# 中国矿业大学计算机学院

# 2019 级本科生课程报告

| | |
|---|---|
| 课程名称 | 文献检索与学术写作 |
| 报告时间 | 2021 年 10 月 25 日 |
| 学生姓名 | 许万鹏 |
| 学　　号 | 05191643 |
| 专　　业 | 信息安全 |
| 任课教师 | 李鸣 |

# 任课教师评语

## 任课教师评语

| 考查点 | 18-20 | 16-17 | 14-15 | 12-13 | 0-11 | 得分 |
|---|---|---|---|---|---|---|
| 文献检索的方法、步骤；检索的有效性（20 分） | 熟悉 4 种以上数据库文献检索的方法、步骤；能正确提取检索的作者；充分检索了相关文献并进行了相应的分析 | 较为熟悉 4 种以上数据库文献检索的方法、步骤；能正确提取检索的关键词和作者；较为充分检索了相关文献 | 较为熟悉 3 种以上数据库文献检索的方法、步骤；检索相关文献不够充分 | 文献检索的方法、步骤不够熟悉，存在缺陷；检索的相关文献不够充分，存在较大片面性 | 文献检索的方法、步骤不熟悉；检索的相关文献较少 | |
| 在各种中英文数据库的熟练使用基础上，能对检索结果进行有效分析（20 分） | 在信息检索的基础上，给出的文献重要性和作者重要性，理由合理；格式规范 | 在信息检索的基础上，给出的文献重要性和作者重要性，理由较为合理；格式规范 | 在信息检索的基础上，给出的文献重要性和作者重要性理由基本合理；格式规范 | 在信息检索的基础上，给出的文献重要性和作者重要性理由不够合理 | 不能正确使用各种中英文数据库；给出的文献重要性和作者重要性理由不合理、牵强 | |
| 总分（40 分） | | | | | | |

# 基于机器学习的 Android 恶意软件检测方法

许万鹏
（中国矿业大学计算机科学与技术学院，江苏徐州 221116）

**摘 要：** 随着最近内置 Android 子系统的操作系统（如鸿蒙、Windows11）出现，以及这些操作系统在各行业的日益普及，若不为针对实体设备的恶意软件进行检测，发生沙盒逃逸的风险也成为可能。由于可用资源和授予用户的特权有限，检测此类恶意软件成为了一种独特的挑战，但也为附加到每个应用程序所需的元数据提供了独特的机会。在本文中，我提出了一个基于机器学习的 Android 子系统恶意软件检测系统。我的检测系统提取了一些特征，并以离线(脱机)的方式训练单类支持向量机，以利用服务器或服务器集群的更高计算能力。

**关键词：** Android 安全；计算机安全；数据挖掘；机器学习；支持向量机；恶意软件检测
**中图分类号：** TP309；TP181 **文献标识码：** A **文章编号：** 0372-2112（2021）08-2502-07
**电子学报 URL：** http://www.ejournal.org.cn **DOI：** 10.12263/j.issn.0372-2112.2021.10.007

## Android Malware Detection Method Based on Machine Learning

XU Wan-peng
(*School of Computer Science and Technology, China University of Mining and Technology, Xuzhou,Jiangsu 221116,China*)

**Abstract:** With the recent emergence of operating system with built-in Android subsystems (such as HarmonyOS, Windows11) and the rising ubiquity of using these operating system in various industries, there is a possibility associated with malware sandbox escape. The problem of detecting such malware presents unique challenges due to the limited resources avalible and limited privileges granted to the user, but also presents unique opportunity in the required metadata attached to each application. In this article, I present a machine learning-based system for the detection of malware on Android devices. My system extracts a number of features and trains a One-Class Support Vector Machine in an offline (off-device) manner, in order to leverage the higher computing power of a server or cluster of servers.

**Key words:** android security;computer security;data mining;machine learning;support vector machines;malware detection

## 1 引言

以智能手机为代表的智能设备已成为人们日常生活中不可或缺的工具。根据 StatCounter 的最新数据[1]，谷歌 Android 系统在移动平台的市场占比达 72.44%，随着新型操作系统的发展，有些系统选择了内置 Android 系统。而 Android 恶意软件的数量正随着互联网应用的爆发而急剧增长[2]，严重侵犯用户权益、 影响产业发展，甚至威胁国家安全，若不对这些恶意软件进行有效的检测，它们的威胁有可能通过沙盒逃逸从移动设备转移至桌面设备，造成进一步的破坏。

使用基于机器学习的分类器检测恶意软件有两个主要的挑战:首先，给定一个应用程序，我们必须提取该应用程序的某种特征表示; 其次，我们有一个几乎完全良性的数据集，所以我们必须选择一个只能在一个类上训练的分类器。为了解决第一个问题，我们提取了一个异构特征集(在第 4 节中描述)，并使用多个核独立处理每个特征。(在第 5 节中描述)。为了解决第二个问题，我们使用了一个单类支持向量机[3]，并且只使用良性应用程序来训它。

本文的组织如下:在第 2 节中，我们讨论了相关工作;在第 3 节中，我们概述了我们的方法;在第 4

收稿日期：2021年10月23日；修回日期：2021年10月25日；责任编辑：孙瑶
基金项目：国家自然科学基金( No.61802194,No.61902190)；江苏省高等学校自然科学研究( No.17KJB520015,No.19KJB520040)；审计信息与工程协同创新中心资助项目( No.18CICA06)

