7 Malware behaviour

Malware behaviour comprises five (5) main elements: infection, activation, payload, operating algorithm, and propagation. These are essential for dynamic analysis to identify malware based on its behaviour [55]. For the enhancement of this paper, this worm classification was integrated with the phylogenetic concept as depicted as input α, as mentioned above.

Infection is the act of malware infiltrating a computer, system, or files [56]. Infection can be accomplished in several ways via the host or network. The host method entails that the malware is in its initial position and typically replicates itself and waits for the user to transfer it without permission. To activate is to make it operational. In a malware context, acti-

**Table 5** Existing works on phylogenetic

| Author | Strength | Challenge | Method |
|---|---|---|---|
| [35] | Automatically detects temporal logic properties designed to detect Android malware families and track the phylogenetic tree | More chances to initiate malicious payload from a collection of 25 events submitted to the application under analysis | Process mining |
| [36] | Using process mining techniques, the programme calls traces gathered from a mobile application for characterizing its conduct to classify associations and repeated execution patterns | The construct validity analysis reveals that the extraction of syscall traces in some instances could be ambiguous | Process mining |
| [27] | Developing mathematical formulation for mobile malware classification using phylogenetic concepts by integrating malware behaviour, vulnerability exploitation, and mobile phone surveillance features | Evaluated using one mobile malware dataset | Phylogenetic concept |
| [37] | The syscalls of a malware programme can be modelled to get a malware fingerprint with several associations and repeated execution patterns | Ambiguity in the process of syscall extraction | Fuzzy clustering algorithm |
| [38] | Nodes correspond to input instances, and edges represent the gain or loss of functional characters | Restrictions of persistent phylogeny tree, such as conflict characters | Persistent phylogeny tree model |
| [39] | Computationally intensive due to pairwise KLD and JSD track | Parallel hardware will theoretically lead to an efficient calculation | Discrete-time Markov chain |
| | Calculation | | |
| [40] | Bayesian network algorithm to learn a directed acyclic graph by statistical inference of conditional dependence from observational data, with an informative prior to partial variables ordering | Graphs are difficult to construct from data and need expert information | Bayesian network algorithm |
| [41] | Predicts the phylogenetic graph by discovering a sparse matrix of precision based on the combined matrix of the kerne | Phylogenetic graphs found manually by domain experts are not 100% accurate | Extension of graphical lasso |

vation is the process for malicious activity to begin. There are four (4) types of basic malware activation: no activation, human trigger, schedule process, and self-activation. In the no activation scenario, the malware remains in and does nothing, taking up the hard drive's storage space. The slowest means of activation is the human trigger. It needs to wait for the user to conduct some activity that triggers malicious action. As mentioned by [57], a malware creator could program malicious behaviour to only activate after a particular amount of time has passed since the application was executed. For a scheduled process, the malware creator must set up the malware to be triggered on a particular date and time. The quickest method to activate malware is self-activation, where the malware executes immediately.

As defined by [58], payload is the action whereby the vulnerability is conducted separately from the main behaviour. The payload can vary from stealing credential information to removing hard disk contents. Several types of payloads can harm devices. Some payloads allow the malware to access the embedded backdoor for the attacker's purpose. A backdoor is a process wherein an attacker bypasses the normal authorization and is given unauthorized access [59].

The operating algorithm is considered to be a malware evasion method. There are several types of malware mechanisms, such as polymorphic, stealth, terminate and stay resident, and anti-antivirus. Polymorphic malware is a type of malware that modifies its identifiable features regularly to avoid detection [60]. The most prominent mechanism is stealth, wherein the malware slowly spreads, not producing any abnormal contact pattern, making identification difficult. As malware variants grow exponentially year-on-year, there is an immediate need to counter stealth malware techniques [61]. Terminate and stay resident is a software program that remains in memory until needed and executes a dedicated function [62]. Anti-antivirus is a technique used to aggressively attack software, making antivirus analysis difficult for researchers, or preventing malware detection by antivirus software.

Malware may attempt to replicate and distribute itself to a specific host or network. As mentioned by [63], malware propagation is the act of malware spreading across a network from one host to another. For example, random scanning, sequential scanning, or passive scanning can be performed in several ways. Random scanning is the most popular among this type of malware. The malware is linked to a particular IP address, and it attempts to scan the IP and commence a connection to propagate the malware.

It can be concluded that all malware exhibits certain behaviour during the exploitation of users' devices, such as infection, activation, payload, operating algorithm, or propagation. The behaviour acts according to the malware's functionality to steal information or be linked to particular IP addresses. In this paper, some malware exhibits similar behaviour: phishing for payload, host for infection, stealth for the operating algorithm, self-activation for activation, and passive monitoring for propagation.

## 2.8 iOS version

Apple supports its customers by providing them with the latest iOS edition. The version must be updated to ensure the user is adequately protected against existing security problems, such as bug fixes, patch vulnerabilities, and issues with the latest features [64]. Beginning with iOS 7.1.2, Apple upgraded its 32-bit processor to a 64-bit processor to improve its applications' performance and graphics. Table 6 shows the iOS version and processor used.

**Table 6** iOS Version and Processor Used

| iOS Version | Processor (Acorn RISC Machine) ARM) |
|---|---|
| 3.1.3 | 32 bits |
| 4.2.1 | |
| 5.11 | |
| 6.1.6 | |
| 7.1.2 | 32/64-bit |
| 9.3.5 | |
| 9.3.6 | |
| 10.3.3 | |
| 10.3.4 | |
| 12.5.3 | 64 bits |
| 14.5.1 | |
| 14.6 beta 2 | |
| 15.0 | |

Malware writers use vulnerabilities in the iOS version to exploit devices that affect productivity, personal information, and victims' finances. This includes the NSO Group's Pegasus spyware that has successfully exploited "zero-day" vulnerabilities affecting the media player, which unidentified bugs or flaws in the mobile phone's operating system that the manufacturer has not fixed. This spyware hacks and monitors iPhones used by potential targets. It is used as a zero-click attack and impacts a range of iPhone and iPad models, such as iPhone 5s, iPhone 6, iPhone 6 Plus, iPad Air, iPad mini version, and iPod touch [65]. To overcome the exploitation of these zero-day vulnerabilities, Apple released iOS 12.5.5, which includes a patch for a CoreGraphics vulnerability that allows maliciously created PDFs to execute arbitrary code on a target device. From 2007 to 2021, there were 2,443 reported vulnerabilities, consisting of denial of services (DoS), code execution, overflow, and memory corruption, as well as others. In 2021 there were 232 vulnerabilities, including 14 DoS, 111 code executions, 22 overflows, 23 memory corruptions, five cross-site scripting, three directory traversals, seven bypasses, six information gains, and seven privilege gains [66]. From the information above, it can be concluded that vulnerabilities in the iOS version can be exploited by attackers and cause damage to users.

## 2.9 Surveillance features

The smartphone has five (5) main surveillance features: SMS, call log, GPS, audio, and camera. Some applications that exploit SMS will anonymously charge users to send premium text messages. These can also prevent SMS operations from device operators to avoid users receiving fee messages [67]. Work by [68] tested 50 anonymous mobile applications from

the Google Play store, of which 36% matched the proposed classification, composed of 16 new SMS Android package index (API) classifications that detect SMS exploitation. Call log is one of a phone's main functions that lead to malware attacks. The attackers typically use the call log to gain useful contact list information and call history. According to study results by [60], 7% of the mobile applications evaluated were linked to possible call log exploitation, with mobile applications from the communication, entertainment, and game categories obtaining the highest scores.

A satellite navigation device used to determine the land location of an object or unit with GPS is known as a global positioning system. As stated in the work of [69], GPS is one of the surveillance features of a mobile phone that attackers frequently target. As a result, 10% of the mobile applications evaluated were considered to be high-risk and vulnerable to attack. Most attackers use GPS to exploit smartphone malware to gain information such as satellite data and track a victim's movements [70]. Work by [71] indicated that audio and camera could be exploited, wherein new malware was found, allowing attackers to steal data, record audio and video, and even infect the device with ransomware. In conclusion, malware writers commonly use surveillance features to exploit devices to collect the credential information of the users. These features are mapped by the phylogenetic concept, malware behaviour, and the iOS version.

### 2.10 Accuracy

Accuracy is one parameter for classification evaluation. Informally, accuracy is the percentage of the classification's predictions. The following definition refers to accuracy in evaluating the pattern developed in detecting social media and online banking exploitation:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ numbers\ of\ predictions}$$

## 3 Methodology

Figure 4 shows the detailed process from the beginning to the end of this paper.

This work started with data collection, which comprised of downloading malware dataset from the Contagio dump website. The laboratory environment was set up by installing Mac OS Catalina as the operating system. The hopper disassembler and frida dump tool were run in the virtual machine. Then, the malware dataset were cleaned, and a hybrid analysis was conducted. For this analysis, several steps were necessary to unencrypt and decipher binary code using frida dump. This step could be skipped if the malware sample was already in .deb or. dylib format. Next, the parent process was identified and retrieved from each iOS application. Then, the malware files were monitored and documented. The functions for each feature were identified during this step. From the malware sample, classifications were created based on their malicious payload. The classifications were integrated into the phylogenetic concept for the classification detection process. Lastly, for evaluation, the classification developed was compared with 150 applications from the AppStore and third-party platforms to identify the applicability of the patterns developed with current iOS applications that possess malicious scripts related to social media and online banking exploitation.

Figure 5 shows the lab architecture used, while Table 7 shows the software and hardware used.

The similarity between iPhone 5S and the latest iPhone is the 64-bit architecture and processor. This paper used the iPhone 5S because it was easier to jailbreak the device to acquire full access to the operating system's root and all its features. For this paper, 12 malware families consisting of 50 malware dataset were collected from the Contagio website for training, while 150 applications were taken from the AppStore and third-party platforms for testing. Several works, such as those by [16, 72, 73], have used the Contagio dataset as training and evaluation samples. Based on the previous studies and implications on iOS exploitation, the 12 malware families were selected due to their capacity to perform iOS exploitation. By choosing this malware, the fundamental concept of iOS architecture exploitation was captured, and by integrating phylogenetic, future malware evolution can be predicted.

