Web 进程链及 SSH 操作序列异常检测技术设计文档

1. 问题描述

针对 Web 进程链/SSH 异常操作序列，检测上下文关系中的子进程异常。

Web 进程链：同一 Web 服务应用的进程调用关系中，父进程为 JAVA，其派生的所有子进程序列。

SSH 操作序列：同一台主机系统中，父进程为 TTY，其派生的所有子进程序列。（可以当作是管理员的输入）

异常定义：根据历史行为建立基线模型，识别有异于正常行为的异常行为

2. 研究意义

如今绝大多数企业及个人用户都选择使用云计算来部署、运营业务。

Web 进程链及 SSH 异常操作检测技术研究将应用在的主机型入侵检测系统，通过分析程序访问的内存、文件系统、日志存储等信息，结合算法能力，精确检测程序异常或恶意入侵行为，以为系统提供主动性的防御机制。

该项技术可通过捕获 Linux 系统下用户态和内核态通信过程中产生的信息，挖掘更深层次的特征，以进程为粒度进行入侵或异常行为的检测，能够抵御新型攻击手段，特别是隐蔽而又高危的 APT 攻击，为用户提供安全可靠的云计算环境。

3. 已有最新技术描述

实现恶意指令检测的技术，传统一般采用的是基于模式匹配和规则的方法。原理就是基于黑名单的命令匹配，网络管理员构造一些恶意命令集合，通过对比执行的指令与恶意命令集合中的指令，来判断该指令是否是恶意的。此种类型的检测算法依赖于模式和规则的构造，而且对新出现的恶意行为检测能力较差，攻击者可能通过简单地混淆措施，绕过规则检查。

此外，一些异常检测算法也可以应用于异常数据识别。比如基于统计的方法，现在正常数据集上计算出一些统计数据，当测试数据的相关数值与原数据集相差较多（比如正常数据集的长度均值为 35， 但是测试命令的长度为 150），就将该测试数据识别为恶意指令。基于密度的异常检测方法，数据集中的一些数据点与最近 n 个邻居的距离，一定程度上反映了该数据点附近的密度，如果某个数据点所在位置密度较小，我们有理由认为该数据点是异常数据。基于聚类的异常检测，对数据进行聚类分析，分析测试数据与聚类中心的距离，如果距离过大，表示该数据是恶意数据。

机器学习算法在分类任务与自然语言处理方面取得的成就可以应用到此问题中。机器学习算法在分类任务中取得了令人惊喜的成绩，T. Minárik, S.Alatalu, S.Biondi 等人在 Detection of Malicious Remote Shell Sessions 提出了使用基于 n-gram 的 knn 算法识别恶意指令的方法[1]。使用 n-gram 衡量两条 shell 指令之间的距离，随后使用 knn 算法进行有监督的学习与预测，模型准确率达到 98%。YifanTian，Jiabao Wang 等人在 CNN-Webshell: Malicious Web

Shell Detection with Convolutional Neural Network 中提出了使用卷积神经网络识别恶意 web shell 的方法[2]，将 http 请求文本当做原始数据，经过 word2vec 向量化之后，输入卷积神经网络进行有监督的训练与预测，模型准确率达到 98%。Danny Hendler，Shay Kels 等人在 Detecting Malicious PowerShell Commands using Deep Neural Networks 中提出了使用深度神经网路识别恶意命令的方法[3]，他们将 powershell 指令进行 base64 编码后进行正则化，输入神经网络进行训练与预测，模型准确率在 87%左右。

可以看到，当前的机器学习在恶意命令识别的应用模式，主要是：使用自然语言处理的文本向量化方法（n-gram, Tfidf, WordBag, word2vec），将文本转换为向量，进行机器学习算法或神经网络模型训练。

4. 拟采取的技术

特征提取

文本向量化的方式有 Tfidf，词袋模型，因为 tfidf 考虑单词顺序信息，相比词袋模型更加精确，所以在初步测试中我们选择 TFidf 向量化方法。将指令当做文本，使用 Tfidf 方法提取向量特征。在进行某些算法测试的过程中，由于向量化以后的数据规模庞大，也需要我们使用降维算法对数据进行处理。

有监督学习

Knn、svm、bayes 与 dt 都是机器学习中常用的分类算法，在数据分类方面具有广泛的应用。神经网络在分类任务中有着不输上述分类算法的能力，全连接神经网络作为神经网络的代表，也参与到我们的初步测试中。在初步测试中我们会使用 knn、svm、dt、bayes 算法、全连接神经网络进行有监督学习测试。

无监督学习

我们使用适用于 novelty detection 的检测方法，我们已有的样本都是正常的，通过这一点，我们可以根据已有的正常数据构建模型，将所有异于正常样本的数据都视作异常数据，达到恶意命令执行检测的目的。基于聚类的算法也值得尝试，web 进程链执行的进程类型数量有限，使用聚类算法将数据集中的命令分成数量有限的中心，我们测试待测指令与聚类中心的距离，从而判断该指令是否恶意。

5. 方案设计

1. 验收标准

交付件

- 原型系统
- 算法源码
- 设计文档及验证报告

功能目标

- 可对多个业务产生的 web 进程建立正常基线检测异常进程
- 可针对业务调整，模型具有动态更新能力，或根据新的业务行为进行历史相关性学习
- 在检测出已知威胁的基础上，需对未知的威胁有检测能力

