# Web进程链及SSH操作序列异常检测技术设计文档

## 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

## 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。

## 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。

## 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。此外在18年的一篇论文中也提到了使用基于n-gram距离的knn算法进行恶意shell指令识别的方法。

## 方案设计

### 验收标准

#### 交付件

* 原型系统
* 算法源码
* 设计文档及验证报告

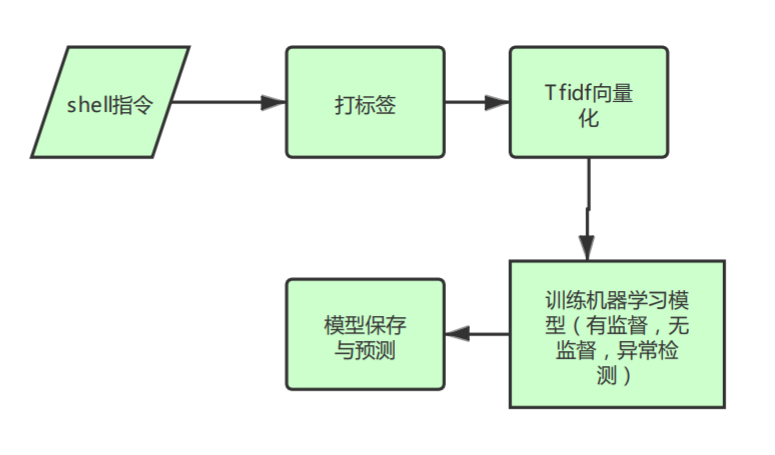
#### 功能目标

* 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

### 整体方案

#### 有监督学习

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。整体流程如下

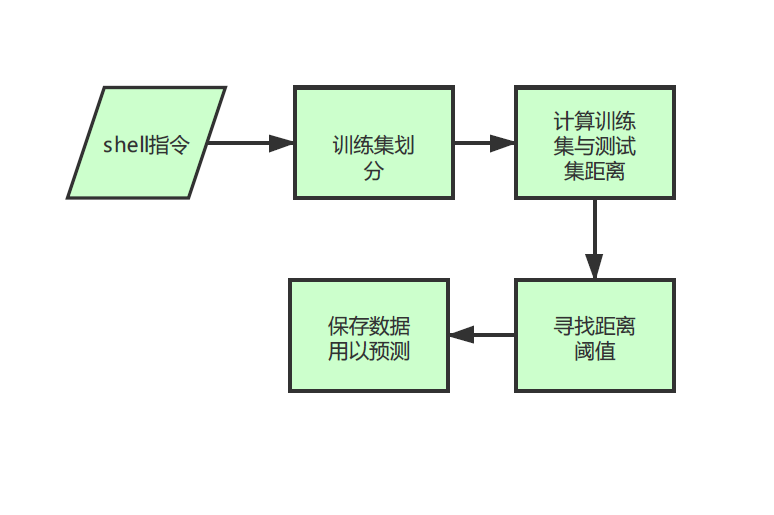


图表 1 传统命令分类流程

* 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

#### 无监督学习

无监督学习主要讨论基于n-gram距离的knn算法，其整体流程如下



图表 2 n-gram based knn算法流程

使用相对的n-gram距离衡量字符串差异，参考knn算法思想，如果一条指令与正常样本中的某条指令很相似（n-gram距离很小），这条指令大概率是一条正常的指令。我们计算待测指令与训练样本的距离，通过与距离阈值比较，判定指令是否恶意。