节中，我们介绍了我们从数据集中提取的各种特征;在第 5 节中，我们给出了我们用来训练模型的核;在第 6 节中，我们讨论了我们的实验结果;最后，在第 7 节中，我们分析了我们的结果，并提出了未来可能的研究方向。

## 2 相关工作

在检测 Andriod 系统的恶意软件的问题上已经有了大量的工作。有几种方法[4,5,6]监视应用程序的功耗，并报告异常功耗。其他的[7,8]监视系统会调用并尝试检测异常的系统调用模式。其他方法使用与已知恶意软件[9]或其他启发式方法[10]进行更传统的比较。

更广泛的恶意软件检测领域有更广泛的方法。传统的静态分析方法，如文献[11,12]，侧重于根据程序代码将程序与已知恶意软件进行比较，寻找签名或使用其他启发式方法。其他方法[13,14,15]侧重于使用机器学习和数据挖掘方法检测恶意软件。在文献[15]中，Tesauro 等人训练了一个神经网络，以检测基于字节字符串三角形的引导扇区病毒。Schultz 等人[14]比较了三种基于三个特征训练的机器学习算法：程序进行的 DLL 和系统调用、程序二进制中的字符串以及二进制的原始十六进制表示形式。在文献[13]中，Kolter 和 Maloof 在字节串 n-grams 上训练了几种机器学习算法。

## 3 方法

我们使用开源项目 Androguard[16]从打包的 Android 应用程序(APKs)中提取特性。然后，我们通过 scikit-learn 框架[17]，使用这些提取的特征来训练单类向量机[3]，该框架为 LIBSVM[18]提供了方便的接口。

文献[3]的主要思想是生成一个分类器，该分类器将大部分训练数据分类为正数据，并且仅当训练或测试数据与训练数据有足够的差异时，才将其分类为负数据，这样非常适合我们的目的，因为良性 Android 应用程序比恶意 Android 应用程序更容易访问。作为支持向量机，单类支持向量机是基于约束二次优化问题的高维特征空间中的线性分类器：

$$\min_{\omega,\vec{\xi},\rho} \frac{1}{2}\|\omega\|^2 + \frac{1}{vl}\sum_i \xi_i - \rho \qquad (1)$$

$$\text{such that } \big(\omega \cdot \phi(x_i)\big) \geq \rho - \xi_i, \xi_i \geq 0.$$

这样就给出了分类器函数

$$f(x) = sign\big(\omega \cdot \phi(x)\big) - \rho \qquad (2)$$

这里，$\phi(x)$是一个特征空间变换，目的是将$x$的特征映射到一个类是线性可分离的空间，而$\omega$是该空间的某个元素。然后，优化问题(1)试图找到一个线性分离，使训练数据与原点分离，且与原点的距离最大。

我们不直接提供$\phi(\cdot)$，而是提供一个核，

$$k(x,y) = \langle \phi(x), \phi(y) \rangle \qquad (3)$$

核的必要性质，以及我们对$k(\cdot,\cdot)$的特殊选择在第五节中描述。

## 4 特征提取

为了应用于任意的核，我们必须首先从应用程序中提取重要的特征。Android 应用程序被打包为 APK 文件，这类似于标准的 Java jar 文件。我们使用开源项目 Androguard[16]来处理这些文件并提取特征。Androguard 提供了一个页面友好的，用于分析和逆向工程 Android 应用程序。

一旦我们提取了请求的权限列表，我们将它们分为两组:标准内置权限和非标准权限。对于标准权限，我们生成一个二进制向量，其中每个条目对应一个内置权限，如果应用程序请求该权限，则将其设置为 1，否则设置为 0。

对于非标准权限，我们将字符串分成三个部分:前缀(通常是“com”或“org”)、组织和产品部分以及权限名称。我们忽略了“android”或“permission”这些无处不在的词。

### 4.1 权限

每个 APK 都必须包含一个清单文件，其中包括请求访问 Android 操作系统某些受限制元素的权限。这些元素包括访问各种硬件设备(如 GPS、相机)、操作系统的敏感特性(如联系人)，以及访问其他应用程序的某些暴露部分。例如，权限“android.permission.”“INTERNET”请求访问 INTERNET 的权利，以及“android.permission”。“READ CONTACTS”请求访问用户电话联系人数据库[19,36]的权限。

### 4.2 控制流图

对于给定应用程序中的每个方法，我们从方法的原始字节码中提取一个控制流图(CFG)。CFG 是一个程序的抽象表示，其中顶点表示非跳转指令的原子块，边表示程序流程的可能路径。每个顶点都基于块中的最后一条指令进行标记，因为正是这条指令决定了程序如何离开块。例如，无条件跳转表

示为单个边;条件跳转表示为在同一起始节点上关联的两条边。但是，许多病毒使用变形技术生成具有相同语义但控制流[20]不同的代码的修改副本。为了对抗这些突变，我们应用文献[21]的图重写规则:

- 合并连续的指令块
- 将无条件跳转与它们跳转到的指令块合并
- 合并连续条件跳转

这些重写或删减有一个额外的优点，即在不破坏有关程序流的重要语义信息的情况下减小了所提取的图的大小。

一旦我们提取并处理了这些图，我们就会丢弃那些具有 5 个或更少节点的图，因为只有相对较少的可能的 5 个或更少节点的 cfg(因此它们编码很少的语义信息)，而丢弃它们可以大大加快处理速度。然后我们将这些图合并成一个大型的不相连的图(有时这样的图被称为森林)。

