设计研究与应用

基于隐马尔可夫模型的 web 异常检测案例分析

朱骊安

（北京邮电大学理学院，北京 100876）

摘 要： 如何将数据科学在网络安全领域内应用一直是一个火热的话题，本文介绍了如何用隐马尔可夫模型（HMM）建立 web 参数模型，检测注入类的 web 攻击。以及这种方法在某航空公司的实施情况，准确率最终达到约 80%。

关键词： 隐马尔可夫模型；异常检测；隐含序列；概率模型；web 威胁

中图分类号： TP181 **文献标识码：** A **DOI：** 10.3969/j.issn.1003-6970.2017.11.022

本文著录格式： 朱骊安. 基于隐马尔可夫模型的 web 异常检测案例分析[J]. 软件，2017，38（11）：114–118

Case Analysis of Web Anomaly Detection Based on Hidden Markov Model

ZHU Li-an

(1. *Beijing University of Posts and Telecommunications, Beijing* 100876, *China*;

【Abstract】: How to apply data science in the field of network security has been a hot topic, This paper describes how to use the hidden Markov model (HMM) to establish a web parameter model to detect injection attacks. As well as the implementation of this method in an airline, the accuracy rate eventually reached about 80%.

【Key words】: Hidden markov model; Anomaly detection; Implicit sequence; Probability model; Web threat

0 引言

随着互联网的发展，企业的传统网络边界在逐渐消失。工业界特别是大型互联网公司，平均每日活跃用户上千万，每个应用系统的日志都会高达几百 G 甚至上 T 字节。同时，以灰产，黑产为代表的恶意访问占比依然居高不下，并且攻击手段在不断推陈出新。传统 web 入侵检测技术，无论是 Firewall、Web 应用防火墙（WAF）、入侵防御系统、入侵防御系统（IPS）还是入侵检测系统（IDS）本质上都是依据白名单或已发现攻击总结出的规则，通过维护规则集对入侵访问进行拦截。一方面，硬规则在灵活的黑客面前，很容易被绕过，且基于以往知识的规则集难以应对攻击；另一方面，攻防对抗水涨船高，防守方规则的构造和维护门槛高、成本大。

基于机器学习技术的新一代 web 入侵检测技术有望弥补传统规则集方法的不足，为 web 对抗的防守端带来新的发展和突破。Web 异常检测归根结底还是基于日志文本的分析，因而可以借鉴自然语言处理中的一些方法思路，进行文本分析建模。基于隐马尔科夫模型（HMM）的参数值异常检测，就是借助自然语言处理的方法，发现 web 日志中的异常序列，从而在线检测出未知异常行为。

1 模型原理

1.1 隐马尔可夫模型

隐马尔可夫模型（HMM）是马尔可夫链的一种，它的状态不能直接观察到，但能通过观测向量序列观察到，每一个观测向量是由一个具有相应概率密度分布的状态序列产生。

马尔可夫链是满足马尔可夫性质的随机过程。$X_1X_2X_3\cdots$ 马尔可夫链，描述了一种状态序列，其每个状态值取决于前面有限个状态。马尔可夫链是具有马尔可夫性质的随机变量的一个数列。这些变量的范围，即它们所有可能取值的集合，被称为“状态空间”，而 X_n 的值则是在时间 n 的状态。如果 X_{n+1} 对于过去状态的条件概率分布仅是 X_n 的一个函数，则

$$P(X_{n+1}=x\mid X_1=x_1, X_2=x_2,\cdots, X_n=x_n)=$$
$$P(X_{n+1}=x\mid X_n=x_n)$$

