扫描探测工具行为识别实验报告

> 作者：赵奕翔
>
> 导师：周舟

一、背景介绍

在《扫描探测工具指纹库构建报告》[1]中，介绍了基于扫描器指纹信息的识别方法，并详细阐述了指纹库的构建过程。

然而，目前越来越多的扫描器支持自定义数据包的参数（如窗口大小），对于开源的扫描器，使用者也可以自行修改源码后重新编译以绕过检测。另外，由于 HTTPS 对原始数据包进行加密，无法获取相关指纹数据，因此基于指纹的方法并不能检测此类扫描行为。

本文从网络日志中提取流级特征，通过训练机器学习模型，对不同扫描器产生的网络流量进行分类，实现基于行为特征的扫描器流量识别。

二、分析与调研

对于网络流量分类任务，相关综述[2]分析了以下几种主流的解决方法。

分类方法	优点	缺点
基于端口	易于实现	应用不使用标准端口号
基于单个数据包	结果准确	无法分类加密流量，计算开销较高
机器学习方法	可以处理加密和非加密流量	性能依赖于人为选择的特征，普遍性受限
深度学习方法	自动学习输入输出之间的非线性关系，无需先进行特征选择	需要足够的标注数据和充足的计算能力

针对网络扫描器识别场景，近年来一些论文进行了研究，总结如下表所示。

论文	年份	识别方法	实验结果
ScannerHunter [3]	2014	构造bipartite graph识别（附录1）	precision: 96.5%（附录2）
DarkHunter [4]	2018	将 payload 信息构成二维矩阵使用 CNN 模型分类识别（附录3）	accuracy: 94.6%, recall: 95.0%（附录4）
WSDBBD [5]	2019	通过被动检测、主动注入、主动检测多种方式结合识别（附录5）	见附录6
CWSDBHC [6]	2019	分层关联检测，将语义内容识别和时间序列特征识别相关联（附录7）	precision: 100%, metric Rand Index for clustering: 98.4%（附录8）

由上表可知，机器学习/深度学习用于基于特征的扫描器识别方案可行，本文主要使用机器学习算法进行分类模型的训练和评估。

三、实验设计

根据综述 [2] 中提出的网络流量分类的一般性框架，本文从数据收集、数据预处理、特征选择、模型选择等几个方面介绍实验。

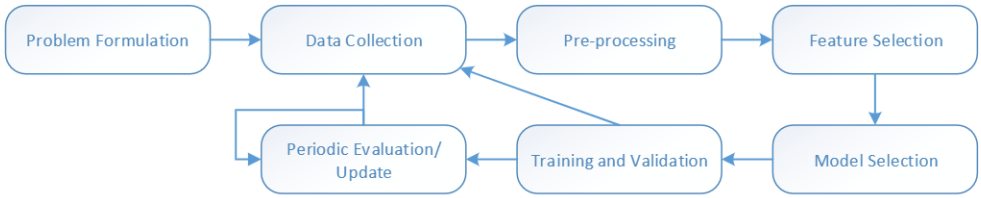


Fig. 1. General Framework to build a network classifier.

3.1 数据收集

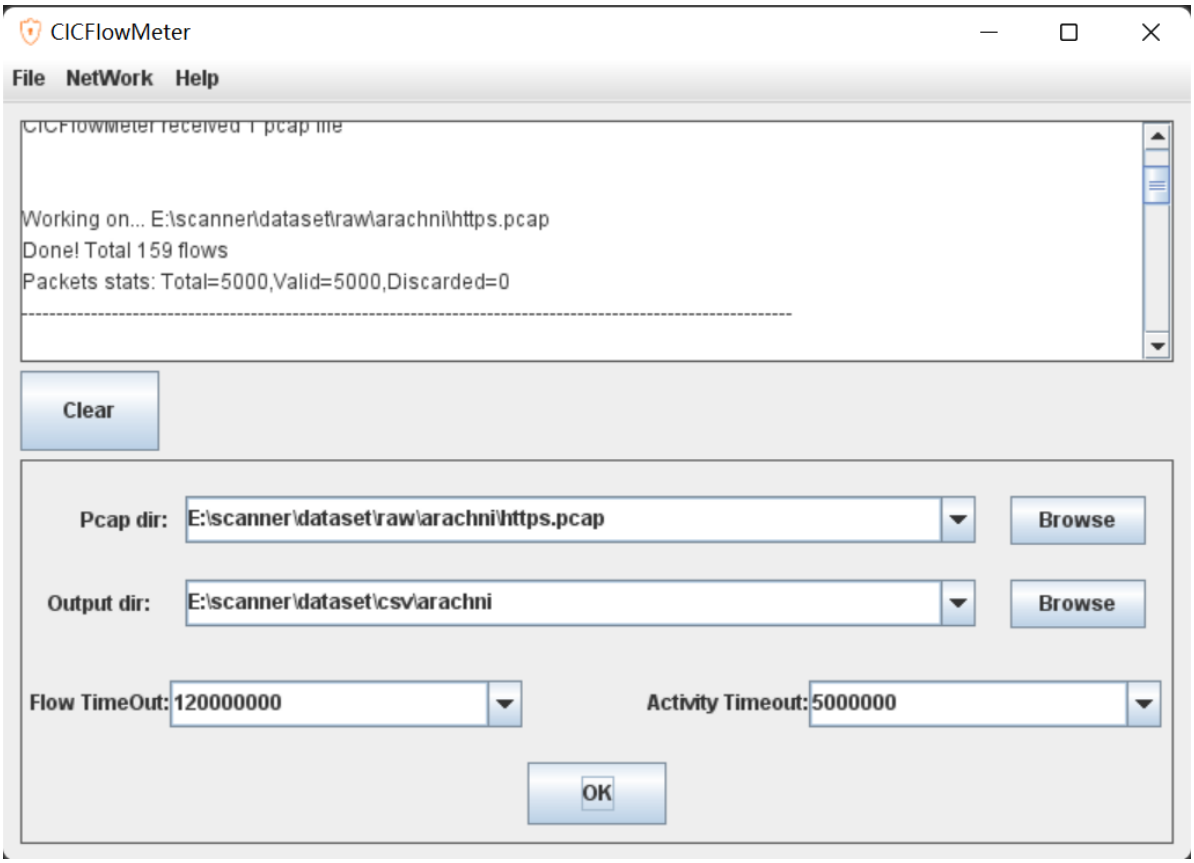
实验收集了5种广泛使用的扫描器的网络流量，分别为 Arachni、AWVS、Nessus、Nikto 和 Nmap。

