Mobile Phone Firmware and Hardware Hacking Detection System

Michael Nhyk Ahimbisibwe and Noluntu Mpekoa
University of Johannesburg, South Africa

nhykmichael@live.com
noluntum@uj.ac.za

Abstract: Mobile devices have become prevalent due to the features they offer to their users, such as browsing the internet, digitising notes, sending and receiving invoices, asset management, recording signatures, checking emails and accessing social media platforms. In 2021, the number of mobile devices operating worldwide stood at almost 15 billion, expected to reach 18.22 billion by 2025. The sheer volume of sensitive information stored on these devices, from personal data to corporate credentials, makes them an enticing prospect for malicious actors. The increasing reliance on these mobile devices for personal and professional purposes underscores the importance of robust security measures. Modern hacking techniques often target mobile hardware and firmware vulnerabilities, jeopardising user privacy and data integrity. This research introduces the "Mobile Phone Firmware and Hardware Hacking Detection System", a comprehensive solution built with Python to detect unauthorised firmware and hardware modifications in mobile devices. The system integrates various modules, including tools for secure user interaction, machine learning-based for Android applications analysis, desktop user interface, and real-time threat detection. A meticulous review of existing research was conducted to gauge the current landscape of mobile phone hacking detection. The proposed system showcases innovative features like firmware attack detection, application behaviour analysis, and hardware integrity checks. This research addressed the escalating issue of mobile phone security by providing a system that can potentially thwart unauthorised access and data breaches. The system's implementation details include the user interface, Android app analysis, threat detection algorithm, firmware hack detector and the phone's low-level connector. Comparative analysis with existing solutions reveals the model's robustness in detecting hacking attempts while highlighting potential improvement areas. Although the system demonstrates significant capabilities, it is crucial to consider the potential challenges posed by more sophisticated firmware and hardware hacking techniques, such as those exploiting previously unknown vulnerabilities.

Keywords: Mobile phone hacking detection, Mobile firmware analysis, Mobile hardware analysis, Machine learning, Android application analysis

1. Introduction

Smartphones and mobile phones play a crucial role in our daily lives. They are used in various fields, including government, agencies, education, entertainment, and even the military. (Dhalaria and Gandotra, 2021; Senanayake *et al,* 2021) observed that the number of individuals using smartphones and their performance increases yearly. Because of their ease of use, users can store and perform many sensitive and significant tasks, making them a profitable target for attackers. In early 2024, South Africa had 118 million cellular mobile connections, which is 195% of the total population (Simon, 2024). This ubiquitous use of mobile devices in today's digital age has heightened the need for robust security measures (Van Heerden *et al,* 2016). Many personal and professional applications use these devices to store sensitive data, including social media and private and professional information. Users also perform sensitive tasks like mobile or internet banking, making these devices a prime target for unauthorised access and usage (Alqahtani *et al,* 2019; Donner, 2004).

Hacking attacks and the spread of malware have drastically increased over the years. Modern hacking attempts are more sophisticated, often exploiting vulnerabilities in mobile hardware and firmware (Friedman and Hoffman, 2008; Senanayake *et al,* 2021). Currently, mobile malware detection and defence technologies are still not sufficient. Chowdhury *et al* (2020) emphasised that mobile security depends not solely on the operating system and devices used but also encompasses factors such as internet communication, data encryption, data summarisation, and user privacy awareness. The availability of enticing features like unlimited internet access and many application options has created significant opportunities for malware developers (Ashawa and Morris, 2021; Raghuvanshi and Singh, 2022). According to the Kaspersky Security Network, over 9 million malware, adware, and riskware (legitimate but potentially harmful software) attacks were blocked on mobile devices in the third quarter of 2021 (Kaspersky, 2021). An attacker can exploit vulnerabilities in a mobile device security system by triggering any action that can lead to downloading an infected application that can compromise the device (Hossain and Riaz, 2022). This malware can disrupt networks and steal data, creating significant user cyber risks. Furthermore, Droos *et al* (2020) suggest that hackers may pose threats by exploiting network components or connection vulnerabilities within Bluetooth, Wi-Fi, or GPS.

Given this context, the paper aims to develop and enhance a mobile phone hacking detection system that incorporates detecting firmware and hardware hacking. This project seeks to address the escalating issue of

mobile phone security by providing a system that can potentially thwart unauthorised access and data breaches. Section 2 provides an in-depth exploration of the existing literature pertaining to mobile phone hacking detection systems, encompassing the contributions made by many researchers on this topic. Section 3 details the research methodology used in this study. Section 4 introduces the proposed solution, which includes a description of the system's key components, how they interact, and how they address the problem. Section 5 details the implementation of the model, providing insights into how the proposed system was brought to life. Section 6 presents the results obtained. Section 7 concludes the paper, summarising the research project's achievements and implications.

