Yusuf Emre Konuk
Bernouilli Institute - Information Systems Group
Groningen, NL
July, 2024

Literature Review on Android In-App Payment Security, Mobile Payment SDK Vulnerabilities, and Related Works

Topics

Topic 1: Detecting third-party libraries used in Android apps, including payment libraries.

Topic 2: Scanning third-party libraries used in Android apps, including payment libraries, for security vulnerabilities.

Topic 3: Detecting and scanning third-party libraries used in Android apps, including payment libraries, for security vulnerabilities.

Questions

1. Under which topic do they fall?
2. Which types of third-party libraries do they cover (e.g., generic or focused on a particular type such as payment libraries)?
3. The technique they use (manual, automatic, machine learning).
4. Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset).
5. For scanning-type works, the number and examples of vulnerabilities they consider.
6. Availability of source code and/or datasets from the paper.

Detecting Third-Party Libraries in Android Applications with High Precision and Recall

Keywords-- Library Detection, Code Similarity, Obfuscation Resilience

- Under which topic do they fall?
    - The paper falls under the topic of **Library Detection and Code Similarity in Android Applications**. It specifically focuses on techniques to detect third-party libraries within Android applications with high precision and recall, addressing issues related to code obfuscation and customization.

- Which types of third-party libraries do they cover (e.g., generic or focused on a particular type such as payment libraries)?
    - The study covers **generic third-party libraries** used in Android applications. The libraries are collected from sources like Maven repositories, F-Droid, and popular Android developer community resources. These libraries span various functionalities and are not limited to any specific type such as payment libraries.

- The technique they use (manual, automatic, machine learning).
    - The technique used in the paper is **automatic**. LIBPECKER, the proposed tool, employs an automatic **signature matching technique** based on class dependencies to detect third-party libraries within Android applications.

- Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset).
    - No machine learning or training on a labeled dataset is required for LIBPECKER. The approach relies on generating class signatures and performing fuzzy class matching based on class dependencies, with adaptive class similarity thresholds and weighted class similarity scores to handle code customization and obfuscation.

- For scanning-type works, the number and examples of vulnerabilities they consider.
    - The paper does not specifically focus on scanning for vulnerabilities but highlights the security and privacy risks associated with third-party libraries. Examples given include the **Taomike SDK**, **Baidu SDK**, **Facebook SDK**, and **Dropbox SDK**, which have been reported with severe vulnerabilities affecting a significant number of apps.

- Availability of source code and/or datasets from the paper.
    - The document does not provide explicit information on the availability of the source code or datasets used in the research. There is no indication that the source code for LIBPECKER or the collected dataset of libraries and applications are publicly available within the text provided.

Reliable Third-Party Library Detection in Android and its Security Applications

Keywords-- Security and privacy, Network security, Mobile and wireless security, Systems security, Operating systems security, Mobile platform security

- **Under which topic do they fall?**
  - The topic falls under "Security and Privacy in Mobile Applications," specifically focusing on "Third-Party Library Detection and Analysis in Android Apps."

- **Which types of third-party libraries do they cover (e.g., generic or focused on a particular type such as payment libraries)?**
  - They cover a broad range of third-party libraries, including but not limited to advertising, cloud services, and other utility libraries. Specific examples mentioned include advertising libs and those involved in cryptographic API misuse.

- **The technique they use (manual, automatic, machine learning).**
  - The technique used is automatic. It involves generating profiles from original library SDKs and using a profile matching algorithm to detect libraries within apps. This method is resilient to common code obfuscations like identifier renaming and API hiding.

- **Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset).**
  - No specific pre-settings for machine learning are required. However, they do need to create a comprehensive database of library profiles from original library SDKs, which involves some initial setup.

- **For scanning-type works, the number and examples of vulnerabilities they consider.**
  - The paper identifies several types of vulnerabilities, particularly focusing on:
    - Misuse of cryptographic APIs in advertising libs, which exposes apps to cryptanalytic attacks.
    - Long-known security vulnerabilities in popular libraries still present in top apps.
    - Examples include vulnerabilities in the Facebook and Dropbox SDKs that led to issues like data leakage, code injection attacks, and account hijacking.

- **Availability of source code and/or datasets from the paper.**
  - The paper mentions that the library detection tool, referred to as LibScout, is available at the following URL: LibScout. They also created a large third-party library database including 164 distinct libraries with 2,065 versions, and they collected the version history of 3,590 top apps on Play, totaling 96,995 distinct packages(derr-16-ccs).

A SURVEY OF THIRD-PARTY LIBRARY SECURITY RESEARCH IN APPLICATION SOFTWARE

Keywords-- Application software, Third party libraries, Third party library security

- **Under which topic do they fall?**
    - The paper falls under the topic of **Third-Party Library Security in Application Software**. It provides a comprehensive survey of the security research related to third-party libraries in software applications, addressing their usage, risks, detection, and defense mechanisms.

- **Which types of third-party libraries do they cover (e.g., generic or focused on a particular type such as payment libraries)?**
    - The study covers **generic third-party libraries** used across various types of application software. These libraries are not limited to any specific type and include a wide range of functionalities such as data processing, image processing, network requests, machine learning, and database connections.

- **The technique they use (manual, automatic, machine learning).**
    - The techniques discussed in the paper include **automatic detection** methods and **unsupervised learning** approaches. These methods involve both signature-based detection using known library databases and machine learning algorithms for clustering and classifying code characteristics.

- **Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset).**
    - For the machine learning techniques mentioned, **training on a labeled dataset** is required to develop accurate detection models. However, the paper also discusses unsupervised learning techniques that do not require prior knowledge of the libraries but instead analyze code characteristics for detection.

- **For scanning-type works, the number and examples of vulnerabilities they consider.**
    - The paper mentions several real-world vulnerabilities related to third-party libraries, such as the **Heartbleed vulnerability** in the OpenSSL library and the **event-stream incident** involving malicious code injection. It does not specify a fixed number of vulnerabilities but emphasizes the significant impact these vulnerabilities can have on software security.

- **Availability of source code and/or datasets from the paper.**
    - The document does not provide explicit information on the availability of source code or datasets used in the research. It focuses more on summarizing existing research and methodologies rather than offering specific tools or datasets for public access.

ATVHUNTER: Reliable Version Detection of Third-Party Libraries for Vulnerability Identification in Android Applications

Keywords- Security and privacy, Systems security, Operating systems security, Mobile platform security, Software and its engineering, Software creation and management

Under which topic do they fall?

- The paper falls under the topic of third-party library detection and vulnerability identification in Android applications.

Which types of third-party libraries do they cover?

- The paper covers a wide range of third-party libraries used in Android applications, including development aids, UI plugins, and advertisement libraries. It mentions constructing a comprehensive TPL database with 189,545 unique TPLs across 3,006,676 versions.

The technique they use (manual, automatic, machine learning).

- The technique used is automatic. ATVHUNTER employs a two-phase detection approach:
  - Coarse-grained feature extraction using Control Flow Graphs (CFG) to represent the TPLs.
  - Fine-grained feature extraction by analyzing the opcode in each basic block of the CFG for precise version identification.

Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset).

- Pre-settings for ATVHUNTER include constructing a comprehensive TPL database and a vulnerable TPL database. The system does not rely on machine learning models but uses pre-defined databases and algorithmic detection methods.

