International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 05 | May - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Analysis and Identification of Malicious Mobile Applications Software

Lokesh J [1], S. Yeswanth sai [2], Sheetal Ramesh [3], Veerendragouda H [4], Dr Pakruddin B [5]
[1234] UG student, Dept. of Computer Science & Technology, Presidency University, Bengaluru [5] Assistant Professor, Dept. of Computer Science & Engineering, Presidency University, Bengaluru.

Abstract - The widespread adoption of mobile applications has made smartphones an indispensable part of our life. But as customers' reliance on mobile applications has grown, they are now increasingly susceptible to rogue apps that pose a severe threat to device security and data privacy. The average user might not have the technical know-how to tell the difference between safe and dangerous software, therefore it might be challenging for them to identify such programmes before installing them.

A web-based malware detection tool called Cyber Rakshak File Scanner is provided by the "Analysis and Identification of Malicious Mobile Applications" project as a remedy for this issue. The Virus Total API is used to scan mobile application files that users upload to this system. The output of multiple antivirus engines is combined by Virus Total to produce a comprehensive and accurate assessment of the file's security.

Real-time analysis and thorough reports that emphasise the threat category, antivirus engine, and detection method are provided by Cyber Rakshak. This technology decreases the risk of cyberattacks and fosters a safer mobile environment by enabling users to make educated decisions before installing apps.

1.INTRODUCTION

In the current digital world, where mobile devices are indispensable to daily life, criminal attacks on mobile apps are becoming commonplace. Apps are regularly downloaded by users without their knowledge, putting their personal data, privacy, and device integrity at risk. The necessity for practical technology that can assess and identify potentially harmful mobile apps before they are loaded is highlighted by this expanding threat.

Cyber Rakshak File Scanner is a simple yet powerful web-based tool that is presented by the project "Analysis and Identification of Malicious Mobile Applications" to detect malware in mobile application files. Users can submit mobile app files (such as APKs) to this user-friendly application, which will analyse the files using the Virus Total API, a reputable and well-known platform that aggregates results from multiple antivirus engines.

Cyber Rakshak provides real-time scanning results along with detailed reports on the threats associated with the uploaded material. Through this platform, users may find out crucial details about a mobile application's security before installing it. By offering a useful and reliable method of spotting risky apps early on, we hope to promote safer mobile use by empowering people to take charge of their digital health and make informed choices.

2. LITERATURE SURVEY

2.1 Abstract :

As mobile applications play a bigger role in everyday life, they have become a prime target for cyberattacks. The growing sophistication of mobile infections is making it tough for traditional antivirus technologies to keep up. Cyber Rakshak, a web-based file scanner that analyses malware using the Virus Total API, was developed with the help of this literature review, which looks at existing techniques for finding and classifying harmful mobile phone applications.

A variety of methodologies are covered in the articles under review, including hybrid systems that integrate static and dynamic analysis, machine learning classifiers like Support Vector Machines (SVM), and deep learning models like Gated Recurrent Units (GRUs). Memory dumps, permission tracking, and API calls are used for behavioural analysis, which significantly increases the accuracy of detection. Furthermore, cutting-edge solutions to current cybersecurity issues include federated learning and fine-grained access control in mobile edge computing, which offer scalable and privacy-conscious solutions.

2.2 Introduction :

The rapid growth of mobile applications, particularly on open platforms like Android, has increased the potential for cyberattacks. Malicious apps have the ability to disrupt services, jeopardise user privacy, and steal crucial information. Having effective and user-friendly detection systems is more crucial than ever because malware can readily mimic legitimate applications.

Scholars have explored a range of methods to address this issue, from machine learning and deep learning to hybrid models that incorporate both static and dynamic analysis of applications. In addition to showing the range of attack pathways, these techniques offer insights into developing dependable, scalable malware detection programmes.

For instance, studies that use Support Vector Machines (SVM) and Gated Recurrent Units (GRUs) to classify malware based on app behaviour have demonstrated exceptional detection accuracy. Federated learning could be used in decentralised situations, particularly in the healthcare industry, by facilitating model training without compromising data privacy. These methods are the basis for the architecture of Cyber Rakshak, which prioritises security and privacy in malware detection procedures.