2. Literature Review

A substantial body of research exists on various aspects of mobile phone security. Some existing solutions focus on individual components of the problem, such as hardware security (Sadeghi, 2008; Tam *et al,* 2017) or firmware analysis (Liu *et al*, 2017; Nadir *et al,* 2022; Srivastava *et al,* 2019). In contrast, others emphasise specific types of hacking detection, such as signature-based, found in (Mos and Chowdhury, 2020; Muzaffar *et al,* 2022). MADAM, developed by Dini et al (2012), is a multi-level detector that identifies malicious behaviours on Android Mobile Phones. It monitors the system kernel and user space, including signature-based, static and dynamic analysis methods to catch malware effectively. The authors initially developed a specialised prototype to detect signs of malicious attacks and a technique to identify previously unknown malware for the Android Operating System (OS). Through rigorous testing, the system demonstrated low false favourable rates. MADAM utilises machine learning techniques alongside a dataset with compact parameters and 13 proven effective characteristics to describe phone behaviours. The MADAM-based system evaluated over 50 typical applications during testing and exhibited low memory, battery, and Central Processing Unit (CPU) consumption, proving performance efficiency (Dini *et al,* 2012).

To examine firmware for potential hacking detection, Hernandez et al (2019) developed a system to extract and analyse firmware images from mobile phones. The extracted Android includes file systems such as rootfs, which grants access to user space. They faced challenges in collecting and unpacking firmware images from various vendors due to the fragmentation of the Android platform. Using domain knowledge and specialised tools, the study extracted over 3,500 unique AT commands ("AT" meaning 'attention'). Both static and dynamic analysis were employed to understand the functionality of these commands, leading to the discovery of undisclosed firmware vulnerabilities. The study highlighted the need for standardised firmware images and modular frameworks to facilitate deeper investigation and improve mobile device security. However, this approach does not address specific firmware threats, such as checking whether the entire firmware has been compromised or replaced with a malicious version during manufacturing or shipping. On the hardware security front, studies like Yadav *et al* (2022) have proposed methods for examining the hardware of mobile phones to detect anomalies. Andromaly is a detection technique that monitors the system's behaviour to identify suspicious activities exhibited during the phone's hardware analysis. It achieves this by critically evaluating the CPU and battery consumption rates and network packet transfer. However, the system has limitations, particularly its ability to detect malware and respond accordingly. This is especially true for system classes that need rapid response times. Additionally, high-speed network capacities make it more challenging to process supervised data samples effectively. As a result, there is a clear need for an adaptable model to meet better the extensive requirements of anomaly detection in hardware behaviours.

Vulnerabilities in mobile devices stem from flaws in the operating system and hardware, making the device susceptible to attacks (Yadav and Gupta, 2022). The proposed solution is an innovative software-based approach to improve mobile phone security against firmware and hardware hacking efforts. The solution mimics specialised hardware without the accompanying costs by merging hardware-based detection techniques with the versatility of the software. Polymorphic adaptability is included in this paradigm for the system's real-time reaction to new and evolving threats, making it more resilient against sophisticated attacks. Core features include firmware attack detection, which uses machine learning to analyse pulled files to make predictions, application behavioural analysis, processor backdoor defect discovery, hardware integrity check, and firmware integrity verification.

3. Research Methodology

The main aim of this study was to design and develop a mobile phone hacking detection system that incorporates detecting firmware and hardware hacking. A literature review was used to get an in-depth examination of the current state of mobile phone security (presented in section 2). This method helped the

study identify existing mobile phone security gaps and best practices for developing a defensive algorithm that fills those gaps. The detection tool was implemented using Python as the programming language of choice. Running a mobile phone hacking detection system in Python necessitated the use of several libraries and tools that interact with one another:

- Scapy is a packet manipulation library that allows one to craft custom packets and analyse network traffic. It can be used to monitor and study mobile phone network traffic. This helps examine network traffic and the system's functionality to detect suspicious activities (Brahmanand *et al,* 2022).
- PyUSB is a Python library that provides a simple and easy-to-use interface for USB devices. It allows the system to communicate using the USB protocol and offers a range of functions for device discovery, configuration, and data transfer. It supports USB 1. x and USB 2.0 devices and works on various platforms, including Windows, Linux, and macOS. Additionally, it can interact with other USB devices, including storage, webcams, and smartphones. This interaction with USB devices assists the proposed system in detecting mobile phone hardware-based attacks (GitHub, 2024; Hale et al*.,* 2020; Mohd Roki, 2013).
- PySerial is a library that provides a simple interface for hardware devices to communicate with serial ports. It can be used to detect attacks that involve USB-to-serial adapters (Ma *et al,* 2021; Maged *et al,* 2023).
- The Volatility Framework is an open-source memory forensics framework that can extract digital artefacts from a volatile memory image of a system. It includes a library called Volatility Library, a Python-based library that can be used to analyse memory images using the Volatility Framework programmatically. This library can analyse memory dumps, which can help detect malware running in memory. This library was used to detect attacks on the hardware level (Mohanta *et al,* 2020; Song *et al,* 2018).
- The Android Debug Bridge (ADB) command-line tool was used to communicate with an Android device over a USB or network connection to extract device logs and activity (Amarante and Barros, 2017).
- Capstone library and Interactive Disassembler (IDA) Pro library were used to disassemble binary code to analyse firmware images so that the system can detect malicious code embedded in or any firmware-compromising files (Rays, n.d.; Solovev *et al,* 2018).
- Tkinter is a Python binary module that, when accessed, the developer can use its low-level interface for Graphical User Interface (GUI) design (Python Software Foundation, 2019). The main interface in this project was developed using this library.

Python was a preferred programming language due to its powerful and high-level nature, which has been established over time. In addition to being an interpreted and object-oriented language, Python's syntax resembles plain English text, making it simple to write intricate algorithms (Saabith *et al,* 2019). For the proposed system evaluation, various metrics, such as the tool's accuracy, detection rate, recall, F-measure, false negative rate (FNR), and false-positive rate (FPR), to measure the tool's effectiveness in detecting firmware and hardware-level attacks were used.

4. Proposed Solution

This research project focuses on developing and enhancing a mobile phone hacking detection system that incorporates detecting firmware and hardware hacking. This project seeks to address the escalating issue of mobile phone security by providing a system that can potentially thwart unauthorised access and data breaches. The proposed solution is a Python-based system incorporating various methods to detect phone hacking at the firmware hardware level. The system can also manage low-level device connections, display vital components of the phone where possible malicious modifications may occur, and analyse critical firmware behaviours and hardware integrity. The proposed system comprises five vital components ensuring its security and functionality (Figure 1 below).

Figure 1: Components of the proposed solution

1. **LoginUI**: is a dedicated module that facilitates secure user interactions. It enables users to sign up and log in, ensuring only authorised persons can access the system's core functionalities.

2. **PhoneLowLevelConnector**: creates a bridge between the system and mobile phones. This connection is established through Android Debug Bridge (ADB), enabling the developer's mode connections. The PhoneLowLevelConnector supports pulling files from the device execution of low-level operations and ensuring that the system remains in sync with the mobile phone, enabling real-time analysis and action.
3. **FirmwareHackDetector**: is the third major component, the heart of the system. It is responsible for directly interfacing with mobile phones. The FirmwareHackDetector captures logs and core system state, inspects installed packages, and uses these details to detect potential unauthorised activities or tampering.
4. **PermissionBasedHackingDetection**: Triggers device activities logs including a record of requested permissions, which are pulled using "ADB" commands and saved and analysed using a neural network (NN) classifier.
5. **APKManifestoExtractor**: is a utility for extracting manifest information from Android Package (APK) files. The extracted data provides insights into the permissions requested and other metadata of the applications and other configurations, which is essential for the subsequent analysis phase, ensuring a data-driven approach to threat detection. This component is followed by **APKRandomForestTrainer,** a supervised machine-learning component that trains a Random Forest classifier using features extracted from Android application packages. Then, based on behavioural classification, the trained model classifies new application packages as benign or potentially malicious, enabling proactive threat detection (Ferdous *et al,* 2023).
6. **FirmwareHackDetectionInterface**: This is the live state of the user interface when a device has been successfully detected over the USB connection, as indicated in Figure 3. Before the device is connected using debug mode, this interface is not available. It comprises access to all the tool's functionalities and detection options, such as hardware, firmware, permissions and APK analysis.

At the start, users securely access the system through *LoginUI* (see Figure 2). Upon successful login, the *FirmwareHackDetectionInterface* presents various options to scan the firmware or APKs. The system utilises APKManifestoExtractor to offload the application package from the device and extract AndroidManifest.xml before saving it in a local location. The information in AndroidManifest.xml (i.e., permissions requested and other static information related to the app) is then used as input for the Random Forest model, which *APKRandomForestTrainer* has trained for classification purposes. For firmware inspection, the *PhoneLowLevelConnector* establishes a connection with the target device. The *FirmwareHackDetector* then uses adb-shell to pull and analyse device activity logs, including logcat data, system dump information, firmware properties, hardware properties, vendor security settings, and more, to identify any signs of hacking or unauthorised changes on firmware and hardware level. The analysis results are readily available in real-time on the *FirmwareHack-DetectionInterface*, ensuring users get immediate feedback on potential threats.