To detect malware, several variables have been identified for the formulation of malware detection. This includes three (3) main variables for malware phylogenetic: malware behaviour, iOS architecture, and surveillance features. The formula is as follows:

- Let $\alpha$ be malware behaviour $i$, where $\alpha = \bigcap_{i=1}^{p} \alpha_i$, $\beta_j$ be iOS architecture $j$, where $\beta = \bigcup_{i=1}^{m} \beta_i$, and$\gamma$ be a surveillance feature $k$, where$\gamma = \bigcap_{i=1}^{p} \gamma_i$.
- Let M be the malware classification and T be the iOS devices. S is the detection model, and it can be defined as the following function:

$$f(\mathrm{M},\ \mathrm{T}) = \mathrm{S}, \tag{1}$$

where

$$M(\alpha, \beta, \gamma) = \alpha + \beta + \gamma \tag{2}$$

$$\boldsymbol{f(Mi, Tj) = Sij} \tag{1.1}$$

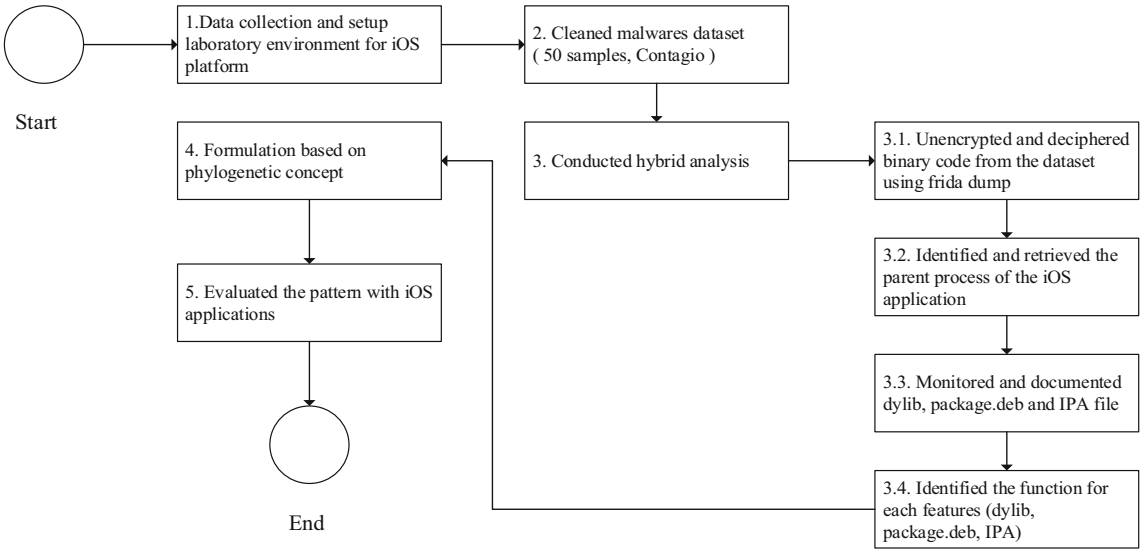


**Fig. 4** Detail Research Process

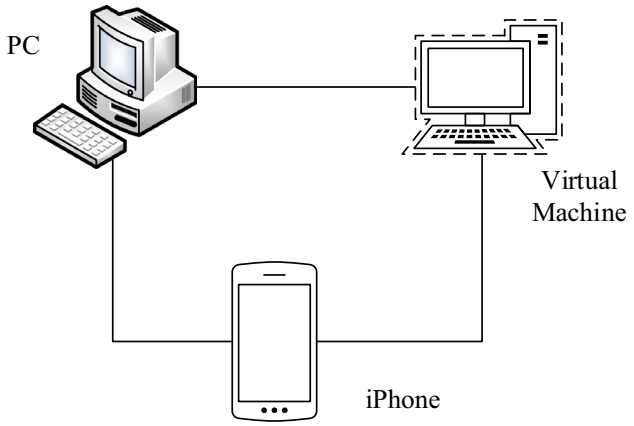


**Fig. 5** Lab architecture

**Table 7** Software and Hardware Used

| Software/Hardware | Function |
|---|---|
| Hopper Disassembler | It acts as an iOS reverse-engineering too |
| Personal Computer (PC) | To run the experiment and hybrid analysis |
| iPhone 5S | To run iOS applications for evaluation purposes |
| VMWare Workstation 14 | It acts as an emulator to run MacOS |
| MacOS Catalina | Mac operating system used for the experiment |
| Frida dump | To get decrypted IPA from a jailbroken device |

- where, M represents malware classification, T represents iOS devices, and S is the detection model.

$$M(\alpha, \beta, \gamma) = \alpha + \beta + \gamma$$

$$\alpha = \alpha_1 \cap \alpha_2 \cap \alpha_3 \cap \alpha_4 \cap \alpha_5 \tag{2.1}$$

$$\beta = \beta_1 \cup \beta_2 \cup \beta_3 \cup \beta_4 \cup \beta_5 \tag{2.2}$$

$$\gamma = \gamma_1 \cup \gamma_2 \cup \gamma_3 \cup \gamma_4 \cup \gamma_5 \tag{2.3}$$

$$\begin{matrix} M_i & \beta & \gamma \\ \vdots & \ddots & \vdots \\ M_n & \ldots & \gamma_n \end{matrix}$$

- where $\alpha_1 - \alpha_5$: payload, infection, operating algorithm, activation, and propagation.

$\beta_1 - \beta_5$ : iOS 10.x, iOS11.x, iOS12.x, iOS13.x, iOS14.x, iOS15.x.

$\gamma_1 - \gamma_5$ : SMS, call log, GPS, audio, and camera.

## 4 Findings

This section further analysed and classified all the extracted functions using the reverse-engineered malware dataset. Then, the extracted functions used for exploitation by the malware were cross-checked and mapped with the iOS framework, as summarized in Table 8. This table is significant for identifying and classifying functions; it can either

**Table 8** Function And Framework Related to Exploitation

| Malware | iOS Framework | Functions | Explanation |
|---|---|---|---|
| Unflod | Security | _replace_SSLWrite | It hooked the Security. frame into SSLWrite and searched the buffer for some strings showing the existence and password of the Apple-ID |
| | | | It attempted to connect the stolen information to Chinese IP addresses 23.88.10.4 and 23.228.204.55 |
| | | | The iOS framework involved: Security framework, located at the Core OS layer |
| Spad | Mobile Substrate | MsHookMessageEx | This function must be implemented from the Cydia substrate to make it possible to change the advertisement ID |
| | | | The iOS architecture involved is related to UI Kit, located at the Cocoa Touch layer |
| | Objective-C Runtime | InstancesRespondToSelector: | Implementation of the hook in this function |
| | | | Its instants the malware writer to the 'GADBannerView YouMi' item. GAD stands for Google Ads, and if examined the software for this feature, it refers to the AdMob package, not to YouMi. There is web information on GADBannerView in the AdMob SDK |
| | | | The attacker could change the ads id after MsHookMessageEx has been implemented |
| | | | The iOS architecture involved is related to UI Kit, located at the Cocoa Touch layer |
| Inception | Foundation | ObjectForKeyedSubscript | It collected The Integrated Circuit Card ID (ICCID)from ro/var/wireless<br>The iOS architecture involved is Foundation, located at the Core Services layer |
| | | NSMutableDictionary-> setObject(_:forKey:) | It collected the victim's address book content |
| | | ObjectForKeyedSubscript | It collected the device battery level, device platform, ICCID, international roaming edge, phone number, carrier bundle name, etc |
| | | RemoveItemAtPath:error: | Deleted deb file |
| XcodeGhost | Foundation | ObjectForKeyedSubscript | It collected system and app information such as timestamp, OS Version, device type, language, and others<br>The iOS architecture involved is Foundation, located at the Core Services layer |
| | | Connection: NSURL URLWithString:@" | It made a connection with the c2 server |
| Wirelurker | System Configuration | ReachabilityWithHostname | It connected to http://www.comeinbaby.com/app/getversion.php?v=%@&adid=%@ to search for new update Wirelurker malware<br>The iOS frameworks involved are System Configuration, Foundation, and Objective C Runtime, located at the Core Services layer and Cocoa Touch |

**Table 8** (continued)

| Malware | iOS Framework | Functions | Explanation |
|---|---|---|---|
| | Foundation | NSMutableArray DatabaseWithPath | It collected the victim's address book content |
| | | NSMutableArray DatabaseWithPath | It got SMS information from the victim's phone |
| | Objective-C Runtime | OperationWithPath | It uploaded the file to the c2 server |
| Xsser | Foundation | StringWithFormat | It made a connection with the c2 server |
| | | | The iOS framework involved is Foundation, located at the Core Services layer |
| Keyraider | Security | Replace_SSLWrite<br>Replace_SSLRead | Most KeyRaider samples hooked SSLRead and SSLWrite functions in the itunesstored process |
| | | | Itunesstored is the system daemon responsible for communicating with the App Store using the iTunes protoco |
| | | | It was re sponsible for stealing apple account data |
| | | | The iOS framework involved in the security and Foundation framework is located at the Core OS and Core Services layer |
| | Foundation | Adding methods to NSData and NSString using categories to provide AES 256 encryption on iOS:<br>AES256EncryptWithKey | It used AES encryption with the fixed key of "mischa07" |
| Muda (AdKing) | AVFoundation | SharedInstances | It requested ads to be displayed<br>The iOS framework involved is the AVFoundation framework, located at the Media layer<br>It requested what types of ads to be displayed<br>It made the ads appear by the time they wanted |
| Clickfraud | UIKit | InitWithNibName:bundle: | It connected to a certain web to install IPA via OTA (Over the Air)<br>The iOS framework involved is related to the UIKit framework, located at the Cocoa Touch layer |
| TinyV | Objective-C Runtime | InstallApplication:withOptions | It allowed the malware to install IPA files on the devices. The iOS framework involved is related to the Objective-C Runtime framework, located at the Cocoa Touch layer |
| | | UninstallApplication:withOptions | It allowed the malware to uninstall IPA files in the devices |
| Yispecter | UIKit | CurrentDevice | It checked the victim's current device version<br>The iOS framework involved is related to UIKit, the Foundation framework located at the Cocoa Touch layer and Core Services |
| | MobileCoreServices | MobileInstallationLookup | It checked whether there are NoIcon installed or not |
| | Foundation | URLWithString | It checked whether the device was jailbroken or not by connecting to Cydia |

**Table 8** (continued)

| Malware | iOS Framework | Functions | Explanation |
|---|---|---|---|
| | UIKit | UIApplicationDelegate | It requested NoIcon be installed on the device |
| | Swift | InstallAdPage | It installed AdPage on the device |
| | | UninstallApps | It uninstalled any apps that the c2 server had told |
| | | InstallNoIconUpdate | It installed the NoIconUpdate on the victim's device |
| ZergHelper | MessageUI Objective-C Runtime | Class TTMessageViewController */-(void)getMessageListServerr | When the app launched, it immediately connected to the URL interface[.]xyzs.com and reacted differently based on the HTTP request result. The webpage was configured to return a 404-not-found error if the access comes from an IP address outside mainland China<br>The iOS framework involved is related to MessageUI, located at the Cocoa Touch layer |

be used for actual operation or for possible exploitation. The iOS framework is a hierarchical directory structure that contains common resources like a dynamic shared library, nib files, picture files, localized strings, header files, and reference material. Multiple applications could simultaneously utilize all of these resources. Frameworks are similar to static and dynamic shared libraries in that they provide a library of routines that an application can access to do a specific task.