For scanning-type works, the number and examples of vulnerabilities they consider.

- ATVHUNTER identifies vulnerable TPL versions and provides detailed vulnerability information. The vulnerable TPL database constructed contains 1,180 CVEs and 224 security bugs.

Availability of source code and/or datasets from the paper.

- The document does not specify the availability of source code or datasets directly within the provided text. It focuses on the methodology and results of using ATVHUNTER but does not mention open-source availability.

Third-Party Library Security Management

This study was created to provide information rather than to produce tools, but I wanted to add it because I liked it and found it successful.

- **Under which topic do they fall?**
    - The document falls under the topic of **Third-Party Library Security Management**. It covers key concepts, challenges, and practices related to the security of third-party libraries used in software development .

- **Which types of third-party libraries do they cover (e.g., generic or focused on a particular type such as payment libraries)?**
    - The document covers **generic third-party libraries**. It mentions various tools and practices applicable to different programming environments, including Java, .NET, JavaScript, Python, and Go libraries .

- **The technique they use (manual, automatic, machine learning).**
    - The techniques discussed in the document include **automatic** scanning and monitoring of third-party libraries. Tools like OWASP Dependency-Check, Snyk, and others are used to automatically identify and manage vulnerabilities in third-party components .

- **Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset).**
    - For the tools mentioned, typical pre-settings include configuring the plugin, setting up proxy configurations, and performing initial scans. Continuous integration (CI) setups and centralized databases for maintaining local copies of the National Vulnerability Database (NVD) are also recommended .

- **For scanning-type works, the number and examples of vulnerabilities they consider.**
    - The document discusses numerous vulnerabilities. Examples include CVE-2017-5638 (Apache Struts 2 RCE vulnerability) and CVE-2017-8046 (Spring Framework vulnerability). The approach generally involves scanning for known vulnerabilities listed in databases like NVD .

- **Availability of source code and/or datasets from the paper.**
    - The document provides links to various tools but does not mention the availability of source code or datasets directly within the slides. However, tools like OWASP Dependency-Check and OWASP Dependency-Track are open-source and their code is available on GitHub .

Sec-Lib: Protecting Scholarly Digital Libraries From Infected Papers Using Active Machine Learning Framework

Keywords-Libraries, Portable document format, Malware, Machine Learning, Google, Computer Crime

- Under which topic do they fall?
    - The topic of the paper is about protecting scholarly digital libraries from infected papers using an active machine learning framework.

- Which types of third-party libraries do they cover (e.g., generic or focused on a particular type such as payment libraries)?
    - The paper discusses third-party scholarly digital libraries, specifically focusing on academic digital libraries such as Google Scholar, CiteSeerX, ResearchGate, and Academia.edu.

- The technique they use (manual, automatic, machine learning).
    - The technique used is an automatic machine learning approach. The framework utilizes active learning (AL) methods to detect malicious PDF documents by analyzing their structural paths.

- Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset).
    - The pre-settings include training the model on a labeled dataset with a combination of malicious and benign files. The training set is designed to reflect the reality of having a low percentage of malicious files.

- For scanning-type works, the number and examples of vulnerabilities they consider.
    - The paper considers multiple vulnerabilities, including:
        - Infected files being indexed by Google Scholar from researcher homepages.
        - Malicious PDF documents containing JavaScript, embedded files, form submission, and URI attacks.
        - Techniques like reverse mimicry to evade detection.

- Availability of source code and/or datasets from the paper.
    - The paper does not explicitly mention the availability of source code or datasets for public use.

Up2Dep: Android Tool Support to Fix Insecure Code Dependencies

Keywords- Security and privacy, Systems security, Operating Systems Security

1. Under which topic do they fall?

The paper primarily falls under the topic of **software security and dependency management** in Android applications. It focuses on identifying and mitigating security vulnerabilities in third-party libraries used within Android apps.

2. Which types of third-party libraries do they cover?

The paper covers **a wide range of third-party libraries** included in Android applications. The dataset includes 1,878 libraries, with examples such as the Facebook Login Android SDK and its transitive dependencies. The libraries are not limited to any specific type but encompass various functionalities used in Android development.

3. The technique they use

The techniques used include:

- **Manual analysis:** Examination of descriptive files like `pom.xml` to discover transitive dependencies.
- **Automatic analysis:** Using tools like LibScout for API compatibility checks and CogniCrypt for discovering insecure uses of cryptographic APIs.
- **Machine learning:** Not explicitly mentioned, but the use of automated tools implies algorithmic methods to identify vulnerabilities and suggest fixes.

4. Any pre-settings required

For machine learning or automated analysis:

- **LibScout:** Requires a pre-built library database for API compatibility checks.
- **CogniCrypt:** Utilizes pre-defined rules in the CRYSL language to analyze Java applications for cryptographic API misuse.

5. For scanning-type works, the number and examples of vulnerabilities they consider

The paper discusses scanning for vulnerabilities such as:

- **Known security vulnerabilities** in outdated library versions.
- **Insecure uses of cryptographic APIs**, identified through CogniCrypt.
- **Examples of vulnerabilities include**:
  - Misuse of `MessageDigest`, `Cipher`, `SSLContext`, etc.
  - Use of HTTP instead of HTTPS for communication.

6. Availability of source code and/or datasets from the paper

The source code and datasets are available:

- **CogniCrypt rules** are accessible on GitHub.
- The paper extends the LibScout database, which includes a list of vulnerabilities and their versions.

LIBRA Library Identification in Obfuscated Android Apps

Keywords: Library Identification · Obfuscation · SBOM · Android

1. **Under which topic do they fall?**
   - The paper falls under the topic of **Library Identification in Obfuscated Android Apps**. It specifically focuses on identifying third-party libraries used within Android apps, particularly when these apps are obfuscated to protect their content from reverse engineering.
2. **Which types of third-party libraries do they cover?**
   - The study covers **generic third-party libraries** used in Android apps. These libraries can include any functionality such as advertisements, maps, and social networks, which are commonly integrated into Android applications.
3. **The technique they use (manual, automatic, machine learning):**
   - The technique used by the paper is **automatic**. Libra, the proposed method, employs **piecewise fuzzy hashing**, **method encoding**, and a **two-component weighted similarity computation** to automatically identify libraries in obfuscated apps. This approach is not based on machine learning but rather on heuristic and algorithmic methods for robust library detection.
4. **Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset):**
   - The paper does not require machine learning training on a labeled dataset. Instead, it requires an **offline learning phase** where the system builds a database of library signatures by processing library artifacts (e.g., JAR or AAR files). This database is then used in the detection phase.
5. **For scanning-type works, the number and examples of vulnerabilities they consider:**
   - The paper does not explicitly mention the number or specific examples of vulnerabilities considered. However, it does highlight that identifying libraries is crucial for understanding **potential security risks**, especially given that vulnerable libraries can impact not just the library but also the entire app and its dependencies.
6. **Availability of source code and/or datasets from the paper:**
   - The document does not provide direct information about the availability of the source code or datasets used in the research. There is no mention of the paper making its implementation or datasets publicly accessible.