作者简介：朱骊安(1992–)，女，研究生，主要研究方向：随机排队网络、物流供应链管理。

这里 x 为过程中的某个状态。上面这个恒等式可以被看作是马尔可夫性质[1]。

随机过程是一连串随机事件动态关系的定量描述。马尔科夫过程是一种随机过程，简单地说，已知现在、将来与过去无关（条件独立），则称此过程为马尔科夫过程[2]。

HMM 主要有以下三类应用：解码问题，根据模型参数和观测序列，找出该观测序列最优的隐含状态序列；评估问题，根据模型参数和观测序列，计算该观测序列是由该模型生成的概率；学习问题，根据一系列观测序列，建立对应该系列序列最优的 HMM 模型。这里我们只用得到后两个。在训练阶段，对应学习问题，用大量正常的参数值训练出站点下的参数 id 的 HMM 模型；在检测阶段，对应评估问题，待检测的参数值带入模型检测是否是正常。

1.2 参数异常模型

Web 威胁中的几大类攻击，SQL、XSS、RCE 等虽然攻击方式各不相同，但基本都有一个通用的模式，即通过对参数进行注入 payload 来进行攻击，参数可能是出现在 GET、POST、COOKIE、PATH 等等位置。所以对于异常模型，能覆盖掉参数中出现的异常，就能覆盖掉很大一部分的常见的 Web 攻击[3]。

假设有这样一条 url：www.xxx.com/index.php?id=123。通过对 url 的所有访问记录分析，不难发现：普通用户的正常请求虽然不一定完全相同，但总是彼此相似；攻击者的异常请求总是彼此各有不同，同时又明显不同于正常请求。如下数据所示：

User: www.xxx.com/index.php?userid=admin123
www.xxx.com/index.php?userid=root
www.xxx.com/index.php?userid=hzq_2017
Attacker: www.xxx.com/index.php?userid= mai06’ union select xxx from xxx
www.xxx.com/index.php?userid=%3Cscript%3E alert(‘XSS’)%3C
www.xxx.com/index.php?userid=125$%7B@print(md5(123))%7D

如果我们能够搜集大量参数 id 的正常的参数值，建立起一个能表达所有正常值的正常模型。由于，正常总是基本相似，异常却各有各的异常。基于这样一条观测经验，如果我们能够搜集大量参数 id 的正常的参数值，建立起一个能表达所有正常值的正常模型，那么一切不满足于该正常模型的参数值，即为异常。

2 工程实现

2.1 系统架构--组件选择与模块关系

模型训练过程我们需要大量的正常历史数据进行训练、数据量会达千万级别以上，因此我们需要一个大数据处理引擎；此外，检测过程中我们希望能够实时的检测数据，及时的发现攻击，这是一个流（streaming）计算过程，需要一个流计算引擎。综合考虑，我们选择 spark 作为统一的数据处理引擎，即可以实现批处理，也可以使用 spark streaming 实现近实时的计算。

