

这是作者的系列网络安全自学教程，主要是关于网安工具和实践操作的在线笔记，特分享出来与博友共勉，希望你们喜欢，一起进步。前文分享了Web渗透的第一步工作，涉及网站信息、域名信息、端口信息、敏感信息及指纹信息收集。这篇文章换个口味，将分享机器学习在安全领域的应用，并复现一个基于机器学习（逻辑回归）的恶意请求识别。

作者作为网络安全的小白，分享一些自学基础教程给大家，希望你们喜欢。同时，更希望你能与我一起操作深入进步，后续也将深入学习网络安全和系统安全知识并分享相关实验。总之，希望该系列文章对博友有所帮助，写文不容易，大神请飘过，不喜勿喷，谢谢！

下载地址：<https://github.com/eastmountyxz/NetworkSecuritySelf-study>

百度网盘：[https://pan.baidu.com/s/1dsunH8EmOB\\_tIHYYXGguOeA](https://pan.baidu.com/s/1dsunH8EmOB_tIHYYXGguOeA) 提取码：izeb

## 文章目录

### 一.安全领域中的机器学习

- 1.身份识别与认证
- 2.社会工程学
- 3.网络安全
- 4.Web安全
- 5.安全漏洞与恶意代码
- 6.入侵检测与防御

### 二.基于机器学习的恶意代码检测

- 1.传统的恶意代码检测
- 2.基于机器学习的恶意代码检测
- 3.机器学习在安全领域的特点及难点

### 三.逻辑回归识别网站恶意请求

- 1.数据集
- 2.N-grams和TF-IDF结合构造特征矩阵
- 3.训练模型
- 4.检测新数据集是恶意请求还是正常请求
- 5.完整代码

### 四.总结

## 前文学习：

[网络安全自学篇] 一.入门笔记之看雪Web安全学习及异或解密示例

[网络安全自学篇] 二.Chrome浏览器保留密码功能渗透解析及登录加密入门笔记  
[网络安全自学篇] 三.Burp Suite工具安装配置、Proxy基础用法及暴库示例  
[网络安全自学篇] 四.实验吧CTF实战之WEB渗透和隐写术解密  
[网络安全自学篇] 五.IDA Pro反汇编工具初识及逆向工程解密实战  
[网络安全自学篇] 六.OllyDbg动态分析工具基础用法及Crakeme逆向破解  
[网络安全自学篇] 七.快手视频下载之Chrome浏览器Network分析及Python爬虫探讨  
[网络安全自学篇] 八.Web漏洞及端口扫描之Nmap、ThreatScan和DirBuster工具  
[网络安全自学篇] 九.社会工程学之基础概念、IP获取、IP物理定位、文件属性  
[网络安全自学篇] 十.论文之基于机器学习算法的主机恶意代码  
[网络安全自学篇] 十一.虚拟机VMware+Kali安装入门及Sqlmap基本用法  
[网络安全自学篇] 十二.Wireshark安装入门及抓取网站用户名密码（一）  
[网络安全自学篇] 十三.Wireshark抓包原理（ARP劫持、MAC泛洪）及数据流追踪和图像抓取（二）  
[网络安全自学篇] 十四.Python攻防之基础常识、正则表达式、Web编程和套接字通信（一）  
[网络安全自学篇] 十五.Python攻防之多线程、C段扫描和数据库编程（二）  
[网络安全自学篇] 十六.Python攻防之弱口令、自定义字典生成及网站暴库防护  
[网络安全自学篇] 十七.Python攻防之构建Web目录扫描器及ip代理池（四）  
[网络安全自学篇] 十八.XSS跨站脚本攻击原理及代码攻防演示（一）  
[网络安全自学篇] 十九.Powershell基础入门及常见用法（一）  
[网络安全自学篇] 二十.Powershell基础入门及常见用法（二）  
[网络安全自学篇] 二十一.GeekPwn极客大赛之安全攻防技术总结及ShowTime  
[网络安全自学篇] 二十二.Web渗透之网站信息、域名信息、端口信息、敏感信息及指纹信息收集

### 前文欣赏：

[渗透&攻防] 一.从数据库原理学习网络攻防及防止SQL注入  
[渗透&攻防] 二.SQL MAP工具从零解读数据库及基础用法  
[渗透&攻防] 三.数据库之差异备份及Caidao利器  
[渗透&攻防] 四.详解MySQL数据库攻防及Fiddler神器分析数据包

### 该篇文章参考了以下文献，非常推荐大家阅读这些大牛的文章和视频：

机器学习在安全攻防场景的应用与分析 - 腾讯云FreeBuf官方

入侵某网站引发的安全防御思考 - 腾讯云“我是小三”大神

用机器学习玩转恶意URL检测 - 腾讯云FreeBuf官方

<https://github.com/exp-db/AI-Driven-WAF>

<https://github.com/foospidy/payloads>

<http://www.secrepo.com/>

<https://github.com/eastmountyxz>

张思思, 左信, 刘建伟. 深度学习中的对抗样本问题[J]. 计算机学报, 2019 (8) .

<http://fsecurify.com/fwaf-machine-learning-driven-web-application-firewall/>

黑产用“未来武器”破解验证码，打码小工都哭了 - FreeBuf

[转载] 机器学习科普文章：“一文读懂机器学习，大数据/自然语言处理/算法全有了”

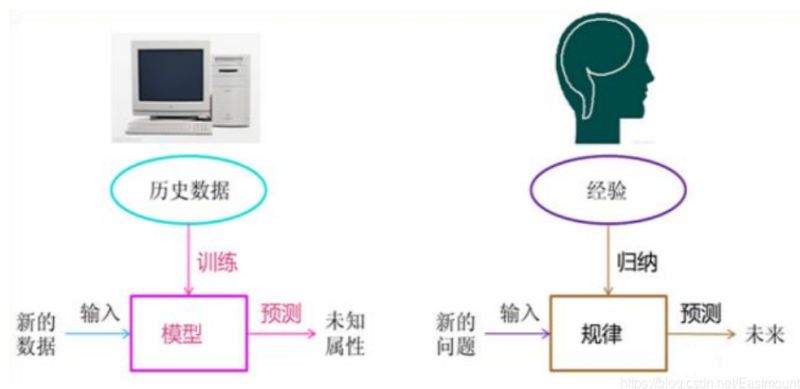
<https://www.bilibili.com/video/av60018118> (B站白帽黑客教程)

<https://www.bilibili.com/video/av63038037> (B站HACK学习)