2. 整体方案

参考自然语言处理中的情感分析方法，将每条命令当做一条文本，向量化以后输入算法模型中做极性分析，得到恶意指令分类结果。

- 数据集
  从互联网上爬取 linux 指令集，将与白样本相同的指令去除，剩余的指令当做恶意指令
- 使用 tfidf 方式将命令数据向量化
- 训练机器学习算法与神经网络模型
- 对比选择最优模型

3. 方案对比

4. 项目名词解释

表格 1 名字解释

DT	Decision Tree 决策树算法
SVM	支持向量机分类算法
KNN	K 临近算法
Bayes	贝叶斯分类算法
n-gram	用于描述字符串差异性的方法
DNN	深度神经网络（Deep Neural Networks）

CNN	卷积神经网络（Convolutional Neural Network）
OneClassSvm	单类别支持向量机，用于异常检测
Iforest	独立森林算法
LOF	离群因子检测（Local Outlier Factor）
PCA	Principal Component Analysis，用于数据降维的算法
LDA	线性判别分析，可用于数据降维与分类
查准率	描述模型正确识别出恶意样本的能力，识别出的恶意样本，有多少是真正的恶意样本
召回率	从数据中识别恶意样本的能力，识别出的恶意样本占整个恶意样本的比例
学习曲线	模型准确性随着训练数据的增加而变化的曲线，用来分析模型是否发生了过拟合，或欠拟合

5. 具体检测方案

5.5.1 研究内容

Web 进程链恶意入侵检测，通过分析程序访问的内存、文件系统、日志存储等信息，结合算法能力，精确检测程序异常或恶意入侵行为，以为系统提供主动性的防御机制。针对线上业务类型的 web 进程链中的指令集，训练机器学习模型，用来进行恶意指令的分类。要求模型准确率高于 95%， 误报率小于 8%，也需要具有检测未知威胁的能力。此外，攻击方式有恶意指令和注入攻击两种，针对注入攻击，我们在检测之前先使用可能的命令分隔符对要预测的数据集进行处理，将注入攻击转换为直接的恶意指令检测。

5.5.2 模型输入

训练阶段

训练阶段的正常白样本为离线的 json 格式的文本，每个 json 字符串一行，json 字段和收集到的日志相同，但是只用到了 ccmdline 字段。

CreateTime	进程创建时间
ppid	父进程 id
pname	父进程名称
pcmdline	父进程命令
pexe	父进程执行路径
cpid	子进程 id
cname	子进程名称
ccmdline	子进程命令
cexe	子进程执行路径

1 白样本 json 字段

训练阶段的黑样本数据是 shell 指令组成的文本文件，每行一条指令。

预测阶段

预测阶段的数据来源于 spark streaming，格式同训练数据白样本格式，同表 1。

5.5.3 模型输出

模型输出为指令的预测到的恶意指令结果

5.5.4 数据来源

5.5.5 检测原理

命令语言字典

命令向量化

异常检测算法

机器学习算法

6. 已进行的工作

数据集

由于恶意数据与正常数据数量相差众多，从互联网上使用爬虫爬取更多的 linux 指令，去除与原有正样本相同的指令，其他指令的当做恶意指令参与模型训练过程。

算法衡量指标

衡量算法我们使用查准率与召回率指标。查准率表征了模型正确识别恶意指令的能力，召回

率表征了模型从海量数据中找到恶意指令的能力。学习曲线，是模型准确率随着训练数据的增加而变化的曲线，可以看出模型训练过程中的性能表现，判断模型是否出现了过拟合的现象。

无监督学习

Oneclass SVM 算法

此算法的思想是通过核函数将训练样本的向量映射到高维向量空间，在此空间中，这些样本收敛在一定区间之内，该算法找到一个包络将这些样本点包含进去，以此包络来划分异常数据与正常数据。