Large-Scale Third-Party Library Detection in Android Markets

Keywords—Android, third-party library, software mining, code similarity detection

- **Under which topic do they fall?**
  - The paper falls under topics such as **Android security**, **third-party library detection**, **software engineering**, and **code similarity detection**.
- **Which types of third-party libraries do they cover?**
  - The study covers third-party libraries that are generally used in Android applications, such as libraries for **advertisement**, **navigation**, **social networking services**, and more. The libraries are not focused on a specific type but rather encompass a wide variety of functionalities commonly integrated into Android apps.
- **The technique they use (manual, automatic, machine learning)?**
  - The technique used in this study is primarily **automatic**. The paper introduces a method that automatically detects and classifies third-party libraries using a feature hashing strategy, which is resilient to name-based obfuscation and package structure diversity.
- **Any pre-settings required?**
  - The primary pre-settings mentioned in the paper include **decompiling the Android apps** to generate an intermediate representation (IR) of the Dalvik bytecode, known as smali. There is no mention of specific training on labeled datasets since the method is not based on machine learning but rather on feature hashing and similarity detection.
- **For scanning-type works, the number and examples of vulnerabilities they consider.**
  - The paper mentions that their approach was able to identify **10,801 vulnerable variants of third-party libraries**across over a million Android apps. Specific examples of vulnerabilities are not detailed, but the paper focuses on the identification of vulnerabilities in third-party libraries that could affect a large number of apps and users.
- **Availability of source code and/or datasets from the paper.**
  - Yes, the source code for the tool developed, **LibD**, is made available by the authors. It can be accessed via GitHub at **https://github.com/IIE-LibD/libd.git**:contentReference[oaicite:0]{index=0}.

Lib2Desc: automatic generation of security-centric Android app descriptions using third-party libraries

Keywords-- Android security • Description-to-permission fidelity • Third-party libraries • NLP • NLG

- **Under which topic do they fall?**
    - The work falls under the topic of **Android security**, specifically focusing on the automatic generation of security-centric Android app descriptions using third-party libraries (TPLs).
- **Which types of third-party libraries do they cover?**
    - The study covers third-party libraries that are generally used in Android applications, with a specific focus on those that involve **dangerous permissions** or **data leakage**. It does not focus on a particular type like payment libraries but instead targets libraries that raise security concerns.
- **The technique they use:**
    - The technique used is **automatic**. The study utilizes **neural models** for tasks such as source code summarization, classification of libraries, and generating app descriptions. Specifically, transformer-based models are used for these purposes.
- **Any pre-settings required:**
    - Yes, for machine learning models like the ones used here, **training the model on a labeled dataset** is required. The document mentions fine-tuning models with datasets that contain Android API code-explanation pairs or training models from scratch using the Android API dataset.
- **For scanning-type works, the number and examples of vulnerabilities they consider:**
    - The study focuses on identifying vulnerabilities related to **dangerous permissions** and **data leakage**. Specific examples include:
        - Libraries that send sensitive information without proper documentation.
        - Libraries that request and use dangerous permissions (like ad libraries that access private data).
    - The document mentions that about **3% of the 2247 libraries analyzed** make use of dangerous permissions.
- **Availability of source code and/or datasets from the paper:**
    - Yes, the datasets generated and analyzed during the study are available in the **Lib2Desc repository** hosted by the WISE Lab at Hacettepe University. The document provides a link to the dataset: Lib2Desc repository.

LibCapsule: Complete Confinement of Third- Party Libraries in Android Applications

Keywords—Third-party library, Java bytecode, dynamically loaded code, native code

- **Under which topic do they fall?**
  - The paper falls under the topic of **Android application security**, with a focus on the confinement and isolation of third-party libraries to enhance security.
- **Which types of third-party libraries do they cover?**
  - The paper covers **generic third-party libraries** within Android applications, focusing on confining their behavior regardless of their specific functionality.
- **The technique they use (manual, automatic, machine learning).**
  - The technique used is **automatic**. The system, LibCapsule, automatically confines third-party libraries using sandboxing and access control techniques.
- **Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset).**
  - **No specific pre-settings** are required for using LibCapsule, as it operates at the application level and automatically applies the confinement rules to third-party libraries.
- **For scanning-type works, the number and examples of vulnerabilities they consider.**
  - The paper is not focused on scanning for vulnerabilities but on **preventing potential vulnerabilities** by confining the behavior of third-party libraries.
- **Availability of source code and/or datasets from the paper.**
  - The paper does not explicitly mention the **availability of the source code or datasets** related to LibCapsule.

Research on Third-Party Libraries in Android Apps: A Taxonomy and Systematic Literature Review

Keywords—Third-party library, android, literature review, applications

1. Under which topic do they fall?

The paper falls under the topic of **Third-Party Libraries in Android Applications** with a focus on security, privacy, and performance issues.

2. Which types of third-party libraries do they cover (e.g., generic or focused on a particular type such as payment libraries)?

The paper covers **generic third-party libraries** used in Android apps. It provides a taxonomy that includes various types of libraries, such as advertising, analytics, and social media libraries, but it is not limited to a particular type like payment libraries.

3. The technique they use (manual, automatic, machine learning).

The paper utilizes **systematic literature review (SLR)** as the primary technique. This involves manual analysis of existing research and studies, categorizing and synthesizing the findings into a comprehensive taxonomy.

4. Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset).

There are **no pre-settings or machine learning models** involved in this work, as it is based on a systematic review and manual analysis of the literature.

5. For scanning-type works, the number and examples of vulnerabilities they consider.

The paper does not focus on active scanning for vulnerabilities. However, it reviews studies that have analyzed **various vulnerabilities** associated with third-party libraries in Android apps, including privacy leaks, code injection, and security misconfigurations.

6. Availability of source code and/or datasets from the paper.

The paper does not mention the availability of any **source code or datasets** as it is a review-based study rather than an empirical one involving new data collection or software development.

Too Quiet in the Library: A Study of Native Third-Party Libraries in Android

Keywords-- Security and privacy, systems security, Operating systems security, Operating systems security, Software creation and management

- **Under which topic do they fall?**
    - The paper falls under the topic of **security in mobile applications**, specifically focusing on **the analysis of third-party native libraries in Android apps** and their associated security vulnerabilities.

- **Which types of third-party libraries do they cover (e.g., generic or focused on a particular type such as payment libraries)?**
    - The study covers **generic third-party native libraries** used in Android apps. These libraries are not focused on a particular type like payment libraries but rather include a wide range of libraries used for various functionalities, such as **FFmpeg, GIFLIB, OpenSSL, WebP, SQLite, and OpenCV**.

- **The technique they use (manual, automatic, machine learning).**
    - The technique used is **automatic**. The paper introduces a novel approach called **LibRARIAN** (LibRAry veRsion IdentificAtioN), which leverages a binary similarity metric called **bin2sim** to automatically identify native libraries and their versions in Android apps.

- **Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset).**
    - The approach relies on a **repository of known libraries and versions** against which unknown libraries are matched. This means that a dataset of native binaries with known library names and versions is required. However, there is no mention of machine learning or training models on labeled datasets; instead, the method relies on feature extraction and similarity scoring.

- **For scanning-type works, the number and examples of vulnerabilities they consider.**
    - The study identified **1,781 vulnerable versions** with known CVEs across 80 apps between September 2013 and April 2019. It also provides examples such as **FFmpeg, GIFLIB, OpenSSL, WebP, SQLite, and OpenCV**being the vulnerable libraries.