Security features such as fine-grained access control in mobile edge computing further emphasise the importance of limiting access and safeguarding sensitive data during the detection phase. Despite being a web-based scanner at the moment, Cyber Rakshak may use these techniques in future iterations to enhance user authentication and secure file processing.

In conclusion, there are many techniques and resources available in the literature that have a direct impact on the development and conception of Cyber Rakshak. The goal of this project is to combine proven methods from behaviour analysis, machine learning, and safe system architecture to create a malware detection platform for mobile applications that is easy to use, accurate, and efficient. Because of this, users will be able to identify threats before they compromise the security of their data or devices.

2.3 Methodologies :

2.3.1 Hybrid Malware Classification Using LLMs and Knowledge Graphs:

The method uses calibrated large language models (LLMs) (e.g., LLaMA) in conjunction with a cybersecurity knowledge graph (KG) augmented with MITRE ATT&CK data.
working: Network packets and memory dumps are routed into optimised LLMs for first classification. The KG is used to produce reasoning through prompt chaining. The final forecasts are enhanced by the KG's contextual understanding and entity association.
High classification accuracy was attained by striking a balance between detection performance and interpretability (91.2% for network data, 94.35% for memory data).

2.32 IoHT Cyberattack Detection Using Federated Learning:
Gated Recurrent Units (GRU) are used in this decentralised federated learning architecture to identify time-series patterns.
working: IoHT devices, or edge clients, collaboratively train a global model without sharing sensitive local data. This program can recognise five types of cyberattacks and was trained on the ECU-IoHT dataset.
Benefit provides outstanding generality and scalability, lowers the chance of data leaks, and has a stated accuracy of 99.65%.

2.3.3 Static Analysis with SVM Classifier:
Support Vector Machine (SVM) categorisation is utilised for static reverse analysis of Android APKs.
operating: Permissions, API calls, and directory structures are extracted. An beneficent or malevolent binary classification vector is produced by combining these traits.

Result: SVM's effectiveness in feature-based classification was proven by its high accuracy, low false positives, and high true positive rates.

2.3.4 End-to-End Mobile Malware Behavior Modeling:
An execution engine for virtual machines with both static and dynamic analysis was part of its usage techniques.
Reverse engineering and knowledge extraction from APKs are operational. Malware behaviour is investigated over a variety of transmission channels, including as SMS, Bluetooth, and 5G. Knowledge maps and behaviour graphs are used to represent threat spread and traceability.

result: was the availability of detailed modelling of malware behaviour, which was useful for predicting and demonstrating attack paths.

3. RESEARCH GAPS OF EXISTING METHODS

3.1 Limited Real-Time Implementation and User Accessibility:

While many existing systems employ sophisticated models such as federated learning, LLMs, or graph-based behavioural analysis to detect malware accurately, they often lack implementations that are visible to users in real-time. Most research does not offer simple, easy-to-use platforms that let people test apps before installing them, and it is restricted to enterprise-level or experimental technologies. This limits the applicability of these solutions, especially for non-technical audiences and everyday mobile users.

3.2 Over-reliance on Known Signatures and Static Features :

Many studies use classifiers like SVM for identification and primarily use static analysis (such permissions, directory structures, and API requests). These techniques often fall short when it comes to detecting unknown or disguised malware. Due to their inability to keep up with quickly evolving malware versions that use evasion techniques to evade standard inspections, static-only systems are extremely vulnerable.

3.3 Lack of Integrated Multi-Layered Analysis in Lightweight Platforms:

Although these integrations are typically computationally intensive and inappropriate for web-based, lightweight applications, static and dynamic analysis are occasionally merged in research. Insufficient research has been done on web-deployable, lightweight architectures that offer comprehensive analysis without requiring a large amount of hardware resources or system-level integration.

3.4 Explainability and Transparency in Malware Detection:

One study uses LLMs and knowledge graphs to explore explainable AI, although most research do not provide interpretable insights into the decision-making process. End users often have binary outcomes (malicious/benign), with no indication of the behaviours or traits that lead to the classification. This lack of transparency hinders trust in the detection process, especially for consumers who are worried about security.