The framework involved the following: two (2) security, five (5) UIKit, six (6) foundation, one (1) AVFoundation, one (1) system configuration, and one (1) MessageUI, while for iOS architecture, there were two (2) Cocoa Touch, one (1) media, two (2) core services, and one (1) core OS layer involved. All details are summarized in Fig. 6

Patterns were created using the functions and frameworks obtained from the Contagio dataset. If the features are the same, these patterns can be analysed to recognize any possible malicious behaviour from other malware dataset. Table 9 displays the patterns that have been constructed. At this point, data from the static analysis were used to create patterns. All the functions were extracted from the malware dataset, and a combination of functions was used to make patterns. The commonly used function in the applications are _Unwind_SjLj_Unregister, Objc_msgSend, and _Unwind_SjLj_Register, where their occurrences in malware dataset are 14, 12, and 11, respectively. As mentioned above, patterns were developed using a combination of functions extracted from the malware. Table 9 shows the patterns for iOS malware classification.

The next section further evaluates all the patterns that were developed. Table 10 shows the similarities of functions from the malware dataset based on their malware types.

The similarities or roots in the malware dataset are depicted in Table 10. The exploit classification was then mapped into the concept of phylogenetics in order to complete the iOS malware detection model. Based on the mapping results, the malware might lead to the possible exploitation of social media or online banking. For example, suppose surveillance features such as SMS or phone calls are being used. In that case, it might consider that online banking is being exploited because the applications usually use SMS and phone calls for verification and transaction purposes. The attack will constitute social media exploitation if the attackers are using those five features. Next, Fig. 7 shows the mapping of library function similarities between malware dataset.

As in Table 10 and Fig. 7, a conclusion can be drawn that each malware type has its functional similarities. As a result, these newly developed iOS malware classifications have overcome the challenges related to iOS platform exploitation. Hence, this classification has fulfilled the first objective of this paper. The iOS malware model for detecting social media and online banking exploitation was developed, as illustrated in Fig. 8.

Figure 8 shows the iOS model developed from malware dataset, functions extraction, and functions combination until integration with the phylogenetics concept. The classification resulted from combining malware functions based on the types. When the model matches the applications tested, it will be detected as a possible exploitation of social media and online banking. An example of mapping malware classification and phylogenetic is depicted in Fig. 9, while the detailed classification description can be referred to in Table 11.

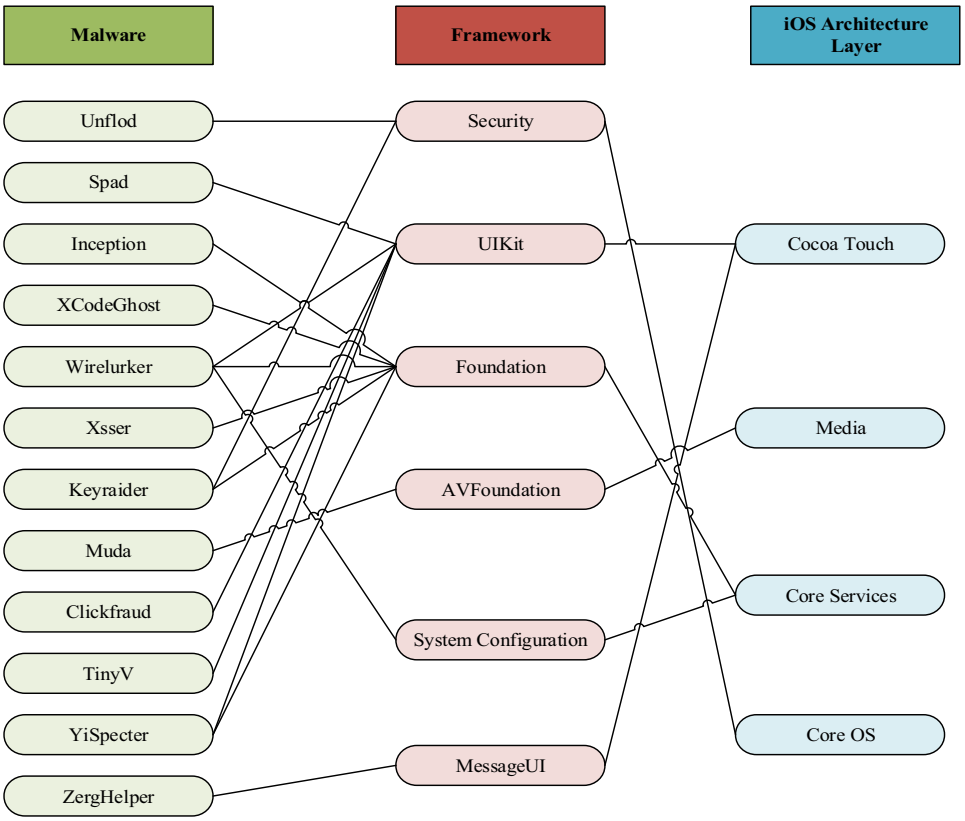


**Fig. 6** Summarisation of Malware Mapped to the framework and iOS Architecture

Figure 9 shows Unflod malware, which can exploit social media and online banking. The type of malware behaviour exhibited by Unflod is phishing, host, stealth, self-activation and passive monitoring for payload, infection, operating algorithm, activation, and propagation. For the iOS chain, Unflod uses Chain 1 (iOS 10. x and below); for surveillance features, it uses GPS, SMS, call, audio, and camera.

Table 11 displays the phylogenetic mapping of 22 of 30 malware classifications, which can be involved in exploitation against social media and online banking. These show that exploitation could have occurred if malware behaviour, iOS architecture, and surveillance features matched the classification. If the attackers exploit SMS or phone calls, generally, this is online banking exploitation, because these applications commonly use SMS and phone calls for verification and transaction purposes. Attackers using those five (5) elements are exploiting social media. The malware behaviour of each classification needs to be examined deeply to identify the malware act. In summary, E1 (Unflod malware), E4 + E5 + E6 + E7 (Inception malware), E8 + E9 (XCodeghots malware), E10 + E11 + E12 + E13 (Wirelurker malware), E14 (Zerghelper malware), E16 (Xsser malware), E22 + E23 + E24 + E25 + E26 + E27 + E28 (Yispecter malware), and E29 + E30 (Keyraider malware) are the classifications that involve possible exploitation of social media and online banking.

## 5 Evaluation

The dataset evaluation involved three (3) categories of iOS applications: social media and online banking, none of the social media and online banking (games), and, lastly, a combination of social media and online banking with other applications. Table 12 shows the dataset categories' details that match the pattern developed.

Based on the analysis, 30 patterns were created. These were used as input for iOS malware model detection. For evaluation purposes, 150 applications from the AppStore and third-party platforms were compared with the patterns developed to identify applications that possess malicious scripts related to social media and online banking exploitation.

Figure 10 shows the occurrence of functions from 150 evaluated applications. The functions most often used in the applications are FC127 (Init) with 146 occurrences, FC61 (MainBundle) with 145 occurrences, and FC14 (setDelegate) with 144 occurrences. It can be concluded that most of the applications evaluated have these three essential functions. An application would become malicious if several functions were exploited, such as:

- ObjectForKeyedSubscript
- URLWithString
- StringWithFormat

**Table 9** Patterns For iOS Malware Classification

| Pattern | Exploitation |
|---|---|
| FC1 + FC2 + FC3 + FC4 + FC5 + FC6 + FC7 + FC8 | E1 (Unflod Malware) |
| FC9 + FC10 + FC11 | E2(Spad Malware) |
| FC10 + FC11 + FC12 + FC13 + FC14 | E3(Spad Malware) |
| FC15 + FC16 + FC17 + FC18 + FC19 + FC20 + FC21 + FC22 + FC23 + FC24 + FC25 + FC26 + FC27 + FC28 + FC29 + FC30 + FC31 | E4(Inception Malware) |
| FC27 + FC28 + FC32 + FC33 + FC34 + FC35 + FC36 + FC37 | E5(Inception Malware) |
| FC19 + FC21 + FC22 + FC24 + FC26 + FC27 + FC28 + FC37 + FC38 + FC39 + FC40 + FC41 + FC42 + FC43 + FC44 + FC45 + FC46 + FC47 + FC48 + FC49 + FC50 FC51 + FC52 + FC53 + FC54 + FC55 | E6(Inception Malware) |
| FC19 + FC26 + FC37 + FC40 + FC41 + FC56 + FC57 + FC58 + FC59 + FC60 | E7(Inception Malware) |
| FC42 + FC61 + FC62 + FC63 + FC64 + FC65 + FC66 + FC67 + FC68 + FC69 + FC70 + FC71 + FC72 + FC73 + FC74 + FC75 | E8(Xcodeghost Malware) |
| FC19 + FC34 + FC54 + FC73 + FC74 + FC75 + FC76 + FC77 + FC78 + FC79 + FC80 + FC81 + FC82 + FC83 | E9(Xcodeghost Malware) |
| FC26 + FC32 + FC37 + FC84 + FC85 + FC86 + FC + FC88 + FC89 + FC90 + FC91 | E10 (Wirelurker Malware) |
| FC19 + FC26 + FC32 + FC36 + FC37 + FC40 + FC84 + FC92 + FC93 + FC94 + FC95 + FC96 | E11(Wirelurker Malware) |
| FC19 + FC26 + FC32 + FC37 + FC84 + FC88 + FC92 + FC96 + FC93 + FC97 + FC98 + FC99 + FC100 + FC101 | E12(Wirelurker Malware) |
| FC32 + FC37 + FC84 + FC90 + FC91 + FC102 + FC103 + FC104 + FC105 + FC106 + FC107 | E13(Wirelurker Malware) |
| FC15 + FC33 + FC37 + FC34 + FC64 + FC67 + FC61 + FC43 + FC79 + FC86 + FC108 + FC109 + FC110 + FC111 + FC112 + FC113 + FC114 + FC115 + FC116 + FC117 + FC118 + FC119 + FC120 + FC121 + FC122 + FC123 + FC124 + FC125 | E14(Zerghelper Malware) |
| FC126 + FC127 | E15(Oneclickfraud Malware) |
| FC26 + FC128 + FC129 | E16(Xsser Malware) |
| FC19 + FC130 + FC131 | E17(Muda Malware) |
| FC76 + FC77 + FC84 + FC95 + FC115 + FC117 + FC132 + FC133 + FC134 + FC135 + FC136 + FC137 | E18(Muda Malware) |
| FC32 + FC37 + FC138 + FC139 + FC140 + FC141 | E19(Tinyv Malware) |
| FC19 + FC32 + FC37 + FC138 + FC139 + FC140 + FC141 + FC142 + FC143 + FC144 | E20(Tinyv Malware) |
| FC32 + FC37 + FC15 + FC20 | E21(Tinyv Malware) |
| FC43 + FC19 + FC26 + FC45 + FC46 | E22(Yispecter Malware) |
| FC15 + FC19 + FC43 + FC111 + FC145 + FC146 + FC147 + FC148 + FC149 + FC150 + FC151 | E23(Yispecter Malware) |
| FC19 + FC152 + FC153 | E24(Yispecter Malware) |
| FC15 + FC32 + FC37 + FC123 + FC111 + FC124 + FC125 + FC154 + FC155 + FC156 + FC157 + FC158 | E25(Yispecter Malware) |