The proposed system uses an integrated approach to thoroughly evaluate the applications (APKs) and firmware for potential threats. By utilising machine learning and low-level device interactions, it provides a complete solution to identify any unauthorised hardware and firmware modifications, possible malware app, or suspicious Android firmware activities, which ultimately leads to improved mobile security (Liu *et al,* 2022; Renjith *et al,* 2022).

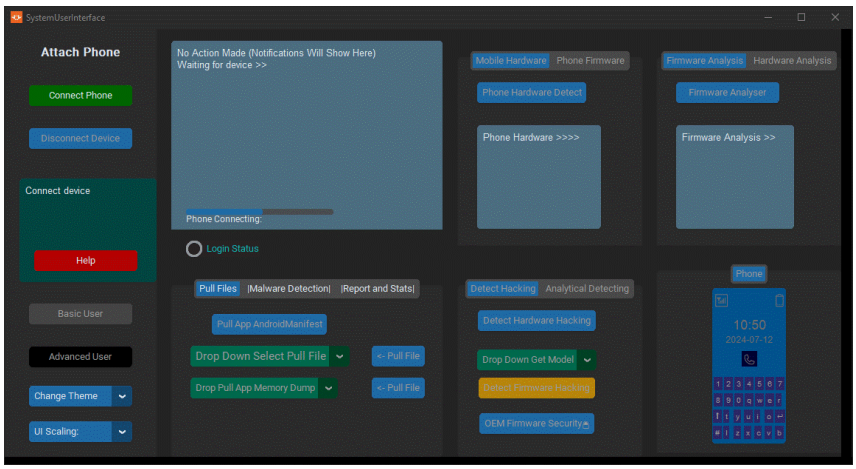

Figure 2: User interface – disabled device (firmware hack detection disabled because no device detected)

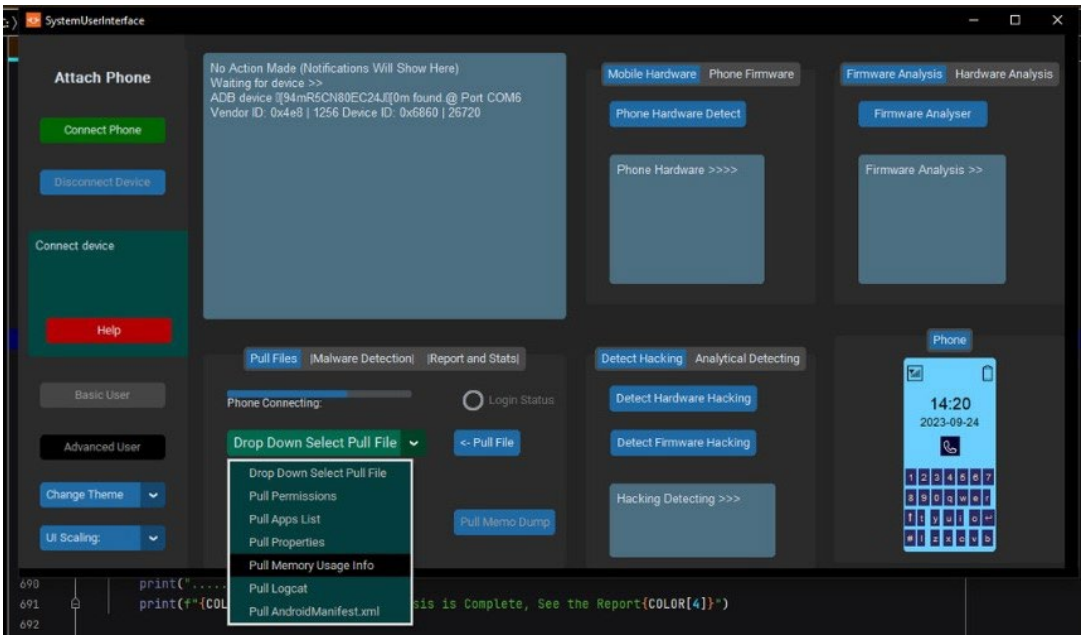

Figure 3: User interface – device detected (all functionalities activated)

