**算法源代码说明文档模板**

修订记录

| 日期 | 修订版本 | 修改章节 | 修改描述 | 作者 |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1. 原理介绍 4](#_Toc512525127)

[2. 总流程图 4](#_Toc512525128)

[3. 分模块介绍 4](#_Toc512525129)

[3.1 逻辑模块1: 4](#_Toc512525130)

[3.1.1 代码 4](#_Toc512525131)

[3.1.2 流程图 5](#_Toc512525132)

[3.1.3 详细说明 5](#_Toc512525133)

[3.2 接口模块1: 5](#_Toc512525134)

[3.2.1 代码 5](#_Toc512525135)

[3.2.2 功能描述 5](#_Toc512525136)

[3.2.3 接口描述 5](#_Toc512525137)

[3.2.4 核心原理 6](#_Toc512525138)

# 原理介绍

MalGAN (Malware Generative Adversarial Network) 是一种生成式对抗网络用于产生对抗病毒样本来免疫基于机器学习的查杀系统。这个对抗网络被训练，用来最小化对抗样本被检测机检测的概率。

# 总流程

具体步骤如下：

1. 从virusshare, malmr 等网站获取大量的PE格式的病毒样本和非恶意程序样本
2. 安装并配置cuckoo沙盒的环境

3）使用cuckoo沙盒运行并提取详细的所有API使用情况

1. 对API使用数据进行重要性权重排序，筛选并精简输入维度

5.）将得到M 维的特征向量输入到神经网络

6）训练调参

7）对各种分类器进行实验，比较其有效性和迁移性

8）测试在可以重复学习的黑盒检测器上的效果

9）优化降低检测率

# 分模块介绍

## 逻辑模块1:

### 代码

|  |
| --- |
| 核心代码  # Select a random batch of malware examples idx = np.random.randint(0, xtrain\_mal.shape[0], batch\_size) xmal\_batch = xtrain\_mal[idx] noise = np.random.uniform(0, 1, (batch\_size, self.z\_dims)) idx = np.random.randint(0, xmal\_batch.shape[0], batch\_size) xben\_batch = xtrain\_ben[idx] yben\_batch = ytrain\_ben\_blackbox[idx]  # Generate a batch of new malware examples gen\_examples = self.generator.predict([xmal\_batch, noise]) ymal\_batch = self.blackbox\_detector.predict(np.ones(gen\_examples. shape)\*(gen\_examples > 0.5))  # Train the substitute\_detector d\_loss\_real = self.substitute\_detector.train\_on\_batch(gen\_examples, ymal\_batch) d\_loss\_fake = self.substitute\_detector.train\_on\_batch(xben\_batch, yben\_batch) d\_loss = 0.5 \* np.add(d\_loss\_real, d\_loss\_fake)  # Train Generator idx = np.random.randint(0, xtrain\_mal.shape[0], batch\_size) xmal\_batch = xtrain\_mal[idx] noise = np.random.uniform(0, 1, (batch\_size, self.z\_dims))  # Train the generator g\_loss = self.combined.train\_on\_batch([xmal\_batch, noise], np.zeros((batch\_size, 1))) |

### 流程图

MalGAN的训练过程

1: 当算法未收敛

2: 采样生成小批量malware样本M

3: generator输入M生成对抗样本M’

4: 采样生成小批量benign样本B

5: 利用blcak-box detector对M’及B打上标签

6: 利用梯度下降法更新substitute detector的参数

7: 利用梯度下降法更新generator的参数

8: 结束循环

### 详细说明

Generator是一个生成malware的网络，它接收一个随机的噪声z，通过这个噪声生成对抗malware样本，记做G(z)。

Substitute Detector是一个判别网络，判别一张图片是不是“真实的”。它的输入参数是x，x代表一张图片，输出D（x）代表x为malware的概率，如果为1，就代表100%是malware，而输出为0，就代表不可能是malware。

在训练过程中，生成网络G的目标就是尽量生成对抗样本去欺骗判别网络D。而D的目标就是尽量把G生成的样本和真实的样本分别开来。这样，G和D构成了一个动态的“博弈过程”。

## 接口模块1:

### 代码

|  |
| --- |
| 核心代码  def \_\_init\_\_(self, blackbox='RF', same\_train\_data=1, filename='data.npz'):  self.apifeature\_dims = 128  self.z\_dims = 20  self.hide\_layers = 256  self.generator\_layers = [self.apifeature\_dims+self.z\_dims, self.hide\_layers, self.apifeature\_dims]  self.substitute\_detector\_layers = [self.apifeature\_dims, self.hide\_layers, 1]  self.blackbox = blackbox # RF LR SVM MLP VOTE  self.same\_train\_data = same\_train\_data # MalGAN and the black-boxdetector are trained on same or different training sets  self.filename = filename |

### 功能描述

apifeature\_dims：决定了输入特征的维数，默认为128；

噪声维数z\_dims：决定了输入generator的噪声维数，默认为20；

隐层维数hide\_layers：决定了generator以及subsitite detector的隐层节点数；

blackbox：决定了选用的分类器类型，可选参数RF、LR、SVM、MLP以及多种算法的加权组合VOTE；

same\_train\_data：决定了black detector以及substitude detector是否使用相同的训练数据；

filename：决定了训练数据文件名，训练数据为二维矩阵，其中每一行代表一个样本，每一列代表不同特征。

### 接口描述

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | In/Out/parameter | Size/precision | Description/Value |
| apifeature\_dims | in | int | 输入特征维数 |
| z\_dims | in | int | 输入噪声维数 |
| hide\_layers | In | int | 隐层节点数 |
| blackbox | in | string | blackbox detector名 |
| same\_train\_data | in | bool | 是否用相同训练集 |
| filename | in | string | 数据文件名 |

### 核心原理

1）generator以及substitute detector的结构由输入参数apifeature\_dims、z\_dims以及hide\_layers共同决定。generator的输入节点数等于apifeature\_dims+z\_dims，隐层节点数等于hide\_layers，输出节点数等于apifeature\_dims。substitute detector的输入节点数为apifeature\_dims，隐层节点数等于hide\_layers，输出节点为1，输出值范围0-1，决定了该样本是malware的概率。  
2）blackbox决定了选用的分类器类型，可选参数RF（随机森林）、LR（逻辑回归）、SVM（支持向量机）、MLP（多层感知器）以及多种算法的加权组合VOTE算法；

3）采用两种方式进行训练集和测试集的分割：第一种是将80%的数据集作为训练集，其余作为测试集，并且black detector以及substitude detector共用这部分训练集；另一种是将80%的训练集进一步对半分，分别作为black detector以及substitude detector的训练集。same\_train\_data等于1则使用第一种分割方法，等于0则使用第二种分割方法。

4）filename：决定了训练数据文件名，训练数据为二维矩阵，其中每一行代表一个样本，每一列代表不同特征。