**Table 9** (continued)

| Pattern | Exploitation |
|---|---|
| FC111 + FC115 + FC150 + FC28 + FC61 + FC63 + FC64 + FC100 + FC56 + FC23 + FC19 + FC15 + FC165 + FC166 + FC167 + FC159 + FC160 + FC161 + FC162 + FC163 | E26(Yispecter Malware) |
| FC100 + FC65 + FC66 + FC67 + FC68 + FC163 + FC165 + FC166 + FC167 | E27(Yispecter Malware) |
| FC23 + FC28 + FC57 + FC61 + FC63 + FC64 + FC29 + FC15 + FC100 + FC111 + FC150 + FC160 + FC162 + FC163 | E28(Yispecter Malware) |
| FC32 + FC37 + FC4 + FC100 + FC170 + FC171 + FC172 + FC173 + FC174 + FC175 | E29(Keyraider Malware) |
| FC176 + FC177 + FC178 + FC179 + FC180 + FC181 + FC32 + FC37 | E30(Keyraider Malware) |

**Table 10** Similarities of Functions from The Malware Dataset

| Malware | Function Label | Function |
|---|---|---|
| Unflod | FC1 + FC2 + FC3 + FC4 + FC5 + FC6 + FC7 + FC8 | MSHookFunctions + Write + Int_addr + Strstr + Strcpy + Connect + Socket + Close |
| Spad | FC10 + FC11 | GetClass_GADBannerView_Youmi + initwithAdSize |
| Inception | FC19 + FC26 | Objc_msgSend + StringWithFormat |
| | FC27 + FC28 | NSMutableDictionary + ObjectForKeyedSubscript |
| XCodeghost | FC73 + FC74 + FC75 | DictionaryWithObjectsAndKeys + dataWithJSONObject + Objc_autoreleaseReturnValue |
| Wirelurker | FC32 + FC37 + FC84 | _Unwind_SjLj_Register+ _Unwind_SjLj_Unregister + Alloc |
| Zerghelper | FC15 + FC37 + FC108 + FC109 + FC110 + FC112 + FC113 + FC114 + FC116 + FC118 + FC119 + FC120 + FC121 + FC122 | _stack_chk_fail + _Unwind_SjLj_Unregister + SharedChannel + Certificate + ChannelID + Instance + UserID + Length + Token + SetEnabled + InitWithBaseURL + SetParameterEncoding + RequestWithMethod + RegisterHTTPOperationClass |
| OneClickFraud | FC126 + FC127 | Swift_getInitializedObjCClass + Init |
| Xsser | FC26 + FC128 + FC129 | StringWithFormat+ _ZNSsC1Ev + cxa_atexit:F |
| Muda | FC19 + FC130 + FC131 | Objc_msgSend + RequestAdType + RequestAdWithType |
| | FC132 + FC133 + FC134 + FC135 + FC136 + FC137 | TypeCanRequest + Dispatch_async + SendRequest + setIsCustomer + setType + StartShowAd |
| TinyV | FC32 + FC37 | Unwind_SjLj_Register+ _Unwind_SjLj_Unregister |
| | FC32 + FC37 + FC138 + FC139 + FC140 + FC141 | Unwind_SjLj_Register+ _Unwind_SjLj_Unregister + PublicPath + FrameworksPath + safeClass + NSSelectorFromString |

**Table 10** (continued)

| Malware | Function Label | Function |
|---|---|---|
| Yispecter | (FC15 + FC19 + FC43 + FC100) | _stack_chk_fail + Objc_msgSend + CurrentDevice + NSLog |
| | FC19 + FC43 | Objc_msgSend + CurrentDevice |
| | FC15 + FC19 + FC43 + FC145 + FC146 + FC147 + FC148 + FC149 + FC150 + FC151 | _stack_chk_fail + Objc_msgSend + CurrentDevice + SystemVersion + objectAtIndexedSubscript + jailBreakIdentify + Isequaltostring + FloatValue + MobileInstallationLookup + NSClassFromString |
| | FC19 + FC152 + FC153 | Objc_msgSend + SharedApplication + CanOpenURL |
| | FC15 + FC154 + FC155 + FC156 + FC157 + FC158 | _stack_chk_fail + SharedOwner + Version + SharedClient + GetPath+ _block_object_dispose |
| | FC15 + FC19 + FC100 + FC159 + FC160 + FC161 + FC162 + FC163 + FC165+ FC166 + FC167 | _stack_chk_fail + Objc_msgSend + NSLog + NSHomeDirectories + stringByAppendingPathComponent + PathForResource + CopyItemAtPath + InstallIPA + GetNowDateStr + Save + MobileInstallationUninstall |
| | FC100 + FC163 + FC165 + FC166 + FC167 | NSLog + InstallIPA + GetNowDateStr + Save + MobileInstallationUninstall |
| | FC15 + FC100 + FC160 + FC162 + FC163 | _stack_chk_fail + NSLog + stringByAppendingPathComponent + CopyItemAtPath + InstallIPA |
| Keyraider | FC32 + FC37 | Unwind_SjLj_Register+ _Unwind_SjLj_Unregister |

- MobileInstallationUninstall
- DatabaseWithPath
- NSMutableDiction52ary
- dataWithJSONObject
- SharedApplication
- CanOpenURL

To conclude, 4% percent of the evaluation dataset (seven out of 150 applications) matched the patterns developed. Table 13 shows the details of possible exploitation by the applications evaluated mapped to phylogenetics.

This table concludes that all seven applications have similarities with the pattern developed and exposed possible vulnerabilities based on phylogenetic classification. Apart from the result obtained based on matched patterns, further analysis was conducted to verify the proposed detection model. A few points to be considered when categorizing the mobile application as malicious or not are as follows.

False alarms could occur for any mobile applications analysed in this paper. Based on the analysis, some features might not be used for exploitation, specifically surveillance features. In this case,, theoretically, seven mobile applications should be using the surveillance features based on the mobile application named and described by the developer. The functions exploited that matched the evaluated applications are shown in Table 14.

It can be concluded that this model is able to identify features used for exploitation. For example, the pattern produced for this application matched with App 48, App 71, App 80, App 84, App 97, App 137, and App 144. The applications were under the media and health categories. Thus, all seven (7) applications have the possibility of exploiting social media and online banking.

## 6 Discussion

This paper developed an iOS malware classification focusing on social media and online banking. From the malware dataset, 30 patterns were created. Based on the analysis conducted, 22 of the 30 patterns have been linked to social media and online banking exploitation: E1 (Unflod malware), E4, E5, E6, and E7 (Inception malware), E8 and E9 (XCodeghost malware), E10, E11, E12, and E13 (Wirelurker malware),

