CAPTURE: Customizable Android and IoT Protection with Threat Understanding, Response, and Evaluation

**Why?**

1. growing sophistication of malware.

Malware and adware targeting Android devices have grown in complexity, making traditional detection methods less effective. Attackers frequently adapt to bypass static detection mechanisms, requiring real-time and adaptive solutions.

1. Threats to IoT Devices:

The rapid proliferation of IoT devices has led to an increase in malware and adware targeting their software, exploiting weak security protocols and outdated systems. These attacks compromise personal data, disrupt services, and even threaten critical infrastructure. Addressing IoT threats alongside Android device security is crucial for safeguarding connected environments.

1. The goal is to provide a **state-of-the-art, adaptable, and user-centric** malware detection solution that safeguards Android and IoT ecosystems.

**What’s new?**

1. Hybrid Detection Approach

By combining static analysis (e.g., permissions, API calls) and dynamic analysis (e.g., runtime behavior, network flows), the proposed system offers comprehensive coverage, improving detection rates and reducing false positives.

1. Customizable Cataloging Models

Recognizing that users have different preferences and requirements, the system will allow users to select from multiple cataloging models to fit their specific needs. This customization enhances the system's usability and effectiveness for various contexts.

1. Real-Time Learning Mechanism: The app will employ incremental learning techniques to adapt dynamically to emerging threats, ensuring continuous protection for users.
2. Provides users with detailed insights into detected threats and security recommendations.

**Cataloging Models Users Can Choose From:**

1. **Static Feature-Based Model:** Focuses on permissions, API calls, and manifest files for quick identification of malware using predefined rules and models.

Best for users preferring straightforward, fast scanning.

1. **Dynamic Behavior-Based Model:** Analyzes runtime behavior, network flows, and app interactions to detect anomalies and advanced threats. Ideal for users targeting deeper and more comprehensive analysis.
2. **Hybrid Model (Static + Dynamic):** Combines both static and dynamic analysis for a balanced and highly accurate detection approach. Recommended for users who want maximum protection.
3. **IoT-Specific Security Model:** Tailored for IoT devices, focusing on firmware-level analysis, network traffic patterns, and usage anomalies. Perfect for users with interconnected devices.

**Literature:**

1. DroidMat: Android Malware Detection through Manifest and API Calls Tracing

<https://ieeexplore.ieee.org/document/6298136>

1. Android malware detection method based on highly distinguishable static features and DenseNet

<https://pmc.ncbi.nlm.nih.gov/articles/PMC9683612/>

1. HyDroid: android malware detection using network flow combined with permissions and intent filter

<https://www.inderscience.com/offer.php?id=131799>

1. Android Malware Detection Using Deep Learning

<https://www.sciencedirect.com/science/article/pii/S1877050921007481>

1. NATICUSdroid: A malware detection framework for Android using native and custom

<https://www.semanticscholar.org/paper/Poster%3A-NATICUSdroid%3A-A-malware-detection-framework-Mathur-Podila/9ad96aa48a93baf10638f0e3f93c471aa0030797>

1. Significant Permission Identification for Machine-Learning-Based Android Malware Detection

<https://ieeexplore.ieee.org/document/8255798>

1. A Machine Learning Approach to Android Malware Detection

<https://ieeexplore.ieee.org/document/6298824>

1. MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention

<https://ieeexplore.ieee.org/document/7422770>

1. Kernel-based Behavior Analysis for Android Malware Detection

<https://ieeexplore.ieee.org/document/6128277>

1. Deep Android Malware Detection

<https://dl.acm.org/doi/10.1145/3029806.3029823>

1. MalDozer: Automatic framework for android malware detection using deep learning

<https://www.sciencedirect.com/science/article/pii/S1742287618300392?via%3Dihub>

1. AMDDLmodel: Android smartphones malware detection using deep learning model

<https://pmc.ncbi.nlm.nih.gov/articles/PMC10798489/>

1. M0Droid: An Android Behavioral-Based Malware Detection Model

<https://www.researchgate.net/publication/282197701_M0Droid_An_Android_Behavioral-Based_Malware_Detection_Model>

1. A Proposed Artificial Intelligence Model for Android-Malware Detection

<https://www.mdpi.com/2227-9709/10/3/67>

**Datasets:**

1. <https://www.unb.ca/cic/datasets/andmal2017.html>

<https://www.unb.ca/cic/datasets/index.html>

1. <https://www.kaggle.com/datasets/malikbaqi12/cic-invesandmal2019-dataset/data>
2. <https://archive.ics.uci.edu/dataset/722/naticusdroid+android+permissions+dataset>
3. Drebin Dataset <https://drebin.mlsec.org/>
4. <https://www.kaggle.com/datasets/subhajournal/android-malware-detection/data>
5. AndroZoo: Collecting Millions of Android Apps for the Research Community

<https://androzoo.uni.lu/>

**System Design**

**Data Collection and Preprocessing**

**Static Analysis:** Extract permissions, API calls, and manifest files from APKs using tools like APKTool and Androguard.

**Dynamic Analysis:** Monitor runtime behavior using sandbox environments using tools like DroidBox and Cuckoo Sandbox.

**IoT-Specific Features:** Gather firmware-level data, network logs, and device-specific activity using IoT monitoring frameworks.

**Feature Engineering:** Combine static and dynamic features for a comprehensive dataset. Use methods like Recursive Feature Elimination (RFE) and feature embedding via DenseNet for identifying critical attributes.

**Model Training**

Model Selection: Experiment & Combine classical models like Random Forest and SVM with advanced deep learning architectures such as CNNs (for image-based APK analysis) and DenseNet (for static features).

**Real-Time Adaptability**

Incremental Learning: Use online learning algorithms to update the model with new data.

Feedback Loop: Incorporate user feedback to refine predictions.

**Application**

**Backend**:

Use cloud-based services for model inference and updates. (Goole Cloud)

Databases (e.g., Firebase) for storing malware signatures and detection logs.

**Frontend:**

Develop a user-friendly Android app for malware detection, reporting and cataloguing.

Options for real-time scanning of APKs, files and IoT devices.

**Threat Reporting and Visualization**

* Generate actionable insights, including threat summaries, mitigation steps, and real-time alerts for users.
* Provide visualizations such as bar charts, heatmaps, and flow diagrams for threat activity.

**Security Measures**

Implement encryption for data transmission.

Use secure authentication mechanisms for app access.

**Deployment:**

Use containerized environments (e.g., Docker) for scalable and reliable app deployment.