声明：本人坚决反对利用教学方法进行犯罪的行为，一切犯罪行为必将受到严惩，绿色网络需要我们共同维护，更推荐大家了解它们背后的原理，更好地进行防护。

## 一.安全领域中的机器学习

机器学习方法是计算机利用已有的数据（经验），训练得出某种模型，并利用此模型预测未来的一种方法。机器学习学科融合了数学中的多个领域，主要包括统计学、概率论、线性代数以及数学计算。机器学习中的“训练”与“预测”过程可以对应到人类的“归纳”和“推测”过程，如下图所示。



机器学习和模式识别、统计学习、数据挖掘、计算机视觉，语音识别，自然语言处理等领域有着很深的联系。从范围上来说，机器学习跟模式识别、统计学习、数据挖掘是类似的，同时，机器学习与其他领域的处理技术的结合，形成了计算机视觉、语音识别、自然语言处理等交叉学科。一般说数据挖掘时，可以等同于说机器学习，我们平常所说的机器学习应用，应该是通用的，不仅仅局限在结构化数据，还有图像、音频、视频等应用。

- 模式识别 ≈ 机器学习 + 工业应用
- 数据挖掘 ≈ 机器学习 + 数据库
- 统计学习 ≈ 机器学习 + 数理统计
- 计算机视觉 ≈ 机器学习 + 图像处理 + 视频处理
- 语音识别 ≈ 机器学习 + 语音处理
- 自然语言处理 ≈ 机器学习 + 文本处理



机器学习能够深入挖掘大数据价值，被广泛用于各个领域，同时在网络安全领域也有相关的应用。为了更清晰地阐述机器学习在安全攻防领域的实际应用与解决方案，如下图所示，FreeBuf官网汇总了六大安全领域，分别是身份识别与认证、社会工程学、网络安全、Web安全、安全漏洞与恶意代码、入侵检测与防御，且在每一领域列举了典型的应用案例。



PS：下面这小部分内容引用FreeBuf的文章，推荐大家阅读。作者也尝试了总结，但总不尽如人意，看看大牛写得吧！

## 1.身份识别与认证

身份识别与认证是AI运用较多的领域，除了现有的各种人脸图像识别，语音声波识别，异常行为检测等AI应用之外，本部分将列举“验证码破解”与“恶意用户识别”两例。

### 身份认证——验证码破解

2017年6月，腾讯守护者计划安全团队协助警方打掉市面上最大打码平台“快啊答题”，挖掘出一条从撞库盗号、破解验证码到贩卖公民信息、实施网络诈骗的全链条黑产。在验

证时识别时，黑产运用 AI，极大提升了单位时间内识别验证码的数量，2017年一季度打码量达到259亿次，且识别验证码的精准度超过 80%。

在网络黑产中，不法分子窃取网站数据库后，需要确认帐号对应的密码是否正确，用撞库将有价值的数据通过验证的方式筛选出来，在这一过程中，最核心的障碍就是验证码安全体系。打码平台的AI系统，能将一张验证码图片作为一个整体，将单字识别转换成单图多标签、端到端的识别出验证码中的所有字符。此外还会通过搜集反馈回来的失败样本，以及人工打码的标定数据，来实时训练和更新识别网络，不断迭代训练进行优化，进一步提高神经网络模型的识别能力。因此，在面对网站验证时，还需要多种不同类型的验证方式，如图片选取，文字选择，图片填补等等，才能应对黑客日新月异的攻击破解手段。

### 行为分析——恶意用户识别

在分析用户行为时，从用户点击流数据中分析恶意用户的请求，特别地，可采用孤立森林（Isolation Forest）算法进行分类识别。在用户点击流数据中，包括请求时间、IP、平台等特征。孤立森林模型首先随机选择用户行为样本的一个特征，再随机选择该特征取值范围中的一个值，对样本集做拆分，迭代该过程，生成一颗孤立树；树上叶子节点离根节点越近，其异常值越高。迭代生成多颗孤立树，生成孤立森林，识别时，融合多颗树的结果形成最终的行为分类结果。

由于恶意用户仅占总体用户的少部分，具有异常样本“量少”和“与正常样本表现不一样”的两个特点，且不依赖概率密度，因此此异常检测模型不会导致高维输入的下溢出问题。该模型可识别异常用户盗号、LBS/加好友、欺诈等行为。随着样本增加，恶意请求的uin、类型、发生时间通过分析端通过线下人工分析和线上打击，达到良好的检测效果。

## 2.社会工程学

社会工程学是指攻击者利用某些手段使他人受骗的行为。除了现有的信用卡欺诈，信贷风险评估等AI应用，本部分将列举“鱼叉式网络钓鱼”与“欺诈电话识别”两例。

### 反钓鱼——鱼叉式网络钓鱼

2017年5月，Google利用机器学习技术，其垃圾邮件和网络钓鱼邮件的识别率已经达到了 99.9%。Google建立了一个系统。该系统可通过延迟Gmail信息的时间以执行更详细的网络钓鱼分析。当用户在浏览邮件的过程中，有关网络钓鱼的信息会更快被检测出来。利用 Google的机器学习，该系统还能随着时间的推移实时更新算法，从而可对数据和信息进行更深入的分析。不过，该系统仅适用于0.05%的信息。

区别于普通网络钓鱼，鱼叉式网络钓鱼是针对特定目标进行定制的网络钓鱼攻击。黑客会从社交媒体、新闻报道等资料中对攻击目标的信息中，采用机器学习的方法进行前期的分析，包括姓名、邮箱地址、社交媒体账号或者任何在网上参与过的内容等。攻击对象通常不对于普通用户，而是特定的公司或者组织的成员，窃取的资料也并非个人的资料，而是其他高度敏感性资料。面对鱼叉钓鱼，一方面企业会加强网站的数据保护，防



各种爬虫工具，通过逆向分析，并采用机器学习进行垃圾/钓鱼邮件的检测过滤，另一方面用户自身提高安全意识注意个人隐私泄露，保持警惕性。

反欺诈——欺诈电话识别

这几年，在通信诈骗方面的犯罪愈演愈烈，仅2015年的报案数据，如“猜猜我是谁”，“冒充公检法”此类涉及电话诈骗的案件，全国用户损失就约220亿左右。在应对通信欺诈，通常分为事后处置与实时阻断两种解决方法，而由于事后处置的时效性太低，诈骗资金往往已被转移，无法很好地起到保护公民财产的作用。因此实时阻断十分必要，当用户接打电话，通过机器学习，能够实时发现是否属于诈骗电话，并立刻发出实时告警。

从号码活跃特征数据、号码的社交网络、号码的行为事件流、号码的行为特征、号码信用度、号码异常度等方面来进行特征抽取，根据机器学习架构检测。此外，再结合事件模型与行为模式的关联分析，能更准确地对欺诈电话进行监测。