# 5 核

本节将专门描述我们方法中使用的核。对于我们使用的每一个核，我们都希望保证一个名为 Mercer 条件的属性，或称正确定性。为了提高可读性，在附录中给出了 Mercer 核理论的概述，以及定理 5.2 的证明和更有效的字符串核公式。

## 5.1　二进制向量上的核

给定一个二元向量的特征空间（即对于某些$n$的$\{0,1\}^n$），如文献[22]中所讨论的那样，有许多可能的核。特别是，合取、反析取和当且仅当的标准布尔运算适用于直接的 Mercer 核。我们使用当且仅当内核：

**定义 1** 当且仅当内核中，两个二进制向量的$k_{\leftrightarrow}$由下式确定：

$$k_{\leftrightarrow}(A,B)=\sum_{i=1}^{n} a_i \leftrightarrow b_i\,, \tag{4}$$

其中,如果$x$和$y$的值相同，则$x \leftrightarrow y$为 1。也就是说，当且仅当内核返回等价位数。

**定理 5.1(文献[22])** $k_{\leftrightarrow}$是 Mercer 核。

## 5.2　字符串上的核

对于我们的字符串特征（第 4.1 节），我们使用了文献[23]中的内核，它是在一些字母表$\sum$上的任意长度字符串上定义的。其基本思想是计算字符串之间公共子序列的数量，并根据原始字符串中子序列的长度进行加权。

**定义 2** 字符串$s$是从字母表$\sum$中提取的符号的有序列表。我们用$\sum^n$表示所有长度为$n$的字符串的集合，用$\sum^*$表示所有字符串（任意长度）的集合。两个字符串$s$和$t$的串联表示为$s \circ t$。对于$s$的子串$t$，从$s_i$开始到$s_j$结束表示为$s[i:j]$。给定一个字符串$s$，子序列$u$是满足条件$u_j = s_{i_j}$,其中$1 \le i_j < i_j + 1 \le |s|$的任意字符串，我们将这些索引记为$\vec{\imath} = \{i_1, \dots, i_{|u|}\}$，如此即可将子序列表示为$u = s[\vec{\imath}]$，我们将子序列的长度表示为$l(\vec{\imath}) = i_{|u|} - i_1 + 1$; 也就是说，长度被测量为$s$中子序列开始和结束之间的距离，而不是$u$中的符号数

将$n^{th}$子序列内核定义为

$$k_n(s,t)=\sum_{u\in\Sigma^n}\sum_{\vec{\imath}:u=s[\vec{\imath}]}\sum_{\vec{\jmath}:u=t[\vec{\jmath}]}\lambda^{l(\vec{\imath})+l(\vec{\jmath})}$$

那么，$k_n(s,t)$测量$s$和$t$之间的公共$n$符号子序列的数量，并根据它们的长度进行加权。如果$s=t$，那么它们将具有长度$n$的每一个子序列，从而对所有$n \le min(|s|,|t|)$，$k_n(s,t)$都是最大化的。另一方面，如果$s$和$t$最大不同（即它们没有共同的符号），那么对所有$n \le N$，$k_n(s,t)=0$。我们可以将不同$n$的$k_n(\cdot,\cdot)$与一个简单的以$n$为权重的和结合起来：

$$k_{str}(s,t)=\sum_{n=0}^{N}\mu(n)k_n(s,t)$$

为了简单起见，我们对所有$n \le N$使用一个简单的加权函数$\mu(n)=1$。

**定理 5.2** $k_{str}(\cdot,\cdot)$是 Mercer 核。

## 5.3　图上的核

给定带标签的有向图的特征空间，我们使用文献[24]中的 Weisfeiler-Lehman 子树核，它基于图同构的 Weisfeiler-Lehman 检验[25]。在文献[24]中，Shervashidze 等人提出了一个图核的一般框架，其中 Weisfeiler-Lehman 图同构测试用于构造一系列标记图，每个标记图都使用其他一些图核进行比较。这种方法允许以一种提高其辨别能力的方式对非常简单（或者不是非常有用）的图形内核应用。

**定义 3** 给定一个带标签的图$G=(V,E,l_0)$，其中$l_0(v), v \in V$是取自某个字母表$\Sigma$的标签，高度为 h 的 Weisfeiler-Lehman 序列是图的序列。

$$\{G_0, G_1, \dots, G_h\} = \{(V,E,l_0),(V,E,l_1),\dots,(V,E,l_h)\}$$

其中$l_1, \dots, l_h$使用算法 1 构造。

算法中的函数$f(\cdot)$是某种映射$f:\Sigma^*\to\Sigma$，它满足当且仅当$s_i(v)=s_i(u)$时$f(s_i(v))=f(s_i(u))$。压缩函数$f(\cdot)$旨在使算法更节省空间，从而使标签不会随着$i$呈指数增长，文献[24]的作者建议维护一个全局计数器，记录调用$f(\cdot)$的不同字符串的数量。因为我们的实现是设计成分布式的，所以这种方法是不可行的。我们取$f(\cdot)$作为恒等函数。

