# 基于机器学习的网络流量分析审计系统<br>部署文档


**郑宇森** **王鑫** **赵鸿宇**

**韩志鹏** **姜来** **喻路稀**


## 1 引言

为更好的指导部署与测试基于机器学习的网络流量分析审计系统，包括网络嗅探器 WireCat、数据库 MinIO、网络流量特征提取器、机器学习模型以及用于展示分析结果的网页，特编写《基于机器学习的网络流量分析审计系统部署文档》，分别提供本地环境部署与 docker 环境封装两种部署方式，以便本系统的用户能够更加方便地部署与使用本项目。

## 2 本地环境部署

### 2.1 部署说明

本部分主要说明项目部署时需要准备的系统环境要求、软件列表、安装流程。

#### 2.1.1 系统环境要求

本项目基于 GNU/Linux Ubuntu 发行版开发，运行环境如表 1所示。可通过如下命令安装所需依赖。

```
# 由于我们所使用环境已有一定的使用基础，因此此处所列的依赖可能并不完整，如若遇到缺少相应包或工具的问题，请自行安装
$ sudo apt install build-essential
$ sudo apt install qtbase5-dev $tchooser qt5-qmake qtbase5-dev-tools
$ sudo apt install libpcap-dev
$ sudo apt install make
$ sudo apt install tshark
$ pip install minio
$ pip install pandas
$ pip install scikit-learn
$ pip install torch
$ pip install streamlit
```

Table 1: 运行环境

| 环境/依赖 | 版本 |
|---|---|
| Ubuntu Desktop 64 位 | 20.04 |
| Qt | 5.9.0 |
| libpcap | 1.10.1 |
| GNU Make | 4.3 |

### 2.1.2 软件列表

本项目所需的软件如表 2所示。

Table 2: 软件列表

| 所属 | 软件 | 版本 |
|---|---|---|
| 第三方 | MinIO | 5.0.4 |
| 第三方 | streamlit | 1.22.0 |
| 开源库 | ddos | 1.0.0 |
| 开源库 | kdd99_feature_extractor | – |
| 第三方 | dsniff | – |
| 项目软件 | WireCat | – |
| 项目软件 | faashark.pt | – |

### 2.1.3 部署流程

为避免遇到各类安装失败的问题（如：服务间依赖关系、配置失败等），请按照推荐安装流程进行安装。具体流程如图 1所示

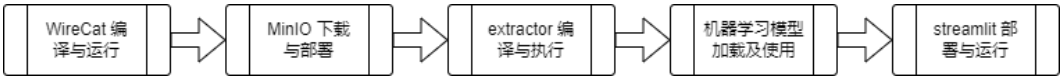


Figure 1: 部署流程

## 2.2 系统部署

本部分主要说明项目在本地进行部署时需要的依赖环境及项目运行的步骤及注意事项。下面按照上一节中部署流程的先后顺序对系统的安装步骤进行描述。

### 2.2.1 WireCat 编译

假设你所在目录为你所想部署本系统的目录下，请打开终端，依次运行如下命令：

```
# 克隆项目到本地
$ git clone https://github.com/ysyszheng/TrafficCat.git
$ cd TrafficCat
# 编译WireCat(GUI模式+CLI模式)
$ cd build
$ bash build.sh clsniff.sh
# 编译完成后注意返回TrafficCat目录
$ cd ..
```

至此，WireCat 的两种运行模式均已编译完成，可使用命令 `sudo ./build/bin/traffic` 运行 GUI 模式，使用命令 `sudo ./build/bin/clsniff` 运行 CLI 模式。

### 2.2.2 MinIO 部署

通过下述步骤在本机完成 MinIO 的部署：

1. **下载和安装**：使用以下命令下载并安装 MinIO 服务器。

```
# create a new directory
mkdir ~/minio
cd ~/minio
# Download the server
```

```
wget https://dl.minio.org.cn/server/minio/release/linux-amd64/minio
# Copy the downloaded minio file to the specified folder and give it executable permissions
sudo chmod +x ~/minio/minio
```

2. **运行 MinIO**：使用如下命令在本地运行 MinIO。命令输入完成后，在浏览器中访问 `http://127.0.0.1:9002`。在用户名和密码字段中输入默认的用户名和密码（minioadmin/minioadmin）即可登录系统。

```
# Create the MinIO storage directory
mkdir -p ~/minio/data
# create a log file
touch ~/minio/minio.log
# Start MinIO, specifying the storage directory and access address:
nohup ~/minio/minio server --address :9001 --console-address :9002 ~/minio/data >~/minio/minio.log 2>&1 &
```