3.网络安全

网络安全是指网络系统软硬件受保护，网络服务不中断。除了现有的隐藏信号识别等AI应用，本部分将列举“大数据DDoS检测”与“伪基站短信识别”两例。

抗DDoS——大数据DDoS检测

近年来，基于机器学习算法的分布式拒绝服务（distributeddenial-of-service，简称DDoS）攻击检测技术已取得了很大的进展。在攻击感知方面，可从宏观攻击流感知与微观检测方法两个角度，分别基于IP流序列谱分析的泛洪攻击与低速率拒绝服务（Low-rate Denial of Service，LDoS）方法进行感知。在此基础上，将DDoS攻击检测转化为机器学习的二分类问题。

从概率点判别角度，基于多特征并行隐马尔科夫模型（Multi-FeatureParallel Hidden Markov Model，MFP-HMM）的DDoS攻击检测方法，利用HMM隐状态序列与特征观测序列的对应关系，将攻击引起的多维特征异常变化转化为离散型随机变量，通过概率计算来刻画当前滑动窗口序列与正常行为轮廓的偏离程度。从分类超平面判别角度，基于最小二乘孪生支持向量机（LSTSVM）的DDoS攻击分类超平面检测方法，采用IP包五

元组熵、IP标识、TCP头标志和包速率等作为LSTSVM模型的多维检测特征向量，以体现DDoS攻击存在的流分布特性。

### 无线网络攻击——伪基站短信识别

为了解决“犯罪分子通过冒充10086、95533等机构发送短信来获得用户的账号、密码和身份证等信息”这一问题。2016年，360手机依托360公司研发的伪基站追踪系统，率先在全球推出了伪基站诈骗短信识别功能，拦截准确度达98%，可有力的确保用户财产安全。360伪基站追踪系统的核心价值就在于它解决上述伪基站打击难题，依托海量的数据、高效的数据分析处理和数据可视化，可以为追查伪基站供精确的信息与准确的判断。

2015年12月，360手机在全球率先推出了伪基站垃圾、诈骗短信精准识别功能。由于垃圾和诈骗短信的识别和分类涉及到自然语言处理技术与机器学习模型，360使用语法规则与统计学方法相结合的方式定义伪基站短信特征，可从海量数据中精确识别出伪基站短信，因而其识别精度可达98%。对于360伪基站追踪系统的发布、部署，以及其在360手机中的成功运用，有力遏制猖獗的伪基站诈骗活动，有助于维护广大手机用户及其他群众的财产安全。

## 4.Web安全

Web安全是指个人用户在Web相关操作时不因偶然或恶意的原因受到破坏、更改、泄露。除了现有的SQL注入检测、XSS攻击检测等AI应用，本部分将列举“恶意URL检测”与“Webshell检测”两例。后续实验部分，作者将详细描述Python实现该过程。

### 安全网站检测——恶意URL检测

在市面上，Google的Chrome已将检测模型与机器学习相结合，支持安全浏览，向用户警示潜在的恶意网址。结合成千上万的垃圾邮件、恶意软件、有启发式信号的含勒索软件的附件和发送者的签名（已被标识为恶意的），对新的威胁进行识别和分类。

目前大多数网站检测方式是通过建立URL黑白名单的数据库匹配进行排查，虽然具有一定的检测效果，但有一定滞后性，不能够对没有记录在案的URL进行识别。而基于机器学习，从URL特征，域名特征，Web特征的关联分析，使恶意URL识别具有高准确率，并具有学习推断的能力。一些开源工具如Phinn提供了另一个角度的检测方法，如果一个页面看起来非常像Google的登录页面，那么这个页面就应该托管在Google域名。Phinn使用了机器学习领域中的卷积神经网络算法来生成和训练一个自定义的Chrome扩展，这个Chrome扩展可以将用户浏览器中呈现的页面与真正的登录页面进行视觉相似度分析，以此来识别出恶意URL（钓鱼网站）。

### 注入攻击检测——Webshell检测

Webshell常常被称为匿名用户（入侵者）通过网站端口对网站服务器的某种程度上操作的权限。由于Webshell其大多是以动态脚本的形式出现，也有人称之为网站的后门工具。在攻击链模型中，整个攻击过程分为：踩点、组装、投送、攻击、植入、控制、行

动。在针对网站的攻击中，通常是利用上传漏洞，上传Webshell，然后通过Webshell进一步控制web服务器。

常见传统的Webshell检测方法主要有静态检测、动态检测、语法检测、统计学检测等。随着AI的兴起，基于AI的Webshell文件特征检测技术要较之传统技术更胜一筹，通过词袋&TF-IDF模型、Opcode&N-gram模型、Opcode调用序列模型等特征抽取方式，采用合适的模型，如朴素贝叶斯和深度学习的MLP、CNN等，实现Webshell的检测。类似地，也可进行SQL注入、XSS攻击检测等。

## 5.安全漏洞与恶意代码

安全漏洞是指漏洞是在硬件、软件、协议的具体实现或系统安全策略上存在的缺陷；恶意代码是指具有安全威胁的代码。除了现有的恶意软件检测与识别等AI应用，本部分将列举“恶意代码分类”与“系统自动化漏洞修补”两例。

### 代码安全——恶意代码分类

早期反病毒软件无论是特征码扫描、查找广谱特征、启发式扫描，这三种查杀方式均没有实际运行二进制文件，因此均可归为恶意代码静态检测的方法。随着反恶意代码技术的逐步发展，主动防御技术、云查杀技术已越来越多的被安全厂商使用，但恶意代码静态检测的方法仍是效率最高，被运用最广泛的恶意代码查杀技术。

2016年在Kaggle上微软发起了一个恶意代码分类比赛，冠军队采用了一种恶意代码图像绘制方法。将一个二进制文件转换为一个矩阵（矩阵元素对应文件中的每一个字节，矩阵的大小可根据实际情况进行调整），该矩阵又可以非常方便的转换为一张灰度图。再基于N-gram，统计概率模型。最后代入分类决策树与随机森林进行训练与测试。这个方法能够发现一些静态方法发现不了的变种，并且也可推广应用到Android和IOS平台的恶意代码检测中。

### 漏洞修复——系统自动化漏洞修补

2016年8月，DARPA在DEFCON黑客大会上举办Cyber Grand Challenge挑战赛，要求参赛者在比赛中构建一套智能化的系统，不仅要检测漏洞，还要能自动写补丁、并且完成部署。当今的软件漏洞平均发现周期长达312天，发现后还需要对漏洞研究、开发补丁程序，到最后公布，在这期间，攻击者很有可能已经利用这个漏洞发起网络攻击。因此系统自动化漏洞修复十分必要。