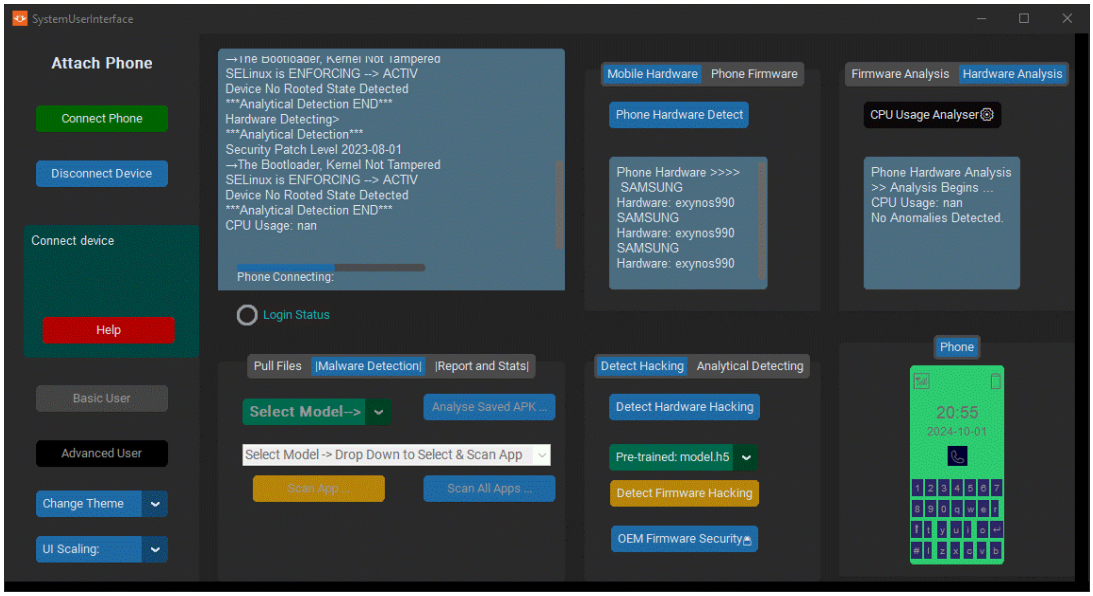

Figure 4: User interface – no malicious activities identified

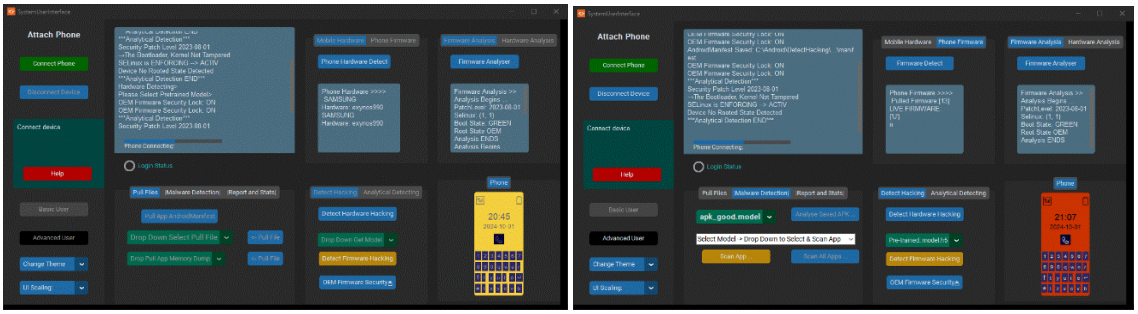

Figure 5: User interface – the image on the left indicates minor suspicious activities found, and on the right indicates major malicious activities found

The system user interface facilitates mobile phone analysis and security assessment by providing an intuitive and interactive platform. It enables users to inspect the hardware and firmware of a connected phone, detect potential hacking attempts, and evaluate the overall security posture of the device. The user interface is built

using the *tkinter* library augmented with *customtkinter* to offer a tailored user experience. This interface offers interactive features and serves as the gateway to the system's core functionalities.

5. Results

The "Mobile Phone Firmware Hardware Hacking Detection" system results from the seamless integration of software engineering principles, modern machine learning techniques, and direct mobile device interactions. The system benefits from the python's versatility and extensive library support to provide a robust and scalable solution. The Random Forest algorithm, facilitated by the *sklearn* library, is the heart of app threat detection. By training on features extracted from APKs, the system distinguishes between benign and potentially malicious applications. The Android Debug Bridge (ADB) makes direct interactions with mobile devices possible. The SimpleShiftEncryption module safeguarding user credentials and sensitive data was incorporated to ensure security.

The system was designed to conduct firmware and hardware analysis with the following sample results on an Android phone: Patch Level: 2023-06-01; SELinux Status: Enforcing (indicated by the tuple (1,1)); Boot State: GREEN, indicating a secure and uncompromised boot process; Root State: The device is in its Original Equipment Manufacturer (OEM) state, suggesting that it has not been tampered with or rooted.