3.5 Decentralization and Data Privacy Trade-Offs:

Although federated learning preserves anonymity, it is not practical for public use cases due to its complexity and lack of federated setup infrastructure. A research gap is identifying a platform that offers privacy protection without the disadvantages of total decentralisation or federated learning protocols.

3.6 Insufficient Attention to Real-World Propagation Scenarios and Social Engineering:

While various research look into this problem, few integrate virus transmission via SMS, Bluetooth, or app stores into a comprehensive detection and reporting system. There are not enough proactive defences that consider the behavioural and social ways that malware spreads and provide users with helpful information or preventative recommendations based on the threat's origin.

3.7 Security Limitations in Mobile Edge Computing and IoT Contexts:

In mobile edge computing, FGAC addresses access control, although it does not expressly address virus detection. The literature has not fully explored the seamless integration of access security techniques with real-time malware analysis for edge and mobile devices.

3.8 Lack of Unified Benchmarking Across Approaches:

Even though most studies have acceptable accuracy metrics, it is difficult to evaluate their effectiveness since they do not follow conventional assessment procedures or use standardised datasets. A fragmented study area results from this, and generalisability across different kinds of mobile malware is called into doubt. For more accurate evaluations, your project can make use of well-known scanning tools (like Virus Total) that pull data from numerous malware databases.

3.9 Low Emphasis on Real-World Usability Testing :

Many studies use algorithmic performance instead of end-user testing or feedback techniques to assess usability, learnability, and efficacy in real-world scenarios. Malware detection platforms do not have a human-centered design. Cyber Rakshak, which is web-based and visually guided, has the potential to bridge this gap with its user-friendly results display and intuitive user interface.

3.10 Scalability Challenges in Complex ML Architectures:

Even though deep learning methods (such knowledge graphs or GRU in federated setups) are quite accurate, they are not lightweight or easily scalable. Their high requirements for computing infrastructure limit their usage to consumer-level tools or organisations with modest resources. The need for a hybrid paradigm that balances performance and accessibility is highlighted by this, and Cyber Rakshak aims to deliver it.

3.11 Lack of Cross-Platform Compatibility for Mobile Threat Analysis:

Even if your current project is primarily focused on APK files, identifying this gap can help you plan for future improvements or modular architecture. Because Android is an open environment, most articles focus exclusively on Android APKs, leaving a gap in malware detection techniques that work with other mobile systems, such as iOS, HarmonyOS, etc.

3.12 Limited Focus on Post-Detection User Empowerment:

Even once malware has been detected, the majority of systems end there. There are not enough useful solutions provided, such as how to delete harmful files, safeguard data, or alert authorities. Such post-scan defensive or educational capabilities could be added by Cyber Rakshak to make the service more complete.

3.13 Inadequate Protection for Non-App-Based Threats:

Malware is only found in mobile apps, according to several studies. In reality, additional sources of danger include malicious PDFs, URLs, email attachments, and embedded website content. This expands on the variety of risks that app-centric research currently may ignore. Cyber Rakshak may use VirusTotal's broad file-type compatibility to manage a greater variety of threats.

3.14 Absence of Community-Driven Threat Sharing:

Except for knowledge graphs and MITRE ATT&CK frameworks, none of the studies discuss crowdsourced or community-contributed threat intelligence. This has the potential to revolutionise the detection of zero-day or region-specific threats. In the future, Cyber Rakshak may use anonymous scan records submitted by users to grow a community-driven threat feed (with permission)..

4. METHODOLOGY

4.1 PROPOSED METHODOLOGY :

To efficiently detect malicious mobile applications, the Cyber Rakshak project uses a web-based scanning system. Virus Total API-based virus analysis, file handling and preparation, and web development integration are the primary methods employed. The frontend ensures simple file uploading and interaction with its HTML, CSS, and JavaScript structure. File transfers, Virus Total API calls, and backend interpretation of scan findings are all handled by Flask (Python). This approach accelerates malware detection by automating the file analysis process and rapidly giving users thorough scan data. A lightweight, scalable, and effective mobile security awareness system is produced by combining these strategies.

