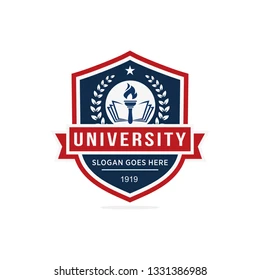
**Project Report**

**GIF Malware Detection Project**

**First four page [front page thats include acknowledgment, support and logo of your college]**

## **PROJECT REPORT**

## **GIF Malware Detection**



**Student Names**

**Year/Sem**

**Email ID**

**GIF Malware Detection Project**

##### PROJECT REPORT

SUBMIT BY

##### **Student Name**

##### ROLL NUMBER

##### UNIVERSITY NAME

##### Under the guidance of MR. TEACHER NAME



**UNIVERSITY**

**ADDRESS**

Year 2023



BONAFIDE CERTIFICATE

Certified that this project titled **‘GIF Malware Detection Project’** is a bonafide work of  **STUDENT NAME** who carried out the research under my supervision.

HEAD OF DEPARTMENT FACULTY IN CHARGE

(SIGNATURE WITH DATE) (SIGNATURE WITH DATE)

## **Acknowledgement**

This document: I thank all my colleagues who provided a deep insight and expertise that greatly assisted in the document development.

I personally thank my Professor for the assistance with the initial drafts and the core structure of the proposal for the comments that greatly improved the manuscript.

I would also like to show my gratitude to the University for sharing its pearls of wisdom with me during the course of the development of this proposal. I am immensely grateful to everyone.



**GIF Malware Detection Project**

**ABSTRACT**

As the use of mobile devices continues to increase, so does the risk of malware attacks. GIF malware has become a significant threat to mobile security, and detecting it has become a challenge for security researchers. In this study, we propose a malware detection framework to classify GIF applications as malware or benign. The proposed framework uses a combination of static and dynamic features extracted from the Android application's code and behaviour.

The study also conducted feature importance analysis, revealing the most significant features contributing to the detection of malware GIF.

**INTRODUCTION**:

Idealistic hackers attacked computers in the early days because they were eager to prove themselves. Malicious actors often want to get information of interest from targeted computer environments. To achieve this goal, they usually decide to plant some kind of software that will provide that information continuously. Throughout history, the most common way of doing that was to plant an executable file and make it run. Over time, the defensive systems improved and became more successful at detecting such executable implants. In this cat-and-mouse game, both sides try to improve their tools and, as defensive tools get better, malware actors try to find new ways of smuggling malicious software into a system. There are several popular ways of doing this, suchs as embedding malicious code into various document formats or executing malicious code in memory without saving anything on disk. As time passes, security solutions are becoming increasingly more aware of such threats.  
  
ReversingLabs continuously improves its malware-detection capabilities. One of the more novel methods that caught our eye is hiding malware inside image formats like PNG, BMP, GIF or JPEG. Recently, we enhanced our platform support for unpacking these image formats, and listed some of those improvements in one of our previous blog posts.

In this blog, we will demonstrate how these new enhancements can be used to discover novel malware threats and showcase several examples of images with hidden PHP executable content. Most of them try to fetch additional resources from a remote server and use different kinds of obfuscation to hide their malicious intents. As an example of threat hunting via this new functionality, a hidden web shell which led to discovery of a vulnerable web site will also be shown.

### **Malware hiding in images**

Image formats are interesting to malware authors because they are generally considered far less harmful than executable files. Images can be used to deploy malware in combination with a dropper, where the dropper acts as a benign executable which parses malicious content hidden inside of an image.  
  
One area where this technique can be used are web uploads. Many websites enable uploading image content, but improperly filter out executables and scripts. In such cases, malicious code can be packed into an image and uploaded to a web server containing a potential vulnerability which enables execution of its contents. Probably the most familiar type of such payloads are PHP web shells.  
  
Threat actors discover and exploit vulnerabilities in applications used to parse image formats. To remain undetected and avoid attracting the attention of security tools, they typically try to create files which adhere to the image format specification whenever possible.  
  