Table 1: Shows diagnostic outcomes from significant parts of hardware and firmware

Diagnostic type	Level	Results
Detect hardware	Hardware	Return chipset installed
Firmware analysis	Firmware	Return firmware version
Battery usage	Hardware	Return battery usage stats
Security patch level	Firmware	Return date
SELinux	Firmware	Enforcing or otherwise
Boot state analysis	Firmware	Green, Yellow, Unknown
Root state	Firmware	Must return OEM
Detect hardware	Hardware	Return chipset installed

Table 1 above presents the firmware analysis results and provides a snapshot of the device's security posture, emphasising its up-to-date patch level and secure boot and root states. These details are vital for ensuring that mobile devices are safe and trustworthy. The Central Processing Unit (CPU) Usage Analysis is presented in Figure 6 below, indicating the CPU load percentage between intervals separated by one second. After every second, the analysis computes CPU percentage using a z-score (i.e., a measure of how many standard deviations above or below the population mean).

```
Raw Output: Load: 13.1 / 13.25 / 14.15
CPU usage from 58670ms to 12417ms ago (2023-10-23 16:26:37.326 to 2023-10-23 16:27:23.578):
  3.2% 1477/system_server: 1.5% user + 1.6% kernel / faults: 726 minor
  1.1% 2138/com.android.systemui: 0.8% user + 0.3% kernel / faults: 504 minor 2 major
  0.7% 31471/com.android.chrome: 0.6% user + 0% kernel / faults: 550 minor
  0.6% 1121/surfaceflinger: 0.3% user + 0.3% kernel / faults: 3 minor
  0.6% 5312/com.sec.android.app.samsungapps: 0.3% user + 0.2% kernel / faults: 43 minor
```

Figure 6: CPU usage analysis results

Figure 7 indicates the final z-score of standard deviations below or above the population mean. Suppose the z-score is greater than 2 standard deviations away from the mean. In that case, CPU usage can be considered suspicious, and there are no anomalies if the z-score is within 2 standard deviations.

```
  +0% 32566/kworker/u19:2-thermal_check: 0% user + 0% kernel
1.7% TOTAL: 0.8% user + 0.8% kernel + 0% softirq

No Anomalies Detected.
   CPU Usage (%)  z-score
0            1.7      NaN
```

Figure 7: Standard deviations final score

The final two steps of the system evaluation involved creating and assessing two unique Random Forest models to detect malicious mobile applications and manually implementing the Neural Network model to perform permission-based behavioural analysis. The random forest model underwent training on distinct sets of Android packages to assess their performance across diverse situations. Model 1 was trained using application packages of 1000 malware samples and 1000 benign apps. Model 2, on the other hand, was trained on packages with 50 malware samples and 20 benign apps.

Table 2: Model evaluation results

Random forest classification model results				
	Accuracy	*Recall*	*Precision*	*F-Measure*
Model 1	93.10%	91.67%	91.67%	91.67%
Model 2	97.67%	100%	91.56%	98.77%

Upon comparing the two models, it became evident that Model 1 exhibits superior performance across all metrics compared to Model 2 (Table 2 above). The disparity highlights the possible impact of the training dataset's composition on the model's performance. Model 1, when trained on a more evenly distributed dataset, has superior precision, recall, and total accuracy compared to Model 2. For phone permission-based analysis, Table 3 presents the training results. Table 3 indicates that the model accurately distinguishes malicious behaviours within extracted logged permissions. With a loss of 0.0639 and an accuracy of 97.84%, the model effectively identifies devices exhibiting malicious behaviours. If the model predicts a class of 'S', the device is considered compromised; if it predicts a class of 'B', the device is considered clean.

Table 3: Neural network classifier results

Permission-based neural network classifier results				
Training samples	*Testing Samples*	*Classes*	*Loss*	*Accuracy*
12028	3008	['B','S']	0.0639	0.9784

The proposed system, "Mobile Phone Firmware Hardware Hacking Detection," aims to identify unauthorised modifications or potential threats in mobile phone firmware hardware and applications. The system integrates various modules, each fulfilling a specific role, ensuring a comprehensive and user-friendly approach to mobile security. The evaluation of the system demonstrated that Model 1, with a larger and balanced dataset, produces high performance across most metrics without drawbacks. Model 2 showed that the highest accuracy possible can be achieved even with limited applications and training time.

6. Conclusion

The current digital environment, marked by a growing dependence on mobile devices, highlights the need for robust security measures. The increasing sophistication of hacking methods, particularly those aimed at exploiting weaknesses in mobile applications, firmware and hardware, underscores the necessity of adopting a proactive stance towards mobile security. This research project aimed to develop a system known as the "Mobile Phone Firmware and Hardware Hacking Detection System" in order to address the aforementioned problem. The system's architecture comprises many components: user interface modules, machine learning-based APK analysis, direct mobile device inter-facing, and permission-based threat detection. This integration showcases the system's potential to serve as a holistic solution for the problem at hand. The performance indicators, particularly those pertaining to Model 1, further validated the system's capabilities. The model demonstrated robustness in differentiating between benign and malicious behaviours, with an accuracy rate of 97.67% and flawless recall.

