Science World Journal Vol. 20(No 1) 2025 https://dx.doi.org/10.4314/swj.v20i1.22
www.scienceworldjournal.org
ISSN: 1597-6343 (Online), ISSN: 2756-391X (Print)
Published by Faculty of Science, Kaduna State University

AN EXPERIMENTAL MALWARE FOR SMARTPHONES

*Nyamtswam Ngunengen, Agaji Iorshase, Gbaden Terlumun

Department of Computer Science, Joseph Sarwuan Tarka University, Makurdi, Benue State, Nigeria

*Corresponding Author Email Address: nnengen@gmail.com

ABSTRACT
Android operating system has become one of the platforms developers used to introduce their malicious activities into the smartphone world through Android Applications (App). Although the Google Play Store implements security countermeasures against Android malware, these measures have vulnerabilities. A major weakness is that users often accept all requested permissions as mandatory when installing an application, without understanding the risks. This gives developers the basis to achieve their illicit actions. The aim of the study is to develop a malware application for identification and to exploit vulnerabilities within the android operating system. The work adopted the object-oriented analysis and design methodology (OOAD). Context diagram was used to represent data flow in the malware application and Sequence diagram was used to show the interactions between objects in the application and the sequential order that those interactions occurred. Further, a randomized algorithm was used for the detailed design. The work developed a malware application that kept track of user tasks but at the background modified contacts list causing inconveniences to the user. The malware application replaced the contact list with random strings from set of alphanumeric characters. The malware application simulated a real-world cyber threat, contacts modification, to uncover vulnerabilities that evade detection through conventional security approaches. By exploring this attack vector, the study provided empirical evidence of vulnerabilities that was exploited by the malicious application developed. This study contributed to the broader field of cyber security research by providing experimental evidence and insights into the specific vulnerabilities and attack vectors targeting Android operating system.

Keywords: Android Operating System, Smartphone, Permissions, vulnerabilities, Attacks, Malware.

INTRODUCTION
According to Statista (2024), the global smartphone penetration rate was estimated at 69 percent in 2023, up from 2022. This estimate is based on roughly 6.7 billion smartphone subscriptions worldwide. Android is the leading operating system, powering an impressive 3.3 billion device with a dominant market share of 70.79%. Android surpasses other mobile operating systems like iOS and Windows (Yunmar *et al*., 2024). The operating system is developed by Google, and is designed mainly for devices like smartphones. It utilizes touch inputs, including dragging, tapping and pinching, to interact with on-screen elements and a virtual keyboard (Lazareska & Jakimoski, 2017).
A smartphone is a mobile phone that offers advanced computing ability and connectivity. A smartphone run applications on platforms like Android, iOS, Symbian, Windows, BlackBerry providing a platform for application developers. All smartphones have lots of features in common however, based on their type, they use different operating system with different design and functionality (Ahvanooey *et al*., 2017).
Android smartphones are becoming a dominant form of mobile computing. Its openness provides a favourable atmosphere for developers. The operating system offers a rich, flexible and user-centric experience. Its open-source nature, extensive device range, vast application ecosystem and high level of customization make it an appealing choice for a wide variety of users. However, the extensive use of Android has made it a prime target for cybercriminals. In the third quarter of 2022, Kaspersky Security Network reported over five million instances of mobile malware, with new threats appearing daily and approximately 98% of mobile malware targets Android smartphones (Yunmar *et al*., 2024). These malicious applications perform activities such as unauthorized access, data theft, surveillance, downloading additional code, and sending unauthorized messages. An example is Valentine's Day attack in which attackers distributed a mobile picture sharing application that secretly sent text messages from the user's mobile phone without the user's knowledge (Zaidi *et al*., 2016).

Despite existing security countermeasures, Android malware continues to evolve, exploiting permission vulnerabilities and bypassing detection mechanisms. This study investigates these weaknesses through an experimental malware application.