对于每种扫描器，在Ubuntu环境中向国科大信息门户（http://sep.ucas.ac.cn/）和 MesaLab（https://mesalab.cn/）进行默认选项的网络扫描，使用 Wireshark 分别采集非加密流量和加密流量，为了使每种扫描器采集到的流量总数大致相同，对于产生流量较少的扫描器，还对百度（https://www.baidu.com/）等常用网站进行扫描。

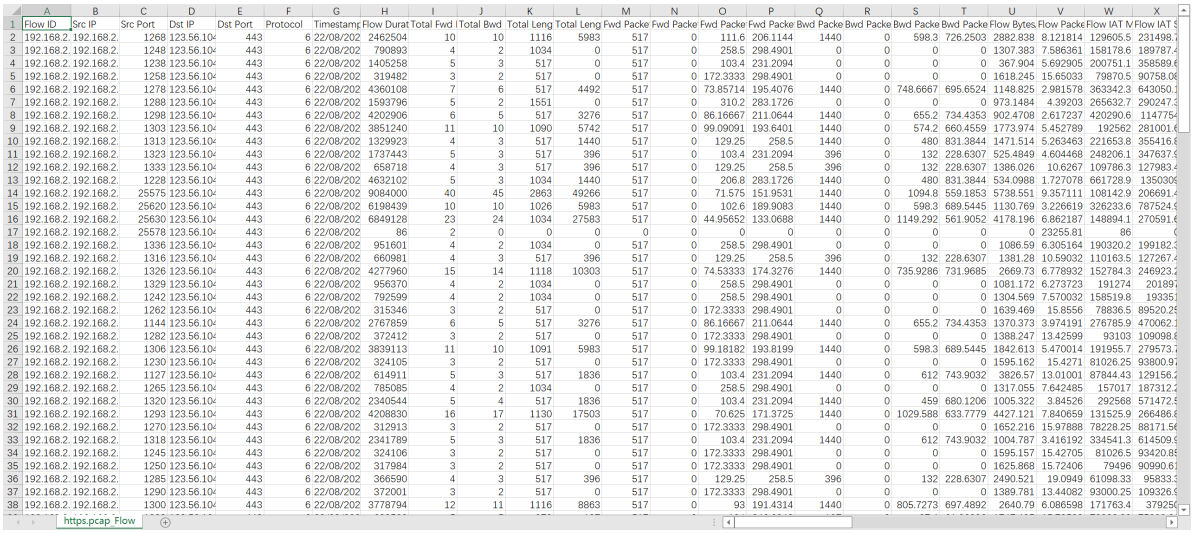
为了确保非扫描流量的复杂性，实验收集了华严网关上一段时间内的流量，与上述扫描器流量共同进行分类。

3.2 数据预处理

实验主要选取流量的流特征，特征提取使用 CICFlowMeter[7] 工具，CICFlowMeter 能够以 TCP 流或 UDP 流为单位完整划分出每条流的多维特征，使用图形化界面的 offline mode 导入 pcap 包即可处理导出 csv 文件。（注：文件路径不能有中文，否则会闪退。）