The simplest way to embed malicious content into an image is to append it to the image end, or, as it’s commonly referred to, the overlay. Malicious actors typically just take a benign image file and append some content. This makes it a well-known method that is quite easy to detect.  
  
For example, in the case of a GIF file, all bytes after the GIF’s trailer byte *(0x3B)* can be considered an overlay. In the case of a PNG file, everything after the end of the *IEND* chunk can be considered an overlay. This is conceptually the same as appending content to any other regular file format, so we won’t go into more details about overlays in this blog post.  
  
Another interesting place to look for malware when analyzing image samples are the EXIF tags. These tags are metadata fields used to store additional descriptive data about the image, like the model of the camera used to take the picture, the date and time when the picture was taken, or even the geolocation of the place where the image was taken. This data is part of the image format, but it isn’t required for the image's visual interpretation and some tools used to view the images opt to not present all of these tags to the users, which makes them a great hiding place.

Cracking machines, however, is an industry in today's world. Despite recent improvements in software and computer hardware security, both in frequency and sophistication, attacks on computer systems have increased. Regrettably, there are major drawbacks to current methods for detecting and analysing unknown code samples. The Internet is a critical part of our everyday lives today. On the internet, there are many services and they are rising daily as well. Numerous reports indicate that malware's effect is worsening at an alarming pace. Although malware diversity is growing, anti- virus scanners are unable to fulfil security needs, resulting in attacks on millions of hosts. Around 65,63,145 different hosts were targeted, according to Kaspersky Labs, and in 2015, 40,00,000 unique malware artefacts were found. Juniper Research (2016), in particular, projected that by 2019 the cost of data breaches will rise to $2.1 trillion globally. Current studies show that script-kiddies are generating more and more attacks or are automated. To date, attacks on commercial and government organisations, such as ransomware and malware, continue to pose a significant threat and challenge. Such attacks can come in various ways and sizes. An enormous challenge is the ability of the global security community to develop and provide expertise in cybersecurity. There is widespread awareness of the global scarcity of cybersecurity and talent. Cybercrimes, such as financial fraud, child exploitation online and payment fraud, are so common that they demand international 24-hour response and collaboration between multinational law enforcement agencies. For single users and

organisations, malware defence of computer systems is therefore one of the most critical cybersecurity activities, as even a single attack may result in compromised data and sufficient losses.

Mobile phones have become increasingly important tools in people’s daily life, such as mobile payment, instant messaging, online shopping, etc., but the security problem of mobile phones is becoming more and more serious. Due to the open source nature of the Android platform, it is very easy and profitable to write malware using the vulnerabilities and security defects of the Android system. This is the main reason for the rapid increase in the number of malware on the Android system. The malicious behaviors of Android malware generally include sending deduction SMS, consuming traffic, stealing user’s private information, downloading a large number of malicious applications, remote control, etc., threatening the privacy and property security of mobile phones users.

The number of Android malware is growing rapidly; particularly, more and more malicious software use obfuscation technology. Traditional detection methods of manual analysis and signature matching have exposed some problems, such as slow detection speed and low accuracy. In recent years, many researchers have solved the problems of Android malware detection using machine learning algorithms and had a lot of research results. With the rise of deep learning and the improvement of computer computing power, more and more researchers began to use deep learning models to detect Android malware. This paper proposes an Android malware detection model based on a hybrid deep learning model with deep belief network (DBN) and gate recurrent unit (GRU). The main contributions are as follows:

(i) In order to resist Android malware obfuscation technology, in addition to extracting static features, we also extracted the dynamic features of malware at runtime and constructed a comprehensive feature set to enhance the detection capability of malware.

(ii) A hybrid deep learning model was proposed. According to the characteristics of static features and dynamic features, two different deep learning algorithms of DBN and GRU are used.

(iii) The detection model was verified, and the detection result is better than traditional machine learning algorithms; it also can effectively detect malware samples using obfuscation technology.