2017年10月，MIT研究团队研发了一个称为“创世纪”的系统，能够对以前的补丁进行自动学习，生成补丁模板，并对候选补丁进行评估。据研究者说，“创世纪是第一个自动推理补丁生成转换或根据先前成功的补丁搜索候选补丁空间的系统”，它修复的bug几乎是最好的手编模板系统的两倍，同时也更精确。这些模板是根据真实补丁的特定类型“订制”而成，因此不会产生尽可能多的无用备选。

## 6.入侵检测与防御



入侵检测与防御是指对入侵行为的发现并采取相应的防御行动。除了现有的内网入侵检测等AI应用，本部分将列举“APT检测与防范”与“C2链接分析”两例。

### 高级攻击入侵检测——APT检测与防范

进行APT攻击的攻击者从侦查目标，制作攻击工具，传递攻击工具，利用漏洞或者弱点来进行突防，拿下全线运行工具，后期远端的维护这个工具，到最后达到了长期控制目标的目的。针对这种现在日益广泛的APT攻击，威胁情报存在于整个攻击的各个环节。

威胁情报是基于证据的描述威胁的一组关联的信息，包括威胁相关的环境信息，如具体的攻击组织、恶意域名。恶意域名又包括远控的IOC、恶意文件的HASH和URL以及威胁指标之间的关联性，时间纬度上攻击手法的变化。这些信息汇总在一起形成高级威胁情报。除此之外，所关注的情报，还包括传统威胁种类的扩充，包括木马远控，僵尸网络，间谍软件，Web后门等。利用机器学习来处理威胁情报，检测并识别出APT攻击中的恶意载荷，提高APT攻击威胁感知系统的效率与精确性，让安全研究人员能更快实现APT攻击的发现和溯源。

### DGA域名检测——C2链接分析

DGA（域名生成算法）是一种利用随机字符来生成C2域名，从而逃避域名黑名单检测的技术手段。而有了DGA域名生成算法，攻击者就可以利用它来生成用作域名的伪随机字符串，这样就可以有效的避开黑名单列表的检测。伪随机意味着字符串序列似乎是随机的，但由于其结构可以预先确定，因此可以重复产生和复制。该算法常被运用于远程控制软件上。

首先攻击者运行算法并随机选择少量的域（可能只有一个），然后攻击者将该域注册并指向其C2服务器。在受害者端恶意软件运行DGA并检查输出的域是否存在，如果检测为该域已注册，那么恶意软件将选择使用该域作为其命令和控制（C2）服务器。如果当前域检测为未注册，那么程序将继续检查其它域。因此，安全人员可以通过收集样本以及对DGA进行逆向，来预测哪些域将来会被生成和预注册并将它们列入黑名单中。

---

## 二.基于机器学习的恶意代码检测

### 1.传统的恶意代码检测

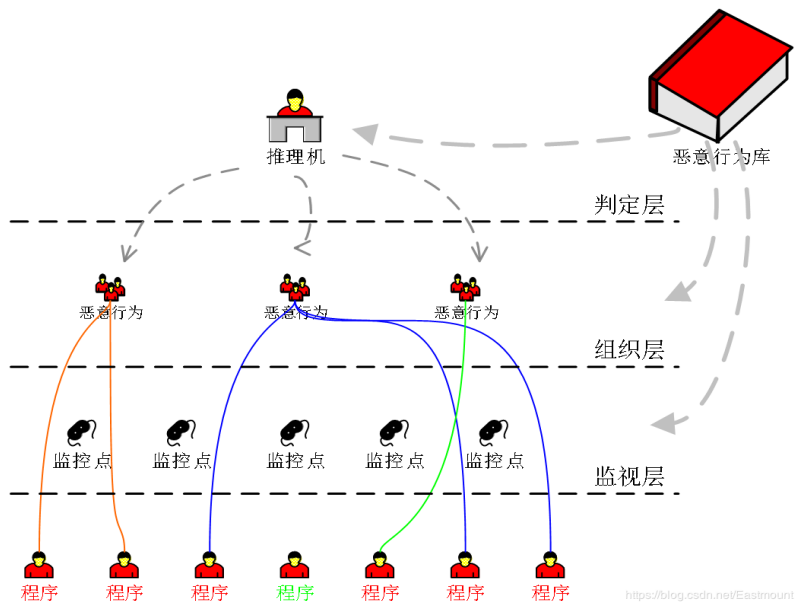
传统的恶意代码检测包括基于签名特征码（signature）的检测和基于启发式规则（heuristic）的检测，在应对数量繁多的未知恶意代码时，正面临越来越大的挑战。

#### (1) 基于签名特征码的检测

签名特征码检测方法通过维护一个已知的恶意代码库，将待检测代码样本的特征码与恶意代码库中的特征码进行比对，如果特征码出现匹配，则样本为恶意代码。该方法需要耗费大量的人力、物力对恶意代码进行研究并要求用户及时更新恶意代码库，检测效率和效果越来越力不从心，并且很难有效抵御未知恶意代码。

(2) 基于启发式规则的检测

启发式规则检测方法通过专业的分析人员对现有的恶意代码进行规则提取，并依照提取出的规则对代码样本进行检测。但面对现阶段恶意代码爆炸式的增长趋势，仅依赖人工进行恶意代码分析，在实施上变得愈发困难。



2.基于机器学习的恶意代码检测

基于机器学习算法的防护技术为实现高准确率、自动化的未知恶意代码检测提供了行之有效的技术途径，已逐渐成为业内研究的热点。根据检测过程中样本数据采集角度的不同，可以将检测分为：静态分析与动态分析。

静态分析不运行待检测程序，而是通过程序（如反汇编后的代码）进行分析得到数据特征，而动态分析在虚拟机或仿真器中执行程序，并获取程序执行过程中所产生的数据（如行为特征），进行检测和判断。

根据 Cohen 对恶意代码的研究结果，可知恶意代码检测的本质是一个分类问题，即把待检测样本区分成恶意或合法的程序。其核心步骤为：

- 采集数量充分的恶意代码样本
- 对样本进行有效的数据处理，提取特征
- 进一步选取用于分类的主要数据特征
- 结合机器学习算法的训练，建立分类模型
- 通过训练后的分类模型对未知样本进行检测