**算法 1** Weisfeiler-Lehman 重标记

```
1: FOR i ← 0 到 h DO
2:     FOR ALL v ∈ V DO
3:         IF i = 0 THEN
4:             M_i(v) ← l_0(v)
5:         ELSE
6:             M_i(v) ← {l_{i-1}(v)|u ∈ N(v)}
7:                 ▷N(v) = {u|(v,u) ∈ E}是v的邻域集
8:             s_i(v) ← l_{i-1}(v) ∘ sort(M_i(v))
9:                 ▷排序M_i(v)，并将其元素连接至前一个标签
10:            l_i(v) ← f(s_i(v))
11:        END IF
12:    END FOR
13: END FOR
```

**定义 4** 给定一个图核$k_1(\cdot,\cdot)$，具有基核$k_1(\cdot,\cdot)$和$h$迭代的 Weisfeiler-Lehman 核定义如下：

$$k^{(h)}_{WL(k_1)}(G,G')=k_1(G_0,G'_0)+\cdots+k_1(G_h,G'_h), \quad (5)$$

其中$\{G_0,\ldots,G_h\}$和$\{G'_0,\ldots,G'_h\}$是$G$和$G'$的 Weisfeiler-Lehman 序列。

**定理 5.3(文献[24],定理 3)** 如果$k_1(\cdot,\cdot)$是图上的 Mercer 核，那么$k^{(h)}_{WL(k_1)}(\cdot,\cdot)$也是 Mercer 核。

**定义 5** 具有$h$迭代的 Weisfeiler-Lehman 子树核是一个具有计算$G$和$G'$之间匹配节点标签数量的基本内核的 Weisfeiler-Lehman 核：

$$k_g(G,G')=\sum_{v\in V}\sum_{v'\in V'}\delta\big(l(v),l(v')\big), \quad (6)$$

其中$\delta(a,b)$是狄拉克核，当$a=b$时为 1，否则为 0，$l(v)$是节点$v$的标签。

### 5.4 集合上的核

考虑任意基数的无序集的特征空间，其中从集合 X 中提取元素，我们有一个 Mercer 核$k_0(\cdot,\cdot)$；也就是说，对于某些 X，连同$k_0:X\times X\to\mathbb{R}$，有特征$x\in\mathcal{P}(X)$，我们希望从$k_0(\cdot,\cdot)$中构造一个 Mercer 核$k:\mathcal{P}(X)\times\mathcal{P}(X)\to\mathbb{R}$。

理想情况下，为了找到$k(x,y)$，我们应该找到$x$和$y$的最佳匹配，换言之，使$k_0(x_i,y_j)$最大的对$(x_i,y_j)$的集合。在文献[26]中，提出了一种简单的方法：

$$k_{match}(x,y)=\frac{1}{2}\frac{1}{|x|}\sum_{i=1}^{|x|}\max_j k_0(x_i,y_j)+\frac{1}{2}\frac{1}{|y|}\sum_{j=1}^{|y|}\max_i k(x_i,y_j). \quad (7)$$

我们的想法对每个$x_i\in x$是找到最匹配的$y_j\in y$，然后是每个$y_j$是找到最匹配的$x_i$，并对这些匹配中的每个$k_0(x_i,y_j)$求和。不幸的是，（7）已被证明不是 Mercer 内核。特别地，使用 max 运算不能保持正定性。作为替代方案，我们使用文献[27]的指数匹配内核：

$$k^{k_0}_{set}(x,y)=\frac{1}{|x|}\frac{1}{|y|}\sum_{i=1}^{|x|}\sum_{j=1}^{|y|}k_0(x_i,y_j)^p. \quad (8)$$

通过将$k_0(x_i,y_j)$提高到指数级，我们可以有效地赋予较大的值更多的权重。事实上，$\lim_{p\to\infty}k(\cdot,\cdot)=k_{match}(\cdot,\cdot)$。作为 Mercer 核的指数的和，$k(\cdot,\cdot)$本身就是 Mercer 核。

### 5.5 基于非标准权限的核

在第 IV-A 节中，我们注意到我们将非标准权限字符串分为三个部分：前缀、包含组织和/或产品名称的中间部分以及包含实际权限名称的后缀。我们将这三个部分视为独立的字符串集，并应用 V-D 部分的集核，然后将这三个值的平均值作为我们的核值（注意，一组 Mercer 核的平均值是这些核的正线性组合，因此是 Mercer 核本身）：

$$k_p(x,y)=\frac{k^{k_{str}}_{set}(x_0,y_0)+k^{k_{str}}_{set}(x_1,y_1)+k^{k_{str}}_{set}(x_2,y_2)}{3}, \quad (9)$$

其中$x_0$是前缀集，$x_1$是中间字符串集，$x_2$是与$x$的权限关联的权限名称集。

### 5.6 应用程序上的核

对前面每个内核的分析表明，它们倾向于更长或更大的输入：例如，通过比较比较向量的长度，$k_{\leftrightarrow}(\cdot,\cdot)$的最大值是$n$。类似地，字符串和图形内核都根据输入的大小执行求和。因此，我们应该使其正常化：

$$\hat{k}(s,t)=\frac{k(s,t)}{\sqrt{k(s,s)k(t,t)}}$$

然后，我们得到了最后一个内核，$k: X \times X \to \mathbb{R}$，其中$X$是可能的 Android 应用程序的空间。

让$v_x$、$p_x$和$g_x$分别作为应用程序$x$中提取的位向量（第 4.1 节），权限字符串集数组（第 4.1 节）和控制流图（第 4.2 节）。那么，我们有：

$$k(x,y)=\hat{k}_{\leftrightarrow}(v_x,v_y)+\hat{k}_p(p_x,p_y)+\hat{k}_g(g_x,g_y), \quad (10)$$