**ANDROID APPLICATIONS**

Android applications, or simply "apps," are software programs designed to run on mobile devices powered by the Android operating system. Since its introduction in 2008, Android has become the most widely used mobile operating system, powering over 70% of smartphones and tablets worldwide. As a result, Android apps have become an essential component of modern life, used for everything from communication to entertainment, productivity, and health and fitness.

Android apps can be downloaded from the Google Play Store, which offers a wide range of free and paid apps. They can also be installed manually by downloading the APK (Android Package) file from a third-party source and installing it on the device. However, it is recommended to download apps from trusted sources only, such as the Google Play Store, to avoid malware or other security risks.

Android apps are built using the Java or Kotlin programming languages, with the Android SDK (Software Development Kit) providing the necessary tools and resources for developers to create apps that can run on Android devices. The SDK includes a range of libraries, APIs (Application Programming Interfaces), and tools that allow developers to create apps with features such as multimedia support, location-based services, push notifications, and more.

One of the key advantages of Android apps is their flexibility and customization. Developers can create apps for a variety of purposes and target specific audiences, from casual users to professionals. Apps can be designed to meet different needs, such as communication, entertainment, productivity, and education. This flexibility also allows developers to create apps that work across different devices, from smartphones and tablets to wearables and smart TVs.

Another advantage of Android apps is their availability. The Google Play Store offers a vast selection of apps, with over 3.5 million apps available as of 2021. This means users can find apps for almost anything they need, from social media and messaging to gaming, finance, and fitness. Many of these apps are also free or offer a free version with limited features, allowing users to try before they buy.

However, with so many apps available, it can be challenging to find the right app for a particular need. Users should research apps before downloading to ensure they are safe, reliable, and meet their needs. They should also be cautious when granting permissions to apps, as some apps may collect personal information or access device features without the user's knowledge.

In conclusion, Android apps have become an integral part of modern life, providing a range of features and services that can enhance productivity, entertainment, and communication. With the flexibility and customization offered by the Android SDK, developers can create apps for a wide range of purposes and target specific audiences. However, users should be cautious when downloading and using apps to ensure they are safe, reliable, and meet their needs.

**MALWARES AND THREATS IN ANDROID APPLICATIONS**

Malware and security threats are a growing concern for Android users as the popularity of Android devices continues to increase. Malware is malicious software that can infect an Android device and cause various issues such as data loss, unauthorized access, and financial loss. In this article, we will discuss some of the common types of malware and security threats that Android users may encounter while using apps.

**Trojan malware:**

1. Trojan malware is one of the most common types of malware that infects Android devices. It disguises itself as a legitimate app, and once it is installed, it can steal personal information, monitor the user's activities, and even control the device remotely. Trojan malware is often spread through social engineering techniques, such as phishing emails or fake app stores.

**Adware:**

1. Adware is a type of malware that displays unwanted ads on the device. These ads can be intrusive and annoying, and in some cases, they may also contain malicious content. Adware is often bundled with free apps or games, and it can be difficult to detect and remove.

**Spyware:**

1. Spyware is a type of malware that is used to spy on the user's activities. It can track the user's location, monitor their calls and messages, and even record their keystrokes. Spyware is often used for malicious purposes, such as stealing sensitive information or blackmailing the user.

**Ransomware:**

1. Ransomware is a type of malware that encrypts the user's files and demands payment in exchange for the decryption key. Ransomware attacks are becoming more common on Android devices, and they can cause significant financial loss and data loss.

**Malicious apps:**

1. Malicious apps are apps that are designed to infect the device with malware. They can be disguised as legitimate apps, such as games or productivity tools, and they may contain hidden malware that can harm the device or steal personal information.