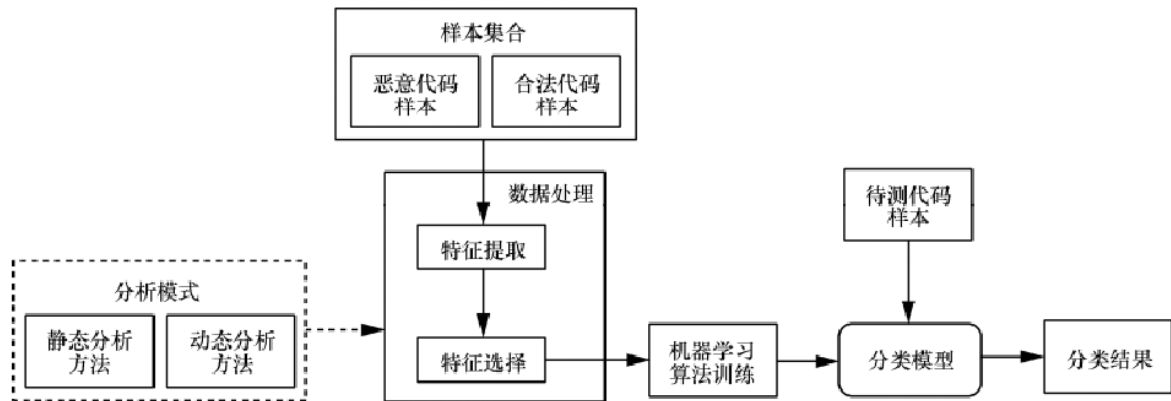


图1 基于机器学习算法的恶意代码检测步骤

<https://blog.csdn.net/Eastmount>

详见作者文章：[\[网络安全自学篇\] 十.论文之基于机器学习算法的主机恶意代码](#)

### 3.机器学习在安全领域的特点及难点

机器学习是个多元学科，其本质是在数据中进行学习，通过合适的算法建模，最终在无规则的情况下，实现分类、聚类或是预测。从第一部分的案例可以看出，机器学习在安全攻防最常应用于在于恶意代码识别、社工安全防范，入侵攻击检测这三大方向。

- **在恶意代码识别方面：** 区别传统的黑白名单库、特征检测、启发式等方法机器学习的安全应用从反病毒的代码分类、恶意文件检测、恶意URL的网页代码识别等
- **在社工安全防范方面：** 区别传统的技术与业务经验分析、安全宣传、金融模型等评估方法，机器学习的安全应用从鱼叉式网络钓鱼检测，恶意用户点击流识别，欺诈电话与短信分析，到金融信用欺诈等
- **在入侵攻击检测方面：** 区别传统的基于规则与策略、正则匹配等，机器学习的安全应用从DDoS防御，webshell检测，DGA防范到APT检测等等。

总体上，即使机器学习在训练模型后无法达到百分百的效果，但相比传统手段，均有不同程度地检测效果提升。

虽然机器学习技术在安全领域已有诸多场景应用，为现有的用户安全防护策略提供了新的视角。从上述的案例中不难看出，机器学习在安全与风控方面应用难点主要包括如：

- 机器学习需要尽可能**平衡的高质量数据集**，而在安全领域，无论是风险欺诈、网络钓鱼、恶意软件等，通常包含大量的正常样本与极少量的安全隐患，因此恶意访问、攻击样本的不充分，导致模型训练后的检测准确率有待提高。
- 机器学习的模型一般均为**黑盒分析**，无法得到足够的信息。不像其他AI应用（如商品推荐系统），在应用安全领域的模型分类错误具有极高的成本，并且在面对网络威胁与隐患时，安全分析人员希望在网络对抗中取得对形势的了解与情报的掌握，以作出相应的人工干预。
- 现阶段所有需监督学习的机器学习模型，均需要输入合理且高相关的特征集，即需要从源数据到特征空间映射的特征工程。在安全领域，会产生网络监控到实际的检

测对象之间的抽象成本，如**软件缺陷与底层实现代码与结构之间的对应关系有一个抽象、翻译的难度**。

与此同时，机器学习作为新兴的前沿技术，即使解决或克服传统安全攻防技术的问题与难点，在一些场景与环境下，仍有无法避免的缺陷或者是即使解决了问题也无法满足实际需求，即无法采用机器学习算法进行安全攻防的盲点。

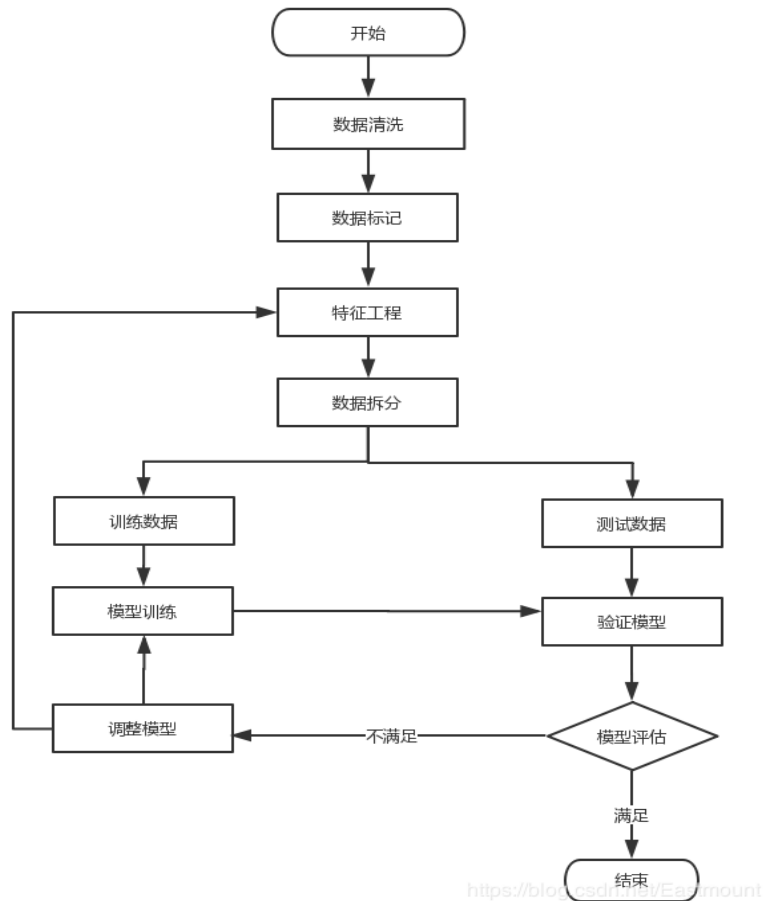
- 无法发现未知模式的恶意行为
- 误报大量测试异常的正常行为
- 对数据数量与质量有强依赖性

---

## 三.逻辑回归识别网站恶意请求

接下来作者复现了Github上exp-db大神的代码，推荐大家阅读之前的参考文献中大神的作品。该代码的基本思想是通过机器学习（逻辑回归）建立检测模型，从而识别网站的恶意请求和正常请求。基本流程如下图所示：