这是 Mercer 核的正线性组合，因此是 Mercer 核。

## 6 实验结果

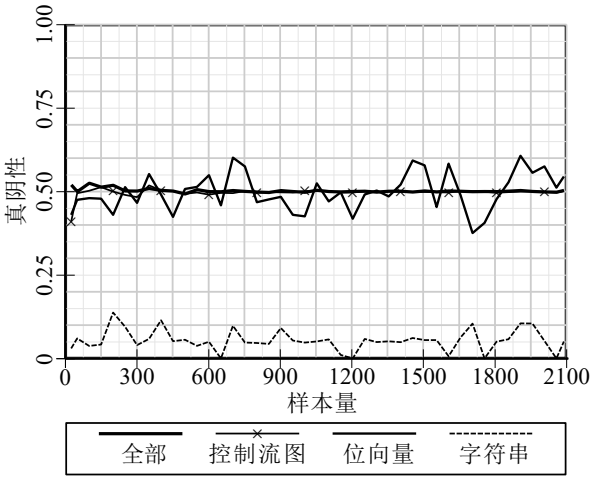


图 1　样本量与真阴性

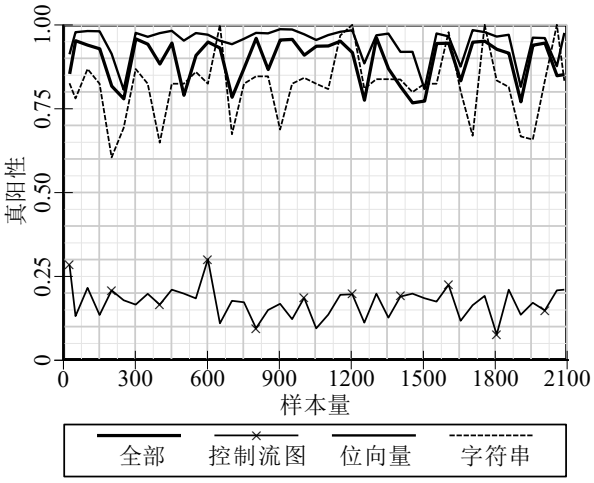


图 2　样本量与真阳性

我们针对 2081 个良性和 91 个恶意 Android 应用程序的集合测试了我们的系统。对于每个数据点，我们选择了训练（良性）应用程序的随机子集，并执行 k 倍交叉验证。我们对每个数据点做了四次，并对结果进行平均。除了完整的核外，我们还分别针对每个单独的核进行训练。图 1 和图 2 分别显示了真阴性（良性分类为良性）和真阳性（恶意软件分类为恶意软件）与样本量的关系。

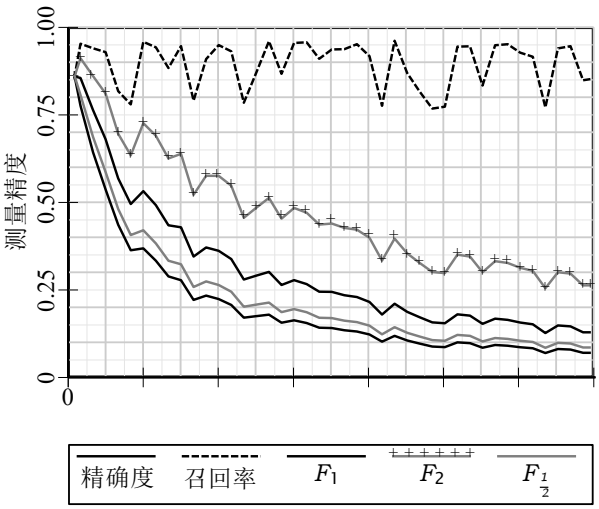


图 3　样本量与测量精度

图 3 显示了系统准确度的各种度量：精度、召回率、$F_1$度量（精度和召回率的调和平均值）、$F_2$度量（与$F_1$类似，但召回率加权为精度的两倍）和$F_{\frac{1}{2}}$（与$F_1$类似，但精度加权为召回率的两倍）。请注意，随着良性样本的增加（以及$F$度量值的增加），增加的良性样本量与固定的恶意样本量相结合会导致精度降低。

这些数据表明我们的系统有一些有趣的特性。首先，完整核的性能永远不会超过比较内置权限的位向量核。位向量核似乎将特征空间分为两部分：请求恶意软件所需权限的应用程序和不请求权限的应用程序。虽然它正确地分类了大多数恶意软件，但大约一半的良性应用程序请求相同的权限，因此位向量核和通常的完整核将其分类为恶意软件。

其次，字符串核似乎没有什么辨别能力，通常将几乎所有东西都归类为恶意软件。这可能是因为根据定义，作为其输入的非标准权限并不常见。这导致特征空间非常稀疏，通常为零，因此大部分训练数据位于原点，使得 1C-SVM（单类支持向量机）无法将训练数据与原点分离。

最后，图形核显示了一些不寻常的行为：它的假阴性率高于真阴性率。这表明，恶意软件距离原始数据的距离远远超过了训练数据的很大一部分，使它们位于 1C-SVM 训练的分离超平面的错误一侧。