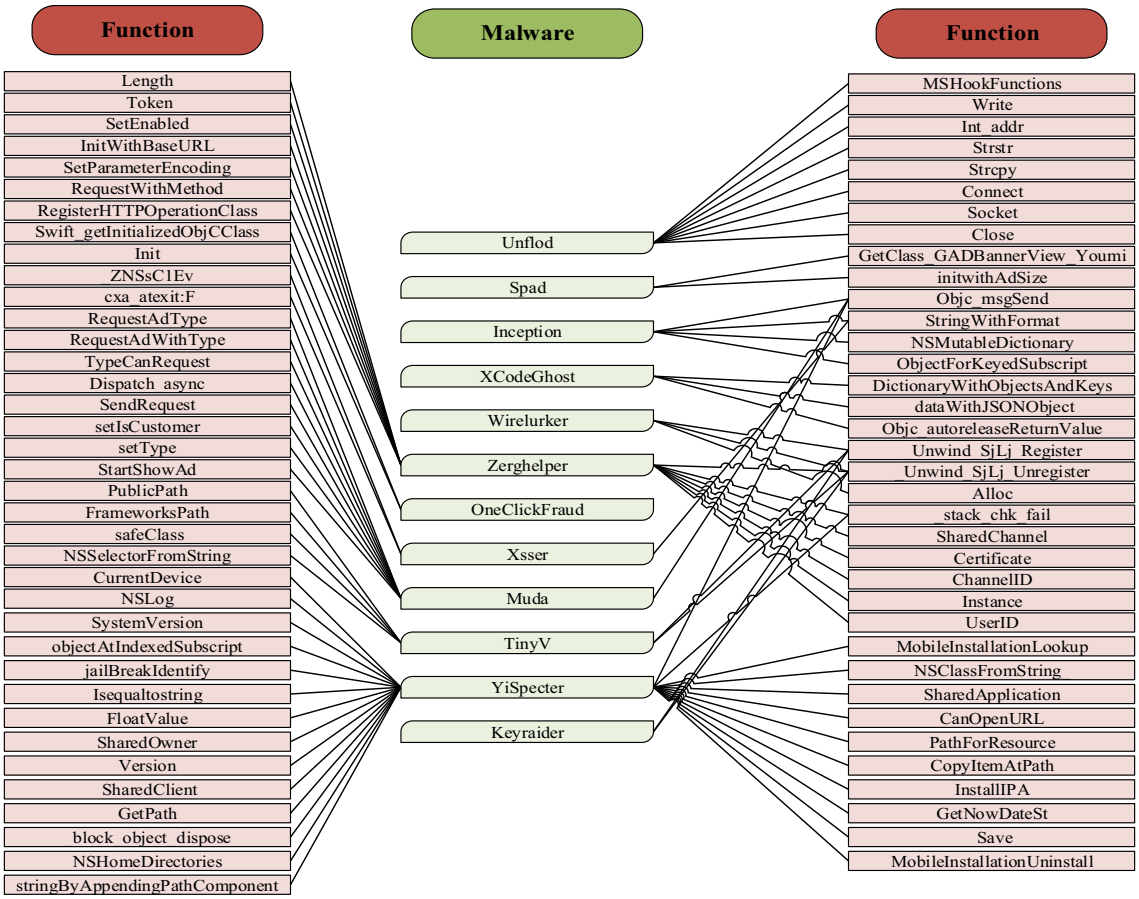


**Fig. 7** Mapping of Library Function Similarities between Malware Dataset

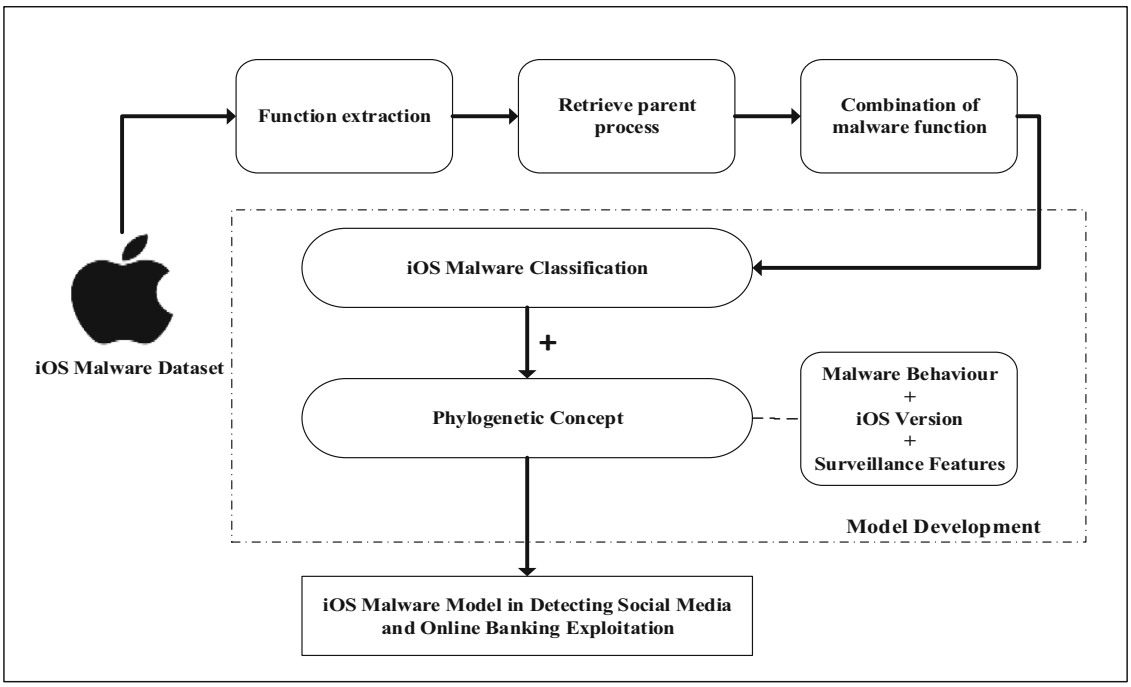


**Fig. 8** iOS Malware Model Methods in Detecting Social Media and Online Banking Exploitation

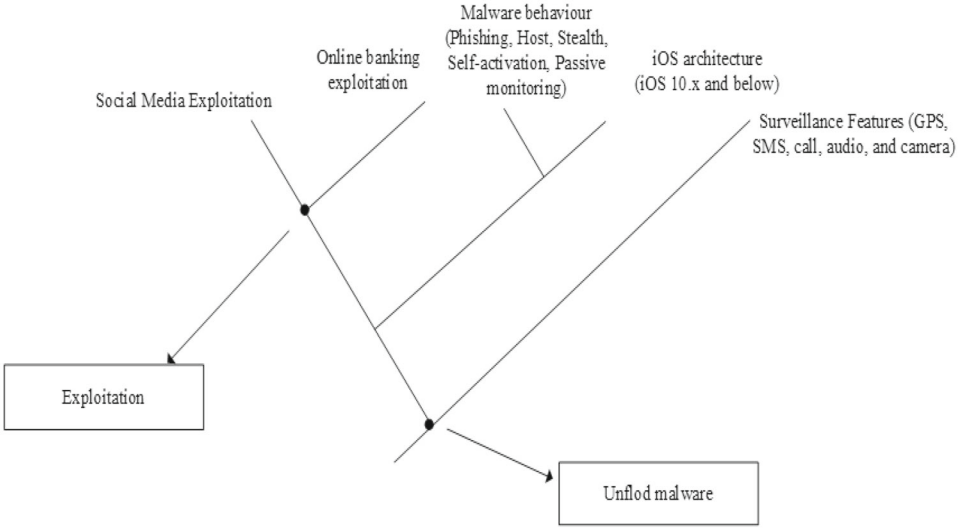


**Fig. 9** Example Of Mapping Malware Classification and Phylogenetic Using Unflod Malware

E14 (Zerghelper malware), E16 (Xsser malware), E22, E23, E24, E25, E26, E27, and E28 (Yispecter malware), and E29 and E30 (Keyraider malware). These classifications matched their functions, which involved possible command and control, monitoring devices, and exploiting banking information.

This paper also developed an iOS model detecting social media and online banking exploitation. The model was devised by integrating classifications and the phylogenetic concept, including malware behaviour, iOS version, and surveillance features. During the evaluation, this model successfully detected seven out of 150 mobile applications with possible exploitation vulnerabilities related to social media and online banking. It matched with App 48, App 71, App 80, App 84, App 97, App 137, and App 144. The classifications involved were E8 (Xcodeghost malware) and E24 (Yispecter malware). The applications that matched fall under the categories of media and health. All seven applications used five surveillance features: GPS, SMS, call log, audio, and camera. These applications have revealed that several functions can be exploited, such as dataWithJSONObject, SharedApplication, and CanOpenURL.

Based on the behaviour of malware, there are many possibilities for social media and online banking exploitation. Today, malware such as Pegasus spyware can activate the phone's camera and microphone and record messages, texts, emails, and phone calls, including those sent via encrypted messaging and phone applications. All this information is sent back to the spyware's clients. This spyware can execute all iPhone users' functions on their smartphones. The Pegasus spyware involves the surveillance features of GPS, SMS, call log, audio, and camera. The malware behaviour can be connected to a c2 server, where the devices are remotely controlled. Thus, social media and online banking exploitation can occur.

The model created in this paper has successfully detected 4% of the mobile applications with possible exploitation vulnerability during its evaluation. This demonstrates that seven out of 150 applications downloaded from the App-Store matched the pattern developed. This proves that the model developed in this paper can detect any possible security exploitation related to social media and online banking for iOS mobile applications.

### 6.1 Limitations and points of improvement

There are some limitations to this paper. First, the constraint of the availability of a malware dataset related to this paper's scope, focusing on social media and online banking exploitation. Currently, the data are more focused on diverse areas requiring more effort to perform the analysis. As the iOS versions continue to be updated and released, new frameworks and functions may be introduced. Collecting recent iOS malware dataset is crucial to identify new behaviour and malware actions using the phylogenetic concept. A larger and dedicated malware dataset will result in a more comprehensive and accurate classification that represents future malware evolution.

The second limitation is the method used to analyse decrypted applications. At the moment, it is evident that there is an increase of applications being encrypted with malicious intention. Automation and efficient techniques will help to hasten the analysis of encrypted applications. This research used manual analysis to identify the features' functions. Hence, there is a need to automate the process to identify functions for each feature in the applications.

**Table 11** Malware Classification Mapped to Phylogenetic

| Pattern | $\alpha_1$- $\alpha_5$ (Malware Behaviour) | $\beta_1$-$\beta_5$ (iOS Chain) | $\gamma_1 - \gamma_5$ (Surveillance Features) | Social Media Exploitation | Online Banking Exploitation |
|---|---|---|---|---|---|
| E1: Unflod Malware | Payload: Phishing. Steal the device's apple id and a corresponding password and send them in plaintext to the server<br>Infection: Host. From Chinese Cydia repositories<br>Operating Algorithm: Stealth. Runs in the background<br>Activation: Self-activation. Initiate their execution by exploring vulnerabilities in services that are available<br>Propagation: Passive monitoring. The malware only can infect those jailbroken devices that have download piracy Chinese repositories | Chain 1 (iOS 10. x & below) | GPS, SMS, call, audio, and camera | Possible | Possible |
| E2 + E3: Spad Malware | Payload: Destructive. Change the actual owner id for the ads into their id to obtain the revenue<br>Infection: Host. Result of some action taken by a user. The criminal hooks the legitimate functions and adds their tweaks<br>Operating Algorithm: Stealth. Runs quietly in the background<br>Activation: Human trigger. The malware will be functioning whenever the ads are operated<br>Propagation: Passive monitoring. The malware only can infect jailbroken devices | Chain 1 (iOS 10.x & below) | No surveillance features are involved, as the attacker only exploits the ads section | Not possible | Not possible |
| E4 + E5 + E6 + E7: Inception Malware | Payload: Phishing. Collect device information, record audio, and send it to the c2 server<br>Infection: Host, Result of some action taken by a user. The malware will inject into the host whenever the user clicks a malicious link or downloads documents<br>Operating Algorithm: Stealth. Runs quietly in the background<br>Activation: Human activation. The malware will be executed when the file has been opened<br>Propagation: Passive monitoring. The malware only can infect those jailbroken devices that click the link or open the trojanized file | Chain 1 (iOS 10.x & below) | GPS, SMS, Call Log, Audio, and Camera | Possible | Possible |