2 Oneclass SVM **初步测试**

	查准率	召回率
正常指令	99%	93%
恶意指令	75%	96%

模型初步表现说明，漏报率非常高。

Iforest 算法

算法结果描述如下

表格 3 **独立森林算法测试结果**

	查准率	召回率
正常指令	80%	88%
恶意指令	12%	7%

模型在该数据集上的查准率与召回率均较低

LOF 算法

算法结果描述如下

表格 4 LOF **算法测试结果**

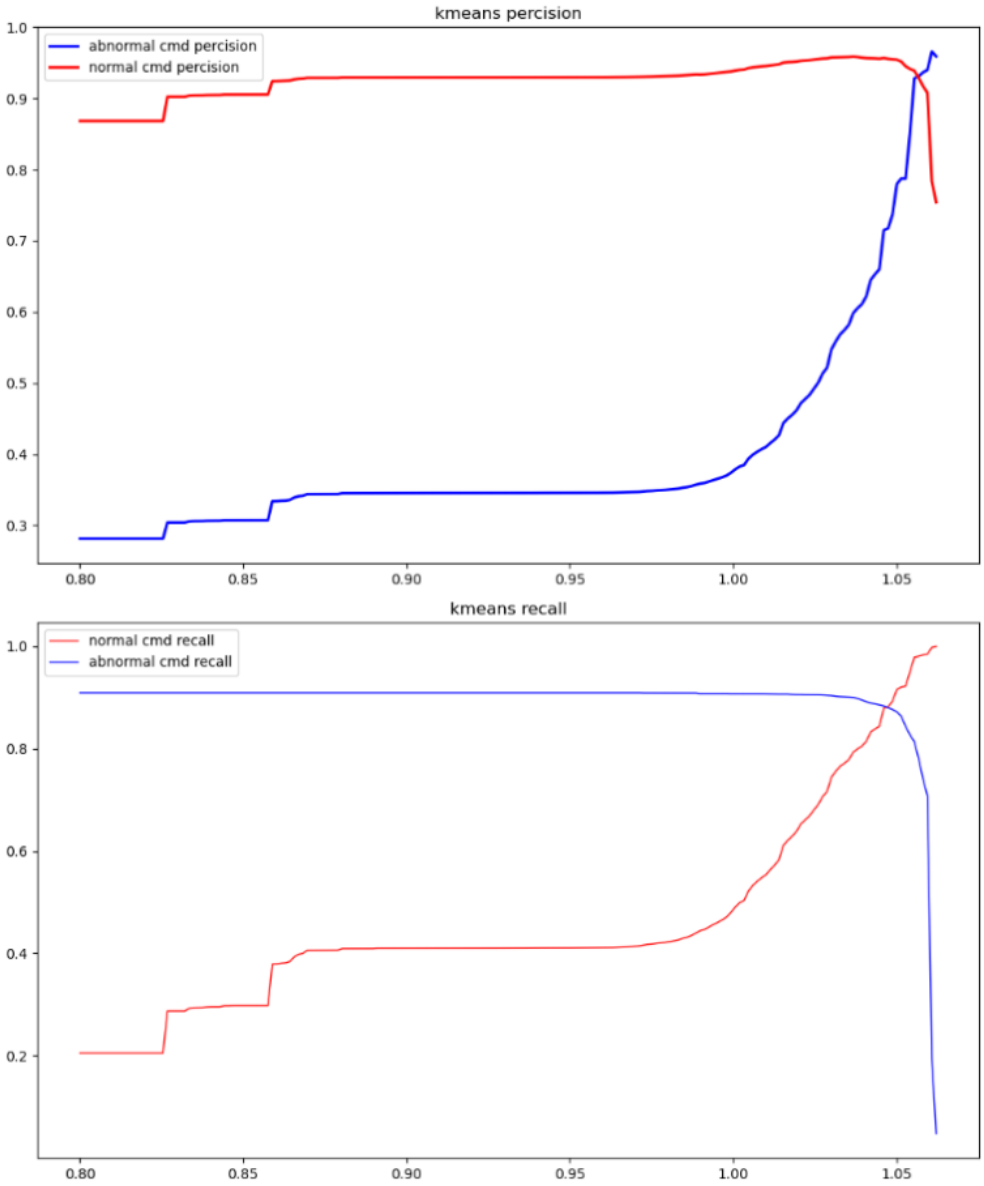
	查准率	召回率
正常指令	87%	87%
恶意指令	44%	44%

模型在该数据集上的查准率与召回率均较低

基于聚类的异常检测

基于聚类的异常，我们使用了 kmeans 聚类算法，使用单独一个聚类中心，测试的时候计算测试样本向量化后的向量与聚类中心的几何距离，通过修改距离阈值，得到准确率和召回率