4.1.1 Web-Based File Upload and Scanning Interface:

Technology Used: HTML, CSS, JavaScript

The interface is simple and easy to use, allowing users to upload mobile application files, such as apk files, directly from their browser.

JavaScript employs asynchronous queries to communicate with the backend and handles file validation.

aim: ensures that file scanning from any device is accessible and easy to use. Because it removes the need for technical expertise, even non-technical people can identify infections.

4.1.2 Backend API Integration with Virus Total:

Utilised Technologies: Virus Total Public/Private API, Flask (Python)

After the file has been uploaded, the Flask server acts as a middleman to transmit it to Virus Total's API. By merging the results of over 70 antivirus engines with URL/domain blacklists, Virus Total examines files for malware signatures. The backend retrieves and formats the report that was acquired from Virus Total.

aim: enables real-time malware scanning without starting from scratch with a detection engine. utilises cloud-based cybersecurity intelligence to generate remarkably precise results.

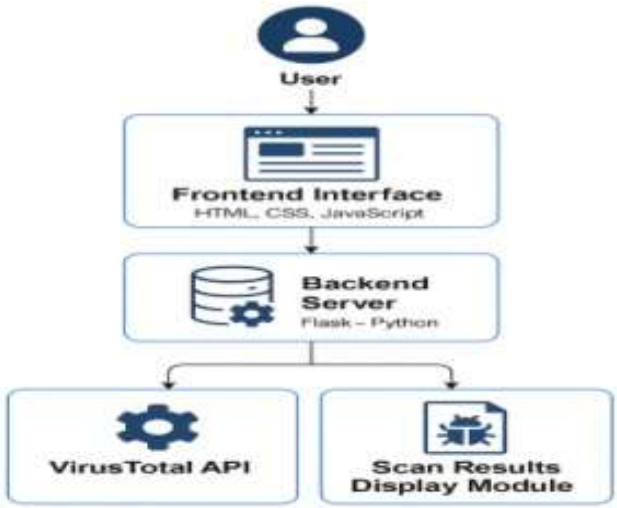
4.1.3 RESTful Architecture:

Flask routes were the technology used (POST, GET).

The application design's RESTful architecture enables separate endpoints to manage various operations, including

1) Uploading a file (scan) via POST

2) Checking the scan's status (GET /report)
It is obvious where presentation and logic diverge.
aim: Promotes scalability and makes it easier to integrate with emerging technologies like mobile apps and browser plugins.



4.1.4 Real-Time Risk Reporting:
JavaScript rendering and JSON parsing were the technologies used.
The Virus Total response includes SHA-256 hashes, threat categories (such trojan, adware, etc.), and scores (like the number of engines that reported the file). A visually appealing presentation of this data is produced by the frontend.
goal: Not only does it assess if a file is dangerous, it also provides users with information about how and why it is classified.

4.1.5 Logging and Anonymized User Interaction:
Utilised technologies: Python logging and Flask sessions
Simple logging mechanisms are used to track scans, timestamps, and file kinds. By not requiring users to register or provide personal information, anonymity is preserved.
aim: maintains a history for administrative insights or system problems while protecting user privacy.

4.1.6 Error Handling and Fail-Safe Mechanisms:
Try-except blocks and flask error handlers were the technologies used.
When the file type is not supported or Virus Total does not reply, the backend handles it gently and shows the appropriate error messages on the frontend.
aim: Ensures dependability and a flawless user experience even in case of a malfunction.

5. SYSTEM DESIGN & IMPLEMENTATION

5.1 components

5.1.1 User Interface (Frontend):
The primary layer of communication between users and the system is the User Interface (UI) of the Cyber Rakshak application. Users can upload Android programme files (typically.apk files) for malware detection thanks to its intuitive and user-friendly design, which was made with HTML, CSS, and JavaScript. The interface includes visual displays of the scan results retrieved from the backend, file input fields, and progress indications. By focussing on ease of use and simplicity, the scanning process is made seamless even for those with less technological knowledge. The frontend is also responsible for basic validation, error alerts, and starting the scanning process by sending backend API calls.