**Table 11** (continued)

| Pattern | $\alpha_1$- $\alpha_5$ (Malware Behaviour) | $\beta_1$-$\beta_5$ (iOS Chain) | $\gamma_1 - \gamma_5$ (Surveillance Features) | Social Media Exploitation | Online Banking Exploitation |
|---|---|---|---|---|---|
| E8 + E9: Xcodeghost Malware | Payload: Destructive. Modifies XCode, infects and steals some device information, and then sends it to the c2 server<br>Infection: Host. Infect apps that Xcodeghost has produced<br>Operating Algorithm: Stealth. Runs quietly in the background<br>Activation: Human trigger. The malware will be functioning whenever the apps are open<br>Propagation: Passive monitoring. The malware can only infect the user of the apps created using Xcodeghost | Chain 1 (iOS 10.x & below) | GPS, SMS, Call Log, Audio, and Camera | Possible | Possible |
| E10 + E11+ E12 + E13: Wirelurker Malware | Infection: Host. Infected through USB when the device connects with the infected Mac<br>Activation: Self-activation. It will start the malicious act once the malware has been injected into the device<br>Payload: Phishing. Exfiltration of user data, application usage and device serial number information<br>Operating algorithm: Stealth. The malware operates quietly in the background<br>Propagation: Passive monitoring. The malware can only infect users connecting their devices with the infected Mac | Chain 1 (iOS 10.x & below) | GPS, SMS, Camera, Audio, and Call<br>The attacker can exploit all the surveillance features by brute-forcing the victim's Apple ID as they already get it. | Possible | Possible |
| E14: Zerghelper Malware | Infection: Host. Spread via host file (mobile app). It camouflaged itself in a genuine mobile app<br>Activation: Scheduled process. It is based on the user's location and activates its payload only in China<br>Payload: Installing backdoor. Installed via a backdoor, requests the user to give Apple ID, shares Apple ID with other users, abuses the Apple ID by running different operations in the background, and abuses enterprise and personal certificates<br>Operating algorithm: Stealth. Runs in the background<br>Propagation: Passive monitoring. The malware camouflaged itself by claiming to resolve stability issues. It then guides the installation for two configurations. Only users in China will see the payload | Chain 1 (iOS 10.x & below) | SMS | Possible | Possible |

**Table 11** (continued)

| Pattern | $\alpha_1$- $\alpha_5$ (Malware Behaviour) | $\beta_1$-$\beta_5$ (iOS Chain) | $\gamma_1 - \gamma_5$ (Surveillance Features) | Social Media Exploitation | Online Banking Exploitation |
|---|---|---|---|---|---|
| E15: Oneclickfraud Malware | Infection: Host. Distributed through an adult site<br>Activation: Human trigger. The malware will be functioning whenever the play button is clicked<br>Payload: Destructive. Installing another app through OTA<br>Operating algorithm: Stealth. Runs quietly in the background<br>Propagation: Passive monitoring. The malware only can infect the visitor of the adult site | Chain 1 (iOS 10.x & below) | No surveillance features are involved | Not possible | Not possible |
| E16: Xsser Malware | Infection: Host. Installed via a rogue repository on Cydia, the most popular third-party application store for jailbroken iPhones<br>Activation: Self-activation. It will start the malicious act once the malware has been injected into the device<br>Payload: Phishing. Act as spyware and harvest information from the user's device, thus sending it to the c2 server<br>Operating algorithm: Terminate and resident. When triggered, xRAT will clean out its installation directory before issuing a package manager command to uninstall itself. The developers behind xRAT created an alert system, flagging the malware operator if any of the following antivirus applications are present on a compromised device<br>Propagation: Passive monitoring. The malware only can infect users that use the repositories | Chain 1 (iOS 10.x & below) | GPS, SMS, and Camera | Possible | Possible |
| E17 + E18: Muda Malware | Infection: Host. It spreads via third-party Cydia sources in China and only affects jailbroken iOS devices<br>Activation: Human trigger. The malware will be functioning whenever the apps are open.Payload: Display advertisements over other apps or in the notification bar and ask users to download the promoted iOS apps<br>Operating algorithm: Stealth<br>Propagation: Passive monitoring. The malware only can infect jailbroken devices that use third-party Cydia repositories | Chain 1 (iOS 10.x & below) | No surveillance features are involved. | Not possible | Not possible |

**Table 11** (continued)

| Pattern | $\alpha_1$- $\alpha_5$ (Malware Behaviour) | $\beta_1$-$\beta_5$ (iOS Chain) | $\gamma_1 - \gamma_5$ (Surveillance Features) | Social Media Exploitation | Online Banking Exploitation |
|---|---|---|---|---|---|
| E19 + E20 + E21: Tinyv Malware | Infection: Hosts. Repackaged into some pirated iOS apps for jailbroken devices<br>Activation: Human trigger. The malware will be executed whenever the apps are being used<br>Payload: Destructive. Connecting with its C2 server to get remote commands, installing a specified IPA file or DEB file(s) in the background, uninstalling a specified IPA app or DEB package(s) in the background, and changing the /etc/hosts file<br>Operating algorithm: Stealth. Runs quietly in the background<br>Propagation: Passive monitoring. The malware only can infect the user that installed the pirated iOS apps | Chain 1 (iOS 10.x & below | No surveillance features are involved. | Not possible | Not possible |
| E22 + E23 + E24 + E25 + E26 + E27 + E28: Yispecter Malware | Infection: Host. Through hijacking traffic from nationwide ISPs, an SNS worm on Windows, and an offline app installation and community promotion<br>Activation: Self-activation. It will start the malicious act once the malware has infected the device<br>Payload: Destructive. Abusing enterprise certificates, installing malicious apps, uninstalling apps, self-monitoring, updating, collecting, and uploading device information, changing safari configurations, hijacking other apps execution, and pretending to be system apps<br>Operating algorithm: Stealth. The malware operates silently in the background<br>Propagation: Passive monitoring. The malware only can infect users that use the repositories | Chain 1 (iOS 10.x & below) | GPS, SMS, Call Log, Audio, and Camera | Possible | Possible |
| E29 + E30: Keyraider Malware | Infection: Host. Distributed through third-party Cydia repositories in China<br>Activation: Human trigger. The malware will be functioning whenever the apps are open<br>Payload: Destructive. Stealing Apple account data, certificates, and private keys<br>.Operating algorithm: Stealth. Runs quietly in the background<br>Propagation: Passive monitoring. The malware only can infect jailbroken devices that use third-party Cydia repositories | Chain 1 (iOS 10.x & below) | GPS, SMS, Call Log, Audio, and Camera | Possible | Possible |

**Table 12** Categories of Dataset Evaluation

| Category | Category A (25 social media & 25 Online Banking) | Category B (50 games) | Category C (10 social media, 15 Online Banking & 25 Games) |
|---|---|---|---|
| Application match | 50/50 | 47/50 | 47/50 |
| Pattern Match | E8 & E24 | E8, E22, E24 | E8, E22 & E24 |
| Description | The pattern developed matches all social media & online banking application | The pattern developed matches all social media & online banking applications<br>Games applications involved with social media as a user ID and sign-in<br>Games applications involved with online banking as item purchase | The pattern developed matches all social media & online banking applications. Match 22 from games apps<br>Games apps involved with social media for user ID and sign-in<br>Games apps involved with online banking for item purchase |

