分类号＿＿＿＿＿＿＿＿　　　　　　　　　密级＿＿＿＿＿＿＿＿

UDC＿＿＿＿＿＿＿＿＿　　　　　　　　　编号 6720230866

 江西理工大学

**开　题　报　告**

**论文题名**  基于潜在语义分析的安卓恶意软件检测

申请学位级别 工程硕士

专业名称 网络与信息安全

研 究 生 姓 名 陈治丞

导师姓名、职称 王振东(副教授)

二Ο二五年 一月

目录

[1 选题背景和意义 3](#_Toc186807631)

[2 文献综述 5](#_Toc186807632)

[3 论文主要研究内容 10](#_Toc186807633)

[3.2 研究内容 10](#_Toc186807634)

[3.2 拟解决的关键问题 11](#_Toc186807635)

[3.3 拟采取的研究方法 12](#_Toc186807636)

[4. 可行性分析 14](#_Toc186807637)

[4.1 技术可行性 14](#_Toc186807638)

[4.2 资料来源或实验设备的可能性 14](#_Toc186807639)

[5. 计划安排 16](#_Toc186807640)

[参考文献 17](#_Toc186807641)

# 选题背景和意义

Android 是由 Google 公司以及开放手机联盟于 2007 年领导及开发的一种基于 Linux 的自由及开放源代码的操作系统，它被广泛使用于移动设备。第一部 Android 智能手机发布于 2008 年 10 月，随后，Android 逐渐扩展到平板电脑以及其他领域上，如数码相机、电视、游戏机、智能手表等。相比于传统的塞班（Symbian）移动操作系统，Android 具有以下三个优点：（1）开源性：开源的代码库、免费的开发软件、社区、第三方开源共享等特点吸引越来越多的 Android 系统开发者，在带来巨大的竞争的同时也使得 Android 系统在开放的平台中显得日益成熟。（2）丰富的硬件选择性：由于其开源性，很多厂商为了吸引更多的用户，对 Android 系统加以改造，推出功能特色各具的各种产品，而不会影响到数据同步、软件兼容等功能，从而不断丰富用户体验。（3）百万级别的 app 应用市场：无论国内国外均有能够为用户提供各种丰富 app 下载的 Android 应用市场，例如官方的 Google Play[1]，第三方市场安智[2]等。截至 2018 年 8月，Google Play 上可供下载应用数量超过 280 万款，总下载量近 3300 亿次。以上优点使得 Android 迅速成为了最流行的移动操作系统。根据研究公司 Gartner 提供的数据，在 2018 年第一季度，Android 已经占据了 85.9% 的移动操作系统市场份额[3]。

然而，Android 的开源性以及流行性也使它成为了 97% 恶意软件的主要攻击目标[4]，现如今各大平台市场上充斥着各种各样的恶意软件。根据 360 公司 2017 年 Android 恶意软件年度专题报告，2017 年全年，360 互联网安全中心累计截获 Android 平台新增恶意软件样本 757.3 万个，平均每天新增 2.1 万个[5]。从用户感染恶意软件情况看，2017 年 Android 用户感染恶意软件数量为 2.14 亿。除此之外，根据 Wang 等[1]在 16 个中国安卓市场以及 Google Play 官方市场上共计 600 万应用的大规模分析结果，中国安卓市场上大约 12.30% 的 app 被至少 10 个反病毒引擎报告为恶意，即使是 Google Play 官方市场，这一数据仍可达 2.09%。恶意软件的快速增长给移动智能手机用户带来了巨大的危害，包括资费消耗、隐私窃取以及远程控制等。资费消耗主要是针对手机用户的资费，强行定制服务并从中牟利。隐私窃取主要实现短信、通讯录和通话记录的获取，定位以及拍照等功能，收集用户的隐私数据、社交数据以及设备数据。远程控制主要是使用 http 协议接收服务器的控制指令实现指令操纵、信息回传以及本地恶意代码更新等。

尽管 Google 和原始设备制造商（OEMs）引入了安全增强功能，但是Android 设备仍然不断成为恶意软件活动的目标，并成功感染[6]。这些设备占针对移动设备的网络攻击的 98% 以上[7]，攻击者通过利用移动设备暴露的动态攻击面的各种攻击媒介来执行攻击[8]。虽然 Android 恶意软件的数量远少于 Windows 恶意软件[9]，但日益复杂化的威胁形势的持续演变[10]，使 Android 最终用户处于危险境地，长期面临感染风险[11]。