系统架构如下图，需要在 spark 上运行三个任务：

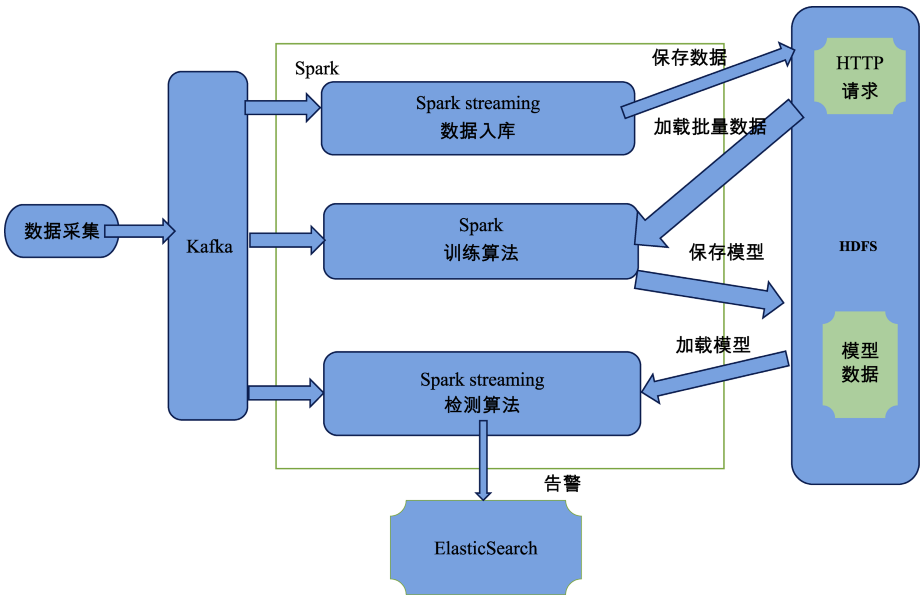

图 1 系统架构图
Fig.1 System architecture diagram

①sparkstreaming 将 kafka 中的数据实时的存入 HDFS；

②训练算法定期加载批量数据进行模型训练，并将模型参数保存到 HDFS；

③检测算法加载模型，检测实时数据，并将告警保存到 ElasticSearch。

在我们的系统中，模型训练算法是在 spark 上开发完成的。用 HDFS 来存储 HTTP 请求数据和模型数据。ElasticSearch 在我们的系统架构中主要用来存储、检索、展示告警数据。

2.2　数据的采集与储存

获取 http 请求数据通常有两种方式，第一种从 web 应用中采集日志，使用 logstash 从日志文件中提取日志并泛化，写入 Kafka；第二种可以从网络流量中抓包提取 http 信息。我这里使用第二种，用 python 结合 Tcpflow 采集 http 数据，在数据量不大的情况下可稳定运行。

与 Tcpdump 以包单位保存数据不同，Tcpflow 是以流为单位保存数据内容，分析 http 数据使用 tcpflow 会更便捷。Tcpflow 在 linux 下可以监控网卡流量，将 tcp 流保存到文件中，因此可以用 python 的 pyinotify 模块监控流文件，当流文件写入结束后提取 http 数据，写入 Kafka。这样数据的采集就完成了，下面开始数据的储存。

开启一个 SparkStreaming 任务，从 kafka 消费数据写入 Hdfs，Dstream 的 python API 没有好的入库接口，需要将 Dstream 的 RDD 转成 DataFrame 进行保存，保存为 json 文件。

2.3　数据的清洗与泛化

抽取器实现原始数据的参数提取和数据泛化，传入一条 json 格式的 http 请求数据，可以返回所有参数的 id、参数类型、参数名、参数的观察状态序列 p_list。

2.3.1　拆解数据生成参数

将 http 请求数据用“请求的 URL 路径”和“GET、POST 的请求参数以及参数名本身”两种方式进行拆解，提取相应的参数值。例如：提取源 IP、目的 IP、host[4]。这一步的难点在于如何正确的识别编码方式并解码。不同的参数，正常的值不同。同时，有参数传递的地方，就有可能发生参数注入型攻击。所以，需要对站点下所有路径下，所有 GET、POST、PATH 中的所有参数都训练各自的正常模型。另外，对参数名本身，也训练其正常的模型。

针对这些情况将参数分成三类：第一类，uri，将每条 uri 的每一个参数对应的参数值泛化后做为 p_state。第二类，uri_pname，将每条数据的所有的参数名拼接起来泛化作为 p_state。第三类，uri_path，将每条数据的 uri 里面的路径泛化作为 p_state。

2.3.2　参数泛化

如果我们把参数 id 的每个参数值看作一个序列，那么参数值中的每个字符就是这个序列中的一个状态。同时，对于一个序列，为 123 或者 345，其背后所表达的安全上的解释都是：数字 数字 数字，我们用 N 来表示数字，这样就得到了对应的隐含序列，取字符的 unicode 数值作为观察序列[5]。泛化的方法如下：