**EVOLUTION OF MALWARE**  
In order to protect networks and computer systems from attacks, the diversity, sophistication and availability of malicious software present enormous challenges. Malware is continually changing and challenges security researchers and scientists to strengthen their cyber defences to keep pace. Owing to the use of polymorphic and metamorphic methods used to avoid detection and conceal its true intent, the prevalence of malware has increased. To mutate the code while keeping the original functionality intact, polymorphic malware uses a polymorphic engine. The two most common ways to conceal code are packaging and encryption . Through one or more layers of compression, packers cover a program's real code. Then the unpacking routines restore the original code and execute it in memory at runtime. To make it harder for researchers to analyse the software, crypters encrypt and manipulate malware or part of its code. A crypter includes a stub that is used for malicious code encryption and decryption. Whenever it's propagated, metamorphic malware rewrites the code to an equivalent. Multiple transformation techniques, including but not limited to, register renaming, code permutation, code expansion, code shrinking and insertion of garbage code, can be used by malware authors. The combination of the above techniques resulted in increasingly increasing quantities of malware, making time-consuming, expensive and more complicated forensic investigations of malware cases. There are some issues with conventional antivirus solutions that rely on signature-based and heuristic/behavioural methods. A signature is a unique feature or collection of features that like a fingerprint, uniquely differentiates an executable. Signature-based approaches are unable to identify unknown types of malware, however. Security researchers suggested behaviour-based detection to overcome these problems, which analyses the features and behaviour of the file to decide whether it is indeed malware, although it may take some time to search and evaluate.

**Malware**, short for malicious software, consists of programming (code, scripts, active content, and other software) designed to disrupt or deny operation, gather information that leads to loss of privacy or exploitation, gain unauthorized access to system resources, and other abusive behaviour. It is a general term used to define a variety of forms of hostile, intrusive, or annoying software or program code. Software is considered to be malware based on the perceived intent of the creator rather than any particular features. Malware includes computer viruses, worms, Trojan horses, spyware, dishonest adware, crime-ware, most rootkits, and other malicious and unwanted software or programs .

In 2008, Symantec published a report that "the release rate of malicious code and other unwanted programs may be exceeding that of legitimate software applications.” According to F-Secure, "As much malware was produced in 2007 as in the previous 20 years altogether.”.

Since the rise of widespread Internet access, malicious software has been designed for a profit, for example forced advertising. For instance, since 2003, the majority of widespread viruses and worms have been designed to take control of users' computers for black-market exploitation. Another category of malware, spyware, - programs designed to monitor users' web browsing and steal private information. Spyware programs do not spread like viruses, instead are installed by exploiting security holes or are packaged with user-installed software, such as peer-to-peer applications.

Clearly, there is a very urgent need to find, not just a suitable method to detect infected files, but too build a smart engine that can detect new viruses by studying the structure of system calls made by malware.

**2. Current Antivirus Software**

Antivirus software is used to prevent, detect, and remove malware, including but not limited to computer viruses, computer worm, Trojan horses, spyware and adware. A variety of strategies are typically employed by the anti-virus engines. Signature-based detection involves searching for known patterns of data within executable code. However, it is possible for a computer to be infected with new virus for which no signatures exist. To counter such “zero-day” threats, heuristics can be used, that identify new viruses or variants of existing viruses by looking for known malicious code. Some antivirus can also make predictions by executing files in a sandbox and analysing results.

Often, antivirus software can impair a computer's performance. Any incorrect decision may lead to a security breach, since it runs at the highly trusted kernel level of the operating system. If the antivirus software employs heuristic detection, success depends on achieving the right balance between false positives and false negatives. Today, malware may no longer be executable files. Powerful macros in Microsoft Word could also present a security risk. Traditionally, antivirus software heavily relied upon signatures to identify malware. However, because of newer kinds of malware, signature-based approaches are no longer effective.

Although standard antivirus can effectively contain virus outbreaks, for large enterprises, any breach could be potentially fatal. Virus makes are employing "oligomorphic", "polymorphic" and, "metamorphic" viruses, which encrypt parts of themselves or modify themselves as a method of disguise, so as to not match virus signatures in the dictionary.

Studies in 2007 showed that the effectiveness of antivirus software had decreased drastically, particularly against unknown or zero day attacks. Detection rates have dropped from 40-50% in 2006 to 20-30% in 2007. The problem is magnified by the changing intent of virus makers. Independent testing on all the major virus scanners consistently shows that none provide 100% virus detection.