处理流量得到的 csv 文件如下图所示。



处理后每种扫描器的流量放入同一个文件夹中，加密流量与非加密流量共同进行分类，处理好的流量特征文件树状图如下所示。

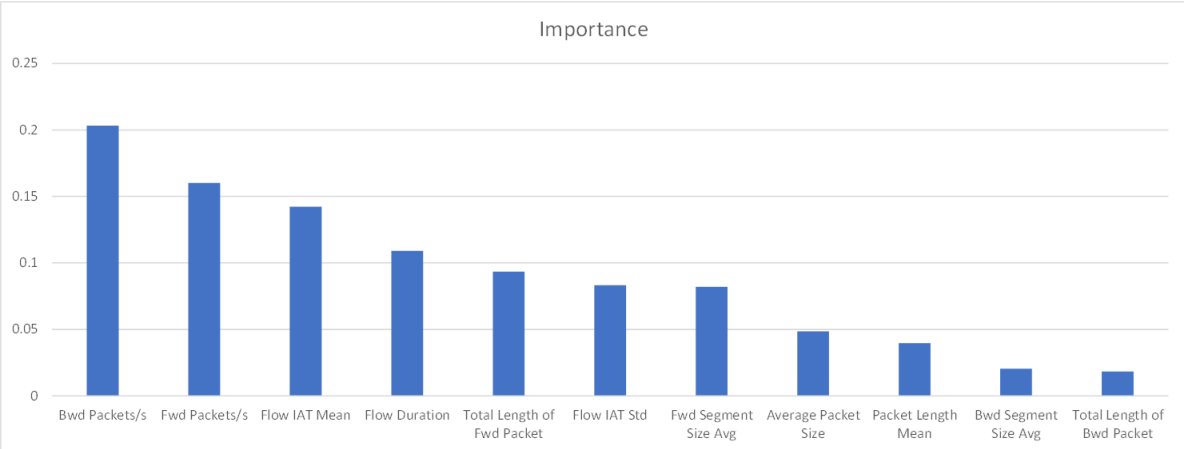
```
├──arachni
|       http.pcap_Flow.csv
|       https.pcap_Flow.csv
|       mesalab.pcap_Flow.csv
|
├──awvs
|       http.pcap_Flow.csv
|       https.pcap_Flow.csv
|
├──nessus
|       http.pcap_Flow.csv
|       https.pcap_Flow.csv
|
├──nikto
|       http.pcap_Flow.csv
|       https.pcap_Flow.csv
|
├──nmap
|       Nmap_host.pcap_Flow.csv
|       Nmap_OS.pcap_Flow.csv
|       Nmap_port.pcap_Flow.csv
|       Nmap_service.pcap_Flow.csv
|       tcp.pcap_Flow.csv
|       udp.pcap_Flow.csv
|
└──white
        huayan.pcap_Flow.csv
```

最后对特征做 Min-Max 归一化处理。

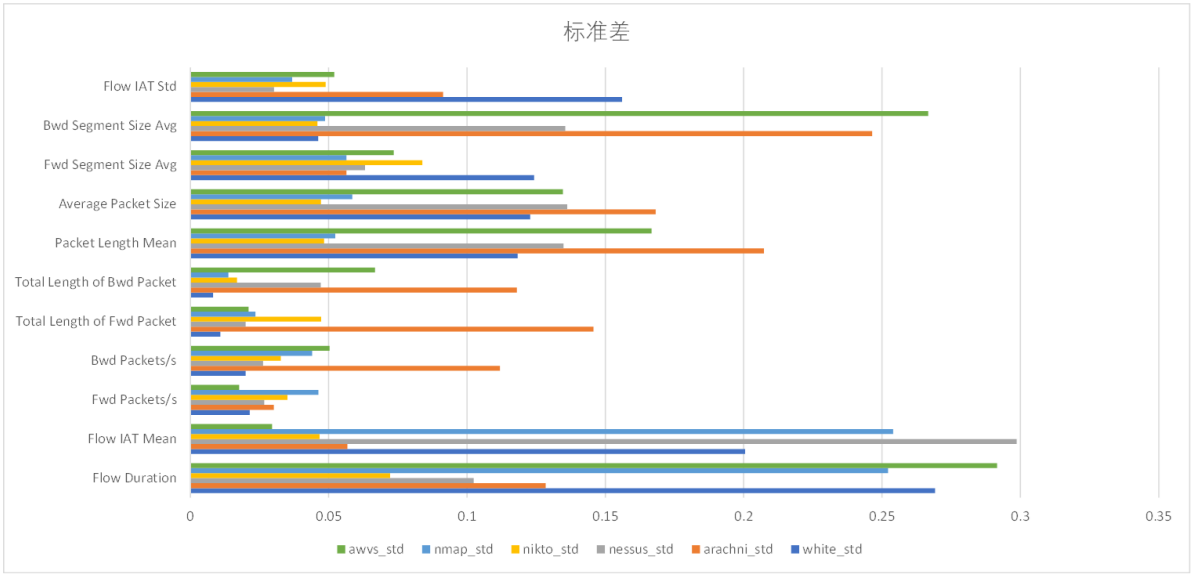
$$X_{\text{norm}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