# 7 总结

我们提出了一种新的基于机器学习的 Android 操作系统恶意软件检测系统。我们的系统显示了令人满意的结果，它具有非常低的假阴性率，但其高假阳性率也有很大的改进空间。有许多可能的改进可以研究。

## 7.1 特性

我们的系统仅限于输入应用程序的权限（内置和非标准）和 CFG。还有许多其他潜在的信息来源丰富的功能。例如，清单文件中还有其他元数据条目包含请求的权限。程序代码本身中还有许多潜在的特性来源，例如常量声明和方法名。

此外，可以改进当前的功能。特别是，我们提取 CFG 的方法放弃了代码中最初存在的大部分信息：我们仅根据 CFG 中表示的块的最后一条指令来标记节点。我们可以根据块中的所有指令进行标记。我们也只有一小部分标签，这些标签可以扩展到包含更多关于当前指令类型的详细信息（例如算术运算、内存访问等）。这样的区别将导致更健壮的标签集，这可能会增加图形核的功能，因为它基于图形标签。

## 7.2 核

除了新特性，以及这些特性所需要的任何新核之外，我们为现有特性选择的核可能还有改进的余地。同样，CFG 特性是最有可能改进的候选特性。当前系统将提取的图形集合视为单个断开连接的图形。这种处理方法可能会通过对这些图形进行总体比较而不是对所有图形进行比较，从而放弃鉴别能力。因此，将文献[24]中的图形核与文献[27]甚至文献[28]中的集合核相结合，可以得到功能更强大的核。不幸的是，每个应用程序中的大量方法使得文献[27]中提出的核的二次时间复杂度不切实际，并且文献[28]中的核在向量集的特征空间上，每个向量集来自相同的向量空间。因为文献[24]的图形核实质上是将图形转换为向量，所以它似乎与文献[28]非常匹配。然而，两个图将被转换为从两个不同的向量空间绘制的向量，并且将这些向量有效地投影到并集或交集空间已被证明是不平凡的。

## 7.3 模式

最后，我们使用单类支持向量机，因为我们的良性示例远远多于恶意示例。然而，我们确实有一些恶意的例子，因此半监督方法可能更强大。例如，在文献[29]中，作者将半监督环境中的新颖性检测问题简化为 Neyman-Pearson 分类问题[30]。这可以与文献[31]相结合，在文献[31]中，作者描述了一种培训 Neyman-Pearson 分类支持向量机的方法。这种半监督分类器可能比 1C-SVM 具有更大的鉴别能力。

# 附录

如第三节所述，我们希望解决约束二次优化问题（1）。为此，（1）转化为拉格朗日函数[3]：

$$L\left(\omega,\vec{\xi},\rho,\vec{\alpha},\vec{\beta}\right)=\frac{1}{2}\|\omega\|^2+\frac{1}{vl}\sum_i\xi_i-\rho-\sum_i\alpha_i\left(\left(\omega\cdot\phi(x_i)\right)-\rho+\xi_i\right)-\sum_i\beta_i\xi_i, \tag{11}$$

这为（2）提供了一种新的形式：

$$f(x)=sign\left(\sum_i\alpha_i k(x_i,x)-\rho\right) \tag{12}$$

由此，我们可以导出一个双目标函数[3]，

$$\min_{\vec{\alpha}}\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j k\left(x_i,x_j\right) \tag{13}$$

$$\text{such that } 0\le\alpha\le\frac{1}{vl},\sum_i\alpha_i=1.$$

然而，为了保证对偶函数（13）有解，我们必须对$k(\cdot,\cdot)$进行某些限制。特别是，如果核不是 Mercer 核（定义见下文），则可能$x_i$存在这样的情况：双目标函数可以变得任意大，因此约束二次优化问题没有解[32]。

**1 Mercer核**

**定义 6** 如果函数$k:X\times X\to\mathbb{R}$是对称的（$k(x,y)=k(y,x)$），并且对任意的$n\in\mathbb{N},x_i\in X,c_i\in\mathbb{R}$，都有

$$\sum_{i=1}^{n}\sum_{j=1}^{n}c_i c_j k\left(x_i,x_j\right)\ge 0 \tag{14}$$

称它为（重值）正定核。

表达这一要求的另一种方式是，由$K_{ij}=K(x_i,x_j)$定义的矩阵$K$是对称的，其所有特征值都是非负的。注意，如果对所有$x_i,x_j\in X$，都有$k(x_i,x_j)\ge$

0且对称，（14）保持不变。

正定核的理论主要基于 Mercer[33]的工作，因此要求核是正定的通常称为 Mercer 条件，这种核称为 Mercer 核。

值得注意的是，有关 Mercer 内核组合的几个关键特性：

**定义 7** 给一个集合$X$如果对所有的$\lambda \geq 0$都有$\{\lambda x | x \in X\} X \subseteq X$，那么称其为锥。

一个集合如果对每个$(x, y) \in X$,线段都连接了$x$和$y$且$\overrightarrow{xy} \subseteq X$，那么称其为凸集。

**定理 1（文献[32]，1.11）** 如果$X$是一个非空集，那么$X \times X$上所有 Mercer 核的集合就是一个凸锥。

**推论 1** 如果$k_1, k_2: X \times X \rightarrow \mathbb{R}$是 Mercer 核，那么$k_+(x, y) = \lambda_1 k_1(x, y) + \lambda_2 k_2(x, y)$也是 Mercer 核。换句话说，在正线性组合下，$X \times X$上的 Mercer 核集是闭合的。