5.1.2 Backend Server (Flask Framework):
To build the backend functionality, Flask, a lightweight Python web framework perfect for building microservices and APIs, is utilised. The Flask server manages interactions with external services, such as the Virus Total API, securely handles file uploads, and reacts to incoming frontend requests. Before forwarding a file to the Virus Total endpoint, the backend verifies its size and format. The contents of the scan report are parsed and arranged into a readable format by the backend before being sent back to the frontend for user viewing. It is also possible to extend the backend to manage user sessions, record scanning activity, and store scan data in a database
.

5.1.3 Virus Total API (Malware Analysis Engine):
A key component of Cyber Rakshak's malware detection capabilities is its integration with the Virus Total API, a cloud-based service that aggregates results from over 70 antivirus engines and other malware detection technologies. An APK file is thoroughly inspected by many scanning engines upon submission to this API, which look for anomalies, odd activity, and known dangerous signatures in the file's contents. The API returns a JSON-formatted report with multiple engines' labels, behavioural analysis, detection ratios, and metadata like as file size, hash, and certificate details. This powerful third-party service significantly boosts Cyber Rakshak's threat detection capabilities' legitimacy and power.

5.1.4 Database (MongoDB or MySQL – Optional/Future Enhancement):

The Cyber Rakshak system benefits from having a database layer like MongoDB or MySQL, even if it was not necessary for the initial configuration. Later iterations might include authentication and history features, in which case this layer would contain scan results, timestamps, uploaded file records, and potentially user-specific data. A well organised database enables analytics, reporting, and historical tracking. Files that are frequently detected or recurring malware patterns, for example, can be identified. The application's scalability and personalisation are further improved by enabling the development of a user dashboard that gives registered users access to previous scans and reports.

5.1.5 Security and File Handling Layer:

Security is an important part of Cyber Rakshak's architecture because it works with potentially dangerous files. To ensure that only APK files are authorised and that size limits are not exceeded, the security and file handling layer integrates server-side validations to thwart denial-of-service attacks. Flask's secure file management features are used to appropriately manage and temporarily store uploaded files before they are examined. The program also closely follows input sanitisation and output encoding requirements and uses HTTPS protocols to protect data while it is in transit. Other features like rate limitation, user authentication, and scan frequency monitoring may be implemented in the future to guard against abuse.

5.2 system architecture :

5.2.1 User (Client-Side):

The user interacts with the programme to begin the journey using a web-based interface. Users can scan Android apps by uploading their.apk files using this user-friendly interface, viewing the scan's status and results, and determining the level of danger depending on the analysis.The user makes the request, and their experience is safeguarded, quick, and simple.

5.2.2 Web Application (Frontend - HTML/CSS/JavaScript):

For frontend development, HTML, CSS, and JavaScript are utilised. It is responsible for: getting the file that the user has submitted; providing feedback messages and upload progress; andsending HTTP requests (via the Fetch API or AJAX) to the Flask server at the backend. providing a clear and visually instructional display of the scan results, including the risk score, threat kind, and engine reports.

This layer ensures that users may engage with the system in an efficient manner.