随阈值变化如图所示

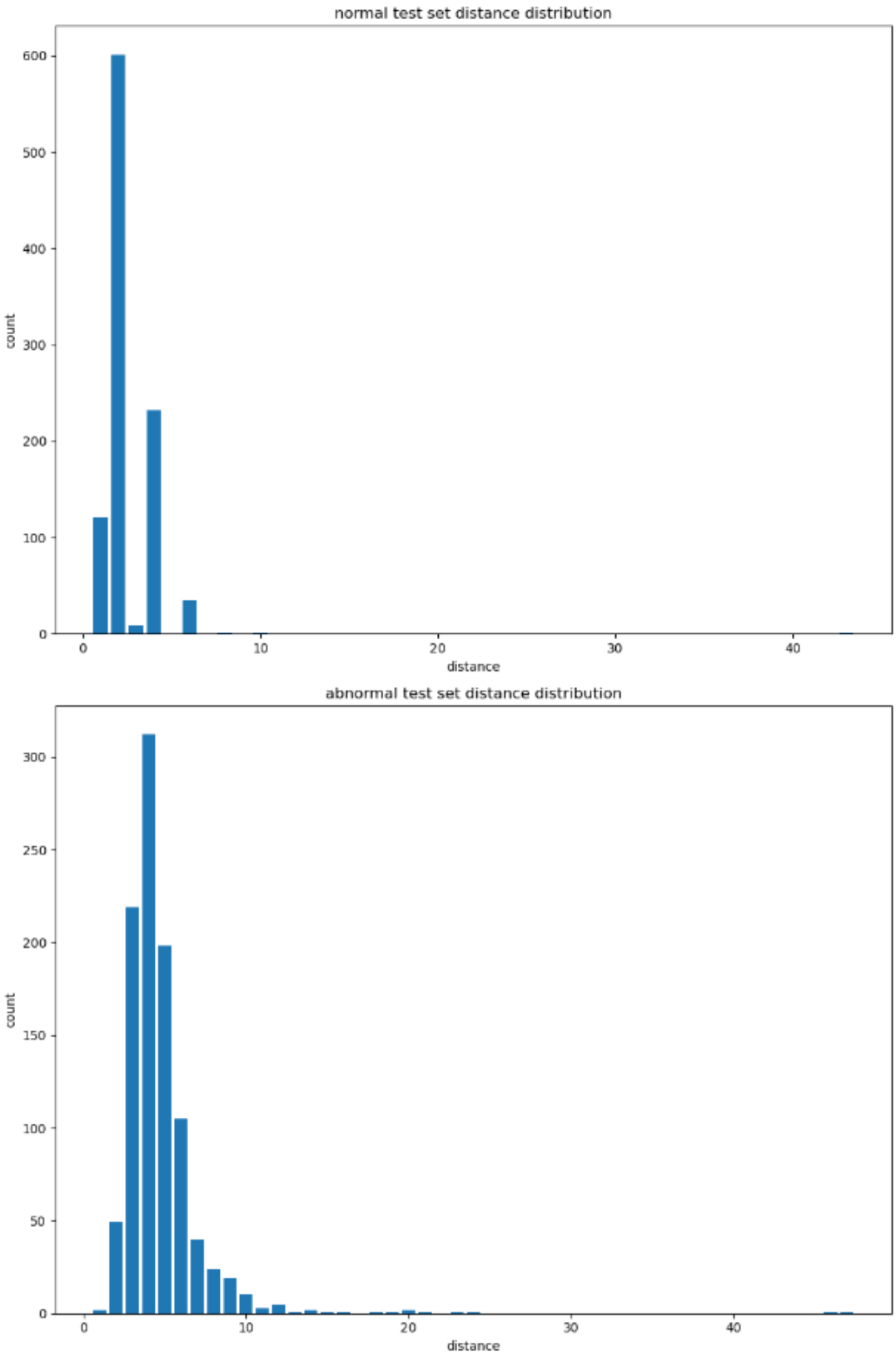


1 kmeans **准确率与召回率变化曲线**

随着距离阈值提高，模型准确率与召回率在阈值为 1.05 左右时达到最优，约为 0.9。但是当阈值大于 1.05 时，模型召回率与准去率曲线变化十分陡峭，阈值的轻微变化就会导致模型准确率与召回率的巨大波动。

基于 n-gram 的 knn

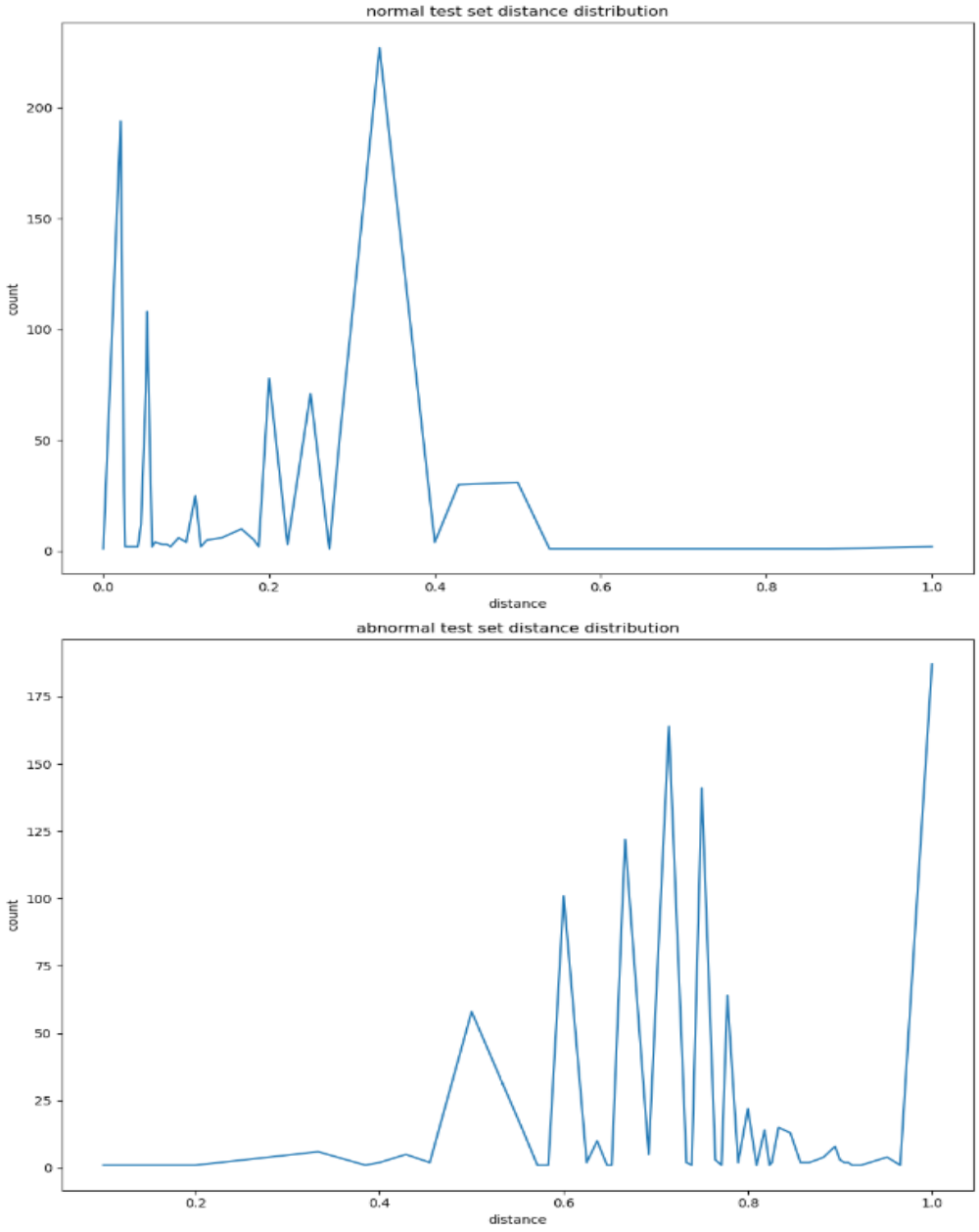
使用 n-gram 方式衡量字符串差异是自然语言处理中常用的手段。我们也可以通过 n-gram 距离来衡量两条 shell 指令之间的差异。当某条指令与正常样本的 n-gram 距离过大，就判定该命令为恶意样本。经过测试该方法准确率 40%。我们随机取 1000 条正常 shell 指令，随机取 1000 条恶意指令，分别计算这些指令到正常数据集的最小 n-gram 距离（n 取 1），了解 n-gram 距离分布，其结果如下图