**Static and dynamic analyses**

Static and dynamic analyses are sources for security features, which the secu- rity investigator uses to decide the maliciousness of a given application. Manual inspection of these features is a tedious task and could be automated using ma- chine learning techniques. For this reason, the majority of the state-of-the-art malware detection solutions use machine learning techniques. Developers must carefully choose between the two analysis techniques depending upon the re- quirements of the model and the kind of applications the model is expected to deal with.

Static analysis is performed in a non-run-time environment. It basically in- volves looking at the application from inside out without executing the program, but rather by examining the source code, byte code or application binaries for signs of security vulnerabilities. In the static analysis, the application data and control paths are modeled and then analyzed for security weaknesses. Dynamic analysis adopts the opposite approach and is executed while a program is in op- eration. It looks at the application by examining it in its running state and trying to manipulate it in order to discover security vulnerabilities. The behaviour of the application in dynamic analysis is studied by emulating it on a sandbox, with a monkeyrunner giving random inputs to the application.

Static analysis, with its white-box visibility, is certainly the more thorough ap- proach and may also prove more time and cost efficient with the ability to detect malware. Static analysis can also unearth some properties that would not emerge in a dynamic test. Dynamic analysis, on the other hand, is capable of exposing a subtle vulnerability too complicated for static analysis alone to reveal, espe- cially in cases of code encryption and obfuscation. A dynamic test, however, will only find defects in the part of the code that is actually executed. The developer must weigh up these considerations with the complexities of their own situation in mind while choosing the best analysis technique. Application type, time, and availability of resources are some of the primary concerns.

## **Images Impact Page Load Time**

Did you know that the images you choose can also impact the functionality and even the security of your website? How many times have you visited a website and for some strange reason the images on the website seem to take forever to load? The reason is not that strange. Sometimes a web designer will forget to reduce the size of the image to something that is more appropriate for a fast-loading website. The result is an image file that’s so large, it can take what seems to be forever to load.

And the longer a page takes to load, the worse it can be for your business. Apart from the fact that your website visitors might just get irritated and leave your site, pages that are slow to respond could also cause you problems with the major search engines. Slow-loading pages can have an impact on search engine ranking. The search engines depend on satisfied surfers who will use their search engine again and again, and poor results can impact that. So as punishment the search engines may not show your website at all. But that’s not the only risk images can present.

## **Hidden Malware in Images**

In their relentless pursuit of the next vulnerability to exploit, hackers have found a way to embed malware in the code that images are made of. For years, hackers and spammers have been renaming their files to end with .jpg, .tif, or .png so that when they emailed malware to their targets, the recipients assumed they were receiving image files and therefore were safe to open. The bad guys used to package their malware in .doc or.zip files, but as more users became more wary of opening such files, the hackers had to get even more creative.

But last year, researchers found that hackers had devised a clever way of hiding malware inside jpeg images. The researchers identified an attack where the hackers would first try to breach the security of a website in order to install their malware, and then hide the malware inside an image where it couldn’t be detected by malware scanners.

And those are not the only tricks hackers have up their sleeves. On many websites, images can include links so that clicking on the image will take the user to a related page or website. Hackers can change the code in that link so that visitors are guided to a page that looks similar to the one you had originally linked to but instead tricks the visitor into downloading malware or revealing sensitive information. This kind of attack was discovered almost a decade ago so you can bet it’s advanced pretty far since then.

That’s why it’s so important not to take your website images for granted. They can slow down page loading speeds, irritate customers, and cost you sales. Worse than that, they can make your website a security minefield for the customers you rely on to survive.