为了在系统重启时自动启动 MinIO 服务，你可以把它配置为一个系统服务。可以通过下述步骤配置自动启动服务（可选，在本项目中并不作要求）：

1. **创建 MinIO 配置文件**：

```
# Create the configuration file /etc/default/minio
sudo gedit /etc/default/minio
```

2. **将以下配置信息写进文件**：写入完成后保存并退出配置文件。

```
# Specify the data storage directory (Note: The directory must exist and have appropriate permissions)
MINIO_VOLUMES="~/minio/data"

# Listening port
MINIO_OPTS="--address :9001 --console-address :9002"

# Specify the default username and password, where the username must be longer than 3 characters
MINIO_ROOT_USER="minioadmin"
MINIO_ROOT_PASSWORD="minioadmin"
```

3. **创建 MinIO 服务文件**：

```
sudo gedit /usr/lib/systemd/system/minio.service
```

4. **将以下配置信息写入服务文件中**：写入完成后保存并退出服务文件。

```
[Unit]
Description=MinIO
Documentation=https://docs.min.io
Wants=network-online.target
After=network-online.target
AssertFileIsExecutable=~/minio/minio

[Service]
WorkingDirectory=~
ProtectProc=invisible

```

```
# Reference the configuration file created in the previous step
EnvironmentFile=/etc/default/minio

ExecStartPre=/bin/bash -c "if [ -z \"${MINIO_VOLUMES}\" ]; then echo \"Variable MINIO_VOLUMES not set in /etc/default/minio\"; exit 1; fi"
ExecStart=~/minio/minio server $MINIO_OPTS $MINIO_VOLUMES

# Always allow systemd to restart the service
Restart=always

# Specify the maximum number of file descriptors the process can open (1M)
LimitNOFILE=1048576

# Specify the maximum number of threads the process can create
TasksMax=infinity

# Disable timeout logic until the process is stopped
TimeoutStopSec=infinity
SendSIGKILL=no
SuccessExitStatus=0

[Install]
WantedBy=multi-user.target
Alias=minio.service
```

5. **使服务配置生效**:

```
# Reload the service configuration file
systemctl daemon-reload

# Set the service to start on boot
systemctl enable minio

# Start the MinIO service
systemctl start minio

# Check the current status of the MinIO service
systemctl status minio
```

如果 MinIO 服务成功启动，你可以在你的浏览器中访问 MinIO 控制台，网址是 `http://127.0.0.1:9002`，用之前配置的用户名和密码登录系统。

### 2.2.3 extractor 编译与执行

我们使用 Github 上的开源软件 kdd99-feature-extractor 对网络流量进行特征提取。假设你现在所处目录为 TrafficCat(主目录下)，依次执行以下命令:

```
# Create a folder to temporal build files
cd kdd/kdd99_cmake
mkdir build-files

# Enter in the folder and compile the cache
cd build-files
cmake -DCMAKE_BUILD_TYPE=Debug -G "CodeBlocks - Unix Makefiles" ..

# Exit the folder of build cache and compile the project
cd ..
cmake --build ./build-files --target kdd99extractor -- -j 4

```

```
# Path to compiled project is:
build-files/src/kdd99extractor

# mv to TrafficCat/kdd/ is:
mv build-files/src/kdd99extractor ../
```

至此，可执行文件 kdd99extractor 编译完成，可在 TrafficCat 目录下运行命令 `kdd/kdd99extractor data/traffic.pcap > data/extractor.txt` 对 traffic.pcap 文件进行处理，提取出特征文件 extractor.txt。

#### 2.2.4 机器学习模型加载及使用

我们将训练好的机器学习模型放在了 kdd/inference/model 目录下，在 kdd/inference/-main.py 中 load 改模型即可进行流量种类判定。

#### 2.2.5 streamlit 部署与运行

Streamlit 提供了一种名为 Streamlit Sharing 的服务，可以用来免费部署公开的 GitHub 仓库中的 Streamlit 应用。只需要在 Streamlit Sharing 的网站上输入 GitHub 仓库的 URL，Streamlit 就会为应用创建一个可公开访问的 URL。我们正是采用了 Streamlit Sharing 的部署方法，以下是 streamlit shareing 一般的部署流程：