5 n-gram **距离数量分布**

横轴是距离大小，纵轴是命令个数，可以看到，即使是异常指令，与正常样本 n-gram 距离最小值也是分布在 0-10 之间，且大量集中在 2 3 4 5 这四个值，当我们选择一个阈值当做 n-gram 距离的上界，比如选择 3 作为划分恶意指令与正常指令的界限，我们会有较高的误

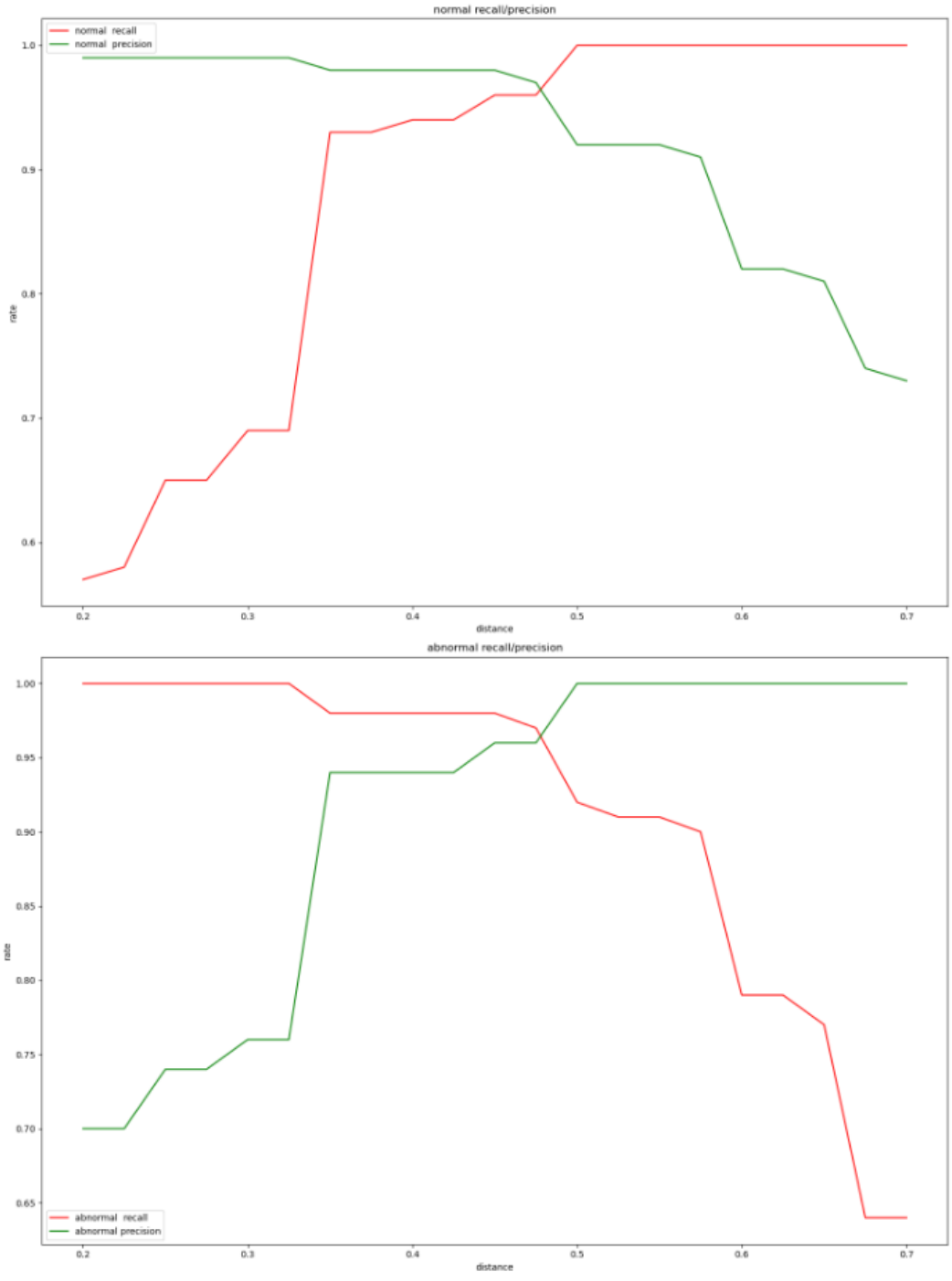
报率与漏报率。结果不理想。使用原始的 n-gram 距离作为分类的界限并不是一个好的选择。假设某条命令用空格分隔后有 100 个关键字，此时在我们的正常数据集中，有一条指令和它只有一个参数不同，它们的 1-gram 距离我们可以计算得 2，我们倾向于认为该指令是一个正常指令，因为它与训练数据集中的一条指令相似度达到了 99%。假设还有一条指令 ifconfig，训练集中有长度为 1 的指令，他们的 n-gram 距离也是 2，但是这条指令显然是恶意的，因为它之前并没有出现过。产生这种问题的原因是因为绝对的 n-gram 距离在衡量指令差异性上存在一些不足，我们如果将指令长度考虑进来，对计算后的 n-gram 距离做归一化处理，那么计算后的距离转换为 0-1 之间的小数，还是上面的两个例子，ifconfig 与训练集的归一化 n-gram 距离将变为 1，而长为 100 的指令，由于有 99 个关键字都相同，归一化距离变为 0.01，一定程度上增加了模型准确性。在我们的数据集上，从正常样本与异常样本中随机各取出 1000 条指令，统计归一化 n-gram 距离如下