1. 大小写英文字母泛化为“A”，对应的 unicode 数值为 65。

2. 数字泛化为“N”，对应的 unicode 数值为 78。

3. 中文或中文字符泛化为“C”，对应的 unicode 数值为 67。

4. 特殊字符和其他字符集的编码不作泛化，直接取 unicode 数值。

5. 参数值为空的取 0。

2.4　训练任务

一个参数对应一条数据，其中包括：一个 p_id 与一个 p_state。这样就得到了对应的隐含序列。Spark 训练任务抽取所有 http 请求数据的参数，并按照参数 ID（p_id）分组，分别进行训练，将训练模型保存到 Hdfs。

2.4.1　得到模型输入

我们需要对经过清洗与泛化（Extractor）后的数据进行分组，并存为字典 p_dict。key 值为参数 ID，将 key 值相同的数据的 p_state 作为其 value 值。p_dict 的 key 值有：p_id，p_name，p_type，p_state。由于模型的输入规则，我们只需要参数 ID（p_id）和隐含序列（p_State）。故将每个 p_dict 中的 p_id 和 p_state 抽取出来得到模型的输入数据。

2.4.2　计算基线并保存--训练器（Trainer）

传入参数的所有观察序列，训练器完成对参数的训练，返回训练好的模型 profile[6]。其中，HMM 模型使用 python 下的 hmmlearn 模块，profile 取观

察序列的最小得分。由于我们假定进入模型的数据是正常数据，建立的是一个能表达所有正常值的正常模型，那么一切不满足于该正常模型的参数值，即为异常。所以，profile 取得是所有 score 的最小值。再将基线结果 profile 值保存起来[7-9]。

2.5 检测任务

Spark Streaming 检测任务实时获取 kafka 流数据，抽取出数据的参数，如果参数有训练模型，就计算参数得分，小于基线输出告警到 Elasticsearch。将得分与基线相比较，低于基线的报警。

3 模型应用案例分析

3.1 背景

某航空公司门户网站发现有不少攻击尝试行为，偶尔会发现有些网页正常用户在常规访问的过程中会发现无法访问的现象。经初步判定，很可能是攻击用户的异常请求造成的。为了进一步分析，航空公司运维人员提供了为期一个月的 web 访问日志，需要把访问日志中的异常行为数据筛选出来。

3.2 研究过程

3.2.1 数据情况

运维人员提供的是 IIS 日志，对应的日志关键属性如表 1 所示。

3.2.2 数据处理与建模

以第一周七天的数据作为训练数据，一周数据大概 1G 左右，我们选择通过 spark 程序把数据处理成我们建模需要的 json 格式，存入 HDFS 中。数据预处理完成后，我们执行训练任务，建立模型。建模结果部分截图如表 2 所示。

3.2.3 检测与结果分析

由于目前拿到的数据都是历史数据，所以我们对实时检测程序 HmmDetectionJob 作了修改，修改 sparkstreaming 代码为 spark 代码，把数据源从 kafka 读数据修改为从 HDFS 读数据。

为了测试模型的准确性及程序的稳定性，我们选取第 8 天的数据为研究对象，用第 8 天数据进行检测。第 8 天一共 13 万条 web 日志，经过检测告警的有 50 天，其中 30 条为有威胁数据，检测准确率为 60%。

表 1 IIS 日志
Tab.1 IIS Logs

s-ip	method	uri-stem	uri-query	s-port	username	c-ip	User-agent	status

表 2 建模结果
Tab.2 Modeling results

p_id	p_name	p_type	profile
3738a96f082d5954431b7	ArrCode	uri	10.84980185735
7b6904e8081f5477f34d9	FlightTabs	uri	4.22543644345871
a1a846c3b246344c985e	DepDate	uri	79.4265417898106
decd0f0cf61735803bff2	Guid	uri	–17518.5852704281
fd6b523b32fsf4424fsb24	CabinLevel	uri	27.1520399722512
aj239c9sj4js5j22sd9af4f	DepCode	uri	10.84980185735
0f0asj28djdskhl8j2342sa	Categoryid	uri	112.497146773755