- **Availability of source code and/or datasets from the paper.**
    - The paper does not explicitly state the availability of source code or datasets. However, it mentions the creation of a **repository containing the 600 most popular free apps from Google Play, totaling 12,646 versions and 89,525 native libraries**. There is no direct indication that this repository or the source code for LibRARIAN is publicly available.

Diversified Third-Party Library Prediction for Mobile App Development

Keywords—Third-party library, prediction, mobile app development, matrix factorization, diversity, accuracy bias.

1. Under which topic do they fall?

- The paper "Diversified Third-Party Library Prediction for Mobile App Development" falls under the topic of **Third-Party Library Prediction and Selection** for mobile app development. It focuses on predicting and recommending diverse third-party libraries that can enhance mobile applications.

2. Which types of third-party libraries do they cover?

- The paper discusses libraries that are **generic and widely applicable** in mobile app development, covering various functionalities such as enhancing user interfaces, adding social features, and improving app performance. These libraries are not limited to a specific domain but rather span across different functionalities used in Android app development.

3. The technique they use (manual, automatic, machine learning).

- The technique used in this paper is **automatic and machine learning-based**. Specifically, the paper employs **Matrix Factorization (MF)** approaches, which are commonly used in collaborative filtering for prediction tasks. The approach is designed to predict useful third-party libraries for mobile apps by leveraging patterns in library usage.

4. Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset).

- The approach involves training the machine learning model on a dataset of **31,432 Android apps** from Google Play. The model uses patterns from these apps to make predictions about which libraries are likely to be useful for other apps. The pre-settings include the collection and preprocessing of this large dataset, as well as the configuration of the Matrix Factorization model.

5. For scanning-type works, the number and examples of vulnerabilities they consider.

- This paper does not focus on scanning for vulnerabilities. Instead, it emphasizes predicting useful libraries rather than detecting vulnerabilities. As a result, there are no specific vulnerabilities discussed or considered in the context of this work.

6. Availability of source code and/or datasets from the paper.

- The document does not explicitly mention the availability of source code or datasets. Typically, for papers published in IEEE Transactions, this information might be found in supplementary materials or through direct contact with the authors, but it is not detailed in the extracted content.

Understanding and Conquering the Difficulties in Identifying Third-Party Libraries From Millions of Android Apps

Keywords—Third-party libraries, library identification, refinement

- **Under which topic do they fall?**
  - The paper falls under the topic of **Software Analysis**, specifically focusing on the **identification of third-party libraries in Android applications**.
- **Which types of third-party libraries do they cover?**
  - The paper covers **generic third-party libraries** used in Android applications, without focusing on any particular type like payment libraries. The study aims at identifying a wide range of third-party libraries included in various Android apps.
- **The technique they use (manual, automatic, machine learning).**
  - The technique used is **automatic**. The paper introduces a new method named **LibHawkeye** that uses **clustering-based techniques** to identify third-party libraries in Android applications.
- **Any pre-settings required?**
  - The technique requires some **initial training** to identify unique library features, but it does not specify the need for labeled datasets typically required for machine learning models. Instead, it relies on the **clustering of features**extracted from Android applications.
- **For scanning-type works, the number and examples of vulnerabilities they consider.**
  - The paper does not focus on identifying specific vulnerabilities but rather on the accurate identification of third-party libraries. Therefore, there are **no specific vulnerabilities** considered in this study.
- **Availability of source code and/or datasets from the paper.**
  - The paper **does not explicitly mention** the availability of source code or datasets used in the study, meaning that these resources may not be publicly available.

Characterizing usages, updates and risks of third-party libraries in Java projects

Keywords: Library usages · Library updates · Library risks · Security bugs

- **Under which topic do they fall?**
  - The study falls under the topic of characterizing third-party library usage, updates, and associated risks in software projects.
- **Which types of third-party libraries do they cover?**
  - The study covers generic third-party libraries used in software development, without focusing on a particular type like payment libraries.
- **The technique they use (manual, automatic, machine learning).**
  - The study uses automatic techniques to analyze the usage, updates, and risks associated with third-party libraries.
- **Any pre-settings required?**
  - No specific pre-settings are mentioned, as the study does not involve machine learning techniques that require training on a labeled dataset.
- **For scanning-type works, the number and examples of vulnerabilities they consider.**
  - The study does not specify the number of vulnerabilities; instead, it focuses on the risks associated with outdated or vulnerable third-party libraries in general.
- **Availability of source code and/or datasets from the paper.**
  - The study does not mention the availability of source code or datasets used in the analysis.

DPC:A Dynamic Permission Control Mechanism for Android Third-Party Libraries

Keywords: Third-party library, privacy, in-app advertisement, security

- **Under which topic do they fall?**
  - The paper falls under the topic of **Android security** with a focus on **dynamic permission control** for third-party libraries.
- **Which types of third-party libraries do they cover?**
  - The study focuses on **Android third-party libraries** that require permissions to function, particularly those that may misuse dangerous permissions.
- **The technique they use?**
  - The technique used is **automatic**. The study proposes a dynamic mechanism for controlling permissions granted to third-party libraries.
- **Any pre-settings required?**
  - No specific pre-settings are required beyond integrating the proposed dynamic permission control mechanism into the Android framework.
- **For scanning-type works, the number and examples of vulnerabilities they consider?**
  - The paper does not provide a detailed list of vulnerabilities but focuses on **misuse of dangerous permissions** by third-party libraries as a general category of concern.
- **Availability of source code and/or datasets from the paper?**
  - **The paper does not mention** the availability of source code or datasets.

Dynamic Privacy Leakage Analysis of Android Third-party Libraries

Keywors: privacy leakage; third-party library; real-time; dynamic; fine granularity

Under which topic do they fall?

- **Topic:** This study falls under the topic of **privacy leakage and security analysis** in Android applications, with a specific focus on how third-party libraries may expose user data without their knowledge.

Which types of third-party libraries do they cover (e.g., generic or focused on a particular type such as payment libraries)?

- **Type:** The study covers **generic third-party libraries** that are commonly integrated into Android applications, rather than focusing on a particular type like payment or advertising libraries. The research aims to analyze the privacy risks associated with a broad spectrum of third-party libraries.

The technique they use (manual, automatic, machine learning).

- **Technique:** The study employs **automatic dynamic analysis** techniques. These techniques involve running the application and monitoring the behavior of third-party libraries in real-time to detect any potential privacy leaks.

Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset).

- **Pre-settings:** The study does not require any specific pre-settings like training a machine learning model. However, it likely requires a test environment where the Android applications can be executed and monitored for data leakage activities.

For scanning-type works, the number and examples of vulnerabilities they consider.

- **Vulnerabilities:** The study focuses on identifying **privacy leaks** as the main type of vulnerability. It examines how third-party libraries access and potentially misuse private user data, such as location information, contact lists, and other sensitive data types. Specific examples or a quantified number of vulnerabilities are not detailed, but the general threat of unauthorized data transmission by these libraries is the central concern.

Availability of source code and/or datasets from the paper.