**证明** 考虑对某些$\rho \in [0,1]$，有$k_\rho(x, y) = \rho k_1(x, y) + (1-\rho) k_2(x, y)$。因为$X \times X$上的 Mercer 核集是凸的，所以$k_\rho(x, y)$是 Mercer 核。然后，考虑核

$$k_{\lambda_1}(x, y) = \frac{\lambda_1}{\rho} k_1(x, y)\text{和}$$

$$k_{\lambda_2}(x, y) = \frac{\lambda_2}{1-\rho} k_2(x, y).$$

因为$X \times X$上的 Mercer 核集是一个锥，所以这两个都是 Mercer 核。把两者结合起来，我们得到

$$\begin{aligned} k_*(x, y) &= \rho \frac{\lambda_1}{\rho} k_1(x, y) + (1-\rho) \frac{\lambda_2}{1-\rho} k_2(x, y) \\ &= \lambda_1 k_1(x, y) + \lambda_2 k_2(x, y). \end{aligned}$$

证毕

**定理 2（文献[34]，1.12）** 如果$k_1, k_2: X \times X \rightarrow \mathbb{R}$是 Mercer 核，那么$k_\times(x, y) = k_1(x, y) k_2(x, y)$ 是 Mercer 核。

## 2 更多关于$k_{str}(\cdot,\cdot)$

$k_n(\cdot,\cdot)$的定义适用于需要$O(|\Sigma|^n)$时间的 naıve 算法，这显然是不可行的。通过构造$k_n(\cdot,\cdot)$的递归定义，我们可以将时间减少到$O(n|s||t|)$。

**定义 8** $k_n(\cdot,\cdot)$的递归版本使用了一种动态规划方法，该方法引入了许多中间计算，$k_i'$和$k_i''$,$1 \leq i \leq n$：

$$\begin{aligned} k_0'(s, t) &= 1 \text{ for all } s, t \\ k_i''(s, t) &= 0 \text{ if } min(|s|, |t|) < i \\ k_i'(s, t) &= 0 \text{ if } min(|s|, |t|) < i \\ k_i(s, t) &= 0 \text{ if } min(|s|, |t|) < i \end{aligned}$$

$$\begin{aligned} k_i''(s \circ x, t \circ y) &= \lambda k_i''(s \circ x, t) \text{ if } x \neq y \\ k_i''(s \circ x, t \circ x) &= \lambda(k_i''(s \circ x, t) + \lambda k_{i-1}'(s, t)) \\ k_i'(s \circ x, t) &= \lambda k_i'(s, t) + k_i''(s \circ x, t) \end{aligned}$$

然后，最后一行：

$$k_n(s \circ x, t) = k_n(s, t) + \sum_{j: t_j = x} k_{n-1}'(s, t[1: j-1]) \lambda^2$$

**定理 5. 2 证明** 考虑每个$u \in \Sigma^n$的特征空间变换

$$\phi_u(s) = \sum_{\vec{i}: u = s[\vec{i}]} \lambda^{l(\vec{i})}$$

它把每个字符串映射为所有出现子序列$u$的子字符串的加权长度。然后，因为这个特征空间是简单的实数，我们可以自然地定义它的内积：

$$\begin{aligned} \langle \phi_u(s), \phi_u(t) \rangle &= \sum_{\vec{i}: u = s[\vec{i}]} \lambda^{l(\vec{i})} \sum_{\vec{j}: u = t[\vec{j}]} \lambda^{l(\vec{j})} \\ &= \sum_{\vec{i}: u = s[\vec{i}]} \sum_{\vec{j}: u = t[\vec{j}]} \lambda^{l(\vec{i}) + l(\vec{j})} \end{aligned}$$

接着，我们可以定义一个由整个$u$上的笛卡尔积组成的完整的特征空间。这给出了内积

$$\begin{aligned} \langle \phi(s), \phi(t) \rangle &= \sum_{u \in \Sigma^n} \langle \phi_u(s), \phi_u(t) \rangle \\ &= \sum_{u \in \Sigma^n} \sum_{\vec{i}: u = s[\vec{i}]} \sum_{\vec{j}: u = t[\vec{j}]} \lambda^{l(\vec{i}) + l(\vec{j})} \\ &= k_n(s, t) \end{aligned}$$

所以$k_n(\cdot,\cdot)$是$\mathbb{R}^{|\Sigma^n|}$的内积，所以它是 Mercer 核。因此，$k_{str}(\cdot,\cdot)$是 Mercer 核的正加权有限和，因此也是 Mercer 核本身。

## 参考文献


[1] StatCounter.Mobile Operating System Market Share Worldwid[OL]. https://gs.statcounter.com/os-market-share/mobile/worldwide.

[2] 360 互联网安全中心.2021 年Q1 手机安全状况报告[OL]. https://zt.360.cn/1101061855.php?dtid=1101061451&did=211171564.

[3] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. Technical Report MSR-TR-99-87, Microsoft Research, 2000.

[4] Bryan Dixon, Yifei Jiang, Abhishek Jaiantilal, and Shivakant Mishra.Location based power analysis to detect malicious code in smartphones. In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, SPSM '11, pages 27–32, 2011

[5] Hahnsang Kim, Joshua Smith, and Kang G. Shin. Detecting energy-greedy anomalies and mobile malware variants. In Proceedings of the 6th international conference on Mobile systems, applications, and services, MobiSys '08, pages 239–252, 2008.