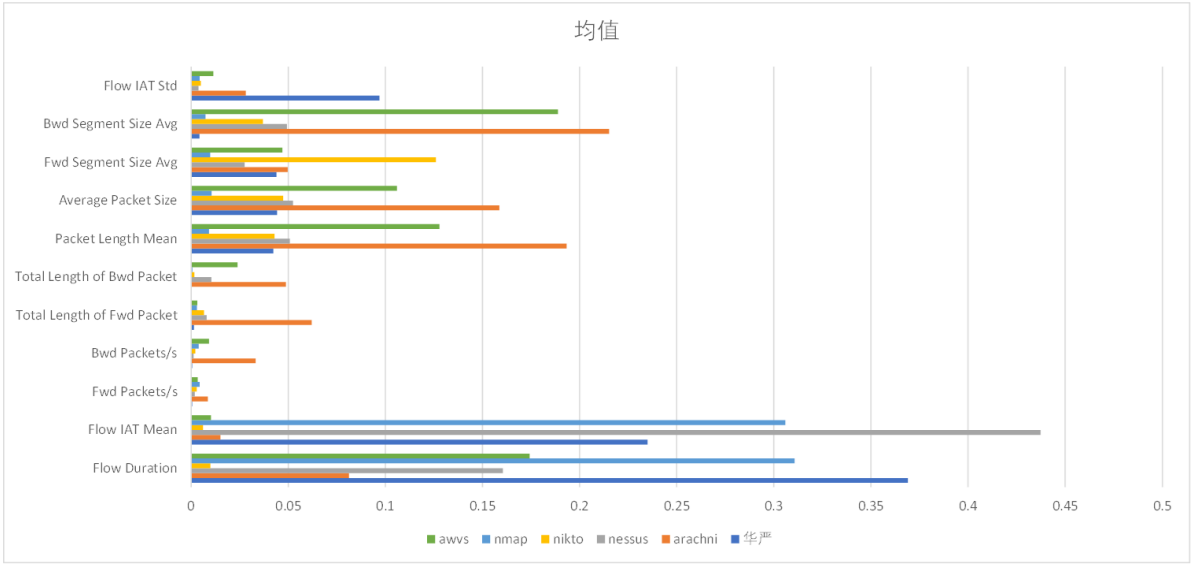
3.3 特征选择

每个分类各取3000条流，使用随机森林模型进行十折交叉检验，输出特征权重，权重最大的10个特征如下图所示。



对这10个特征进行进一步探索，绘图观察其标准差与均值分布，可以发现每个特征不同类别的流量差异较大。





选择其中权重大于0.1的4个特征，分别为下行速度、上行速度、流中数据包平均时间间隔、流持续时间。

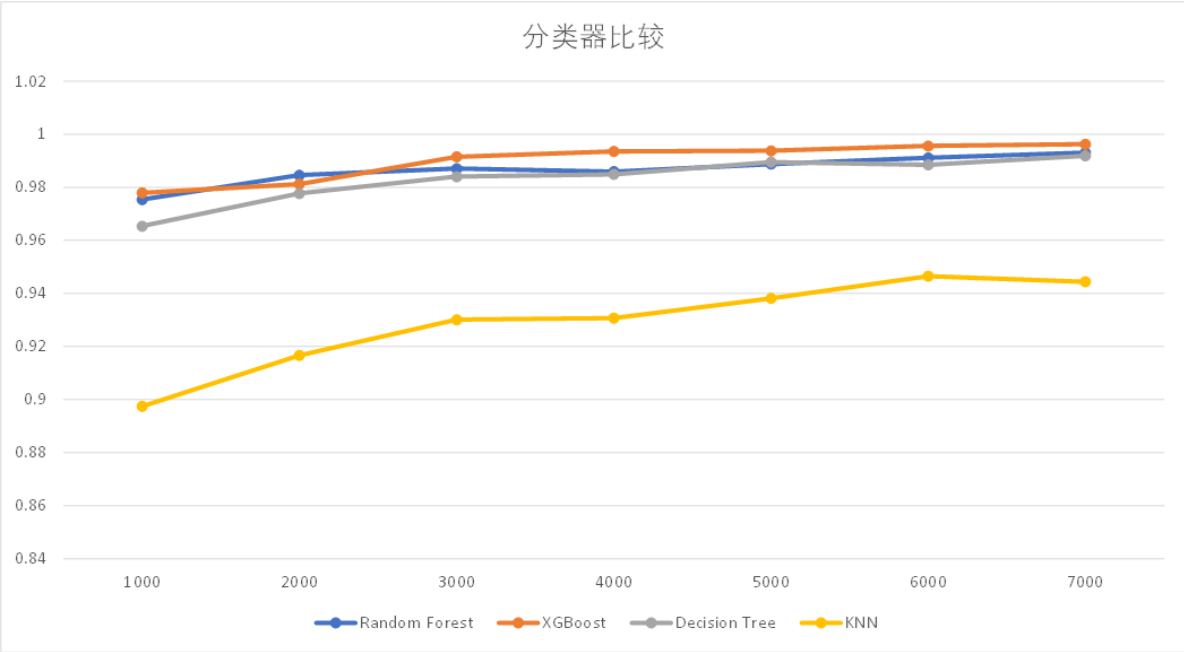
3.4 模型选择

实验选取随机森林、SVM、XGBoost 等主流的机器学习分类模型，训练集与测试集划分比例为7:3，流数目从1000递增至7000，每次增加1000条流。

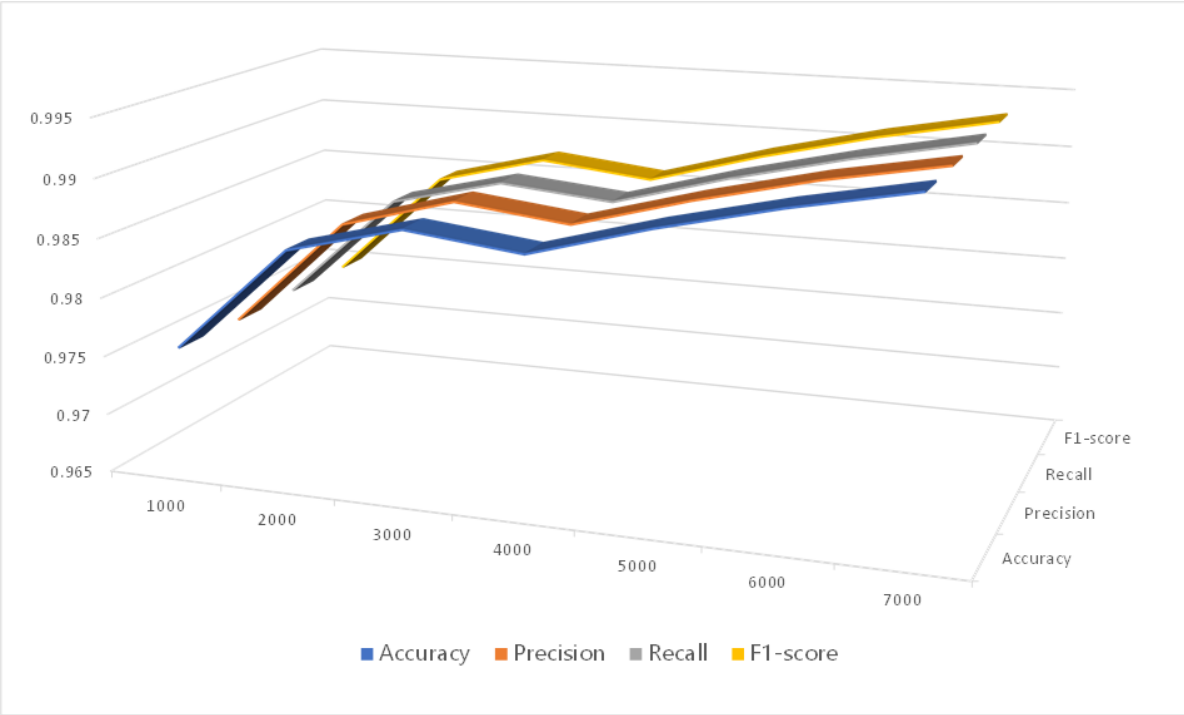