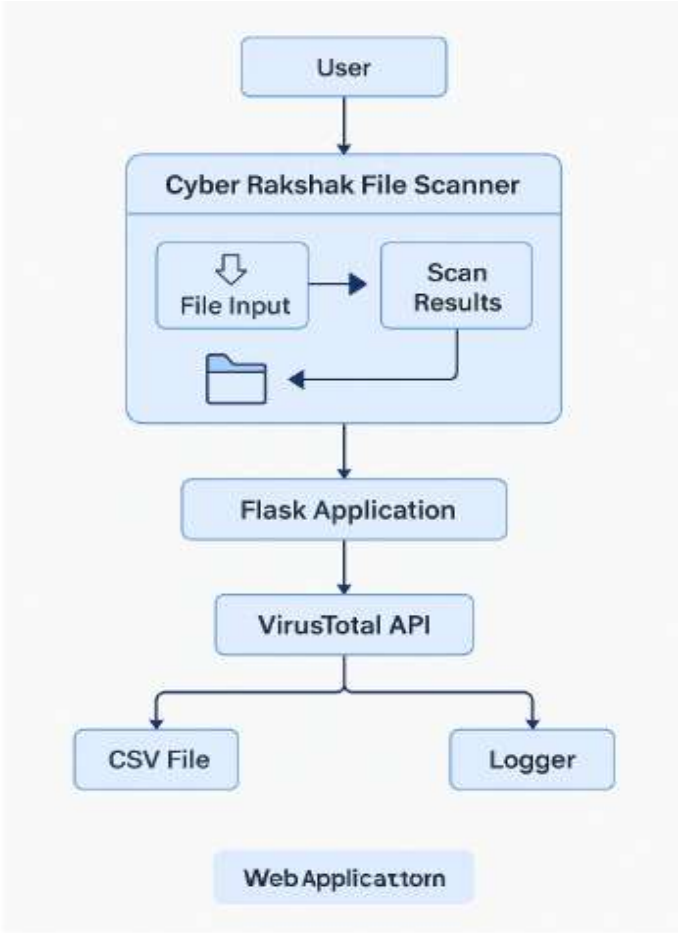


Fig 6.1 ARCHITECTURE DIAGRAM

5.2.3 Flask Backend (Python Server):

The Flask web server is the main component of the program. It acts as a mediator between the frontend and Virus Total, the malware analysis engine. The acquisition of apk files uploaded through the user interface is one of the important tasks.

Using the Virus Total API to securely interact, interpreting the received JSON response, and returning the scan summary to the frontend after confirming and temporarily storing the file.

In addition to managing server-side logic, this component can be improved with error handling, rate restriction, and authentication.

5.2.4 Virus Total API (Malware Detection Engine):

The Virus Total API is a cloud-based virus detection service provided by a third party. It can do the following: >accept the backend's apk file; >run the file through over 70 antivirus and behavioural analysis engines; >identify

odd patterns, anomalies, and known malware signatures; and >provide a comprehensive scan report that includes the names of the threats, their severity levels, and the engines that raised the file's flag.

This component enables malware detection without requiring the system to launch an antivirus engine from scratch.

5.2.5 Response Handling and Visualization (Backend + Frontend):

After Virus Total returns the scan results, the backend parses the raw data and extracts important metrics, such as the number of engines that reported the file and the name of the common malware family.The information is then shown to the user through the frontend in an easy-to-read dashboard-style report.
These include:
>the total number of detections
>the output of each antivirus engine
>the type of threat or malware family and
>file details (size, creation date, hash, etc).

3. CONCLUSIONS

In today's digital environment, malicious mobile apps are posing a growing risk to cybersecurity and consumer privacy. The Cyber Rakshak project was developed as a direct response to this problem; it is a straightforward, dependable, and user-friendly platform that assists in identifying malware in Android program files (.apk).

Many consumers download apps without fully knowing their behaviour or risks due to the open ecosystem of Android and the increasing use of smartphones. In order to close this gap and assist stop any assaults from the outset, Cyber Rakshak enables users to examine app files before installing them.

The web-based platform makes use of the VirusTotal API, which imports potent threat intelligence from a variety of security products and antivirus engines. This makes it possible for Cyber Rakshak to examine program files fast and identify any questionable or harmful trends. A smooth user experience and file uploads are guaranteed by the Flask backend and the HTML, CSS, and JavaScript frontend.

Cyber Rakshak is an excellent illustration of how current technologies and APIs may be used into a full-stack application to provide practical cybersecurity solutions from a technological standpoint. The architecture has been thoughtfully created to manage files securely, communicate with third-party APIs effectively, and provide real-time analysis. File type checks, rate restriction, and automatic cleanup are examples of extra security measures that assist make the system more sustainable.

The user-focused design of the project is one of its best qualities. It makes malware detection easier for non-technical consumers to understand and streamlines the complicated process. It not only scans but also informs users of any dangers related to the programs they plan to install.

Looking ahead, Cyber Rakshak establishes the foundation for next improvements such as dashboards for scan histories, AI-based local malware detection, user login systems, and support for additional file formats. Along with opening the door for more intelligent and scalable malware defence systems, it also emphasises the fascinating possibilities of fusing cybersecurity with AI and cloud computing.
Cyber Rakshak, to put it briefly, is a proactive solution that promotes safer digital practices and lowers mobile security threats. It provides Android users with a crucial layer of security and demonstrates how creative research endeavours can be turned into workable solutions for improved digital wellbeing and online safety.