[6] Lei Liu, Guanhua Yan, Xinwen Zhang, and Songqing Chen. Virusmeter:

Preventing your cellphone from spies. In Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, RAID '09, pages 244–264, 2009.

[7] Burguera I, Zurutuza U, Nadjm-Tehrani S. Crowdroid: Behavior-Based Malware Detection System for Android[C]// 2011.

[8] Liang Xie, Xinwen Zhang, Jean-Pierre Seifert, and Sencun Zhu. pbmds: a behavior-based malware detection system for cellphone devices. In Proceedings of the third ACM conference on Wireless network security, WiSec '10, pages 37–48, 2010.

[9] Bose A, Hu X, Shin K G, et al. Behavioral detection of malware on mobile handsets[C]// International Conference on Mobile Systems. DBLP, 2008:225.

[10] Zhou Y, Zhi W, Wu Z, et al. Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets[J]. proceedings of annual network & distributed system security symposium, 2012.

[11] Twycross J, Williamson M M. Implementing and Testing a Virus Throttle[C]//USENIX Security Symposium. 2003, 285: 294.

[12] Raymond W. Lo, Karl N. Levitt, and Ronald A. Olsson. Mcf: a malicious code filter. Computers & Security, 14(6):541 – 566, 1995.

[13] J. Zico Kolter and Marcus A. Maloof. Learning to detect and classify malicious executables in the wild. J. Mach. Learn. Res., 7:2721–2744, December 2006.

[14] Matthew G. Schultz, Eleazar Eskin, Erez Zadok, and Salvatore J. Stolfo. Data mining methods for detection of new malicious executables. In Proceedings of the 2001 IEEE Symposium on Security and Privacy, S P' 01, pages 38–, Washington, DC, USA, 2001. IEEE Computer Society.

[15] G.J. Tesauro, J.O. Kephart, and G.B. Sorkin. Neural networks for computer virus recognition. IEEE Expert, 11(4):5 –6, aug 1996.

[16] Anthony Desnos. androguard. https://code.google.com/p/androguard/.

[17] F. Pedregosa, G. V aroquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. V anderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Duchesnay E. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[18] Chih-Chung C, Chih-Jen L. LIBSVM: A library for support vector machines[J]. ACM Transactions on Intelligent Systems and Technology, 2011, 2(3):1-39.

[19] Google. Manifest.permission-android developers. http://developer.android.com/reference/android/Manifest.permission.html.

[20] A. Walenstein, R. Mathur, M.R. Chouchane, and A. Lakhotia. Normalizing metamorphic malware using term rewriting. In Source Code Analysis and Manipulation, 2006. SCAM '06. Sixth IEEE International Workshop on, pages 75–84, 2006.

[21] Bonfante G, Kaczmarek M, Marion J Y. Architecture of a Morphological Malware Detector[J]. Journal in Computer Virology, 2009, 5(3).

[22] Francesca Odone, Annalisa Barla, and Alessandro V erri. Building kernels from binary strings for image matching. IEEE Transactions on Image Processing, 14(2), 2005.

[23] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. Journal of Machine Learning Research, 2, 2002.

[24] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. Journal of Machine Learning Research, 12, 2011.

[25] B. Weisfeiler and A. A. Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. Nauchno-Technicheskaya Informatsia, 9, 1968.

[26] Christian Wallraven, Barbara Caputo, and Arnulf Graf. Recognition with local features: the kernel recipe. In IEEE International Conference on Computer Vision (ICCV), pages 257–264, 2003.

[27] Siwei Lyu. Mercer kernels for object recognition with local features. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[28] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Efficient learning with sets of features. Journal of Machine Learning Research, 8, 2007.

[29] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Semi-supervised novelty detection. Journal of Machine Learning Research, 11, 2010.

[30] Clayton Scott and Robert Nowak. A neyman-pearson approach to statistical learning. IEEE Transactions on Information Theory, 51(11), 2005.

[31] Davenport M A, Baraniuk R G, Scott C D. Tuning Support Vector Machines for Minimax and Neyman-Pearson Classification[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2010, 32(10):1888-98.

[32] Burges C. A Tutorial on Support Vector Machines for Pattern Recognition[J]. Data Mining and Knowledge Discovery, 1998, 2(2):121-167.

[33] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character, 209:415 – 446, 1909.

[34] Berg C, Christensen J P R, Ressel P. Harmonic analysis on semigroups: theory of positive definite and related functions[J]. 1984.

[35] Seveniruby．基于appium的app自动遍历工具［OL］．https://github.com/seveniruby/AppCrawler,2020- 03- 01．

[36] 张鹏,牛少彰,黄如强.基于资源签名的Android应用相似性快速检测方法[J].电子学报,2019,47(09):1913-1918.
ZHANG Peng, NIU Shao-zhang, HUANG Ru-qiang. A fast and resource-based detection approach of similar android application[J]. Acta Electronica Sinica, 2019, 47 (9) : 1913－1918．(in Chinese)

## 作者简介

**许万鹏** 男，2001 年生，黑龙江人. 中国矿业大学信息安全专业在读本科生，研究方向为机器学习理论.

E-mail: xwp@cumt.edu.cn