- 读取正常请求和恶意请求数据集，预处理设置类标y和数据集x
- 通过N-grams处理数据集，并构建TF-IDF特征矩阵，每个请求对应矩阵的一行数据
- 数据集拆分为训练数据和测试数据
- 使用机器学习逻辑回归算法对特征矩阵进行训练，得出对应的模型
- 使用训练的模型对未知URL请求进行检测，判断其是恶意请求或正常请求



<https://blog.csdn.net/Eastmount>

## 1.数据集

在<https://github.com/foospidy/payloads>中收集了常见的网站恶意请求，如SQL注入、XSS攻击等的Payload。实验数据包括：

- 正常请求：goodqueries.txt，1265974条，来自<http://secrepo.com>网站日志请求
- 恶意请求：badqueries.txt，44532条，XSS、SQL注入等攻击的payload

注意，资源和精力有限，数据集假定<http://secrepo.com>网站的日志请求全部都是正常的请求，有精力可以进行降噪处理，去除异常的标签数据。

C:\Users\yxz\Desktop\AI-Driven-WAF-master

正常请求: 1265974

/javascript/showcat.lst

/ofnc171120034592o/

/159325/

/dirty\_girl/

/javascript/vol.class

C:\Users\yxz\Desktop\AI-Driven-WAF-master

恶意请求: 44532

/examples/jsp/colors/cgiwrap/cgiwrap\_error\_page\_handling\_xss.nasl

</script/x><scr<script>ipt>alert(1)</script/x>

/examples/jsp/checkbox/directorypro.cgi?want=showcat&show=../../../../../etc/passwd\x00

/actscript2/

/vb/includes/functions\_cron.php?nextitem=XXpathXX

<https://blog.csdn.net/Eastmount>

该部分的核心代码为:

```
import os
import urllib

# 获取文本中的请求列表
def get_query_list(filename):
    directory = str(os.getcwd())
    print(directory)
    filepath = directory + "/" + filename
    data = open(filepath, 'r', encoding='UTF-8').readlines()
    query_list = []
    for d in data:
        # 解码
        d = str(urllib.parse.unquote(d))    #converting url encoded data to
        #print(d)
        query_list.append(d)
    return list(set(query_list))

# 主函数
if __name__ == '__main__':

    # 获取正常请求
    good_query_list = get_query_list('goodqueries.txt')
    print(u"正常请求: ", len(good_query_list))
    for i in range(0, 5):
        print(good_query_list[i].strip('\n'))
    print("\n")

    # 获取恶意请求
    bad_query_list = get_query_list('badqueries.txt')
    print(u"恶意请求: ", len(bad_query_list))
    for i in range(0, 5):
        print(bad_query_list[i].strip('\n'))
    print("\n")

    # 预处理 good_y 标记为0 bad_y 标记为1
    good_y = [0 for i in range(0, len(good_query_list))]
    print(good_y[:5])
    bad_y = [1 for i in range(0, len(bad_query_list))]
    print(bad_y[:5])

    queries = bad_query_list + good_query_list
    y = bad_y + good_y
```

## 2.N-grams和TF-IDF结合构造特征矩阵



本段代码的一个亮点是将N-grams和TF-IDF结合来构造特征矩阵。作者前文：[\[python\]使用scikit-learn工具计算文本TF-IDF值](#)

TF-IDF (Term Frequency-InversDocument Frequency) 是一种常用于信息处理和数据挖掘的加权技术。该技术采用一种统计方法，根据字词的在文本中出现的次数和在整个语料中出现的文档频率来计算一个字词在整个语料中的重要程度。它的优点是能过滤掉一些常见的却无关紧要本的词语，同时保留影响整个文本的重要字词。计算方法如下面公式所示。

$$tfidf_{i,j} = tf_{i,j} \times idf_{i,j}$$

其中，式中tfidf表示词频tf和倒文本词频idf的乘积。TF-IDF值越大表示该特征词对这个文本的重要性越大。其基本思想是将文本转换为特征矩阵，并且降低常用词（如we、all、www等）的权重，从而更好地表达一个文本的价值。如下图示例：

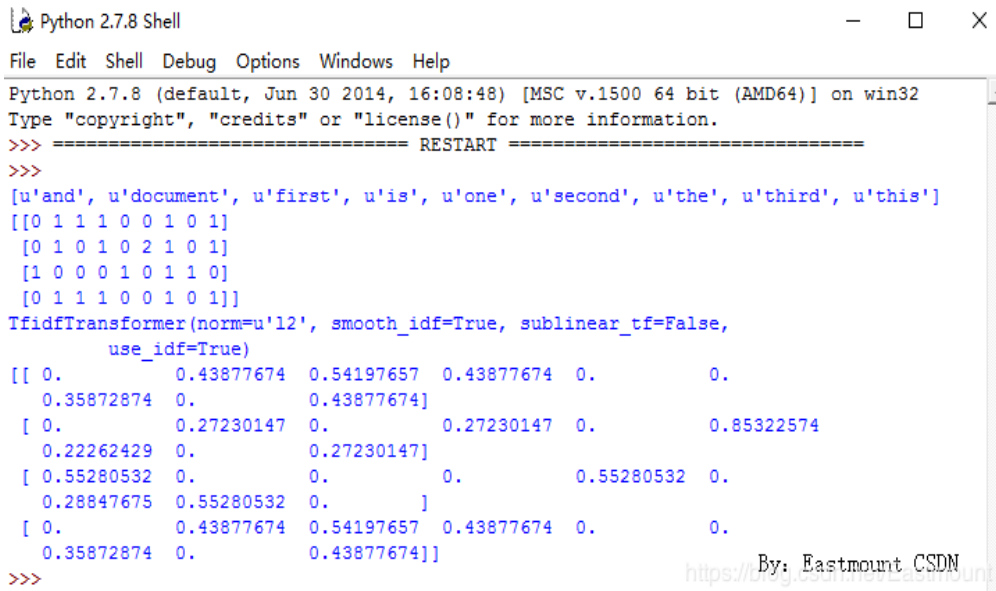
```
# coding:utf-8
from sklearn.feature_extraction.text import CountVectorizer

# 语料
corpus = [
    'This is the first document.',
    'This is the second second document.',
    'And the third one.',
    'Is this the first document?',
]

# 将文本中的词语转换为词频矩阵
vectorizer = CountVectorizer()
# 计算个词语出现的次数
X = vectorizer.fit_transform(corpus)
# 获取词袋中所有文本关键词
word = vectorizer.get_feature_names()
print word
# 查看词频结果
print X.toarray()

from sklearn.feature_extraction.text import TfidfTransformer

# 类调用
transformer = TfidfTransformer()
print transformer
# 将词频矩阵X统计成TF-IDF值
tfidf = transformer.fit_transform(X)
# 查看数据结构 tfidf[i][j]表示i类文本中的tf-idf权重
print tfidf.toarray()
```



```
Python 2.7.8 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.8 (default, Jun 30 2014, 16:08:48) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
[u'and', u'document', u'first', u'is', u'one', u'second', u'the', u'third', u'this']
[[0 1 1 1 0 0 1 0 1]
 [0 1 0 1 0 2 1 0 1]
 [1 0 0 0 1 0 1 1 0]
 [0 1 1 1 0 0 1 0 1]]
TfidfTransformer(norm='l2', smooth_idf=True, sublinear_tf=False,
use_idf=True)
[[ 0.         0.43877674  0.54197657  0.43877674  0.         0.
  0.35872874  0.         0.43877674]
 [ 0.         0.27230147  0.         0.27230147  0.         0.85322574
  0.22262429  0.         0.27230147]
 [ 0.55280532  0.         0.         0.         0.55280532  0.
  0.28847675  0.55280532  0.         ]
 [ 0.         0.43877674  0.54197657  0.43877674  0.         0.
  0.35872874  0.         0.43877674]]
>>>
```