1. **创建 Streamlit 应用**：首先，在本地环境中创建并测试 Streamlit 应用，在 TrafficCat 目录下使用命令 `streamlit run web/dashboard.py`，确保其能够在本地环境中正确运行。
2. **上传到 GitHub**：将本地 Streamlit 应用上传到一个公开的 GitHub 仓库。仓库应该包含 Streamlit 应用的所有代码和依赖项，还应在仓库中包含一个 requirements.txt 文件，列出了应用需要的所有 Python 库。
3. **申请 Streamlit Sharing**：访问 Streamlit Sharing 的网站 (https://streamlit.io/sharing)，然后点击"Sign up for Streamlit sharing" 按钮进行申请。
4. **部署应用**：一旦获得了访问 Streamlit Sharing 的权限，登录到 Streamlit Sharing，然后点击"New app" 按钮来部署一个新的应用。在"GitHub URL" 字段中，输入相应 GitHub 仓库的 URL，然后点击"Deploy" 按钮即可完成部署。

由于我们已将网页代码公开在我们的 Github 仓库中，因此用户不需自行在 Github 部署。在 TrafficCat 目录下运行命令 `streamlit run web/dashboard.py` 即可在本地运行网页；在任何机器上通过访问网址 `https://ysyszheng-trafficcat-webdashboard-ngxdan.streamlit.app/` 都可以访问通过 Streamlit Sharing 部署的网页。

## 3 Docker 封装

为方便项目验收，我们将本系统及其运行环境封装进了一个基于 ubuntu:20.4 镜像的 Docker，基于 Docker 运行本项目方法如下：

### 3.1 获取容器镜像

以下两种方法都能获取到容器镜像 Docker 打包文件下载地址：`https://jbox.sjtu.edu.cn/l/U1Mduw`

```
# 方法一：从打包的tar.gz获取，在traffic_docker.tar.gz对应目录下执行：
docker import traffic_docker.tar.gz futuresjtu/traffic-v6

# 方法二：从Docker hub拉取：
docker pull futuresjtu/traffic-v6
```

### 3.2 进入容器

为了展示运行效果，需要将 docker 的 8501 端口和 9002 端口映射到宿主机，执行以下命令进入 Docker：

```
docker run -it --name trafficat -p 8501:8501 -p 9002:9002 futuresjtu/traffic-v6 bash
```

### 3.3 网络嗅探

依次执行以下命令开始网络嗅探

```
# 1.进入TrafficCat目录
cd TrafficCat/
# 2.开启嗅探器
./build/bin/clsniff
# 3.根据界面提示选择虚拟网卡eth0
eth0
# 4.根据界面提示开始嗅探
start
```

接下来，需要制造一些流量访问，推荐的方法是，另开一个 terminal，操作容器：

```
# 1.另开一个terminal，进入容器
docker exec -it trafficat bash
# 2.制造网络流量，例如ping某些站点,也可以选择其他方式
ping baidu.com
```

制造了一些网络流量后，可以选择结束嗅探，进行后续分析。回到最开始打开嗅探器的终端，执行以下命令：

```
# 1.停止嗅探
stop
# 2.退出嗅探器
exit
# 2.在正常环境下，输入exit会直接退出，在Docker内，可能偶发异常，若没有直接退出可能需要按下control+c，望助教学长海涵
```

### 3.4 进行流量分析

在嗅探完网络流量后，可以运行以下指令对其进行分析：

```
# 运行检测脚本
./run.sh
```

脚本运行成功后，使用宿主机浏览器：

- **访**问网址`http://127.0.0.1:9002`，并使用口令（用户名：minioadmin，密码：minioadmin）登录，即可查看数据库控制面板
- **访**问网址`http://127.0.0.1:8501`，即可查看此次流量捕获的分析结果

按下 control+c 将终止显示界面的呈现，进行后续命令的输入

### 3.5 历史流量合并分析

每次进行流量嗅探与分析时，系统都会自动上传数据到数据库，通过运行以下指令，可以实现对所有历史流量的合并分析：

```
# 为了展示流量分析的实际效果，我们在数据库里事先放了一些之前做测试的数据
./all.sh
```

脚本运行成功后，使用宿主机浏览器：

- **访**问网址`http://127.0.0.1:8501`，即可查看全部历史流量的分析结果，由于数据量较大，加载可能需要一定时间

## 4 总结

以上是在本地部署系统与基于 Docker 封装部署的全部流程，如若在部署系统时遇到问题，欢迎在项目 Github 仓库 [1] 中提出 issue，我们看到后会尽力解答。

[1] `https://github.com/ysyszheng/TrafficCat`