#### 流量告警抖动控制

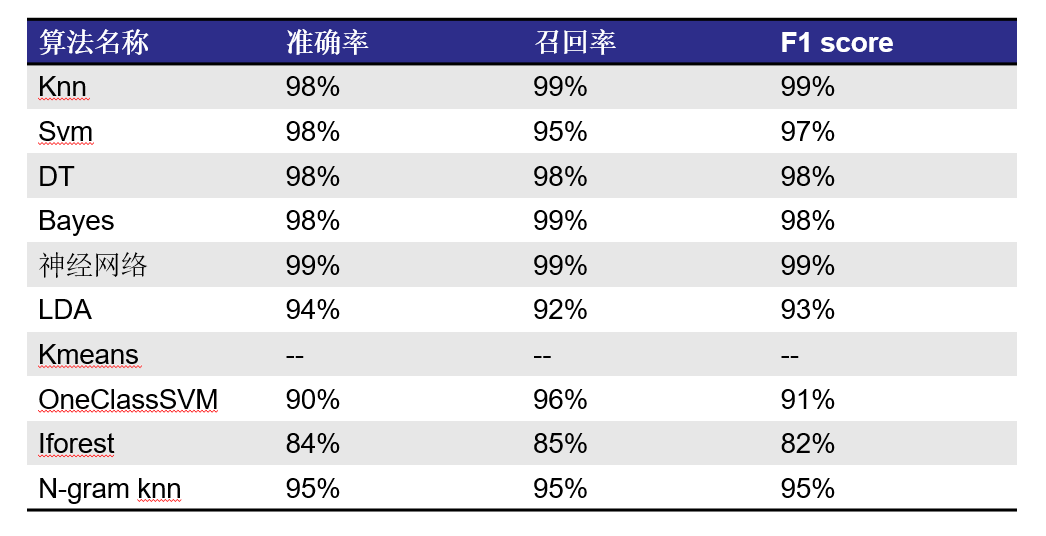
在有新业务产生时，出现数量众多且时间连续的恶意指令报警，此时程序应可识别此类情况。在出现此类业务场景时，恶意指令数量曲线总体呈阶梯状，所以当我们检测到连续五次检测的恶意指令数量大于某个阈值，就判定为新业务，一段时间内不再进行报警。

#### 命令注入

命令注入攻击的方式是将恶意指令附加在正常指令之后，针对此种情况，我们在进行预测时，先使用可能的命令连接符将指令分割，转换为单条指令的预测问题。

### 方案对比

各种检测算法结果如下



图表 3 检测算法准确率对比

众多的有监督学习算法表现良好，准确率与召回率都达到了96%以上，异常检测算法与无监督分类算法在此数据集上的表现不如有监督算法，只有90%。而且，对于完全未出现过的指令，这些算法的行为未知，在spark的算法环境中，无法检测出未出现过的恶意指令。

n-gram knn算法的准确率也达到了95%，而且具有可以检测未知指令的优势，所以我们选择基于n-gram距离的knn算法作为异常指令的解决方案。

### 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 |

### 具体检测方案

#### 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。

#### 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。

上线时将所有训练白样本放入同一个目录，指定白样本目录进行训练。

##### 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。数据为字符串格式的命令条目，字段同表1，每条指令一行。与日志文件格式相同。

#### 模型输出

模型输出为指令的预测到的恶意指令结果。输出数据帧中label列表示最终结果

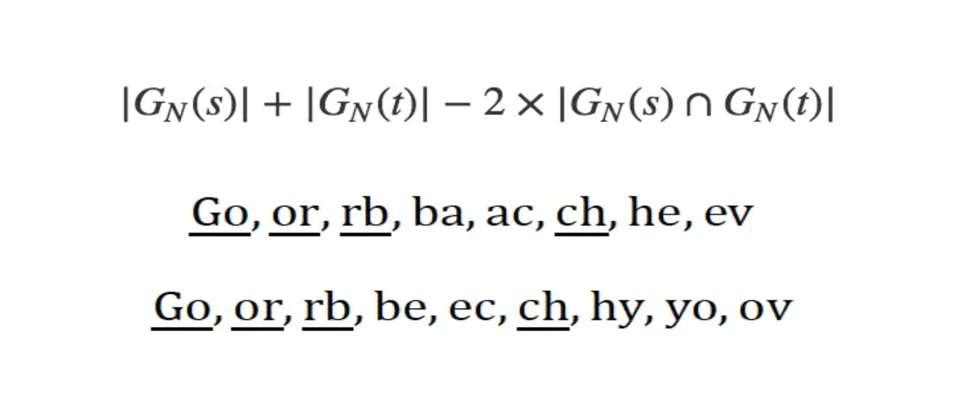
#### 数据来源

恶意样本来源于菜鸟教程和linuxdc.com的linux教程指令集，涵盖了linux系统管理的方方面面。训练时，先去除恶意样本与正常样本中相同的指令。

#### 检测原理

使用相对n-gram距离衡量字符串差异，使用knn算法思想，寻找指令与白样本数据集的最小距离，设定距离阈值，距离大于此阈值时判定指令为恶意。

n-gram距离参考如下图



图表 4 n-gram距离计算

图中的两个字符串之间的1-gram距离为8 + 9 – 2 x 4 = 8.