评估指标选择 accuracy、precision、recall 和 f1-score。

四、实验结果

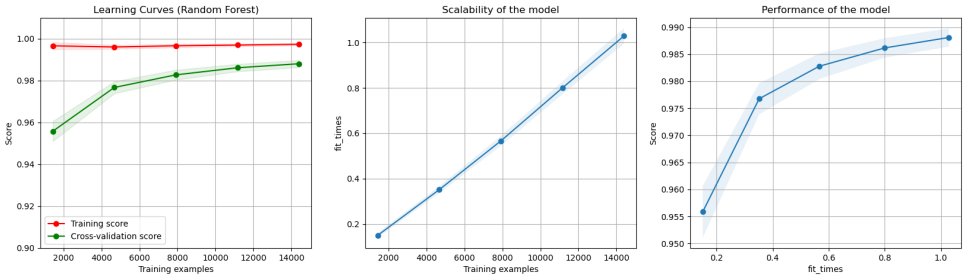
下图为实验所用各个分类模型的实验结果，由于 SVM 和贝叶斯的效果过差，因此去除了这两种模型。可以看到随机森林、XGBoost 和决策树都具有较好的分类效果，在3000条流时准确度超过99%。



下图为随机森林模型在实验中的各个性能指标，各指标之间相差不大，都具有较好的效果。



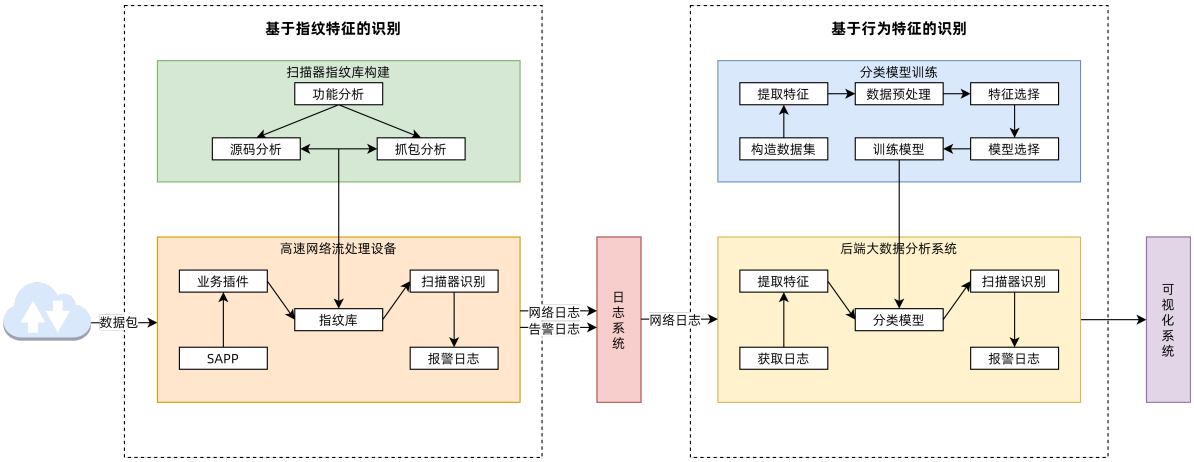
为了检验模型是否过拟合，如下图所示绘制了学习曲线。可以看到图中训练集和验证集上的曲线能够收敛，且偏差和方差都较小。从后两个图可以看出模型的扩展性和性能都比较好。