无论是恶意请求数据集还是正常请求数据集，都是不定长的字符串列表，很难直接用逻辑回归算法对这些不规律的数据进行处理，需要找到这些文本的数字特征，用来训练我们的检测模型。在这里，使用TD-IDF来作为文本的特征，并以数字矩阵的形式进行输出。在计算TD-IDF之前，首先需要对每个文档（URL请求）的内容进行分词处理，也就是需要定义文档的词条长度，这里我们选择长度为3的N-grams，可以根据模型的准确度对这个参数进行调整。

该部分的核心代码如下，详见注释：

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

# tokenizer function, this will make 3 grams of each query
# www.foo.com/1 转换为 ['www', 'ww.', 'w.f', '.fo', 'foo', 'oo.', 'o.c', '.co', '(
def get_ngrams(query):
    tempQuery = str(query)
    ngrams = []
    for i in range(0, len(tempQuery)-3):
        ngrams.append(tempQuery[i:i+3])
    return ngrams

# 主函数
if __name__ == '__main__':
    ....
    # 定义矢量化 converting data to vectors
    # TfidfTransformer + CountVectorizer = TfidfVectorizer
    vectorizer = TfidfVectorizer(tokenizer=get_ngrams)

    # 把不规律的文本字符串列表转换成规律的 ( [i,j], tfidf值) 的矩阵X
    # 用于下一步训练逻辑回归分类器
```

```
X = vectorizer.fit_transform(queries)
print(X.shape)
```

### 3.训练模型

通过构建的特征矩阵作为训练集，调用逻辑回归进行训练和测试，Python中机器学习两个核心函数为fit()和predict()。这里，调用train\_test\_split()函数将数据集随机划分，核心代码如下所示：

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

# 主函数
if __name__ == '__main__':
    ....
    # 使用 train_test_split 分割 X y 列表
    # X_train矩阵的数目对应 y_train列表的数目(一一对应) --> 用来训练模型
    # X_test矩阵的数目对应      (一一对应) --> 用来测试模型的准确性
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

    # 定理逻辑回归方法模型
    LR = LogisticRegression()

    # 训练模型
    LR.fit(X_train, y_train)

    # 使用测试值 对 模型的准确度进行计算
    print('模型的准确度:{}'.format(LR.score(X_test, y_test)))
    print("\n")
```

### 4.检测新数据集是恶意请求还是正常请求

模型训练好之后，发现其精确度挺高的，真实的实验还需要通过准确率、召回率和F值判断。接下来调用Predict()函数对新的RUL进行判断，检测其是恶意请求还是正常请求。核心代码如下：

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

# 主函数
if __name__ == '__main__':
    ....
```

# 对新的请求列表进行预测

```
new_queries = ['www.foo.com/id=1<script>alert(1)</script>',
               'www.foo.com/name=admin\' or 1=1', 'abc.com/admin.php',
               '><svg onload=confirm(1)>',
               'test/q=<a href="javascript:confirm(1)>',
               'q=../etc/passwd',
               '/stylesheet.php?version=1331749579',
               '/<script>cross_site_scripting.nasl</script>.idc',
               '<img \x39src=x onerror="javascript:alert(1)">',
               '/jhot.php?rev=2 |less /etc/passwd']
```

# 矩阵转换

```
X_predict = vectorizer.transform(new_queries)
res = LR.predict(X_predict)
res_list = []
```

# 结果输出

```
for q,r in zip(new_queries, res):
    tmp = '正常请求' if r == 0 else '恶意请求'
    q_entity = html.escape(q)
    res_list.append({'url':q_entity, 'res':tmp})

for n in res_list:
    print(n)
```

最终输出结果如下图所示，可以发现其判断较为准确。

```
{'url': 'www.foo.com/id=1<script>alert(1)</script>', 'res': '恶意请求'}
{'url': 'www.foo.com/name=admin&#x27; or 1=1', 'res': '恶意请求'}
{'url': 'abc.com/admin.php', 'res': '正常请求'}
{'url': '&quot;&gt;&lt;svg onload=confirm(1)&gt;', 'res': '恶意请求'}
{'url': 'test/q=&lt;a href=&quot;javascript:confirm(1)&gt;', 'res': '恶意请求'}
{'url': 'q=../etc/passwd', 'res': '恶意请求'}
{'url': '/stylesheet.php?version=1331749579', 'res': '正常请求'}
{'url': '/&lt;script>cross_site_scripting.nasl&lt;/script>.idc', 'res': '恶意请求'}
{'url': '&lt;img 9src=x onerror=&quot;javascript:alert(1)&quot;&gt;', 'res': '恶意请求'}
{'url': '/jhot.php?rev=2 |less /etc/passwd', 'res': '恶意请求'}
```

<https://blog.csdn.net/Easymount>

## 5.完整代码

完整代码如下，并推荐大家去Github学习很多有些的代码，也推荐大家去FreeBuf、安全客、CVE等网站学习。作者Github有完整代码：

<https://github.com/eastmountyxz/NetworkSecuritySelf-study>

```
# coding: utf-8
import os
import urllib
```

```

import time
import html
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

# 获取文本中的请求列表
def get_query_list(filename):
    directory = str(os.getcwd())
    print(directory)
    filepath = directory + "/" + filename
    data = open(filepath, 'r', encoding='UTF-8').readlines()
    query_list = []
    for d in data:
        # 解码
        d = str(urllib.parse.unquote(d)) #converting url encoded data to normal
        #print(d)
        query_list.append(d)
    return list(set(query_list))