鉴于传统的防病毒措施，例如指纹识别和黑名单，已被证明在保护移动领域的最终用户方面效果极其有限[12]，特别是针对加密和零日恶意软件，需要将Android 恶意软件研究注意力转向机器学习、深度学习，以寻找更有效的恶意软件检测解决方案。针对安卓恶意软件检测研究领域，目前存在许多无法完全解决的困难，例如，采用不平衡、小且旧的数据集进行模型的训练与预测、采用没有更新完善措施的静态模型产生的概念漂移问题、深度学习预测黑盒的可解释性研究。

# 文献综述

目前，Android应用程序的主要安装文件格式是APK。尽管谷歌在2018年启用了新的文件格式AAB（Android APP Bundles）[23]，但AAB格式仅是为开发者上传应用程序而设计的，而用户下载的安装文件仍然是APK格式。因此，几乎所有研究都采用APK格式文件进行恶意软件检测。根据特征提取中是否需要执行应用程序，不同平台（例如Windows和Android）上的恶意软件检测方法可以分为3类，即静态方法、动态方法和混合方法[23]。对于Android恶意软件检测，静态方法从反编译获得的源代码中提取特征。动态方法通过在实际或模拟环境中运行APK文件来提取特征，然后获得相应的行为模式。混合方法集成静态和动态特征来描述应用程序的各个方面。总体而言，动态方法通常提供较高的检测精度并保证对未知恶意软件的检测能力。然而，检测过程很复杂并且需要大量的计算资源。此外，动态方法通常无法遍历应用程序的所有可能的执行路径，这可能会导致信息丢失。静态方法不需要运行应用程序，检测效率高，可有效用于大规模Android应用程序检测，因此大量研究采用静态检测方法。

静态方法主要有三类：图方法，表示APK文件各组成部分之间的关系，基于图结构实现检测；基于语义特征的方法，提取APK代码的语义特征；基于特征工程的方法，提取APK代码的语义特征。使用权限、意图和其他统计特征。对于基于图的方法，Gao、Cheng 和Zhang[24] 以及 Hei[25] 等人构建了一种同构信息网络（HIN）来表示一个数据集中的应用程序及其组件（例如 API 和权限）之间的关系。然后，使用这些关系来检测恶意软件。大多数图方法提取APK文件的结构特征并使用该特征进行检测。这种方法首先构建控制流图（CFG）或函数调用图（FCG）[26][27]等，基于Java函数或类等内部代码区域的关系，利用图卷积网络等模型来获取结构特征并实现分类。由于恶意软件的恶意行为通常包含在特定的代码区域（即图节点）中，因此在检测中融合代码的语义特征非常重要。然而，现有的图方法没有考虑节点的语义特征。一些图方法使用的节点特征是统计特征而不是代码语义特征[28]。 Milosevic、Dehghantanha和Choo[29]证明了代码的语义特征可以很好地反映 Android 恶意软件的恶意行为。然而，这些方法只关注语义特征，而没有考虑结构信息。

基于机器学习的检测系统的性能受到众多变量的影响，例如数据集、特征、标记精度、算法选择、超参数等。尽管重要变量数量众多，但它们可以抽象为两大类：数据相关和模型相关方面。尽管模型选择和建模假设很重要（例如，线性分类器无法对非线性数据集进行建模），但数据数量和质量是使用机器学习算法进行准确且可概括的数据建模的关键因素。Android恶意软件检测系统目前面临的大部分挑战都是与数据相关的方面。

考虑到 Android 恶意软件检测领域提出的基于机器学习的解决方案的高性能指标，数据问题可能看起来相对无关紧要，而与建模方面更相关。然而，对研究中使用的数据集的仔细研究可以发现数据问题的重要性。

表 1 Android 恶意软件研究中最常用的数据集

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 数据集名 | 恶意样本数 | 良性样本数 | 样本收集时间 | 恶意软件家族数 | 被引次数 |
| MalGenome[13] | 1260 | 0 | 2010-2011 | 49 | 2757 |
| Drebin[14] | 5560 | 123453 | 2010-2012 | 179 | 2218 |
| AMD[15] | 24553 | 0 | 2010-2016 | 71 | 412 |
| AndMal2017[16] | 400 | 5065 | 2015-2017 | 42 | 166 |
| AndMal2020[17] | 200000 | 200000 | - | 191 | 22 |
| MalDroid2020[18] | 11598 | 0 | 2017-2018 | 33 | 62 |