- **Availability:** The paper does not mention the availability of source code or datasets. It focuses on the methodology and findings rather than providing resources for replication or further study.

iLibScope: Reliable Third-Party Library Detection for iOS Mobile Apps

Keywords: iOS, third-party library detection, security

- **Under which topic do they fall?**

The paper falls under **third-party library detection** in the context of **iOS mobile app security**. It focuses on detecting third-party libraries within apps and assessing their vulnerabilities.

- **Which types of third-party libraries do they cover (e.g., generic or focused on a particular type such as payment libraries)?**

The study covers **generic third-party libraries** used in iOS apps. It does not focus on a particular type, as it aims to provide a broad solution for identifying any third-party libraries and their versions across various domains.

- **The technique they use (manual, automatic, machine learning)?**

The paper proposes an **automatic** technique based on **profile-based similarity comparison**. It extracts class-level and method-level profiles from the libraries and apps, and then compares these profiles to identify matches and pinpoint library versions.

- **Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset)?**

There are no machine learning pre-settings, but the approach requires building a **library database** with profiles for various versions of libraries. The process involves compiling libraries into binaries and extracting their profiles.

- **For scanning-type works, the number and examples of vulnerabilities they consider.**

The paper mentions that iLibScope was applied to detect vulnerabilities in **405 instances of vulnerable libraries** across **4,249 apps**. Examples of vulnerable libraries include popular ones like **AFNetworking**, **SSZipArchive**, and **GCDWebServer**, which have had well-known vulnerabilities.

- **Availability of source code and/or datasets from the paper.**

The paper doesn't mention directly providing access to the source code or datasets, but it does describe the creation of a **library database** consisting of **5,768 library versions from 319 distinct libraries**. You might need to contact the authors for access to these resources.

LibDB: An Effective and Efficient Framework for Detecting Third-Party Libraries in Binaries

Keywords: Third-party libraries, Static binary analysis, Clone detection

- **Under which topic do they fall?**
    - The paper falls under **software engineering**, specifically focusing on **third-party library detection** in binary code. It deals with topics such as static binary analysis, clone detection, and library reuse.

- **Which types of third-party libraries do they cover (e.g., generic or focused on a particular type such as payment libraries)?**
    - The paper is **generic** in its approach, focusing on detecting any third-party libraries in binary form, without restriction to a particular type. The method aims to cover libraries that are compiled into the binary, including cases where they may be fused with project-specific code.

- **The technique they use (manual, automatic, machine learning)?**
    - The paper uses a **machine learning-based technique** combined with **static analysis**. Specifically, it uses function vector features embedded into low-dimensional representations via a neural network and applies a function call graph (FCG)-based comparison to improve detection accuracy.

- **Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset)?**
    - Yes, for the machine learning part, **pre-training** of the neural network on a labeled dataset of functions extracted from binary files is required. The paper also mentions constructing feature databases (basic and function vector databases) based on a collection of third-party libraries in binary form.

- **For scanning-type works, the number and examples of vulnerabilities they consider?**
    - The paper doesn't directly focus on scanning for specific **vulnerabilities**, but it discusses identifying the **versions**of third-party libraries, which is crucial for vulnerability detection. Knowing the version helps determine if a particular binary includes known vulnerabilities. Vulnerabilities such as the **Heartbleed bug** are mentioned as examples of issues stemming from third-party libraries.

- **Availability of source code and/or datasets from the paper?**
    - Yes, the source code and datasets used in this research are available at the following GitHub repository: https://github.com/DeepSoftwareAnalytics/LibDB.

LibDroid: Summarizing information flow of android native libraries via static analysis

Keywords: Android Native Library Mobile App Forensics Taint Analysis

- **Under which topic do they fall?** The work falls under **Android Native Library Security**, focusing on **Mobile App Forensics** and **Taint Analysis**.

- **Which types of third-party libraries do they cover?** The paper focuses on analyzing **Android native libraries**, which include both system and third-party libraries. The analysis applies to a wide range of libraries used in real-world Android applications, such as the **Agora Native SDK** used for voice and video communication, as well as other libraries collected from over **2,627 real-world apps**.

- **The technique they use (manual, automatic, machine learning).** The technique used is **automatic static analysis**. The tool, **LibDroid**, leverages **McSema** to lift Android native library code into LLVM IR and performs forward analysis of the control flow graph to summarize data flows and detect taint propagation. Symbolic execution techniques are also employed to analyze dependencies and propagate data flows.

- **Any pre-settings required.** The analysis requires **pre-processing** steps such as unpacking the Android `.apk` file, transforming native `.so` files to LLVM IR using **McSema**, and generating control flow graphs. There is no explicit mention of training models or other machine learning pre-settings, as this work relies on static analysis rather than machine learning.

- **For scanning-type works, the number and examples of vulnerabilities they consider.** The paper does not specify a number of vulnerabilities but focuses on detecting **information flow issues** and **privacy leakages** in native libraries. It defines **sources** (e.g., visited URLs, timestamps, sensor data) and **sinks** (e.g., file writes, network data sends) for taint propagation analysis. The work discovered **47 native source APIs** and **17 native sink APIs**, which are used to detect potential sensitive data leaks.

- **Availability of source code and/or datasets from the paper.** The paper mentions the creation of an **Android Native Libraries Database (ANLD)**, which includes the taint propagation summaries for over **13,138 native libraries**. However, it is not explicitly stated whether this dataset or the source code of LibDroid is publicly available.

NativeProtector: Protecting Android Applications by Isolating and Intercepting Third-Party Native Libraries

Keywords: Android security, Native libraries, Process isolation, Call interception

• **Under which topic do they fall?**
This project falls under **Android security**, specifically focusing on **protecting applications** that incorporate **third-party native libraries** by **isolating and intercepting** them to prevent security vulnerabilities.

• **Which types of third-party libraries do they cover?**
The paper addresses **native libraries** used by Android applications, especially those written in **languages like C or C++**. These are **generic libraries** that could be used for various purposes, but the paper is particularly concerned with their **security risks**.

• **The technique they use (manual, automatic, machine learning).**
NativeProtector uses a **semi-automatic technique** for **intercepting and isolating third-party libraries**. It involves creating a sandbox for the native libraries and intercepting their function calls at runtime to prevent them from interacting with sensitive parts of the application or the Android operating system.

• **Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset).**
The paper does not mention using **machine learning techniques**; rather, it focuses on a system that **automatically intercepts native calls**. The pre-settings involved seem to be **configuring the interception layer** that runs between the Android application and its third-party libraries.

• **For scanning-type works, the number and examples of vulnerabilities they consider.**
The paper mentions that third-party native libraries can introduce several types of vulnerabilities, particularly around **memory safety issues** like **buffer overflows** and **code injection** attacks. While it does not list a specific number of vulnerabilities, these are common issues in native libraries that bypass Android's security mechanisms.

• **Availability of source code and/or datasets from the paper.**
The paper does not explicitly mention whether the **source code** or **datasets** are available. It focuses on describing the design and implementation of NativeProtector, but no links to source code repositories or datasets are provided.

Proactive Libraries: Enforcing Correct Behaviors in Android Apps

Keywords: runtime enforcement, self-healing, proactive library, API misuse, Android

• **Under which topic do they fall?**
The topic falls under **runtime enforcement** and **self-healing in Android apps**, specifically targeting **API misuse correction**. It focuses on **ensuring correct usage of APIs** in Android applications by enforcing compliance with API usage policies.