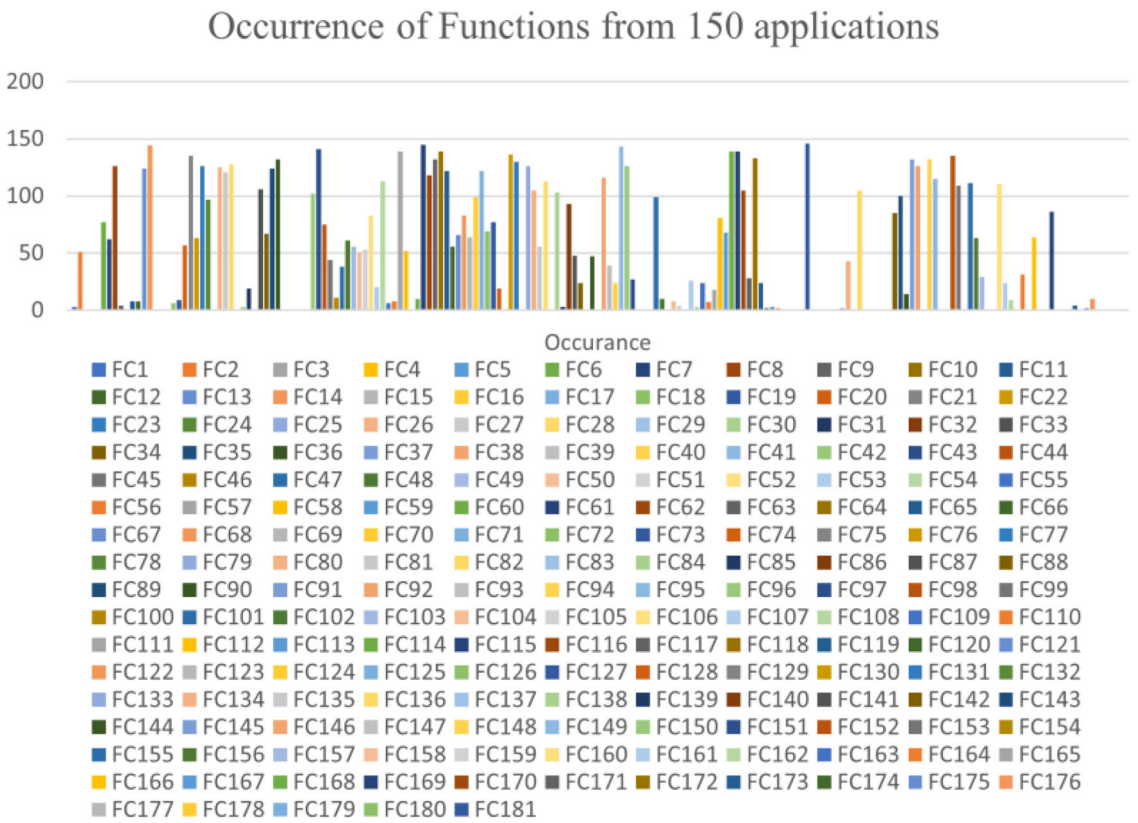


**Fig. 10** Occurrence of Functions From 150 Evaluated Apps

**Table 13** Possible exploitation evaluated application mapped to phylogenetic

| Evaluated Apps | Real-time iOS malware | Description |
|---|---|---|
| App 48 | E24: Yispecter Malware | This app is an app that helps users in taking care of COVID-19<br>For malware behaviour for E24 are<br>Payload: Destructive, Infection: Host, Operating Algorithm: Stealth, Activation: Self-activation, Propagation: Passive monitoring<br>iOS version: Chain 3 (iOS 12.x)<br>Surveillance Features involved: GPS, SMS, Call Log, Audio, and Camera |
| App 71 | E24: Yispecter Malware | This app is for online meetings and webinars or malware behaviour for E24 are<br>Payload: Destructive, Infection: Host, Operating Algorithm: Stealth, Activation: Self-activation, Propagation: Passive monitoring<br>iOS version: Chain 3 (iOS 12.x)<br>Surveillance Features involved: GPS, SMS, Call Log, Audio, and Camera |
| App 80 | E24: Yispecter Malware | This app is for online meetings and webinars<br>For malware behaviour for E24 are<br>Payload: Destructive, Infection: Host, Operating Algorithm: Stealth, Activation: Self-activation, Propagation: Passive monitoring<br>iOS version: Chain 3 (iOS 12.x)<br>Surveillance Features involved: GPS, SMS, Call Log, Audio, and Camera |
| App 84 | E8: XCodeGhost Malware<br>E24: Yispecter Malware | This app is for the community's platform, which can share text, voice, and video<br>For malware behaviour for E8 & E24 are<br>Payload: Destructive, Infection: Host, Operating Algorithm: Stealth, Activation: Self-activation & human trigger, Propagation: Passive monitoring<br>iOS version: Chain 3 (iOS 12.x)<br>Surveillance Features involved: GPS, SMS, Call Log, Audio, and Camera |
| App 97 | E24: Yispecter Malware | This app is for online meetings and webinars<br>For malware behaviour for E24 are<br>Payload: Destructive, Infection: Host, Operating Algorithm: Stealth, Activation: Self-activation, Propagation: Passive monitoring<br>iOS version: Chain 3 (iOS 12.x)<br>Surveillance Features involved: GPS, SMS, Call Log, Audio, and Camera |
| App 137 | E24: Yispecter Malware | This app is for online meetings and webinars<br>For malware behaviour for E24 are<br>Payload: Destructive, Infection: Host, Operating Algorithm: Stealth, Activation: Self-activation, Propagation: Passive monitoring<br>iOS version: Chain 3 (iOS 12.x)<br>Surveillance Features involved: GPS, SMS, Call Log, Audio, and Camera |

**Table 13** (continued)

| Evaluated Apps | Real-time iOS malware | Description |
|---|---|---|
| App 144 | E24: Yispecter Malware | This app is for video-making and sharing platforms<br>For malware behaviour for E24 are<br>Payload: Destructive, Infection: Host, Operating Algorithm: Stealth, Activation: Self-activation, Propagation: Passive monitoring<br>iOS version: Chain 3 (iOS 12.x)<br>Surveillance Features involved: GPS, SMS, Call Log, Audio, and Camera |

**Table 14** Functions Exploited Matched with Evaluated Applications.

| Classification | Function exploited | Description |
|---|---|---|
| E8 (XcodeGhost) | FC74 (dataWithJSONObject) | Return JSON information from the target of the Foundation<br>Using this function, the victim's devices could be remotely controlled when connected to the c2 server |
| E24 (Yispecter) | FC152 (SharedApplication) | It makes a connection to the window server and completes another initialization<br>This function can share applications to the victim's devices, like spyware, whether the applications distributed are benign or malicious |
| | FC153 (CanOpenURL) | Returns a Boolean value indicating whether an app is available to handle a URL scheme<br>This function is considered not malicious if the URL is genuine and vice versa |

## 7 Conclusion

Based on the findings identified in this paper, it can be concluded that the misuse of existing applications' functions and frameworks might lead to the exploitation of online banking and social media. An analysis of the pattern of malicious applications can be used to counteract this issue. Malicious applications require combinations of frameworks and functions to exploit the intended features successfully.

**Acknowledgements** The authors would like to express their gratitude to the Ministry of Higher Education (MOHE) Malaysia and Universiti Sains Islam Malaysia (USIM) for the support and facilities provided. This research paper project is funded by FRGS/1/2019/ICT04/USIM/02/3.

**Funding** All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

**Data availability** The dataset generated during and/or analysed during the current study are available from the corresponding author upon reasonable request.

## Declarations

**Conflict of interest** All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

**Ethical approval** Not applicable

## References

1. Garg, S., Baliyan, N.: Comparative analysis of Android and iOS from security viewpoint. Comput. Sci. Rev. **40**,(2021)
2. Shishkova, T.: IT threat evolution in Q3 2021. Mobile Statistics. Securelist **26**, 448 (2021)
3. McAfee: Labs Threats Report (2021)
4. McAfee: Labs COVID-19 Threats Report (2020)
5. Vulnerabilities and threats in mobile applications. https://www.ptsecurity.com/upload/corporate/ww-en/analytics/Mobile-Application-Vulnerabilities-and-Threats-2019-eng.pdf (2019). Accessed 26 Dec 2021
6. Mobile Cyberattacks Impact Every Businesss. https://blog.checkpoint.com/wp-content/uploads/2017/04/Dimensional_Enterprise-Mobile-Security-Survey.pdf (2017). Accessed 26 Aug 2020
7. Williams, S.: Mobile malware and exploitation amongst biggest cyber threats for 2020. Security Brief Asia. https://securitybrief.asia/story/mobile-malware-and-exploitation-amongst-biggest-cyber-threats-for-2020 (2020). Accessed 26 Aug 2020

8. Khandelwal, S.: Powerful FinSpy Spyware Found Targeting iOS and Android Users in Myanmar. https://thehackernews.com/2019/07/finspy-spyware-android-ios.html (2019). Accessed 13 Aug 2020
9. Khandelwal, S.: 'Exodus' Surveillance Malware Found Targeting Apple iOS Users. The Hacker News. https://thehackernews.com/2019/04/exodus-ios-malware.html (2019). Accessed 13 Aug 2020
10. Facebook disrupts hackers who used iOS exploits, malware to spy on Uyghurs | AppleInsider. Apple Insider. https://appleinsider.com/articles/21/03/24/facebook-disrupts-hackers-who-used-ios-exploits-malware-to-spy-on-uyghurs (2021). Accessed 20 Oct 2021
11. Increased Use of Mobile Banking Apps Could Lead to Exploitation: Internet Crime Complaint Center (IC3). https://www.ic3.gov/Media/Y2020/PSA200610 (2021)
12. Francesco, M., Santone, A.: Deep learning for image-based mobile malware detection. J. Comput. Virol. Hacking Tech. **16**, 157–171 (2020)
13. Zhou, G., Duan, M., Xi, Q., Wu, H.: ChanDet: detection model for potential channel of iOS applications. J. Phys. Conf. Ser. **1187**(4), 214 (2019). https://doi.org/10.1088/1742-6596/1187/4/042045
14. Nisioti, A., Heydari, M., Mylonas, A., Katos, V., Tafreshi, V.H.F.: TRAWL: protection against rogue sites for the masses. Proc. Int. Conf. Res. Challenges Inf. Sci. **21**, 120–127 (2017)
15. Bojjagani, S., Sastry, V.N., (2017) VAPTAi: A threat model for vulnerability assessment and penetration testing of android and IoS mobile banking apps, Proc.: IEEE 3rd Int. Conf. Collab. Internet Comput. CIC. (2017). https://doi.org/10.1109/CIC.2017.00022
16. Cimitile, A., Martinelli, F., Mercaldo, F.: Machine learning meets IoS malware: identifying malicious applications on apple environment, ICISSP 2017 Proc. 3rd Int. Conf. Inf. Syst. Secur. Priv. **2017**, 487–492 (2017). https://doi.org/10.5220/0006217304870492
17. Denis, G.A.D.A., Manuel, M., Carson, W., Eltoweissy, M., Cheng, L.: Biologically inspired safety and security for smart built environments: position paper. In: IEEE Symposium on Security and Privacy Workshops, 2018, pp. 293–298 (2018). https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8424663 Accessed 26 Aug 2020
18. Firdaus, A., Anuar, N.B., Razak, M.F.A., Sangaiah, A.K.: Bio-inspired computational paradigm for feature investigation and malware detection: interactive analytics. Multimed. Tools Appl. **77**, 17519–17555 (2018). https://doi.org/10.1007/s11042-017-4586-0
19. Demertzis, K., Iliadis, L.: Ladon: a cyber-threat bio-inspired intelligence management system. J. Appl. Math. Bioinform. **6**(3), 45–64 (2016)
20. Demertzis, K., Iliadis, L.: Bio-inspired hybrid intelligent method for detecting android malware. Adv. Intell. Syst. Comput. **416**, 289–304 (2016)
21. Saudi, M.M., Sukardi, S., Syafiq, A.S.M., Ahmad, A., Afif, M., Husainiamer: Mobile malware classification based on phylogenetics. Int. J. Eng. Adv. Technol **9**(1), 3661–3665 (2019)
22. Mercaldo, F., Santone, A.: Audio signal processing for Android malware detection and family identification. J. Comput. Virol. Hacking Tech. **17**(2), 139–152 (2021)
23. Imtiaz, S.I., Ur Rehman, S., Javed, A.R., Jalil, Z., Liu, X., Alnumay, W.S.: Deep AMD: detection and identification of android malware using high-efficient deep artificial neural network. Fut. Gener. Comput. Syst. **115**, 844–856 (2021)
24. Mahindru, A., Sangal, A.L.: MLDroid-framework for Android malware detection using machine learning techniques. Neural Comput. Appl. **33**(10), 5183–5240 (2021)
25. Frenklach, T., Cohen, D., Shabtai, A., Puzis, R.: Android malware detection via an app similarity graph. Comput. Secur. **109**, 102386 (2021)
26. Cai, L., Li, Y., Xiong, Z.: JOWMDroid: Android malware detection based on feature weighting with joint optimization of weight-mapping and classifier parameters. Comput. Secur. **100**, (2021)
27. Saudi, M.M., Ahmad, A., Kassim, S.R.M., Husainiamer, M.L., Kassim, A.Z., Zaizi, N.J.: Mobile malware classification for social media application,: Mobile malware classification for social media application. Int. Conf. Cybersecurity, ICoCSec **2019**, 70–75 (2019). https://doi.org/10.1109/ICOCSEC47621.2019.8970800
28. Iadarola, G., Martinelli, F., Mercaldo, F., Santone, A.: Formal methods for android banking malware analysis and detection. In: 2019 6th International Conference on Internet of Things: Systems, Management and Security, IOTSMS Oct. 2019, pp. 331–336 (2019)
29. Cooke, T.N.: Metadata, jailbreaking, and the cybernetic governmentality of ios: or, the need to distinguish digital privacy from digital privacy. Surveill. Soc. **18**(1), 90–103 (2020)
30. Aenurahman Ali, A., Dwi Wahyu, N., Cahyani, Musthofa Jadied, E.: Digital forensic analysis on iDevice: Jailbreak iOS 12.1.1 as a case study. Indones J. Comput. **4**(2), 205–218 (2019). https://doi.org/10.21108/indojc.2019.4.2.349
31. Gui, X., Liu, J., Chi, M., Li, C., Lei, Z.: Analysis of malware application based on massive network traffic. Science **5**, 479 (2016)
32. Gao, B., Wang, Y., Chen, Z., Tang, J.: Data threats analysis and prevention on iOS platform. Sixth Int. Conf. Electron. Inf. Eng. **9794**, 41178 (2015). https://doi.org/10.1117/12.2203437
33. Deore, M., Kulkarni, U.: Malware detection using faster region proposals convolution neural network. Int. J. Interact. Multimed Artif. Intell **7**(4), 146–162 (2022). https://doi.org/10.9781/ijimai.2021.09.005
34. Dhalaria, M., Gandotra, E.: A hybrid approach for android malware detection and family classification. Int. J. Interact. Multimed Artif. Intell. **6**, 174–188 (2021)
35. Cimino, M.G.C.A., De Francesco, N., Mercaldo, F., Santone, A., Vaglini, G.: Model checking for malicious family detection and phylogenetic analysis in mobile environment. Comput. Secur. **90**, 101691 (2020)
36. Bernardi, M.L., Cimitile, M., Distante, D., Martinelli, F., Mercaldo, F.: Dynamic malware detection and phylogeny analysis using process mining. Int. J. Inf. Secur. **18**(3), 257–284 (2019)
37. Acampora, G., Bernardi, M.L., Cimitile, M., Tortora, G., Vitiello, A.: A fuzzy clustering-based approach to study malware phylogeny. IEEE Int. Conf. Fuzzy Syst. **2018**, 1–8 (2018). https://doi.org/10.1109/FUZZ-IEEE.2018.8491625
38. Liu, J., Xie, P.D., Liu, M.Z., Wang, Y.J.: Having an insight into malware phylogeny: Building persistent phylogeny tree of families. IEICE Trans. Inf. Syst. **E10D**(4), 1199–1202 (2018). https://doi.org/10.1587/transinf.2017EDL8172
39. Ghosh, K., Mills, J., Dorr, J.: Phylogenetic-inspired probabilistic model abstraction in detection of malware families. In: AAAI Fall Symposium Technical Report, vol. FS-17-01-, pp. 200–205 (2017)
40. Oyen, D., Anderson, B., Anderson-Cook, C.: Bayesian networks with prior knowledge for malware phylogenetics. In: AAAI Working Technical Report, vol. WS-16-01-, pp. 185–192 (2016)
41. Anderson, B., Lane, T., Hash, C.: Malware phylogenetics based on the multiview graphical lasso. Comput. Sci. **8819**, 1–12 (2014). https://doi.org/10.1007/978-3-319-12571-8
42. Lama, A.F., Alserhani, H.M.: Social media and cybercrimes. Turkish J. Comput. Math. Educ. **12**(10), 2972–2981 (2021)
43. Almalki, S., Alghamdi, R., Sami, G., Alhakami, W.: Social media security and attacks. IJCSNS Int. J. Comput. Sci. Netw. Secur. **21**(1), 4158 (2021)
44. Jain, A.K., Sahoo, S.R., Kaubiyal, J.: Online social networks security and privacy: comprehensive review and analysis. Complex Intell. Syst. **7**(5), 2157–2177 (2021). https://doi.org/10.1007/S40747-021-00409-7
45. Jones, T.: Social Media and the Effects on the Everyday User, Utica College (2020)