图表 1 归一化 n-gram 距离统计

我们使用归一化的 n-gram 距离，均匀选择 0.2-0.7 之间的 20 个界限作为指令是否异常分界

点，绘制准确性曲线得到如下结果



图表 2 n-gram based knn **准确性曲线**

在界限值取 0.45 时模型达到准确率与召回率的平衡，约为 95%。准确率信息如下

	查准率	召回率
正常指令	98%	96%
恶意指令	96%	98%

有监督学习

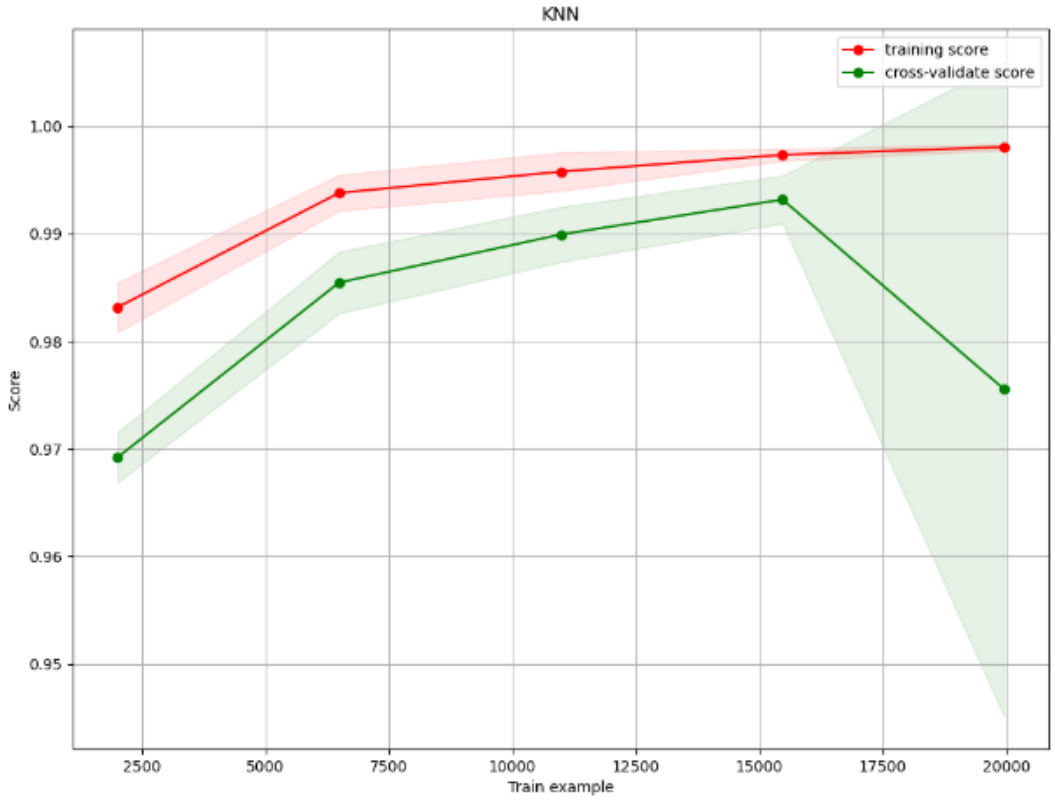
Knn

使用 sklearn 的 knn 算法实现，使用 Tfidf 方式将命令文本向量化，在部分数据集上的测试结果如下

6 knn **算法测试结果**

	查准率	召回率
正常指令	100%	100%
恶意指令	99%	99%

Knn 算法在该数据集上的学习曲线如下



7 knn **学习曲线**

从 knn 的学习曲线可以看出，随着训练样本增加，模型在测试集上的准确率下降，且方差变大，具有过拟合的特征， 在样本达到 15000 时，准确率达到最高的 0.99。