The goal of the study is to develop a malware application specifically to probe the Android Operating System (OS) for potential loopholes such as unauthorized access gain, and data manipulation, the study understands how vulnerabilities manifest and exploited. This was achieved by permission abuse. The malware requested permissions to access contacts, photos, media and files which are not required for the malware functionality but for malicious purpose.
This study provided empirical evidence of vulnerabilities that was exploited by malicious actor. Furthermore, this study contributes to the broader field of cyber security research by advancing knowledge about mobile device security and providing insights into the evolving tactics of cyber threats targeting Android operating system.

RELATED WORKS
The usage of smartphones is on the increase and this increases malware development, exploiting operating system vulnerabilities and applications weaknesses. Malicious applications crashes batteries, take control of the smartphone and performs illicit activities posing dangers to users even with the security countermeasures available (Jannatul *et al.,* 2023).
Jayanti (2024) indicated that smartphone offers connectivity and convenience to users, this made them attractive targets for malware developers to introduce their malicious activities into smartphone world through Android applications (app), and these malwares are introduced in forms, such as ransomware, viruses,

spyware and Trojans causing a significant risk to users' security. Heena & Maria (2022) also stated that botnet, virus, ransomware are malicious softwares that are causing problems to Android operating system these days, they cause harm and exploit other softwares because the detection methods lack accuracy to identify malwares in real time. Malware developers exploit this vulnerability to launch attacks such as denial of service, code injection, unauthorized access etc. leading to security risk that compromised security. Luay & Marwan (2024) published that Android operating system was launched in 2008 and is widely hosting millions of applications. Most of the applications are developed without thorough security checks, making them vulnerable to malware attacks. The target of malware developers is to compromise user's data and denial them access. There is a crucial need for a systematic approach to identify and evaluate vulnerabilities within the Android Operating System (OS) that surpasses traditional testing methods (Yunmar *et al*., 2024). Traditional testing methods, although essential, may not fully uncover all weaknesses within the Android OS due to their dependence on known attacks and vulnerabilities, most smartphone users do not recognize this security shortcoming and some fails to enable the security software that comes with their phones (Asoke &Sneha 2015). Therefore, there is a need for more research on the effective method that will secure the operating system.

Android applications have gradually become part of our daily lives, they are used in sending mails, doing business etc. what facilitate the development of these applications is the free source code, attackers leverage on this to release attractive applications embedded with malwares luring users into disclosing sensitive information and access gain (Albandari *et al*., 2024). Amira *et al*. (2020) highlighted that Android malware has been on increase due to the popularity of Android operating system. Some malwares do not need users' permission to function, once a user download the application, the malware is installed and run on the smartphone making the phone vulnerable to attacks such as stealing of sensitive information, unauthorized access, data theft, spy, and sending unauthorized messages. Also, advancement in technology too made smartphones to face cyber-attacks. Once malware affect a smartphone, it accesses files and cause damages. Atanda *et al*. (2020) also highlighted that Android operating system is an easy target for attackers because the market share of Android has increased. Threats of Android malware have increased due to the increasing popularity of Android smartphones. Once an Android smartphone is infected with malware, the user suffers from various damages, such as remote operations, and data lost.

Dar &Parvez (2016) developed a malicious application (app) in Android that was stored at the play store. This malicious app helped them to track and know the current location of the Android smartphone and also saves the call log which was sent to their application. Before a user install the app, permissions were requested for a successful installation. Their malicious application includes unnecessary permissions which were not required for the app functionality but for malicious activity. Once the app was installed, it automatically sends the position of the device and the call log details. The app acted as a spy as it sends all these details without the knowledge of the user. Their application was a spy, our application modified contact list making them incomprehensible.
From the literature reviewed on Android operating systems vulnerabilities, it is evident that there is no comprehensive solution to fully protect Android users. Moreover, the frequency of malware attacks continues to increase rapidly therefore.