## **How To Reduce Your Risk**

* Talk to your web designers and make sure they’re aware of the risks and taking steps to reduce them.
* Reduce all website images to the smallest size or dimensions you actually need or can get away with. After optimizing the file size of all your images, look into utilizing a Content Delivery Network (CDN) to further accelerate page load times.
* Control access to your image libraries so that hackers can’t alter the images for malicious purposes.
* Make sure all the images you use are appropriate, represent your brand and business the way you want, and are appropriately licensed. Remember, just because someone in your business or who works for you gave you the images doesn’t mean they or you have a right to use them.
* Once loaded on your web pages, check your images regularly to make sure that any links in them go where they’re supposed to.
* Malware in images can’t launch on its own–it needs code in a text file to activate it. Set up a website scanner to monitor your website text files for known malware strains.

**Codes :**

**mode.py**

#Importing some libraries

import binascii

import sys

#Function to display warning message

def unsafe():

print("Warning!!! This Gif is not safe.")

sys.exit()

#Function to skip the extension blocks

def extension(idx):

idx = idx+2 #Skip 21h

while True:

if hexc[idx:idx+2] == 'f9': #Graphics controller extension

idx = idx+14 #Skip the rest of the extension block

if hexc[idx:idx+2] == '01': #Plain Text extension

s = int(hexc[idx+2:idx+4],16) #Calculate the number of bytes to skip

idx = idx+2

idx = idx+(s\*2) #Skip the rest of the extension. After this, there will be an image data block.

return idx

if hexc[idx:idx+2] == 'ff': #Application extension

s = int(hexc[idx+2:idx+4],16) #Calculate the number of bytes to skip

idx = idx+2

idx = idx+(s\*2) #Skip the rest of the extension. After this, there will be an image data block.

return idx

if hexc[idx:idx+2] == 'fe': #Comment extension

idx = idx+2

return idx #After this, there will be an image data block.

if hexc[idx:idx+2] == '2c': #Image separator

packed = hexc[idx+18:idx+20] #Packed field

packed = "{0:08b}".format(int(packed, 16))

LCT = 0 #Number of Local colour table entries

if(packed[7] == '1'): #If Local colour table exists

N = int(packed[0:3],2)

N = N+1

LCT = pow(2,N) #Calculate number of Local colour table entries

idx = idx+20 #Skip Image descriptor

idx = idx+(3\*LCT) #Skip local colour table

idx = idx+2 #Skip LZW minimum code size

return idx

#Giving Input as file names

if \_\_name\_\_ == "\_\_main\_\_":

#filename = 'mix\_gifs/kitty\_m.gif'

filename = 'gifs/dog.gif' #GIF File

try:

with open(filename, 'rb') as f:

content = f.read()

except IOError:

print("File not found!")

sys.exit()

hexc = binascii.hexlify(content) #Load the Hexadecimal version of the GIF file

hexc = hexc.decode("utf-8")

if(hexc[0:4] != '4749'): #Check file type

print("The file is not a gif")

sys.exit()

if(hexc[8:12] != '3961'): #Check GIF file version. We support only GIF version 89a

print("The file is not an 89a gif")

sys.exit()

packed = hexc[20:22] #Packed field of the Logical Screen Descriptor

packed = "{0:08b}".format(int(packed, 16))

GCT = 0 #Number of Global colour table entries

if(packed[0] == '1'): #If Global colour table exists

N = int(packed[5:8],2)

N = N+1

GCT = pow(2,N) #Calculate number of Global colour table entries

skip = (6\*2)+(7\*2)+(3\*GCT) #Skip header, logical screen descriptor and global colour table

idx = skip

while True:

if hexc[idx:idx+2] == '2c': #Image separator

packed = hexc[idx+18:idx+20] #Packed field

packed = "{0:08b}".format(int(packed, 16))

LCT = 0 #Number of Local colour table entries

if(packed[7] == '1'): #If Local colour table exists

N = int(packed[0:3],2)

N = N+1

LCT = pow(2,N) #Calculate number of Local colour table entries

idx = idx+20 #Skip Image descriptor

idx = idx+(3\*LCT) #Skip local colour table

idx = idx+2 #Skip LZW minimum code size

if hexc[idx:idx+2] == '3b': #Trailer

break

elif hexc[idx:idx+2] == '21': #Extension block

idx = extension(idx) #Skip extension blocks

else: #Image data blocks

while hexc[idx:idx+2] != '00': #While block size not equal to zero

s = int(hexc[idx:idx+2],16) #Number of bytes to skip

idx = idx+2

idx = idx+(s\*2)

idx = idx +2

if(len(hexc) != idx+2): #If there exists some data after the trailer 0x3b

unsafe()

else:

print("congratulations!! This Gif is safe to use")