3.2.4 问题与解决方案

通过对检测结果与模型的对比分析，我们发现检出准确率还有进一步提升的空间，主要原因是建模结果的准确率缺失造成的[10-12]，原因如下：

1、样本量偏小

隐马尔可夫模型是一个概率模型，样本量越大、涵盖的 url 数据类型越多，模型的准确度越高。而我们这里只用了一周的 web 日志作为训练数据，样本量偏小，需要提高训练模型的数据量。

2、训练数据中有异常数据

通过对模型中基线分值异常偏低的模型结果所对应的训练数据进一步分析，发现训练数据中混入

一些异常数据，主要有以下三种情况：少量未被认为筛选出的攻击行为数据；中文乱码数据；无法处理的加密数据。建模前我们需要设计一个过滤模块，对这些异常数据进行过滤。

3、缺少避免模型基线过度偏小的修正模块

在对模型基线值 profile 进行训练的过程中，只是简单的把正常数据中评分最低的值当做 profile，而在建模中并没有加入修正过程。

通过对相同 p_id 所对应 score 的分析，我们发现 score 服从正态分布，利用正态分布的特性，我们 profile 距离 score 分布的期望 3 倍方差以内为判断条件，当不满足时，剔除 profile 对应的 score 重新训练，直到满足条件。

通过对以上问题的解决，我们最终把检测准确率提高了 85%以上，并准确的筛选出大量攻击行为数据，部分数据截图如表 3 所示。

表 3 攻击数据
Table 3 Attact Data

URI	攻击行为
/xx-xx/news/Procurement/admin.aspx	猜解后台
/cm.xiamenair.com.tar.gz	获取网站源码
/admin_login.php	猜解后台
/krbcd25667.asp;.jpg iis6	解析漏洞
/browse/category-153.aspx/*><ScriPt>alert(777)</ScriPt>	XSS 漏洞
/templates/master/pc/password.mdb	猜解密码数据库
/Storage/aster/web.config.temp	获取网站配置信息
/flightstatus/database.sql	获取数据库信息

参考文献

[1] 施仁杰. 马尔科夫链基础及其应用[J]. 西安电子科技大学出版社, 1992.

[2] 李裕奇, 刘赫编. 随机过程（第3版）习题解答[M]. 国防工业出版社，2014.09.

[3] 谢逸, 余顺. 争基于Web用户浏览行为的统计异常检测[J]. 软件学报, 2007, 18(4):967-977

[4] DH Schneck, S Cherry, D Goodman. Web interface and method for accessing and displaying directory information[M]. US, 2001.

[5] I Corona, D Ariu, G Giacinto. HMM-web: a framework for the detection of attacks against web applications[J]. IEEE International Conference on Communications , 2009 , 15 (1): 747-752.

[6] 何强, 毛士艺, 张有为. 多观察序列连续隐含马尔柯夫模型的无溢出参数重估[J]. 电子学报, 2000, 28(10): 98-101.

[7] 岳峰, 左旺孟. 基于马尔可夫随机场的弹性掌纹匹配[J]. 新型工业化, 2012, 2(2): 52-61.

[8] 朱靖波, 肖桐. 句法统计机器翻译的一些问题分析[J]. 新型工业化, 2012, 2(1): 1-11.

[9] 张元青, 聂兰顺. 一种BPMN到JPDL的模型转换方法[J]. 新型工业化, 2012, 2(1): 23-31.

[10] 张彦. 动态Web 技术在实时监测系统中的实现[J]. 软件, 2013, 34(12): 265.

[11] 刘晓婉, 胡燕祝, 艾新波. 开源中文分词器在web搜索引擎中的应用[J]. 软件, 2013, 34(3): 80-83.

[12] 黄炳良, 张忠琳. 预测市场技术在机器学习中的应用[J]. 软件, 2014, 35(11): 31-35.