46. Grammatikakis, K.P., Koufos, I., Kolokotronis, N., Vassilakis, C., Shiaeles, S.: Understanding and mitigating banking trojans: from Zeus to Emotet (2021)
47. Jaride, C., Taqi, A.: Mobile banking adoption: a systematic review, and direction for further research. J. Theor. Appl. Inf. Technol. **99**(16), 5899 (2021)
48. Roy, P.K., Shaw, K.: An integrated fuzzy model for evaluation and selection of mobile banking (m-banking) applications using new fuzzy-BWM and fuzzy-TOPSIS. Complex Intell. Syst. **1**, 1–22 (2021). https://doi.org/10.1007/S40747-021-00502-X
49. Wazid, M., Zeadally, S., Das, A.K.: Mobile banking: evolution and threats: malware threats and security solutions. IEEE Consum. Electron. Mag. **8**(2), 56–60 (2019). https://doi.org/10.1109/MCE.2018.2881291
50. Priyanka, M., Kanoi, V.: Internal structure of iOS and Building tools for iOS apps. Int. J. Comput. Sci. Appl. **6**(2), 2020 (2013)
51. Gronli, T.M., Hansen, J., Ghinea, G., Younas, M.: Mobile application platform heterogeneity: android vs windows phone vs iOS vs Firefox OS. Proc. Int. Conf. Adv. Inf. Netw. Appl. AINA **25**, 635–641 (2014)
52. Chen, K., et al.: pp. 357–376 (2016)
53. Chang, Y.T., Teng, K.C., Tso, Y.C., Wang, S.J.: Jailbroken iPhone forensics for the investigations and controversy to digital evidence. J. Comput **26**(2), 19–33 (2015)
54. AI-qershi, F., AI-Qurishi, M., AI-Amri, A.: Android vs iOS: The Security Battle (2014). https://ieeexplore.ieee.org/document/691
55. Saudi, M.M., Tamil, E.M., Md Siti, A.N., Mohd, Y.I.I., Seman, K.: EDOWA Worm Classification. https://www.researchgate.net/publication/44262015_EDOWA_Worm_Classification#fullTextFileContent (2008). Accessed 27 Aug 2020
56. Doroudi, S., Avgerinos, T., Harchol-Balter, M.: To clean or not to clean: malware removal strategies for servers under load. Eur. J. Oper. Res. **292**(2), 596–609 (2021). https://doi.org/10.1016/J.EJOR.2020.10.036
57. Ficco, M.: Malware analysis by combining multiple detectors and observation windows. IEEE Trans. Comput. **9340**, 1–14 (2020). https://doi.org/10.1109/TC.2021.3082002
58. Payload-Definition |Trend Micro, U.S.A. https://www.trendmicro.com/vinfo/us/security/definition/payload (2020). Accessed 26 Aug 2020
59. Solanki, N., Sharma, N.: Malware analysis: types and tools[Online]. http://ijesc.org/ (2019) Accessed 26 Aug 2020
60. Lord, N.: What is polymorphic malware? A definition and best practices for defending against polymorphic malware | digital guardian, Digital Guardian. https://digitalguardian.com/blog/what-polymorphic-malware-definition-and-best-practices-defending-against-polymorphic-malware (2020). Accessed 29 Oct 2021
61. Singh, J., Thakur, D., Gera, T., Shah, B., Abuhmed, T., Ali, F.: Classification and analysis of android malware images using feature fusion technique. IEEE Access **9**, 90102–90117 (2021). https://doi.org/10.1109/ACCESS.2021.3090998
62. TSR, Hope, C.: https://www.computerhope.com/jargon/t/tsr.htm (2020). Accessed 29 Oct 2021
63. Bhunia, S., Tehranipoor, M.: Hardware obfuscation. Hardw. Secur. **2**, 373–396 (2019)
64. Update your iPhone: https://support.apple.com/en-us/HT204204 (2021). Accessed 13 Jun 2021
65. Apple patches iOS zero-day vulnerability exploited by Pegasus spyware|AppleInsider. https://appleinsider.com/articles/21/09/23/apple-patches-ios-zero-day-vulnerability-exploited-by-pegasus-spyware (2021). Accessed 19 Oct 2021
66. Apple Iphone Os: https://www.cvedetails.com/product/15556/Apple-Iphone-Os.html?vendor_id=49 (2021). Accessed 19 Oct 2021
67. Azam, S., Sumra, R.S., Shanmugam, B., Yeo, K.C., Jonokman, M., Samy, G.N.: Security source code analysis of applications in Android OS. Int. J. Eng. Technol. **7**(4), 30–34 (2018)
68. Saudi, M.M., Adli, A., Ismail, C., Ahmad, A., Afif, H.M.: CallDetect: detection of call log exploitation inspired by apoptosis. Int. J. Adv. Sci. Eng. Inf. Technol. **10**(5), 1792–1797 (2021)
69. Istambul, R., Saudi, M.M., Nugraha, U., Yusof, M.: Security exploitation for online meeting applications: proof of concept. Turkish J. Comput. Math. Educ. **12**(3), 1785–1792 (2021)
70. Saudi, M.M., Husainiamer, A.: Mobile malware classification via system calls and permission for GPS exploitation. Int. J. Adv. Comput. Sci. Appl. **8**(6), 277–283 (2017)
71. Yusof, M., Saudi, M.M., Ridzuan, F.: A new mobile botnet classification based on permission and API calls. In: Proceedings–2017 7th International Conference on Emerging Security Technologies, EST 2017, Oct. pp. 122–127, (2017)
72. Taheri, R., Ghahramani, M., Javidan, R., Shojafar, M., Pooranian, Z., Conti, M.: Similarity-based android malware detection using hamming distance of static binary features. Fut. Gener. Comput. Syst. **105**, 230–247 (2020). https://doi.org/10.1016/J.FUTURE.2019.11.034
73. Alam, S., Qu, Z., Riley, R., Chen, Y., Rastogi, V.: DroidNative: automating and optimizing detection of Android native code malware variants. Comput. Secur. **65**, 230–246 (2017). https://doi.org/10.1016/J.COSE.2016.11.011