MATERIALS AND METHODS

The research methodology used in this study is the Object-Oriented Analysis and Design Method (OOADM). It was chosen because it provides a structured, scalable, and reusable approach to software development. This method structures a project into distinct, well-defined tasks and outlines their sequence and interactions. Within the Object-Oriented paradigm, a set of diagramming techniques known as the Unified Modelling Language (UML) is employed. UML emphasizes three key architectural perspectives of a system: functional, static, and dynamic. The functional view characterizes the system's external behaviour as perceived by users, represented through use case diagrams. The static view addresses the system's attributes, methods, classes, relationships, and messages, using tools such as Class Responsibility Collaboration (CRC) cards, class diagrams, and object diagrams. The dynamic view is depicted using sequence diagrams, collaboration diagrams, and state charts. These diagrams are incrementally refined throughout the process until a comprehensive understanding of the system's requirements is achieved.
Context diagram was used to represent data flow in the malware application, the malware application sent a request to Android Operating System (OS) to access Data store where contacts are saved, the OS process the request and send contacts to the application, the malware now access the data storage directly and sent a request to modify the contacts list, the request was granted and the contacts modified as shown in figure 1.

The diagram in Figure 1 consists of the Malware Application (App) Components which accesses the Data Store through the Android OS.

Malware App: A malicious application that interacts with both the **Android OS** and **Data Store.** It requests **contact data** from the **Android OS**. It also sends a **data modification request** to manipulate stored data.

Data Store: A database or storage system where contact data is kept. The data store receives modification requests from the **Malware App** and responds accordingly.

Android OS: The operating system managing access to stored data. It handles requests from the **Malware App** for contact data, verifies and processes data before interacting with the **Data Store** then sends retrieved contact data back to the **Malware App**.

This diagram represents a potential attack where a malware app gains access to user data through unauthorized interactions with the **Android OS** and **Data Store**.

Science World Journal Vol. 20(No 1) 2025 https://dx.doi.org/10.4314/swj.v20i1.22
www.scienceworldjournal.org
ISSN: 1597-6343 (Online), ISSN: 2756-391X (Print)
Published by Faculty of Science, Kaduna State University

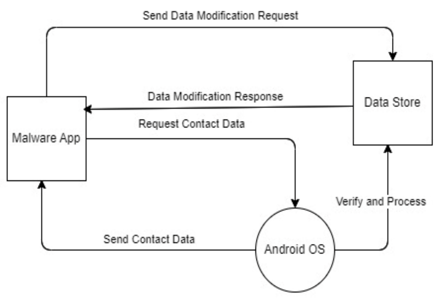


Figure 1: Context Diagram

Figure 2 shows the sequence diagram of the proposed system. Sequence diagram highlights the process flow of activities taking place due to the operation of the malware application (app). The malware app request access permission from the Android Operating system (AOS) to access the contact provider Application Programming Interface (API) that controls access to the contact storage. The AOS accepts the permission and forward the response back to the malware app which now has control of the AOS to access the contact provider API. The malware app then forwards another request to the contact provider which responds due to user permission setting but the response is overridden and bypassed and an update request sent to the contact storage which changes the contact details as shown in figure 2.
This sequence diagram in Figure 2 illustrates how a **malware app** exploits the Android system to access and modify contact information. The diagram includes four key components:
Malware App The malicious application requests access to the **Contact Provider API** to retrieve or modify contacts by sending a request to access the contact provider.
Android OS: The operating system processes the malware app's request. And grants permission if the user has allowed it. The OS then requests access to Android storage for contact data retrieval.
Contact Provider (API): Acts as an intermediary between applications and the **Android Contact Storage**. Once access is granted, it modifies contact data as requested by the malware app.
Android Contact Storage: This is the database where contact information is stored. It allows modifications once the request is verified and processed.
This diagram highlights a potential **security vulnerability**, where the malware app gained access to and modified contact information.

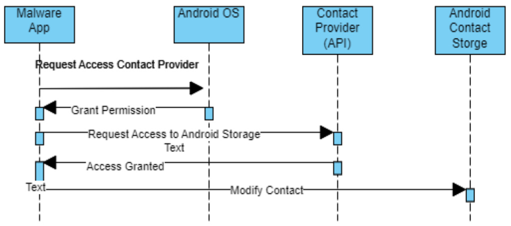


Figure 2: Sequence Diagram

Randomized Algorithm of the proposed system
Input Definition:
Let the input be a list of contact names, denoted as:$C = (c_1, c_2, \dots c_n)$

Where c_n is the last contact's name in the list and c_i represents the i-th contact name in the list.