由于使用绝对n-gram距离在我们的应用场景中具有一定的局限性，所以我们引入相对n-gram距离，计算绝对n-gram距离以后除以两个命令长度之和，做归一化处理。以相对n-gram距离衡量字符串差异。

##### 命令语言字典

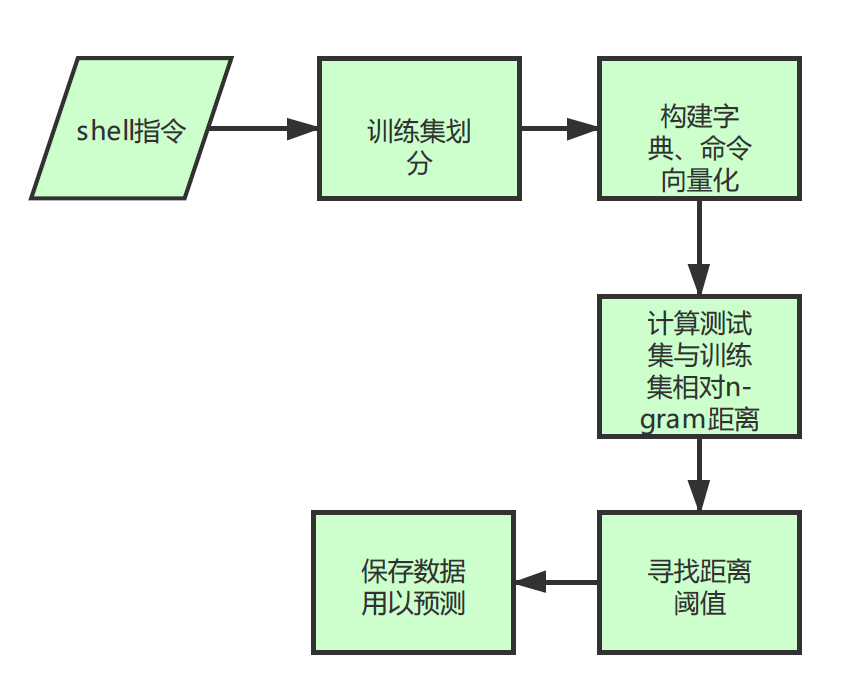
构建语言字典的目的在于将指令向量化为数字，在进行n-gram距离计算时，直接比较字符串会浪费大量不必要的计算资源，所以我们构建字典，将指令转换为向量，再计算n-gram距离。构建字典的过程为：指令分割后得到词，对每个未出现的词进行编号，遍历所有样本。