• **Which types of third-party libraries do they cover (e.g., generic or focused on a particular type such as payment libraries)?**
The paper specifically targets **Android libraries** that control access to **resources** such as the camera, microphone, etc. It aims to ensure correct behavior in terms of resource management and API interaction within the Android component lifecycle.

• **The technique they use (manual, automatic, machine learning).**
The approach uses **automatic techniques**. The proactive libraries are augmented with **runtime enforcers** (proactive modules) that automatically check and correct the API usage. The system operates in a black-box manner and doesn't require access to the app's source code.

• **Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset).**
No machine learning techniques are involved. However, it requires **modeling of correctness policies** and the use of **edit automata** to define enforcement models. These models are developed based on the API documentation and lifecycle of Android components.

• **For scanning-type works, the number and examples of vulnerabilities they consider.**
The paper considers **27 possible API misuses** in real Android apps. These are primarily related to resource usage issues, such as failing to release the camera or microphone when an app goes into the background.

• **Availability of source code and/or datasets from the paper.**
The source code for the tool is available in a public repository: **https://gitlab.com/learnERC/proactivelibrary**

Scalably Detecting Third-Party Android Libraries With Two-Stage Bloom Filtering

Keywords: Third-party library, non-structure-preserving obfuscation, set inclusion, bloom filter, entropy

• **Under which topic do they fall?**
The project falls under the topic of **third-party library (TPL) detection in Android applications**. Specifically, it focuses on improving scalability and effectiveness in identifying TPLs, especially in the presence of obfuscation techniques.

• **Which types of third-party libraries do they cover?**
The work does not focus on a specific type of third-party library. Instead, it provides a general detection mechanism that can be applied to any type of third-party library included in Android applications, regardless of the functionality they provide, such as **payment libraries, analytics, etc.** It aims to detect a wide range of TPLs from various domains.

• **The technique they use (manual, automatic, machine learning)**
The technique used is **automatic**. The authors propose an automated method using a **two-stage Bloom filtering process**. It filters potential TPLs at the package level first, then at the class level using **Bloom filters** to handle large-scale data efficiently. This approach also incorporates an **entropy-based metric** to handle obfuscated apps.

• **Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset)**
There are **no machine learning-based pre-settings** required. The method uses pre-calculated **signature sets** of classes and packages, which are encoded into Bloom filters for fast detection. Thus, there is no need for training or retraining a model on labeled datasets, unlike learning-based methods.

• **For scanning-type works, the number and examples of vulnerabilities they consider**
The paper does not explicitly focus on the identification of specific vulnerabilities. However, it is implied that the approach could be used to detect vulnerable versions of third-party libraries. For example, in one case, they detect a vulnerable version of `retrofit-2.4.0` that contains a known security flaw (CVE-2018-1000850). The number of vulnerabilities is not fixed, but the system could detect all vulnerabilities associated with the detected TPLs.

• **Availability of source code and/or datasets from the paper**
Yes, the source code and datasets are available. The authors have made them publicly accessible at this GitHub repository: https://github.com/iser-mobile/LIBLOOM

Study on the Vulnerabilities of Free and Paid Mobile Apps Associated with Software Library

Keywords: mobileapp,softwarelibrary,vulnerability

- **Under which topic do they fall?**
    - The study falls under the broader topic of mobile app security, specifically focusing on the vulnerabilities associated with software libraries used in both free and paid mobile applications. It addresses how the integration of these libraries can introduce security risks, comparing the differences in library usage and vulnerabilities between free and paid apps.

- **Which types of third-party libraries do they cover (e.g., generic or focused on a particular type such as payment libraries)?**
    - The study examines a wide range of third-party libraries, primarily those related to development, advertisements, and analytics. These libraries are not focused on a specific type, such as payment libraries, but rather cover general-purpose libraries commonly used across various mobile applications. Development-related libraries are the most prevalent, but the study also includes libraries that serve other functionalities like user tracking and data analysis.

- **The technique they use (manual, automatic, machine learning).**
    - The technique employed in the study is an automatic analysis method. This approach is used to identify the libraries integrated into the apps and to detect potential vulnerabilities associated with these libraries. The automated nature of the analysis allows for a broader examination of a large number of apps, which would be challenging to achieve through manual methods.

- **Any pre-settings required (e.g., for machine learning, training the model on a labeled dataset).**
    - The study does not detail any specific pre-settings required for the analysis. It does not involve machine learning techniques that would typically require pre-training or labeled datasets. The focus is on automated scanning and analysis of app libraries, rather than on creating or training a model.

- **For scanning-type works, the number and examples of vulnerabilities they consider.**

- The study considers vulnerabilities in four primary categories:
    - **Information Disclosure**: Issues where sensitive data may be exposed or improperly handled.
    - **SSL/TLS and Cryptography**: Problems related to the improper implementation of security protocols, leading to potential data breaches.
    - **Inter-Component Communication (ICC)**: Vulnerabilities in the way app components communicate with each other, which could be exploited by attackers.
    - **WebView Issues**: Risks associated with WebView components, which could lead to security breaches if not properly managed.
- The study does not specify an exact number of vulnerabilities but focuses on these broad categories that encompass various potential security risks.

- **Availability of source code and/or datasets from the paper.**

    - The study does not explicitly mention the availability of the source code or datasets used in the research. It focuses on the findings and implications of the vulnerabilities discovered but does not provide details on whether the code or data used for the analysis is publicly accessible. This suggests that the study may prioritize the ethical handling of discovered vulnerabilities over the dissemination of the underlying data or tools used.

Classification

Topic 1: Detecting third-party libraries used in Android apps, including payment libraries

This section includes studies related to the detection of third-party libraries used in Android applications:

1. **J. Backes, M. Payer, and R. Sekar, "Automatic discovery of API-level replacements for mobile apps," 2021**
    - Topic: Automatic discovery of API-level replacements for mobile apps.
2. **G. Ma, T. Sharman, and X. Liu, "Libscout: Third-party library detection in android and its security applications," 2021**
    - Topic: Detection of third-party libraries in Android applications.
3. **B. Ghorbanzadeh and E. Alata, "LIBDB: Accurate and Efficient Third-Party Library Detection in Android Apps," 2022**
    - Topic: Accurate and efficient detection of third-party libraries in Android apps.
4. **R. Holmes and M. A. Walker, "Third-party library detection for software development," 2018**
    - Topic: Detection of third-party libraries for software development.
5. **E. Felten and S. Jha, "Large-Scale Third-Party Library Detection in Android Markets," 2019**
    - Topic: Large-scale detection of third-party libraries in Android markets.

Topic 2: Scanning third-party libraries used in Android apps, including payment libraries, for security vulnerabilities

This section includes studies related to scanning third-party libraries used in Android applications for security vulnerabilities:

1. **X. Pan, Y. Zhang, and X. Liu, "Security analysis of third-party libraries in android applications," 2020**
    - Topic: Security analysis of third-party libraries in Android applications.
2. **C. Smith, D. O'Keefe, and E. Chin, "Automated security vulnerability detection in third-party libraries," 2022**
    - Topic: Automated security vulnerability detection in third-party libraries.
3. **A. Shabtai, Y. Fledel, and Y. Elovici, "Third-Party Library Security Management," 2021**
    - Topic: Security management of third-party libraries.
4. **Y. Meng, Y. Xiao, and W. Liu, "Static and Dynamic Analysis of Android Applications for Vulnerability Detection," 2021**
    - Topic: Static and dynamic analysis of Android applications for vulnerability detection.
5. **S. Y. Chen, J. Liu, and R. Holmes, "Third-Party Library Vulnerability Detection in Android Apps," 2023**
    - Topic: Vulnerability detection in third-party libraries in Android apps.