Svm

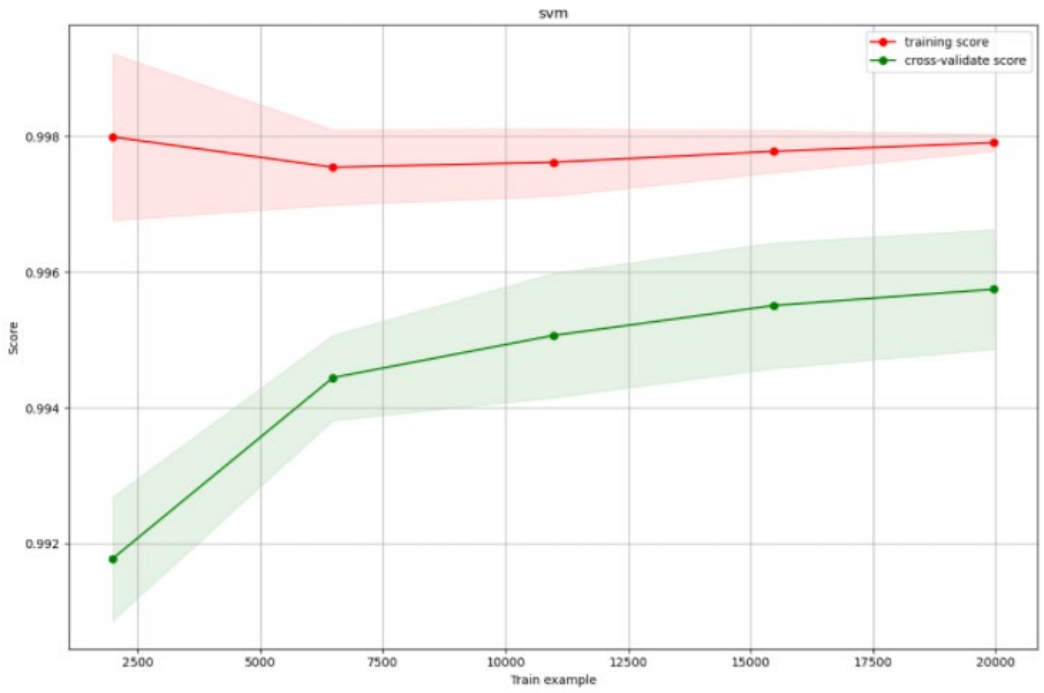
我们使用 sklearn 的 Svm 算法实现，使用 Tfidf 方式将命令文本向量化，在部分数据集上的测试结果如下

8Svm **算法测试结果**

	查准率	召回率

正常指令	100%	100%
恶意指令	99%	100%

Svm 算法在此数据集上的学习曲线如下



9 svm **学习曲线**

svm 的学习曲线较为正常，模型准确率在训练集和测试集的准确率都比较高，而且上升趋势仍未停止，当前准确率 99%，随着数据集增加，准确率有可能进一步提高，并且误差在 1%以内。