表1 总结了 Android 恶意软件研究中最常用的数据集。如表 1 所示，MalGenome 和 Drebin 是迄今为止使用最广泛的数据集。根据统计的被引用次数分析，82.8%的研究都使用了代表了十多年前的威胁状况的2010 年至 2012 年期间收集的数据。考虑到 Android 数据中威胁态势的动态演变，这使得这些数据集过于陈旧和过时，因此不能代表当前（甚至最近）的威胁态势，因为它们是在恶意软件功能和行为与恶意软件功能和行为明显不同时收集的。此外，Drebin 的较大规模数据集也存在重复项的问题[19]，这限制了可用数据，如果不解决，可能会引入数据窥探偏差的模型验证的一个显着问题，即数据泄漏。尽管存在这些重大问题，Drebin 最近依旧用作恶意软件的主要或单一来源[20]。表1中的一半数据集不提供良性样本。因此，为了构建检测模型，必须收集额外的数据样本，这些样本可能属于与恶意软件数据不同的时间范围。例如，一些研究可能使用 Drebin 的恶意样本并从 Google Play 或 AndroZoo 收集良性样本[21]。这可能会在数据之间引入时间偏差，并产生过度夸大且不具有代表性的结果。在提供两个类别（即合法样本和恶意软件样本）的数据集中，数据通常存在显着比例的不平衡，例如，Drebin 的类别比率为 1:2.2，而 AndMal2017 的类别比率为 1:1.2，许多机器学习算法对不平衡数据很敏感，生成有偏差的模型，如果报告不适当的指标，可能会产生误导性结果。为了避免由于类别不平衡问题而生成有偏差的模型，必须包含额外的数据，例如一些研究采用组合数据集或添加来自VirusShare和AndroZoo等软件存储库的额外样本，或模型生成管道中使用的数据平衡技术，这为建模过程增加了额外的复杂性。因此，对于主要基于单个恶意软件数据源的文献中提出的大多数方法都针对 Android 历史恶意软件检测进行了优化。然后，在特定时间段对恶意软件进行模型训练，并推广到未来数据，其中可能包括威胁类型和数据演变的重大变化。在这方面，大多数提出的检测解决方案既没有考虑这种场景，也没有提供模型更新机制。这意味着使用在特定时间点收集且从未更新的数据构建的静态模型被误认为可以很好地检测未来的数据，并能随着时间的推移保持其检测能力。这一现象忽略了恶意软件数据的演变，并使模型容易出现概念漂移及其对模型性能的退化影响。

大多数基于恶意软件检测模型都是静态模型，是不随时间更新的模型，因此用于构建模型的训练数据和用于评估模型的测试数据来自相同的数据分布。虽然这可能适用于某些应用领域，但大多数机器学习问题都面临非平稳数据分布，其中目标变量的统计属性或定义特征随时间以不可预测的方式变化[30]，这种现象称为概念漂移。一般来说，网络安全领域，特别是 Android 恶意软件检测，其特点是威胁形势的不断演变（例如，恶意软件的演变或新家族的出现）。因此，如果不采取自适应措施，Android恶意软件检测的静态模型很容易出现概念漂移问题，随着时间的推移，会导致性能下降和模型过时。尽管如此，研究文献中提出的绝大多数 Android 恶意软件检测解决方案都是静态模型，忽略了概念漂移，并且没有提出或考虑任何自适应措施（例如模型更新机制或再训练计划）。因此，这些模型只是针对 Android特定时间段数据的基于 ML 的优化，将无法有效处理训练数据集中未表示的新数据和进化的数据。此外，随机分割数据集的通常做法忽略了概念漂移和数据样本之间的时间顺序的存在，导致训练集和测试集之间的历史一致性被破坏[31]，产生由数据窥探导致的有偏见、过度夸大和历史不连贯的结果。这是大多数 Android 恶意软件检测研究中存在的主要验证缺陷。