# tokenizer function, this will make 3 grams of each query
# www.foo.com/1 转换为 ['www', 'ww.', 'w.f', '.fo', 'foo', 'oo.', 'o.c', '.co', 'com']
def get_ngrams(query):
    tempQuery = str(query)
    ngrams = []
    for i in range(0, len(tempQuery)-3):
        ngrams.append(tempQuery[i:i+3])
    return ngrams

# 主函数
if __name__ == '__main__':

    # 获取正常请求
    good_query_list = get_query_list('goodqueries.txt')
    print(u"正常请求: ", len(good_query_list))
    for i in range(0, 5):
        print(good_query_list[i].strip('\n'))
    print("\n")

    # 获取恶意请求
    bad_query_list = get_query_list('badqueries.txt')
    print(u"恶意请求: ", len(bad_query_list))
    for i in range(0, 5):
        print(bad_query_list[i].strip('\n'))
    print("\n")

    # 预处理 good_y 标记为0 bad_y 标记为1

```

```

good_y = [0 for i in range(0, len(good_query_list))]
print(good_y[:5])
bad_y = [1 for i in range(0, len(bad_query_list))]
print(bad_y[:5])

queries = bad_query_list + good_query_list
y = bad_y + good_y

# 定义矢量化 converting data to vectors
# TfidfTransformer + CountVectorizer = TfidfVectorizer
vectorizer = TfidfVectorizer(tokenizer=get_ngrams)

# 把不规则的文本字符串列表转换成规律的 ( [i,j], tfidf值) 的矩阵X
# 用于下一步训练逻辑回归分类器
X = vectorizer.fit_transform(queries)
print(X.shape)

# 使用 train_test_split 分割 X y 列表
# X_train矩阵的数目对应 y_train列表的数目(一一对应) --> 用来训练模型
# X_test矩阵的数目对应 (一一对应) --> 用来测试模型的准确性
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=)

# 定理逻辑回归方法模型
LR = LogisticRegression()

# 训练模型
LR.fit(X_train, y_train)

# 使用测试值 对 模型的准确度进行计算
print('模型的准确度:{}'.format(LR.score(X_test, y_test)))
print("\n")

# 对新的请求列表进行预测
new_queries = ['www.foo.com/id=1<script>alert(1)</script>',
               'www.foo.com/name=admin\' or 1=1', 'abc.com/admin.php',
               '><svg onload=confirm(1)>',
               'test/q=<a href="javascript:confirm(1)>',
               'q=../etc/passwd',
               '/stylesheet.php?version=1331749579',
               '/<script>cross_site_scripting.nasl</script>.idc',
               '<img \x39src=x onerror="javascript:alert(1)">',
               '/jhot.php?rev=2 |less /etc/passwd']

# 矩阵转换
X_predict = vectorizer.transform(new_queries)
res = LR.predict(X_predict)
res_list = []

```



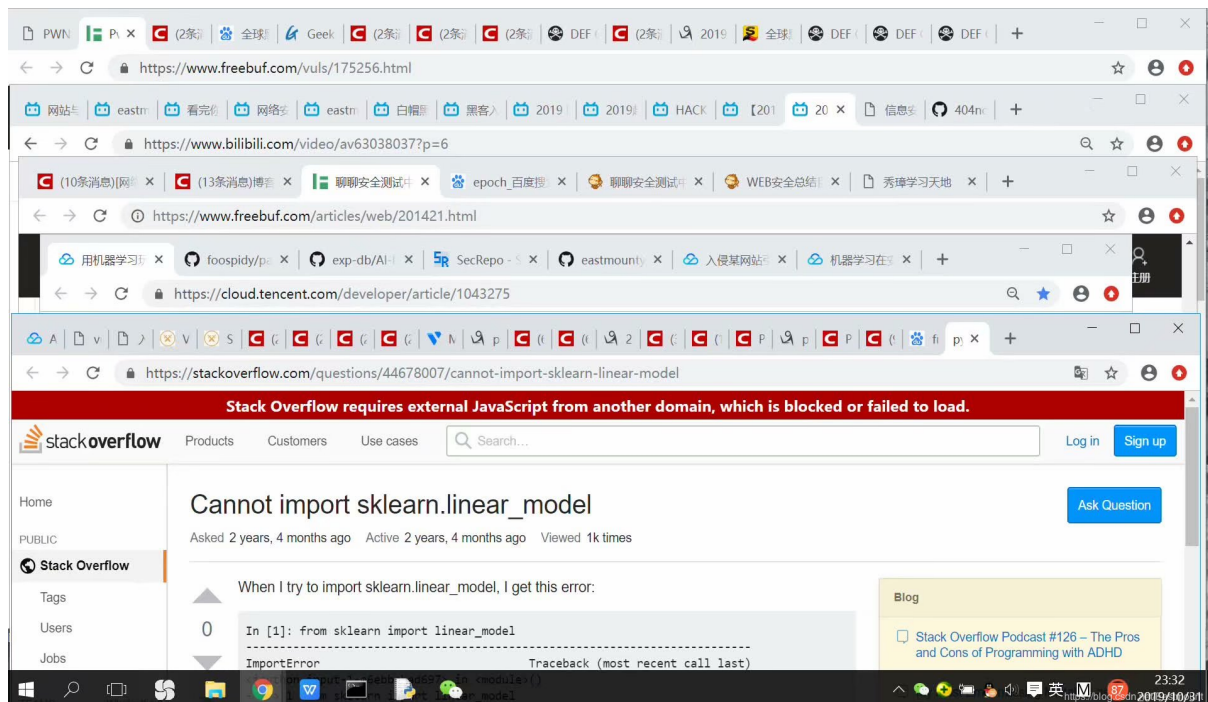
```
# 结果输出
for q,r in zip(new_queries, res):
    tmp = '正常请求' if r == 0 else '恶意请求'
    q_entity = html.escape(q)
    res_list.append({'url':q_entity,'res':tmp})

for n in res_list:
    print(n)
```

## 四.总结

写到这里，一篇基于机器学习的恶意代码请求识别讲述完毕，希望读者喜欢，不喜勿喷。该代码的亮点是N-grams融合到TF-IDF，当然也可以换成其他分类模型，虽然代码很基础，但也花费了作者三个小时时间，并且查阅了大量网页文章复现的（如下图所示）。

一步一个脚印前行，接下来希望通过深度学习实现更多的恶意代码识别和对抗样本，准备开启TensorFlow2.0和更多的安全基础系列的学习。作为安全领域的菜鸟，感觉自己要学习的知识好多、好杂，而且很多收费资料很贵，这系列文章都是作者自学且免费分享给博友们的，希望你们喜欢和点赞，未来继续加油！因为有你的阅读，才有我写作的动力，秀璋共勉。



(By:Eastmount 2019-11-01 中午2点于武汉 <http://blog.csdn.net/eastmount/> )