Random String Generation:
Let the function$R_{rand(k,\Sigma)}$represent the process of generating a random string of length k from a character set Σ. The character set Σ is typically a union of characters (e.g., alphanumeric characters, and special symbols). Mathematically, $R_{rand(k,\Sigma)}$ can be defined as:$R_{rand}(k, \Sigma) = (r_1, r_2, \dots r_n)$,r_j∈ for 1≤j≤k
Where k is the length of the random string, and r_j is the j-th character in the generated string, chosen randomly from the character set Σ.

Random String Replacement Process:
The goal is to replace each contact name c_i with a random string. This is done by applying the function R_{rand} to each contact name. For each i-th contact name c_i, the corresponding random string r_j is generated by:$r_{j=R_{rand(k,\Sigma)}}$
Where: k is the finite length of the random string which can be fixed or determined dynamically by the malware.
Σ is the character set from which the random string is generated (e.g., alphanumeric characters or symbols).
Thus, the list of randomized contact names R(C) is given by: R(C) = $(r_1, r_2, \dots r_n)$
Where each$r_j$is a randomly generated string replacing the corresponding contact name c_i

Output Definition:
We can express the overall process of random string replacement as a function f that takes a list of contact names C and outputs a list of random strings R(C):
F(C) = $R_{rand(k,\Sigma), R_{rand}(k,\Sigma) \dots R_{rand}(k,\Sigma)}$
Where each $R_{rand}(k, \Sigma)$generates a random string for the corresponding contact name in C
.
Summary of the Randomized Algorithm
Step 1 - Start
Step 2 - Generate list of contacts, C= $[c_1, c_2, \dots c_n]$
Step 3 - For each i-th contact in the list generated

Step 4 - generate a new randomized contact name using $r_{i=R}$
Step 5 - Replace the i-th contact with the new contact's name
Step 6 – Stop

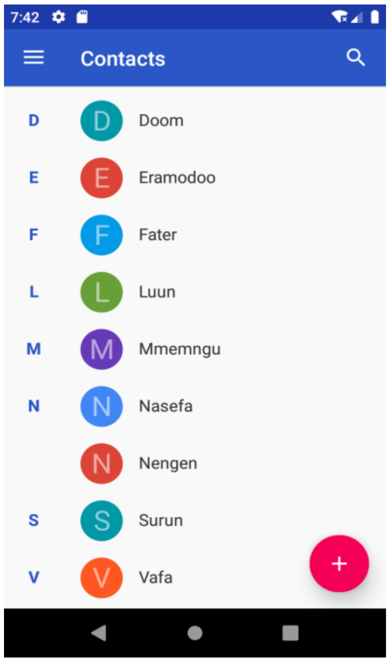
Experiment 1: Reaction of Malware Application (app) on the Contact List saved in the Android Smartphone.
Nine contacts were saved in the Android Smartphone, the user interacted with the malware app by assigning task, in the process, the malware secretly at the background, replaced the contact list with randomly generated string of characters from alphanumeric and special symbols which made the list incomprehensible. This is a function that takes each contact name and output a list of randomised string determined by the malware.

Experiment 2: Reaction of Malware Application (App) on the incomprehensible string from experiment 1.
The user assigned another task using the application, as this was taking place at the foreground, the malware modified the randomly generated string generated from experiment 1 to different strings, the user discovered this when contact list was visited.