The firmware study revealed findings that emphasised the device's adherence to current security patches and implementing a secure boot state. These results offered a concise overview of the device's security status, ensuring the security and integrity of mobile devices. However, while the achievements are notable, it is essential to acknowledge the underlying limitations and challenges. Detection issues may arise from increasingly sophisticated attempts at firmware and hardware hacking, necessitating constant innovation to ensure the system remains adaptable to emerging threats. The proposed solution serves as evidence for the potential of combining software engineering concepts, machine learning techniques, and direct device interactions to enhance mobile security. The research findings provide a substantial contribution to the field of

mobile phone security but also lay the groundwork for future advancements and enhancements in this crucial technology era.

References

Alqahtani, E.J., Zagrouba, R. and Almuhaideb, A. (2019), "A survey on android malware detection techniques using machine learning algorithms", *2019 Sixth International Conference on Software Defined Systems (SDS)*, IEEE, pp. 110–117.

Amarante, J. and Barros, J.P. (2017), "Exploring USB connection vulnerabilities on Android devices breaches using the Android debug bridge", SciTePress.

Ashawa, M. and Morris, S. (2021), "Analysis of mobile malware: a systematic review of evolution and infection strategies", جامعة نايف العربية للعلوم الأمنية.

Brahmanand, S.H., Lal, N.D., Sahana, D.S., Nijguna, G.S. and Nayak, P. (2022), "A Systematic approach of analysing network traffic using packet sniffing with scapy framework", *Computer Networks and Inventive Communication Technologies: Proceedings of Fourth ICCNCT 2021*, Springer, pp. 811–820.

Dhalaria, M. and Gandotra, E. (2021), "Android malware detection techniques: A literature review", *Recent Patents on Engineering*, Bentham Science Publishers, Vol. 15 No. 2, pp. 225–245.

Dini, G., Martinelli, F., Saracino, A. and Sgandurra, D. (2012), "MADAM: a multi-level anomaly detector for android malware", *International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security*, Springer, pp. 240–253.

Donner, J. (2004), "Microentrepreneurs and mobiles: An exploration of the uses of mobile phones by small business owners in Rwanda", *Information Technologies & International Development*, Vol. 2 No. 1, p. pp-1.

Ferdous, J., Islam, R., Mahboubi, A. and Islam, M.Z. (2023), "A State-of-the-Art Review of Malware Attack Trends and Defense Mechanism.", *IEEE Access*, IEEE.

Friedman, J. and Hoffman, D. V. (2008), "Protecting data on mobile devices: A taxonomy of security threats to mobile computing and review of applicable defenses", *Information Knowledge Systems Management*, IOS Press, Vol. 7 No. 1–2, pp. 159–180.

GitHub. (2024), "pyusb/pyusb", available at: https://github.com/pyusb/pyusb (accessed 3 October 2024).

Hale, J., Habib, A., Raval, R., Irvin, R. and Hawrylak, P.J. (2020), "A cyber-physical system testbed for security experimentation", *Cyber Security of Industrial Control Systems in the Future Internet Environment*, IGI Global, pp. 175–209.

Van Heerden, R., Von Soms, S. and Mooi, R. (2016), "Classification of cyber attacks in South Africa", *2016 IST-Africa Week Conference*, IEEE, pp. 1–16.

Hernandez, G., Fowze, F., Tian, D.J., Yavuz, T., Traynor, P. and Butler, K.R.B. (2019), "Toward automated firmware analysis in the iot era", *IEEE Security & Privacy*, IEEE, Vol. 17 No. 5, pp. 38–46.

Hossain, M.S. and Riaz, M.H. (2022), "Android malware detection system: a machine learning and deep learning based multilayered approach", *Intelligent Computing & Optimization: Proceedings of the 4th International Conference on Intelligent Computing and Optimization 2021 (ICO2021) 3*, Springer, pp. 277–287.

Kaspersky. (2021), "Kaspersky Security Bulletin 2021. Statistics", October, available at: https://go.kaspersky.com/rs/802-IJN-240/images/KSB_statistics_2021_eng.pdf (accessed 3 October 2024).

Liu, M., Zhang, Y., Li, J., Shu, J. and Gu, D. (2017), "Security analysis of vendor customized code in firmware of embedded device", *Security and Privacy in Communication Networks: 12th International Conference, SecureComm 2016, Guangzhou, China, October 10-12, 2016, Proceedings 12*, Springer, pp. 722–739.