可解释的人工智能（XAI）旨在揭示机器学习模型预测背后的决策过程，这些模型大多被视为黑盒模型（即特别是基于深度学习的系统）。应用解释方法来理解检测模型及其预测不仅可以提供有关模型行为的相关信息，还可以提供有关数据集行为的相关信息本身（即 Android 恶意软件检测中的威胁态势）。例如，Guerra-Manzanares[32]等人利用特征重要性演化来理解系统调用数据上的概念漂移问题，之后Guerra-Manzanares[33] 等人对此进行了扩展用于基于设备的分析，Guerra-Manzanares[34] 等人提供了针对安全权限和特定恶意软件系列的类似分析。尽管该领域的一些研究强调了预测的可解释性对于审查模型输出以改进检测机制的重要性[35]，但 XAI 技术在网络安全和网络安全领域的范围和应用尤其是 Android 恶意软件检测仍然有限。在这方面，Scala[36]等人使用可解释性方法来查找 Android 应用程序使用的最有歧视性的 API 调用，而 Kinkead[35] 等人使用 LIME 查找总体分类和特定恶意软件家族的最重要的全局特征。Karn[37]等人比较了用于基于云的恶意软件检测的不同 XAI 技术，Iadarola[38] 等人提出了一种用于 Android 恶意软件检测和家族识别的可解释深度学习模型。尽管解释方法并非没有局限性和假设，所以其输出结果必须仔细对比和分析，但当应用于高性能模型时，它们不仅可以用于满足法律要求并进行准确预测，还能帮助理解模型预测结果以及提取相关的数据解释。

# 3 论文主要研究内容

## 3.2 研究内容

为了实现安卓恶意软件检测模型的高效训练和准确检测，提出一种基于潜在语义分析的安卓恶意软件检测模型。研究内容包括一下方面：

1. 构建能体现随时间变化而不断进化的多阶段数据集

针对安卓恶意软件不断进化的特点，构建一个能够反映恶意软件随时间变化的多阶段数据集。该数据集将包含不同时间段的恶意软件样本，以捕捉其演变特征，为模型训练提供丰富的时序信息。实现数据预处理和特征提取流程，以确保数据集能够有效地用于模型训练。

1. 新型调用图设计

设计一种新型调用图，该图不仅包含程序的结构信息，还融合了代码中潜在的语义特征。通过结合传统的控制流图和数据流图，以及引入自然语言处理技术，提取代码中的语义信息，增强模型对恶意行为的识别能力。构建一个深度融合网络，该网络能够同时处理结构信息和语义信息，以提取更全面的图特征。

1. 深度融合网络构建

利用图神经网络（GNN）技术，特别是图注意力网络（GAT），构建一个能够处理结构化图数据的深度学习模型。该模型将对新型调用图中的节点和边进行编码，以学习节点的潜在表示，进而用于恶意软件的二分类任务。研究和优化网络结构，以提高模型对复杂图结构的处理能力和分类准确性。

1. 潜在语义信息的解释机制

开发一种解释机制，用于解释模型的预测结果，特别是模型如何利用潜在语义信息进行恶意软件检测。该机制将帮助研究人员和用户理解模型的决策过程，提高模型的可解释性和可信度。探索可视化技术，以直观展示模型如何关注特定的代码语义特征，以及这些特征如何与恶意行为相关联。

1. 模型训练与评估

设计和实现模型的训练流程，包括损失函数的选择、优化算法的应用以及超参数的调整。采用交叉验证和多个评估指标（如准确率、召回率、F1分数等）来评估模型的性能。与现有的恶意软件检测模型进行比较，以验证所提出模型的有效性和优越性。