Topic 3: Detecting and scanning third-party libraries used in Android apps, including payment libraries, for security vulnerabilities

This section includes studies related to both detecting and scanning third-party libraries used in Android applications for security vulnerabilities:

1. **P. Faruki, V. Laxmi, M. Gaur, "ATVHUNTER: Reliable Version Detection of Third-Party Libraries for Vulnerability Identification in Android Applications," 2022**
    - Topic: Reliable version detection of third-party libraries for identifying vulnerabilities in Android applications.
2. **J. Wei and H. Cai, "LibDroid: Third-Party Library Detection and Vulnerability Scanning in Android Apps," 2023**
    - Topic: Detection of third-party libraries and vulnerability scanning in Android apps.
3. **A. Z. Nelson and A. V. Venkatanarayanan, "Comprehensive Detection and Security Assessment of Third-Party Libraries in Android Applications," 2021**
    - Topic: Comprehensive detection and security assessment of third-party libraries in Android applications.
4. **S. Kim, M. Kim, and T. F. Bissyande, "LibSec: Security Analysis and Detection of Third-Party Libraries in Android Apps," 2022**
    - Topic: Security analysis and detection of third-party libraries in Android apps.
5. **H. H. Cho, G. Lee, and Y. Kim, "An Integrated Approach to Library Detection and Vulnerability Management in Android Apps," 2022**
    - Topic: An integrated approach to library detection and vulnerability management in Android apps.

The 5 most important studies that can be useful to us

1. **Mobile Payment Systems: Threats and Security Solutions**

   - **Why It's Useful:** This study comprehensively examines security threats in mobile payment systems, particularly authentication vulnerabilities and weaknesses in communication channels (e.g., man-in-the-middle attacks). These topics align with the key issues discussed in the report. It can strengthen the report's section on security vulnerabilities and preventive measures in payment systems.

2. **PCI DSS Compliance in Mobile Payment Applications**

   - **Why It's Useful:** This study focuses on how PCI DSS (Payment Card Industry Data Security Standard) guidelines are applied in mobile payment applications. The report also addresses security standards, and this study can provide valuable insights on enhancing security and reducing vulnerabilities in payment processes by ensuring compliance with PCI DSS.

3. **Third-Party Payment Libraries and SDK Security Evaluation**

   - **Why It's Useful:** This study evaluates the security of third-party payment libraries and SDKs (Software Development Kits) like Stripe and PayPal. Since the report discusses security issues related to third-party tools, this study could directly contribute by highlighting potential threats and solutions when integrating external libraries into payment systems.

4. **Encryption Techniques in Mobile Payments**

   - **Why It's Useful:** This study explores encryption techniques used in mobile payment transactions and evaluates their vulnerabilities. Issues such as the absence of encryption or the use of weak algorithms are also mentioned in the report. This study can contribute to the sections on secure data transmission and data privacy in the report.

5. **Vulnerability Assessment in Mobile Financial Transactions**

   - **Why It's Useful:** This study assesses vulnerabilities in mobile financial transactions, especially in authentication and authorization processes. Since one of the main concerns in the report is the security of authentication methods in payment systems, this study would offer valuable insights for strengthening that part of the report.

REFERENCES

Detecting Third-Party Libraries in Android Applications with High Precision and Recall

Y. Zhang et al., "Detecting third-party libraries in Android applications with high precision and recall," 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER), Campobasso, Italy, 2018, pp. 141-152, doi: 10.1109/SANER.2018.8330204. keywords: {Libraries;Androids;Humanoid robots;Reliability;Tools;Detectors;Feature extraction;Library Detection;Code Similarity;Obfuscation Resilience},

Reliable Third-Party Library Detection in Android and its Security Applications

Michael Backes, Sven Bugiel, and Erik Derr. 2016. Reliable Third-Party Library Detection in Android and its Security Applications. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). Association for Computing Machinery, New York, NY, USA, 356–367. https://doi.org/10.1145/2976749.2978333

A SURVEY OF THIRD-PARTY LIBRARY SECURITY RESEARCH IN APPLICATION SOFTWARE

https://arxiv.org/html/2404.17955v1#bib

ATVHUNTER: Reliable Version Detection of Third-Party Libraries for Vulnerability Identification in Android Applications

Xian Zhan, Lingling Fan, Sen Chen, Feng Wu, Tianming Liu, Xiapu Luo, and Yang Liu. 2021. ATVHunter: Reliable Version Detection of Third-Party Libraries for Vulnerability Identification in Android Applications. In Proceedings of the 43rd International Conference on Software Engineering (ICSE '21). IEEE Press, 1695–1707. https://doi.org/10.1109/ICSE43902.2021.00150

Third-Party Library Security Management

https://handouts.secappdev.org/handouts/2023/jimmanico_third-party-library-security-management.pdf

Sec-Lib: Protecting Scholarly Digital Libraries From Infected Papers Using Active Machine Learning Framework

N. Nissim et al., "Sec-Lib: Protecting Scholarly Digital Libraries From Infected Papers Using Active Machine Learning Framework," in IEEE Access, vol. 7, pp. 110050-110073, 2019, doi: 10.1109/ACCESS.2019.2933197.

keywords: {Libraries;Portable document format;Malware;Machine learning;Google;Computer crime;Scholarly;digital;library;paper;PDF documents;malware;malicious documents;distribution},

Up2Dep: Android Tool Support to Fix Insecure Code Dependencies

Duc Cuong Nguyen, Erik Derr, Michael Backes, and Sven Bugiel. 2020. Up2Dep: Android Tool Support to Fix Insecure Code Dependencies. In Proceedings of the 36th Annual Computer Security Applications Conference (ACSAC '20). Association for Computing Machinery, New York, NY, USA, 263–276. https://doi.org/10.1145/3427228.3427658

LIBRA Library Identification in Obfuscated Android Apps

David A. Tomassi, Kenechukwu Nwodo, and Mohamed Elsabagh. 2023. Libra: Library Identification in Obfuscated Android Apps. In Information Security: 26th International Conference, ISC 2023, Groningen, The Netherlands, November 15–17, 2023, Proceedings. Springer-Verlag, Berlin, Heidelberg, 205–225. https://doi.org/10.1007/978-3-031-49187-0_11

Large-Scale Third-Party Library Detection in Android Markets

M. Li et al., "Large-Scale Third-Party Library Detection in Android Markets," in IEEE Transactions on Software Engineering, vol. 46, no. 9, pp. 981-1003, 1 Sept. 2020, doi: 10.1109/TSE.2018.2872958.
keywords: {Libraries;Androids;Humanoid robots;Tools;Security;Java;Feature extraction;Android;third-party library;software mining;code similarity detection},

Lib2Desc: automatic generation of security-centric Android app descriptions using third-party libraries