五、总结展望

本文通过对网络日志的关联分析，使用机器学习模型基于行为特征实现对扫描器的离线检测，填补了指纹失效和加密流量两种情况的空白。

下图所示为扫描器识别的全流程图，针对单包的指纹特征，在 MESA 平台编写了插件，可实现实时的快速识别；针对需要多包关联分析的情况，实现了机器学习模型，可用于在后端进行离线检测。



两种解决方案相结合，可形成完整的扫描器识别能力，通过对数据包的检测还原出其产生的扫描器，以便可以及时发现网络攻击的前兆。

附录

1. 论文3[3] 实验设计

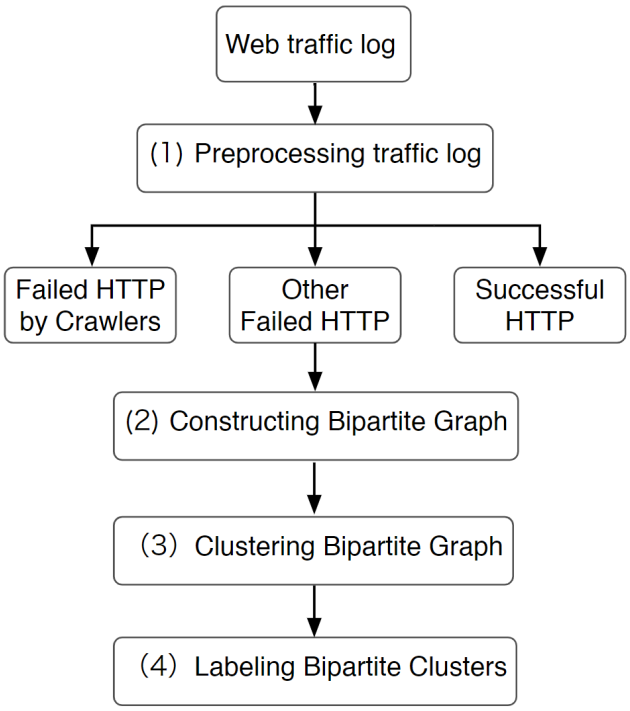


Figure 1: Scanner Hunter's main operations.