## 3.2 拟解决的关键问题

本研究旨在解决以下关键问题，以确保提出的安卓恶意软件检测模型的有效性和实用性：

1. 多阶段数据集的构建与动态特征捕捉：

如何构建一个全面且具有代表性的多阶段数据集，以反映安卓恶意软件随时间的演变特征。以及在数据集中捕捉和表示恶意软件的行为动态和语义变化。

1. 新型调用图的设计挑战：

如何设计一种新型调用图，有效地融合代码的结构信息和潜在的语义特征。如何确保新型调用图能够为深度学习模型提供有用的特征，以提高恶意软件检测的准确性。

1. 深度融合网络的构建与优化：

如何构建一个能够处理结构信息和语义信息的深度融合网络，并确保其有效性。如何优化网络结构和参数，以提高模型对复杂图结构的处理能力和分类准确性。

1. 潜在语义信息的解释与可视化：

如何开发一种有效的解释机制，以解释模型如何利用潜在语义信息进行检测，以及通过可视化技术直观展示模型的决策过程和关注的特征。

1. 模型的泛化能力与实际应用：

如何确保模型在不同类型的恶意软件样本上具有良好的泛化能力。如何在实际应用中评估模型的性能，特别是在面对新型恶意软件时的适应性和鲁棒性。

## 3.3 拟采取的研究方法

本研究将通过以下步骤来实现安卓恶意软件检测模型的研究和验证：



1. 应用程序数据集构建：

从Drebin、VirusShare、CICMalware2020等数据源中，按照样本收集的时间，提取出四个时间段（2010-2012、2013-2016、2017-2018、2019-2023）的恶意软件样本。从AndroZoo软件库中下载与恶意样本相对应的良性样本，按照恶意样本与良性样本1:2的比例组合成四个数据子集，以确保数据集的平衡性和代表性。

1. 敏感API数据集构建：

从安卓开放源代码项目（AOSP）中提取所有安卓系统API信息，包括所需权限敏感度，建立敏感API数据集。该数据集将用于识别和标记应用程序中使用的敏感API，为后续的恶意行为检测提供关键特征。

1. 应用程序反编译与类调用图构建：

使用JADX对应用程序进行反编译，获取Java源代码，并构建应用程序的类调用图。设计构建算法，对属于同一个包的类节点和第三方库的类节点进行动态合并压缩，以简化图结构。删除与敏感API匹配度低的节点，进一步压缩图的大小，生成类调用子图，以提高后续处理的效率。

1. 潜在语义主题提取

对源代码进行词法分析，将源代码转换为可以进行机器学习处理的向量形式。采用LDA（Latent Dirichlet Allocation）或HDP（Hierarchical Dirichlet Process）等主题模型，从源代码的向量表示中提取潜在语义主题。

这些模型能够识别源代码中相似的语义模式，并将它们归类为不同的主题，为后续的特征提取提供丰富的语义信息。考虑使用预训练的语言模型，如BERT或GPT，来进一步增强对源代码语义的理解，尤其是在处理复杂的代码结构和上下文依赖时。

1. 特征提取与融合：

使用带有最大池化的图神经网络（GNN）从类调用子图中提取特征。这个过程涉及潜在语义主题特征和图结构特征的融合，以捕捉代码的全局和局部结构信息，以及代码中蕴含的语义信息。设计图注意力机制，使模型能够自适应地关注对分类任务更重要的节点和边，提高特征提取的准确性。

1. 模型训练与优化：

采用深度学习框架 PyTorch，实现图神经网络模型，并在构建的数据集上进行训练。通过交叉验证和调整超参数，优化模型的性能，确保模型在不同数据子集上具有良好的泛化能力。使用早停法（Early Stopping）和正则化技术防止模型过拟合。

1. 模型评估与验证：

采用准确率、召回率、F1分数等指标对模型进行评估，并与现有恶意软件检测模型进行比较。分析模型在不同类型恶意软件上的检测效果，以及在面对新型恶意软件时的适应性和鲁棒性。