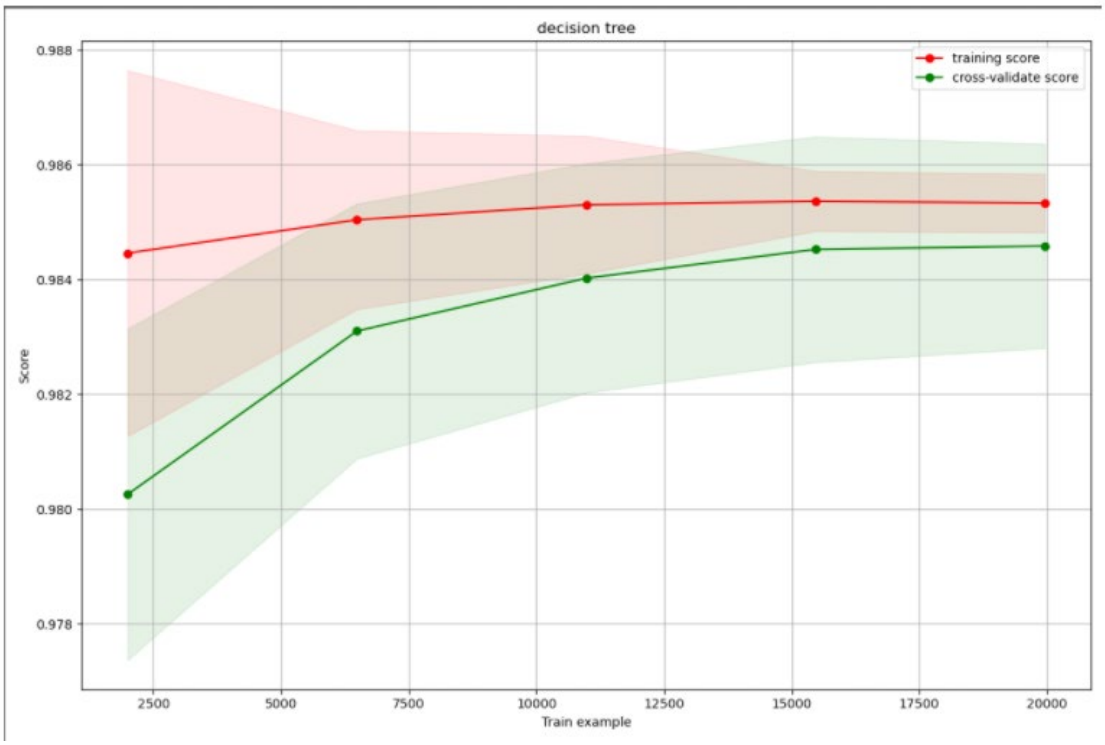
DT

我们使用 sklearn 的决策树算法的实现，使用 Tfidf 进行命令向量化，训练模型。在部分数据上的测试结果如下

10 **决策树算法结果**

	准确率	召回率
正常指令	100%	98%
恶意指令	95%	100%

其学习曲线如下图所示



11 DT **算法学习曲线**

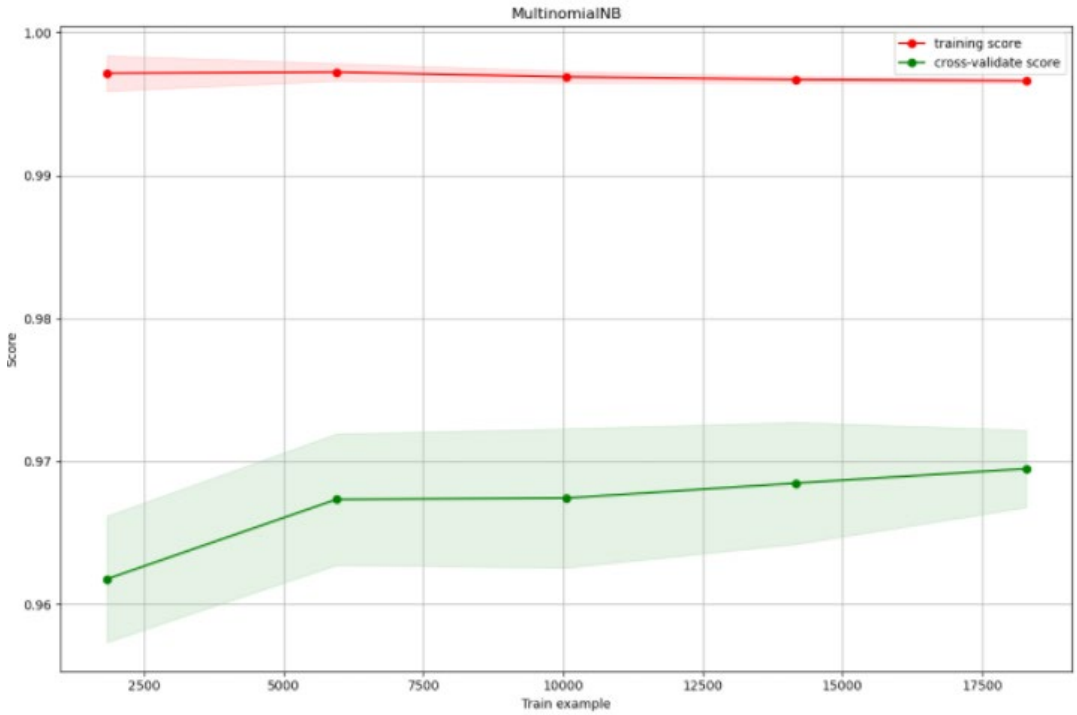
从学习曲线来看，该模型表现与 svm 相当，随着数据集增加，准确率都有逐步提高，当数据到达 17500 时，准确率变化已趋于平缓，在训练集与测试集的准确率达到 98%，未出现过拟合的现象，并且误差范围控制在 1%。

Bayes

使用 sklearn 中的 Bayes 分类器，得到模型准确率如下

	准确率	召回率
正常指令	100%	100%
恶意指令	100%	99%

Bayes 模型在此数据集上的学习曲线



12 Bayes **模型学习曲线**

模型在测试集上的准确率不如训练集上的高，但测试集准确率有上升趋势，模型未表现出过拟合特征，准确率达到 97%。

神经网络

使用 TensorFlow 构建全连接神经网络，隐藏层单元使用 LSTM 神经元。算法测试结果如下

	准确率	召回率
正常指令	100%	100%
恶意指令	99%	100%

算法准确率达到 99%，是可选的解决方案。

算法结果对比

以上测试的有监督学习算法准确性都达到了 95%以上，神经网络、svm 与 bayes 准确率更是达到了 100%，knn 算法经过多次测试，准确性在 90%-98%之间波动。异常检测算法方面我们测试了 OneClassSVM 在此数据集上的表现，当前漏报率高达 70%。与安全相关的问题，使用白名单的方式安全性要高于使用黑名单的方式。异常检测方法相当于白名单，后面对异常检测算法进行测试，尽量优化算法的准确率与召回率。

7. 参考文献

[1]. Dumont, P., Meier, R., Gugelmann, D., & Lenders, V. (2019, May). Detection of Malicious Remote Shell Sessions. In *2019 11th International Conference on Cyber Conflict (CyCon)* (Vol. 900, pp. 1-20). IEEE.

[2]. Tian, Y., Wang, J., Zhou, Z., & Zhou, S. (2017, December). CNN-webshell: malicious web shell detection with convolutional neural network. In *Proceedings of the 2017 VI International Conference on Network, Communication and Computing* (pp. 75-79).

[3]. Hendler, D., Kels, S., & Rubin, A. (2018, May). Detecting malicious PowerShell commands using deep neural networks. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security* (pp. 187-197).