2. 论文3[3] 实验结果

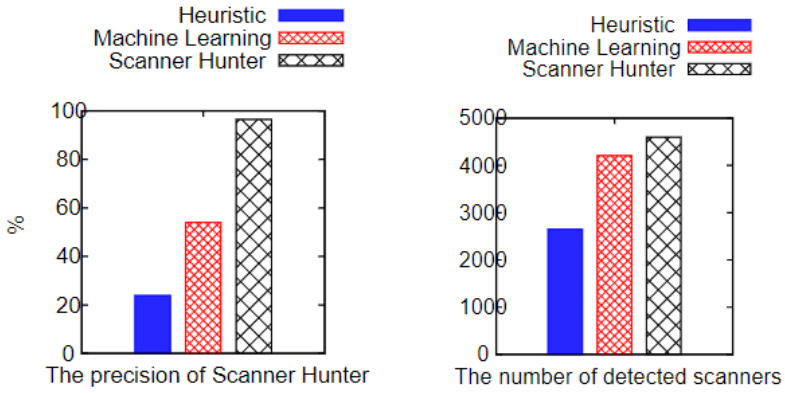


Figure 3: Performances of Scanner Hunter and baselines

3. 论文4[4] 实验设计

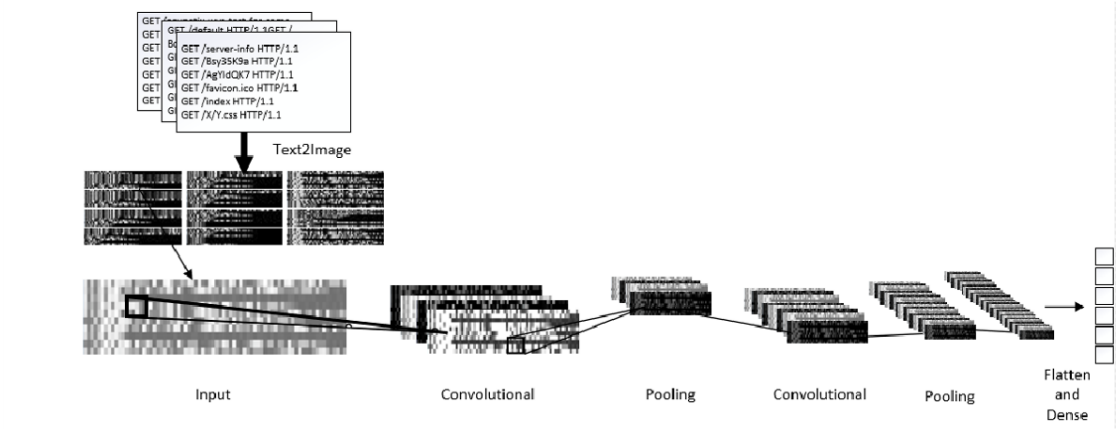


Figure 6. Convolutional Neural Network

4. 论文4[4] 实验结果

Table 4. Comparison of the methods

Methods	Accuracy	Recall	Number of samples
Traditional banner	0.816	0.078	50000
Cosine similarity	0.920	0.922	1200
DarkHunter	0.946	0.950	50000

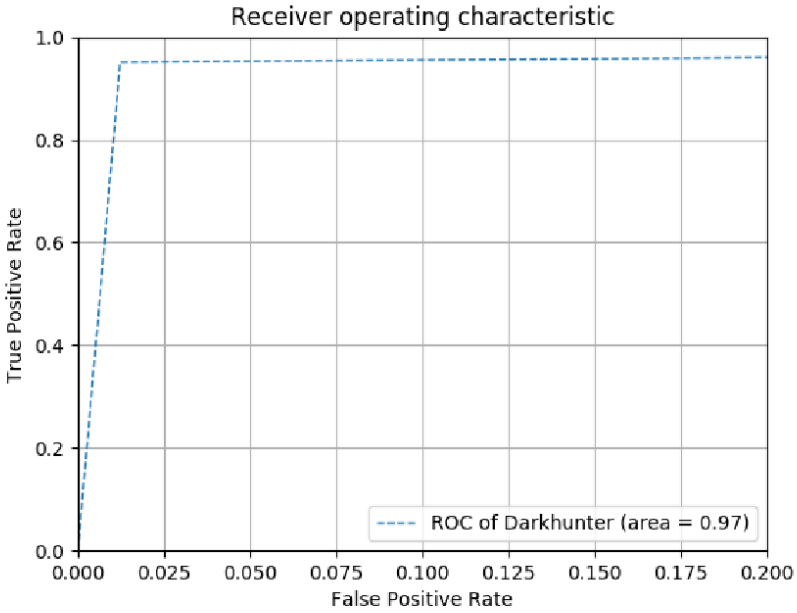


Figure 7. Mean ROC of DarkHunter.

5. 论文5[5] 实验设计

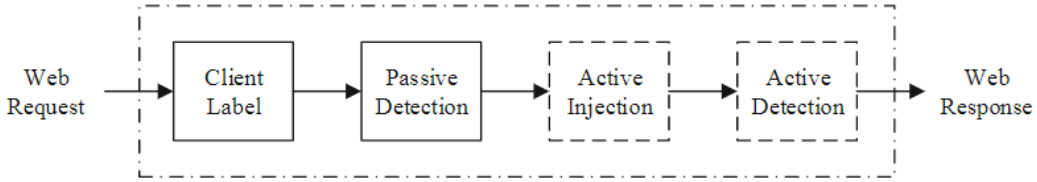


Fig. 1. Workflow of online detection model

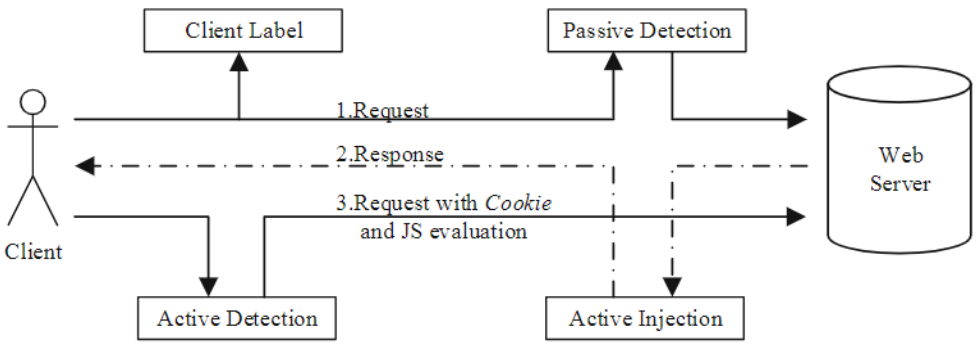


Fig. 2. Overview of the Dome infrastructure

6. 论文5[5] 实验结果

Table 2. Detection result based on function

Detection	Total	Success	Fail	Rate(%)
Passive	21	19	2	90.5
Active	21	21	0	100
Passive & Active	21	21	0	100
Frequency-based	21	16	5	76.2
Fingerprint-based	21	15	6	71.4
Resource-based	21	14	7	66.7
Cookie-based	21	12	9	57.1
JS-based	21	20	1	95.2
Mouse-click-based	21	19	2	90.5