# 4. 可行性分析

## 4.1 技术可行性

**数据集的可用性**：已有的公开数据集如Drebin、VirusShare、CICMalware2020和AndroZoo提供了丰富的恶意软件和良性样本，这些数据集的可用性为本项目的多阶段数据子集构建提供了基础。

**图神经网络技术的成熟度**：图神经网络（GNN），尤其是图注意力网络（GAT），在处理图结构数据方面已经显示出了优异的性能，这为本项目中新型调用图的设计和深度融合网络的构建提供了技术支持。

**自然语言处理技术的进步**：随着自然语言处理（NLP）技术的发展，特别是主题模型如LDA和HDP的应用，使得从源代码中提取潜在语义主题变得可行。

**深度学习框架的支持**：现有的深度学习框架，如TensorFlow和PyTorch，提供了构建和训练复杂神经网络模型的工具和库，这些框架的灵活性和强大的社区支持为本项目的实现提供了便利。

## 4.2 资料来源或实验设备的可能性

**资料来源**：

公开数据集：Drebin、VirusShare、CICMalware2020和AndroZoo。

学术论文和研究报告：通过Google Scholar、IEEE Xplore、ACM Digital Library等数据库获取。

开源项目和代码：GitHub和GitLab等平台提供了大量的开源项目和代码，可用于参考和实验。

**实验设备**：

实验环境：本项目的实验环境将基于实验室提供的工作站电脑

操作系统：Linux Ubuntu 22.04 LTS

计算资源：GPU：RTX4080和CPU：i7-14700k提供了足够的计算能力来训练和测试深度学习模型。

存储资源：至少10TB的硬盘空间。

# 5. 计划安排

|  |  |
| --- | --- |
| 考核时间节点（年/月） | 阶段目标（阶段考核指标） |
| 2025年01月-04月 | 统筹规划，完成项目总体方案论证，包括完成项目的可行性研究、需求分析、项目任务规划， 并完成相关文档的编写 |
| 2025年05月-08月 | 实现所有设计的策略并验证实 验，完成小论文 |
| 2025年09月-12月 | 丰富实验，踏实开展对比工作 |
| 2026年01月-03月 | 撰写毕业论文做好参加毕业答辩之前的材料准备 |