REFERENCES

[1]
Author(s): Neha Mohan Kumar, Fahmida Tasnim Lisa,Sheikh Rabiul Islam
Title: Prompt Chaining-Assisted Malware Detection: A Hybrid Approach Utilizing Fine-Tuned LLMs and Domain Knowledge-Enriched Cybersecurity Knowledge Graphs
Conference: IEEE International Conference
Year: 2024
Link: https://www.computer.org/csdl/proceedings-article/bigdata/2024/10825154/23yldRvUVUY

---

[2]
Author(s): Liyakathunisa, Zoya Riyaz Syeda, Riyaz Sohale Syed,
Title: Cyber Attack Detection for Internet of Health Things through Federated Deep Learning Technique
Conference: Annual Computer Security Applications Conference Workshops (ACSACW)
Year: 2024

Link: https://www.computer.org/csdl/proceedings-article/acsac-workshops/2024/328100a194/25bv0ZqdUBi

---

[3]
Author(s): Li Jing
Title: Mobile Internet Malicious Application Detection Method Based on Support Vector Machine
Conference: International Conference on Smart Grid and Electrical Automation (ICSGEA)
Year: 2017
Link: https://www.computer.org/csdl/proceedings-article/icsgea/2017/2813a260/12OmNARiLZC

---

[4]
Author(s): Long Chen; Chunhe Xia; Shengwei Lei; Tianbo Wang
Title: Detection, Traceability, and Propagation of Mobile Malware Threats
Journal: IEEE Access
Year: 2021
Link: https://ieeexplore.ieee.org/document/9316662

---

[5]
Author(s): Sixian Sun; Xiao Fu; Hao Ruan; Xiaojiang Du; Bin Luo; Mohsen Guizani
Title: Real-Time Behavior Analysis and Identification for Android Application
Journal: IEEE Xplore
Year: 2018
Link: https://ieeexplore.ieee.org/document/8408465

---

[6]
Author(s): Yichen Hou; Sahil Garg; Lin Hui; Dushantha Nalin K. Jayakody; Rui Jin; M. Shamim Hossain
Title: A Data Security Enhanced Access Control Mechanism in Mobile Edge Computing
Journal: IEEE Access
Year: 2020
Link: https://ieeexplore.ieee.org/document/9146646

---

[7]
Authors: Zhenyuan Li, Jun Zeng, Yan Chen, Zhenkai Liang
Title: AttacKG: Constructing Technique Knowledge Graph from Cyber Threat Intelligence Reports
Journal: arXiv preprint
Year: 2021
Link: https://arxiv.org/abs/2111.07093arXiv+1arXiv+1ResearchGate+7arXiv+7Wikipedia+7

---

[8]
Authors: Jian Wang, Tiantian Zhu, Chunlin Xiong, Yan Chen
Title: MultiKG: Multi-Source Threat Intelligence Aggregation for High-Quality Knowledge Graph Representation of Attack Techniques
Journal: arXiv preprint
Year: 2024
Link: https://arxiv.org/abs/2411.08359arXiv

---

[9]
Authors: Vrinda Malhotra, Katerina Potika, Mark Stamp
Title: A Comparison of Graph Neural Networks for Malware Classification
Journal: arXiv preprint
Year: 2023
Link: https://arxiv.org/abs/2303.12812MDPI+5arXiv+5arXiv+5

---

[10]
Authors: Tristan Bilot, Nour El Madhoun, Khaldoun Al Agha, Anis Zouaoui
Title: A Survey on Malware Detection with Graph Representation Learning
Journal: arXiv preprint
Year: 2023
Link: https://arxiv.org/abs/2303.16004arXiv

---

[11]
Authors: Ali Dehghantanha, et al.
Title: Machine Learning Aided Android Malware Classification
Journal: Computers & Electrical Engineering
Year: 2018
Link: https://en.wikipedia.org/wiki/Ali_DehghantanhaWikipedia+1arXiv+1