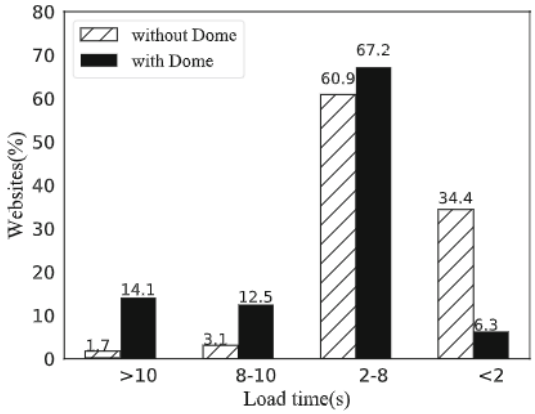


Fig. 3. Diagram of load time

Table 3. Detection delay

Function	Delay (s)
Fingerprint	0.016
Frequency	0.013
Media file	0.015
Cookie	0.0125
JS	0.027
Mouse-Click	0.0195

7. 论文6[6] 实验设计

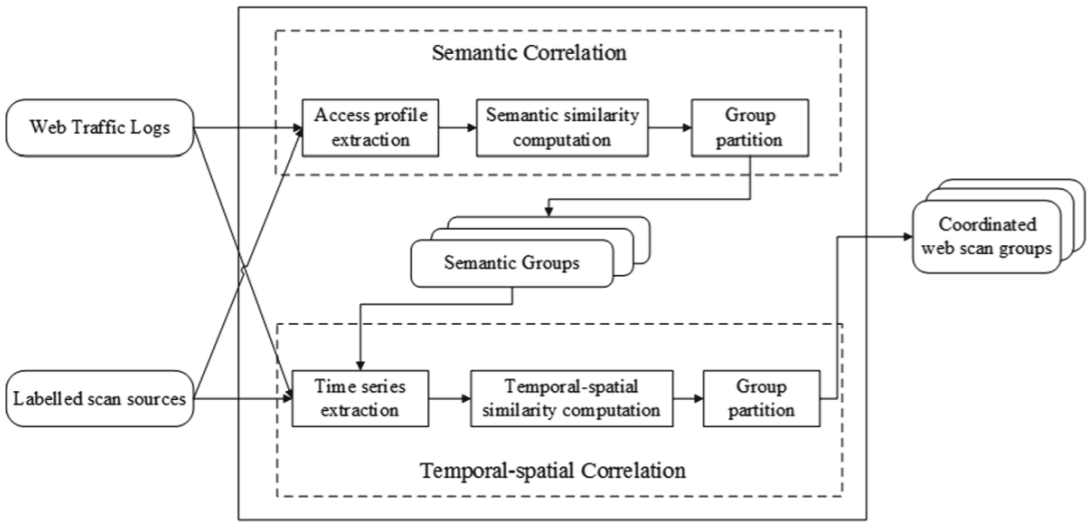


Fig. 2. Overview of our approach

8. 论文6[6] 实验结果

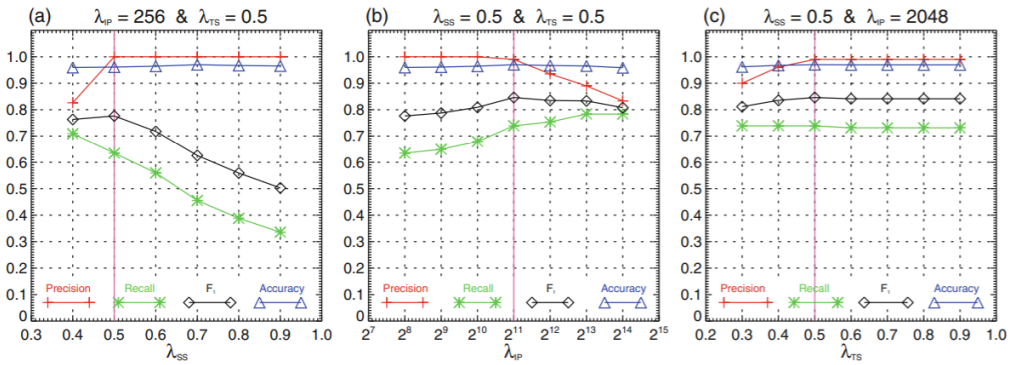


Fig. 3. Results of experiments for parameter selection.

---

1. 扫描探测工具指纹库构建报告

2. Rezaei, Shahbaz, and Xin Liu. "Deep learning for encrypted traffic classification: An overview." *IEEE communications magazine* 57.5 (2019): 76-81.

3. Xie, Guowu, Huy Hang, and Michalis Faloutsos. "Scanner hunter: Understanding http scanning traffic." *Proceedings of the 9th ACM symposium on Information, computer and communications security*. 2014.

4. Fang, Yong, et al. "DarkHunter: a fingerprint recognition model for web automated scanners based on CNN." *Proceedings of the 2nd International Conference on Cryptography, Security and Privacy* . 2018.

5. Fu, Jianming, et al. "Web Scanner Detection Based on Behavioral Differences." *International Symposium on Security and Privacy in Social Networks and Big Data* . Springer, Singapore, 2019.

6. Yang, Jing, et al. "Coordinated Web Scan Detection Based on Hierarchical Correlation." *International Conference on Security and Privacy in New Computing Environments*. Springer, Cham, 2019.

7. https://github.com/ahlashkari/CICFlowMeter