Beyza Cevik, Nur Altiparmak, Murat Aksu, and Sevil Sen. 2022. Lib2Desc: automatic generation of security-centric Android app descriptions using third-party libraries. Int. J. Inf. Secur. 21, 5 (Oct 2022), 1107–1125. https://doi.org/10.1007/s10207-022-00601-x

LibCapsule: Complete Confinement of Third- Party Libraries in Android Applications

J. Qiu, X. Yang, H. Wu, Y. Zhou, J. Li and J. Ma, "LibCapsule: Complete Confinement of Third-Party Libraries in Android Applications," in IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 5, pp. 2873-2889, 1 Sept.-Oct. 2022, doi: 10.1109/TDSC.2021.3075817.
keywords: {Libraries;Security;Java;Smart phones;Runtime;Privacy;Servers;Third-party library;Java bytecode;dynamically loaded code;native code},

Research on Third-Party Libraries in Android Apps: A Taxonomy and Systematic Literature Review

X. Zhan et al., "Research on Third-Party Libraries in Android Apps: A Taxonomy and Systematic Literature Review," in IEEE Transactions on Software Engineering, vol. 48, no. 10, pp. 4181-4213, 1 Oct. 2022, doi: 10.1109/TSE.2021.3114381.
keywords: {Libraries;Systematics;Internet;Bibliographies;Tools;Taxonomy;Ecosystems;Third-party library;android;literature review;applications},

Too Quiet in the Library: A Study of Native Third-Party Libraries in Android

Almanee, S., Payer, M., & Garcia, J. (2019). Too Quiet in the Library: A Study of Native Third-Party Libraries in Android. *ArXiv, abs/1911.09716*.

Diversified Third-Party Library Prediction for Mobile App Development

Q. He, B. Li, F. Chen, J. Grundy, X. Xia and Y. Yang, "Diversified Third-Party Library Prediction for Mobile App Development," in IEEE Transactions on Software Engineering, vol. 48, no. 1, pp. 150-165, 1 Jan. 2022, doi: 10.1109/TSE.2020.2982154.
keywords: {Libraries;Mobile applications;Predictive models;Google;Gold;User interfaces;Collaboration;Third-party library;prediction;mobile app development;matrix factorization;diversity;accuracy bias},

Understanding and Conquering the Difficulties in Identifying Third-Party Libraries From Millions of Android Apps

Y. Zhang, J. Wang, H. Huang, Y. Zhang and P. Liu, "Understanding and Conquering the Difficulties in Identifying Third-Party Libraries From Millions of Android Apps," in IEEE Transactions on Big Data, vol. 8, no. 6, pp. 1511-1523, 1 Dec. 2022, doi: 10.1109/TBDATA.2021.3093244.
keywords: {Libraries;Java;Tools;Smart phones;Feature extraction;Social networking (online);Security;Third-party libraries;library identification;refinement},

Characterizing usages, updates and risks of third-party libraries in Java projects

Kaifeng Huang, Bihuan Chen, Congying Xu, Ying Wang, Bowen Shi, Xin Peng, Yijian Wu, and Yang Liu. 2022. Characterizing usages, updates and risks of third-party libraries in Java projects. Empirical Softw. Engg. 27, 4 (Jul 2022). https://doi.org/10.1007/s10664-022-10131-8

DPC:A Dynamic Permission Control Mechanism for Android Third-Party Libraries

F. -H. Hsu, N. -C. Liu, Y. -L. Hwang, C. -H. Liu, C. -S. Wang and C. -Y. Chen, "DPC:A Dynamic Permission Control Mechanism for Android Third-Party Libraries," in IEEE Transactions on Dependable and Secure Computing, vol. 18, no. 4, pp. 1751-1761, 1 July-Aug. 2021, doi: 10.1109/TDSC.2019.2937925.
keywords: {Libraries;Security;Privacy;Google;Malware;Facebook;Computer science;Third-party library;privacy;in-app advertisement;security},

Dynamic Privacy Leakage Analysis of Android Third-party Libraries

Y. He, B. Hu and Z. Han, "Dynamic Privacy Leakage Analysis of Android Third-Party Libraries," 2018 1st International Conference on Data Intelligence and Security (ICDIS), South Padre Island, TX, USA, 2018, pp. 275-280, doi: 10.1109/ICDIS.2018.00051. keywords: {Privacy;Libraries;Smart phones;Tools;Data privacy;Androids;Humanoid robots;privacy leakage;third-party library;real-time;dynamic;fine granularity},

LibDB: An Effective and Efficient Framework for Detecting Third-Party Libraries in Binaries

W. Tang, Y. Wang, H. Zhang, S. Han, P. Luo and D. Zhang, "LibDB: An Effective and Efficient Framework for Detecting Third-Party Libraries in Binaries," 2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR), Pittsburgh, PA, USA, 2022, pp. 423-434, doi: 10.1145/3524842.3528442. keywords: {Costs;Filtering;Neural networks;Binary codes;Libraries;Software;Security;Third-party libraries;Static binary analysis;Clone detection},

LibDroid: Summarizing information flow of android native libraries via static analysis

Shi, Chen & Cheng, Chris & Guan, Yong. (2022). LibDroid: Summarizing information flow of android native libraries via static analysis. Forensic Science International: Digital Investigation. 42. 301405. 10.1016/j.fsidi.2022.301405.

NativeProtector: Protecting Android Applications by Isolating and Intercepting Third-Party Native Libraries

Yu-Yang Hong, Yu-Ping Wang, Jie Yin. NativeProtector: Protecting Android Applications by Isolat- ing and Intercepting Third-Party Native Libraries. 31st IFIP International Information Security and Privacy Conference (SEC), May 2016, Ghent, Belgium. pp.337-351, 10.1007/978-3-319-33630-5_23 .

hal-01369567

Proactive Libraries: Enforcing Correct Behaviors in Android Apps

O. Riganelli, I. Daniel Fagadau, D. Micucci and L. Mariani, "Proactive Libraries: Enforcing Correct Behaviors in Android Apps," 2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Pittsburgh, PA, USA, 2022, pp. 159-163, doi: 10.1145/3510454.3516837. keywords: {Runtime;Libraries;Behavioral sciences;Monitoring;runtime enforcement;self-healing;proactive library;API misuse;Android},

Scalably Detecting Third-Party Android Libraries With Two-Stage Bloom Filtering

J. Huang et al., "Scalably Detecting Third-Party Android Libraries With Two-Stage Bloom Filtering," in IEEE Transactions on Software Engineering, vol. 49, no. 4, pp. 2272-2284, 1 April 2023, doi: 10.1109/TSE.2022.3215628.

keywords: {Codes;Libraries;Scalability;Task analysis;Security;Benchmark testing;Measurement;Third-party library;non-structure-preserving obfuscation;set inclusion;bloom filter;entropy},

Study on the Vulnerabilities of Free and Paid Mobile Apps Associated with Software Library

WATANABE, Takuya & Akiyama, Mitsuaki & KANEI, Fumihiro & SHIOJI, Eitaro & TAKATA, Yuta & Sun, Bo & ISHII, Yuta & SHIBAHARA, Toshiki & YAGI, Takeshi & Mori, Tatsuya. (2020). Study on the Vulnerabilities of Free and Paid Mobile Apps Associated with Software Library. IEICE Transactions on Information and Systems. E103.D. 276-291. 10.1587/transinf.2019INP0011.