Experiment 3: Reaction of Malware Application (App) on the degenerated string from experiment 2
The user assigned a third task to it daily schedules, as the user was busy scheduling the task, the malware changes the random string again to different ones. At each run of the algorithm, a new string of random bits is produced. It is evidence that provided us with data to verify and validate our results that the contact replacement is from random characters generation. It also supported our investigation that Android operating system is vulnerable

RESULTS

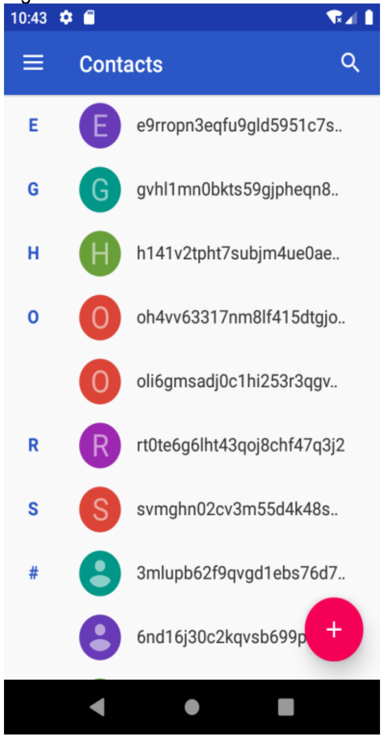
The malware application was developed in Java programming language using Android Studio as the Integrated Development Environment. In other to get results, an Android Emulator was used as Android Virtual Device. Also, three experiments were done to understand the underlying logic governing the application hence the malware is experimentally inclined. Below are results from the study.



D Doom
E Eramodoo
F Fater
L Luun
M Mmemngu
N Nasefa
Nengen
S Surun
V Vafa

Figure 3: Nine contacts in the phone

Figure 3 shows the contact list saved on the smartphone.

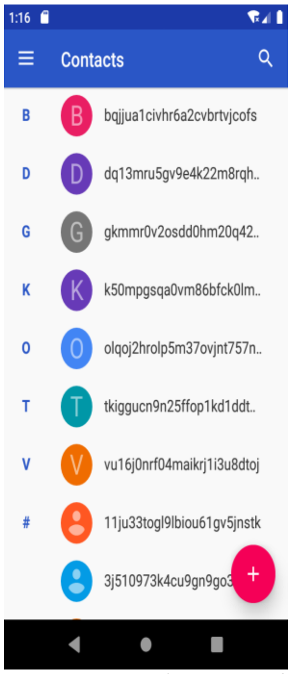


E e9rropn3eqfu9gld5951c7s..
G gvhl1mn0bkts59gjpheqn8..
H h141v2tpht7subjm4ue0ae..
O oh4vv63317nm8lf415dtgjo..
oli6gmsadj0c1hi253r3qgv..
R rt0te6g6lht43qoj8chf47q3j2
S svmghn02cv3m55d4k48s..
3mlupb62f9qvgd1ebs76d7..
6nd16j30c2kqvsb699p

Figure 4: **Result of experiment 1**

Figure 4 is the result of experiment 1. The malware application replaced the contact list with random strings from set of alphanumeric characters.

Science World Journal Vol. 20(No 1) 2025
www.scienceworldjournal.org
ISSN: 1597-6343 (Online), ISSN: 2756-391X (Print)
Published by Faculty of Science, Kaduna State University
https://dx.doi.org/10.4314/swj.v20i1.22

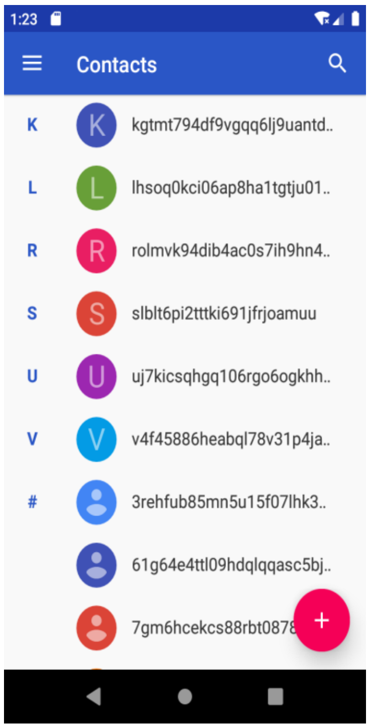

1:16

Contacts

B	bqjjua1civhr6a2cvbrtvjcofs
D	dq13mru5gv9e4k22m8rqh..
G	gkmmr0v2osdd0hm20q42..
K	k50mpgsqa0vm86bfck0lm..
O	olqoj2hrolp5m37ovjnt757n..
T	tkiggucn9n25ffop1kd1ddt..
V	vu16j0nrf04maikrj1i3u8dtoj
#	11ju33togl9lbiou61gv5jnstk
	3j510973k4cu9gn9go3