##### 命令向量化

命令向量化，使用我们构建的字典将指令转换为数字序列，以减少字符串比较的时间。

##### 训练过程

训练流程图如下



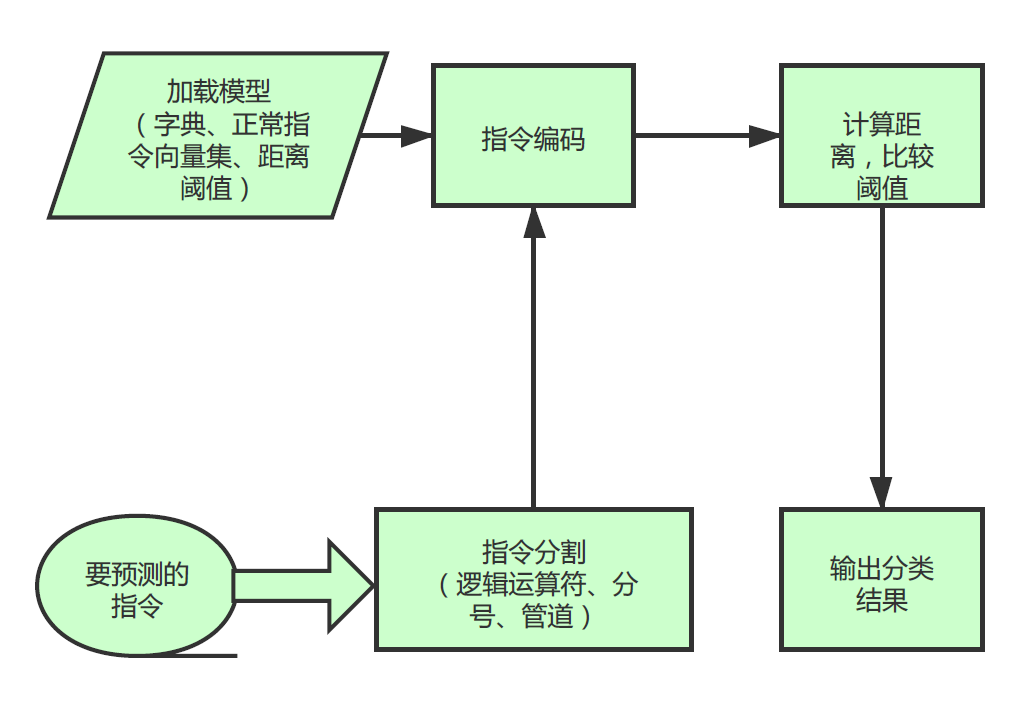
图表 5 算法训练流程

加载shell指令，经数据预处理后得到干净的命令数据，构建字典，对样本进行向量化处理。随后我们无放回的选择部分正常样本，与部分恶意样本作为测试集，计算测试集每条指令与训练集的最小距离，通过设置不同的距离阈值，我们可以得到一系列准确率与召回率，选择使我们数据准确率与召回率最高的距离阈值作为我们的判决标准。

训练完成以后，需要保存的数据包括：命令字典，经过向量化后的正常指令，我们找到的距离阈值。

##### 预测过程

预测流程如下



图表 6 算法预测流程

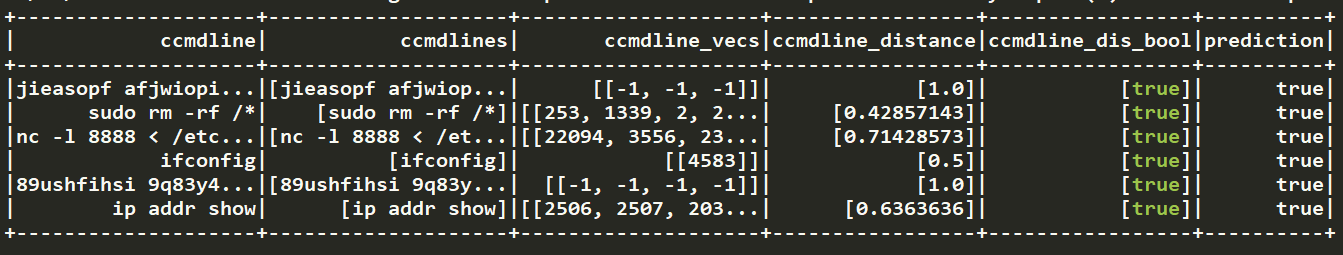
预测时，我们加载命令字典、向量化后的正常指令以及我们找到的阈值，接收到需要预测的指令后，我们使用特殊字符分割以覆盖注入攻击场景，计算指令与训练集的最小距离，大于距离阈值的判定为恶意指令。

## 结果与性能

测试环境：i5-8300H

模型在我们的数据集上达到了96%的准确率，在笔者的计算机spark单机环境处理2000条指令耗时52s。

检测输出格式如下



图表 7 指令检测结果

Prediction为true的列为恶意指令。

### 安全性

无监督学习算法在检测未知指令时具有良好的表现。但是同时也存在一些问题。由于恶意指令与正常样本数据集做的距离计算，如果正常样本中存在有恶意指令，那么会存在恶意指令漏报问题。还有一种攻击场景是，攻击者将正常指令与恶意指令糅合起来，错误的参数虽然会导致命令报错，但是正确参数的部分依然会被正确执行，此种情况单靠指令上的分类器已无法准确区分，需要我们考虑更加完整的解决方案，使用传统的规则匹配与机器学习算法结合，弥补安全性的不足。

## 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).