Liu, Y., Tantithamthavorn, C., Li, L. and Liu, Y. (2022), "Deep learning for android malware defenses: a systematic literature review", *ACM Computing Surveys*, ACM New York, NY, Vol. 55 No. 8, pp. 1–36.

Ma, H., Li, L., Shao, F. and Liu, X. (2021), "Design of a Comprehensive Experimental Platform for Intelligent Robots Based on Machine Vision", *International Workshop of Advanced Manufacturing and Automation*, Springer, pp. 390–396.

Maged, R., Sabry, M., Ali, A., Mesabah, I., Mazhr, A. and Soubra, H. (2023), "An Intrusion Detection System for Smart Autonomous E-Bikes", *2023 Eleventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, IEEE, pp. 233–240.

Mohanta, A., Saldanha, A., Mohanta, A. and Saldanha, A. (2020), "Memory forensics with volatility", *Malware Analysis and Detection Engineering: A Comprehensive Approach to Detect and Analyze Modern Malware*, Springer, pp. 433–476.

Mohd Roki, M.A. (2013), "USB DEVICE CONNECTIVITY USING PYTHON", Universiti Teknologi Petronas.

Mos, A. and Chowdhury, M.M. (2020), "Mobile security: A look into android", *2020 IEEE International Conference on Electro Information Technology (EIT)*, IEEE, pp. 638–642.

Muzaffar, A., Hassen, H.R., Lones, M.A. and Zantout, H. (2022), "An in-depth review of machine learning based Android malware detection", *Computers & Security*, Elsevier, Vol. 121, p. 102833.

Nadir, I., Mahmood, H. and Asadullah, G. (2022), "A taxonomy of IoT firmware security and principal firmware analysis techniques", *International Journal of Critical Infrastructure Protection*, Elsevier, Vol. 38, p. 100552.

Python Software Foundation. (2019), "tkinter — Python interface to Tcl/Tk", available at: https://docs.python.org/3/library/tkinter.html (accessed 3 October 2024).

Raghuvanshi, P. and Singh, J.P. (2022), "Android malware detection using machine learning techniques", *2022 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, pp. 1117–1121.

Rays, H. (n.d.). "ida-pro – Hex Rays", available at: https://hex-rays.com/ida-pro (accessed 3 October 2024).

Renjith, G., Vinod, P. and Aji, S. (2022), "Evading machine-learning-based Android malware detector for IoT devices", *IEEE Systems Journal*, IEEE, Vol. 17 No. 2, pp. 2745–2755.

Sadeghi, A.-R. (2008), "Trusted computing—special aspects and challenges", *International Conference on Current Trends in Theory and Practice of Computer Science*, Springer, pp. 98–117.

Senanayake, J., Kalutarage, H. and Al-Kadri, M.O. (2021), "Android mobile malware detection using machine learning: A systematic review", *Electronics*, MDPI, Vol. 10 No. 13, p. 1606.

Simon, K. (2024), "Digital 2024: South Africa", February, available at: https://datareportal.com/reports/digital-2024-south-africa#:~:text=Data%20from%20GSMA%20Intelligence%20shows,total%20population%20in%20January%202024. (accessed 3 October 2024).

Solovev, M.A., Bakulin, M.G., Gorbachev, M.S., Manushin, D.V., Padaryan, V.A. and Panasenko, S.S. (2018), "Next generation intermediate representations for binary code analysis", *Proceedings of the Institute for System Programming of the RAS*, Institute for System Programming of the Russian Academy of Sciences, Vol. 30 No. 6, pp. 39–68.

Song, W., Ryu, D. and Webb, R.I. (2018), "Volatility dynamics under an endogenous Markov-switching framework: a cross-market approach", *Quantitative Finance*, Taylor & Francis, Vol. 18 No. 9, pp. 1559–1571.

Srivastava, P., Peng, H., Li, J., Okhravi, H., Shrobe, H. and Payer, M. (2019), "Firmfuzz: Automated iot firmware introspection and analysis", *Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things*, pp. 15–21.

Tam, K., Feizollah, A., Anuar, N.B., Salleh, R. and Cavallaro, L. (2017), "The evolution of android malware and android analysis techniques", *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, Vol. 49 No. 4, pp. 1–41.

Yadav, C.S. and Gupta, S. (2022), "A review on malware analysis for iot and android system", *SN Computer Science*, Springer, Vol. 4 No. 2, p. 118.

Yadav, C.S., Singh, J., Yadav, A., Pattanayak, H.S., Kumar, R., Khan, A.A., Haq, M.A., *et al.* (2022), "Malware analysis in IoT & android systems with defensive mechanism", *Electronics*, MDPI, Vol. 11 No. 15, p. 2354.