Figure 5: Result of experiment 2

Figure 5 is another random string generated when the second task was assigned, it helps us to understand the underlying logic governing the application. The malware functions whenever task is assigned by the user.


1:23

Contacts

K	kgtmt794df9vgqq6lj9uantd..
L	lhsoq0kci06ap8ha1tgtju01..
R	rolmvk94dib4ac0s7ih9hn4..
S	slblt6pi2tttki691jfrjoamuu
U	uj7kicsqhgq106rgo6ogkhh..
V	v4f45886heabql78v31p4ja..
#	3rehfub85mn5u15f07lhk3..
	61g64e4ttl09hdqlqqasc5bj..
	7gm6hcekcs88rbt0878

Figure 6: **Result of Experiment 3**

Figure 6 showed the degenerated string from the third experiment.

DISCUSSION

Figure 3 showed the contact list of people stored on the smartphone. It enables the user to store and manage the contact information of people. It is the contacts list the malware made incomprehensible from experiment 1.

Figure 4 is the result of the first task assigned by the user, as the user was busy assigning the task, secretly, the malware replaced each contact name with randomly generated string of characters from alphanumeric and special symbols. It is a function that takes each contact name and output a list of randomised string determined by the malware.

Figure 5 is the result of experiment 2, the malware application changed the result of experiment 1 to another set of randomized string when the user specified another task. This helps us to understand the underlying logic governing the application. The malware functions whenever task is assigned.

Figure 6 is the result of experiment 3. As the user was busy scheduling its task, the malware changes the random strings from experiment 2 to again, different ones. At each run of the algorithm, a new string of random bits is produced. This provided us with evidence to verify and validate our results that the contact replacement is from random characters generation. It also supported our investigation that Android operating system is vulnerable.

Malware Performance Evaluation

Metric Category	**Metric Name**	**Measured Value**	**Notes**
Resource Consumption	CPU Usage (%)	70%	High CPU usage indicates heavy processing.
	Memory Usage (MB)	250 MB	Unusual memory consumption for background apps.
	Battery Drain (mAh/hr)	350 mAh/hr	Malware consumes high battery power.
	Network Bandwidth Usage	500 KB/sec (upload)	Large amounts of data being sent externally.
Execution Time & Latency	Time to Execute Malicious Code	5 seconds	Quick execution of payload.
	System Response Delay	3 seconds	Noticeable lag in system responsiveness.
Stealth & Persistence	Detection Rate by Antivirus (%)	30%	Low detection rate means high stealth.
	Rootkit Persistence	Yes	Malware survives reboots.
Impact Analysis	User Data Leaked (MB)	250 MB	Large amount of sensitive data stolen.

Science World Journal Vol. 20(No 1) 2025
www.scienceworldjournal.org
ISSN: 1597-6343 (Online), ISSN: 2756-391X (Print)
Published by Faculty of Science, Kaduna State University
https://dx.doi.org/10.4314/swj.v20i1.22

Conclusion

The study developed a malware application that kept track of user tasks but at the background modified contacts list causing inconveniences to the user whenever the user wants to make calls or view contacts list. This was achieved by the flexibility of Android operating system. The malware application simulated a real-world cyber threat, contacts modification, to uncover vulnerabilities that evade detection through conventional security approaches. By exploring this attack vector, the study provided empirical evidence of vulnerabilities that was exploited by the malicious application developed.