# 6. 参考文献

1. Wang H Y, Liu Z, Liang J Y, et al. Beyond google play: a large-scale comparative study of Chinese Android app markets. In: Proceedings of the Internet Measurement Conference (IMC), Boston, 2018. 293–307
2. Avdiienko V, Kuznetsov K, Gorla A, et al. Mining apps for abnormal usage of sensitive data. In: Proceedings of IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE), Florence, 2015. 426–436
3. Chen K, Liu P, Zhang Y J. Achieving accuracy and scalability simultaneously in detecting application clones on Android markets. In: Proceedings of the IEEE/ACM 36th International Conference on Software Engineering (ICSE), Hyderabad, 2014. 175–186
4. Li M H, Wang W, Wang P, et al. Libd: scalable and precise third-party library detection in Android markets. In: Proceedings of the IEEE/ACM 39th International Conference on Software Engineering (ICSE), Buenos Aires, 2017. 335–346
5. Feng Y, Anand S, Dillig I, et al. Apposcopy: semantics-based detection of Android malware through static analysis. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE), Hong Kong, 2014. 576–587
6. Dassanayake, D., 2021. Millions of Android phones infected by dangerous malware, these phones are at risk. [Online] Available at: https://www.express.co.uk/life-style/science-technology/1527645/Millions-Android-phones-infected-dangerous-malware-these-phones-at-risk.
7. Kaspersky, 2020. Mobile security: Android vs iOS - which one is safer? https://www.kaspersky.com/resource-center/threats/android-vs-iphone-mobile-security.
8. Townsend, K., 2020. How smartphones have become one of the largest attack surfaces. https://blog.avast.com/smartphones-and-increasing-mobile-threats-avast.
9. Ruth C., 2022. Over 30 million new malware samples found in 2022 as cyber threats evolve. https://atlasvpn.com/blog/over-30-million-new-malware-samples-found-in2022-as-cyber-threats-evolve.
10. Gurubaran, 2022. Hackers use new sophisticated version of Android spyware to conduct mobile surveillance. https://cybersecuritynews.com/sophisticated-version-of-android-spyware/.
11. Spadafora, A., 2022. Millions of Android users at risk of attack after widespread security issue uncovered. https://www.techradar.com/news/millions-of-android-users-at-risk-of-attack-after-widespread-security-issue-uncovered.
12. Timothy, M., 2022. Why you should uninstall your Android antivirus software. https://www.makeuseof.com/uninstall-android-antivirus/.
13. Zhou, Y., Jiang, X., 2015. Malgenome project. [Online] Available at: <http://www.malgenomeproject.org/>.
14. Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., Siemens, C., 2014. Drebin: effective and explainable detection of Android malware in your pocket. In: Ndss, vol. 14, pp. 23–26.
15. ArgusLab, 2020. Amd dataset - Argus Cyber Security Lab.
16. U. of New Brunswick, 2020c. Android malware dataset (CIC-ANDMAL2017).
17. U. of New Brunswick, 2020e. Cccs-CIC-ANDMAL-2020. [Online] Available at: <https://www.unb.ca/cic/datasets/andmal2020.html>.
18. U. of New Brunswick, 2020f. CicMalDroid 2020. [Online] Available at: <https://www.unb.ca/cic/datasets/maldroid-2020.html>.
19. Irolla, P., Dey, A., 2018. The duplication issue within the Drebin dataset. J. Comput. Virol. Hacking Tech. 14 (3), 245–249.
20. Reddy, R., Swamy, M.K., Kumar, D.A., 2021. Feature and sample size selection for malware classification process. In: ICCCE 2020. Springer, pp. 217–223.
21. Allix, K., Bissyandé, T.F., Klein, J., Le Traon, Y., 2016. Androzoo: collecting millions of Android apps for the research community. In: 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR). IEEE, pp. 468–471.
22. Reddy, R., Swamy, M.K., Kumar, D.A., 2021. Feature and sample size selection for malware classification process. In: ICCCE 2020. Springer, pp. 217–223.
23. Odusami, M., Abayomi-Alli, O., Misra, S., Shobayo, O., Damasevicius, R., & Maskeliunas, R. (2018). Android malware detection: A survey. In International conference on applied informatics (pp. 255–266).
24. Gao, H., Cheng, S., & Zhang, W. (2021). Gdroid: Android malware detection and classification with graph convolutional network. Computers & Security, 106, Article 102264.
25. Hei, Y., Yang, R., Peng, H., Wang, L., Xu, X., Liu, J., et al. (2021). Hawk: Rapid android malware detection through heterogeneous graph attention networks. IEEE Transactions on Neural Networks and Learning Systems.
26. Allix, K., Bissyandé, T. F., Jérome, Q., Klein, J., Le Traon, Y., et al. (2016). Empirical assessment of machine learning-based malware detectors for android. Empirical Software Engineering, 21(1), 183–211.
27. Xu, Z., Ren, K., Qin, S., & Craciun, F. (2018). Cdgdroid: Android malware detection based on deep learning using CFG and DFG. In International conference on formal engineering methods (pp. 177–193).
28. Vinayaka, K., & Jaidhar, C. (2021). Android malware detection using function call graph with graph convolutional networks. In 2021 2nd International conference on secure cyber computing and communications (pp. 279–287).
29. Milosevic, N., Dehghantanha, A., & Choo, K.-K. R. (2017). Machine learning aided android malware classification. Computers & Electrical Engineering, 61, 266–274.
30. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G., 2018. Learning under concept drift: a review. IEEE Trans. Knowl. Data Eng. 31 (12), 2346–2363.
31. Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., Cavallaro, L., 2019. {TESSERACT}: eliminating experimental bias in malware classification across space and time. In: 28th {USENIX} Security Symposium ({USENIX} Security 19), pp. 729–746.
32. Guerra-Manzanares, A., Luckner, M., Bahsi, H., 2022b. Android malware concept drift using system calls: detection, characterization and challenges. Expert Syst. Appl. 206, 117200.
33. Guerra-Manzanares, A., Luckner, M., Bahsi, H., 2022c. Concept drift and cross-device behavior: challenges and implications for effective Android malware detection. Comput. Secur., 102757.
34. Guerra-Manzanares, A., Bahsi, H., Luckner, M., 2022a. Leveraging the first line of defense: a study on the evolution and usage of Android security permissions for enhanced Android malware detection. J. Comput. Virol. Hacking Tech., 1–32.
35. Kinkead, M., Millar, S., McLaughlin, N., O’Kane, P., 2021. Towards explainable cnns for Android malware detection. Proc. Comput. Sci. 184 (2019), 959–965
36. Scalas, M., Maiorca, D., Mercaldo, F., Visaggio, C.A., Martinelli, F., Giacinto, G., 2019. On the effectiveness of system API-related information for Android ransomware detection. Comput. Secur. 86, 168–182
37. Karn, R.R., Kudva, P., Huang, H., Suneja, S., Elfadel, I.M., 2021. Cryptomining detection in container clouds using system calls and explainable machine learning. IEEE Trans. Parallel Distrib. Syst. 32 (3), 674–691.
38. Iadarola, G., Martinelli, F., Mercaldo, F., Santone, A., 2021. Towards an interpretable deep learning model for mobile malware detection and family identification. Comput. Secur. 105, 102198