**Conclusion:**

Any smartphone which uses android applications is potentially vulnerable to security breaches, but Android devices are more lucrative for attackers. This is due to its open-source nature and the larger market share compared to other operating systems for mobile devices.. We have proposed a malware detection module based on advanced coding for checking from different colours of gif as images. It differs from the images containing the colours with images with malware. This will not only easily detect known viruses, but act as a knowledge that will detect newer forms of harmful files. While a costly model requires costly infrastructure, it can help in protecting invaluable enterprise data from security threats, and prevent immense financial damage.

**REFERENCES:**

[1]http://www.us-cert.gov/control\_systems/pdf/undirected\_attack0905.pdf

[2] "Defining Malware: FAQ". http://technet.microsoft.com. Retrieved 2009-09-10.

[3] F-Secure Corporation (December 4, 2007). "F-Secure Reports Amount of Malware Grew by 100% during 2007". Press release. Retrieved 2007-12-11.

[4] History of Viruses. http://csrc.nist.gov/publications/nistir/threats/subsubsection3\_3\_1\_1.html [5] Landesman, Mary (2009). "What is a Virus Signature?” Retrieved 2009-06-18.

[6] Christodorescu,M., Jha, S., 2003. Static analysis of executables to detect malicious patterns. In: Proceedings of the 12th USENIX Security Symposium. Washington .pp. 105-120.

[7] Filiol, E.,2005. Computer Viruses: from Theory to Applications. New York, Springer, ISBN 10: 2- 287-23939-1.

[8] Filiol, E., Jacob, G., Liard, M.L., 2007: Evaluation methodology and theoretical model for antiviral behavioral detection strategies. J. Comput. 3, pp 27–37.

[9] Cai, H.; Ryder, B.G. Understanding Android application programming and security: A dynamic study. In Proceedings of the 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, 17–22 September 2017; pp. 364–375.

[10] Liu, K.; Xu, S.; Xu, G.; Zhang, M.; Sun, D.; Liu, H. A Review of Android Malware Detection Approaches Based on Machine Learning. *IEEE Access* **2020**, *8*, 124579–124607.

[11] Gilski, P.; Stefanski, J. Android os: A review. *Tem J.* **2015**, *4*, 116.

[12] Garg, S.; Baliyan, N. Comparative analysis of Android and iOS from security viewpoint. *Comput. Sci. Rev.* **2021**, *40*, 100372.

[13] Odusami, M.; Abayomi-Alli, O.; Misra, S.; Shobayo, O.; Damasevicius, R.; Maskeliunas, R. Android malware detection: A survey. In *International Conference on Applied Informatics*; Springer: Cham, Switzerland, 2018; pp. 255–266.

[14] Bhat, P.; Dutta, K. A survey on various threats and current state of security in android platform. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–35.

[15] Tam, K.; Feizollah, A.; Anuar, N.B.; Salleh, R.; Cavallaro, L. The evolution of android malware and android analysis techniques. *ACM Comput. Surv. (CSUR)* **2017**, *49*, 1–41.

[16] R. Surendran, T. Thomas, and S. Emmanuel, “A TAN based hybrid model for android malware detection,” *Journal of Information Security and Applications*, vol. 54, article 102483, 2020.

[17] W.-C. Kuo, T.-P. Liu, and C.-C. Wang, “Study on android hybrid malware detection based on machine learning,” in *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, pp. 31–35, Singapore, 2019.

[18] H. Zhang, S. Luo, Y. Zhang, and L. Pan, “An efficient android malware detection system based on method-level behavioral semantic analysis,” *IEEE Access*, vol. 7, pp. 69246–69256, 2019.