We strongly advise Android smartphone users to be careful installing new applications because malicious application can slip past Google's security checks and take control of their device that is why Google remove malicious application from time to time to show it does not always catch everything before it gets on phone.

Also, there should be proactive methods that can systematically identify and mitigate potential vulnerabilities within the Android operating system to protect Android smartphone users against malware.

.

REFERENCES

Ahvanooey, M.T., Li, Q., Rabbani, M. & Rajput, A.R. (2017). A Survey on Smartphones Security: Software Vulnerabilities, Malware and Attacks. *International Journal of Advanced Computer Science and Application*8(10): 30-45.

Albandari, A., Heba, E., Mohamed, E., Zeyad, A., Fatemah, H. A., Majid, A., Sumayh, S. A., Sarah, A., Shahad, A. & Khadijah, A. (2024). A Study of Android Security Vulnerabilities and Their Future Prospects. *HighTech and Innovation Journal. 5(3): 855-869*

Amira, B. S., Mohammed, A., Sadeeq, M., Rizgar, R. Z., Maiwan, B. A., Mayyadah R. M.,Hanan, M. S.& Lailan, M. H. (2020). An Investigation for Mobile Malware Behavioral and Detection Techniques Based on Android Platform. *IOSR Journal of Computer Engineering (IOSR-JCE).* 22(4): 14-20.

Asoke, N. & Sneha, M. (2015). Impact of Mobile Phone/Smartphones. A Pilot Study on Positive and Negative Effects. *International Journal of Advance Research in Computer Science and Management Studies*3(5): 294-302

Atanda, A. O., Obi, A. M., Anyaorah, C. C., Idoko, N. A., Udechukwu, P. E., Anusiobi, C. L., Asogwa, S. & Senu, J. F. (2020) Design and Implementation of a Malware System on Smartphones. *International Journal of Research and Innovation in Applied Science* V(X): 63-68.

Dar, M. A. & Parvez. (2016). Smart phone Malware Threat, An Experimental Evaluation of Smart Phone Security. *International Journal of Computer Science and Information Security*14(8): 109 – 113.

Heena, K.S.K. & Maria A. (2022). A Hybrid Model for Android Malware Detection using Decision Tree and KNN. *International Journal on Recent and Innovation Trends in Computing and Communication.* 10*(*1): 321-328.

Jannatul. F., Rafigul. I., Arash, R. M. & Zahidul, M.D. I (2023). A Review of State-of-the-Art Malware Attack Trends and Defense Mechanisms in IEEE Access, vol 11, pp.121118 -121141. 2023, Digital Object Identifier*: 10.1109/ACCESS.2023.3328351.*

Jayanti, K. S. N. B. (2024). Detection of Malware in Android Smartphones Using Machine Learning. *International Journal of Research Publication and Reviews.*5(1) 232-234.

Luay A. & Marwan, O. (2024). Improving mobile security: A study on android malware detection using LOF*. International Journal of Mathematics and Computer in Engineering.**3**(2): 241-252.*

Lazareska, L. & Jakimoski, K. (2017). Analysis of the Advantages and Disadvantages of Android and iOS Systems and Converting Applications from Android to iOS Platform and Vice Versa. *American Journal of Software Engineering and Applications*, **6**(5): 116-120. *doi: 10.11648/j.ajsea.20170605.11*

Statista (2024). Smartphone penetration worldwide as share of global population 2016-2023. *Statista*. Accessed 14/06/2024 from *https://www.statista.com/statistics/203734/global-smartphone-penetration-per-capita-since-2005*

Yunmar, R. A., Kusumawardani, S. S., Widyawan & Mohsen, F. (2024). Hybrid Android Malware Detection: A Review of Heuristic-Based Approach in IEEE Access, vol. 12, pp. 41255-41286, 2024, doi: 10.1109/ACCESS.2024.3377658.

Zaidi, S. F. A., Shah., M. A. & Kamran, M., Javaid, Q. and Zhang, S. (2016). A Survey on Security for Smart Phone Device. *International Journal of Advanced Computer Science and Application*: **7**(4): 206 - 219.