doi:10.11959/j.issn.1000-436x.2016034

面向网络环境的 SQL 注入行为检测方法

赵宇飞[1]，熊刚[2]，贺龙涛[3]，李舟军[1]
(1. 北京航空航天大学计算机学院，北京 100083；2. 中国科学院信息工程研究所，北京 100093；
3. 国家计算机网络应急技术处理协调中心，北京 100029)

摘 要：SQL 注入攻击是 Web 应用面临的主要威胁之一，传统的检测方法针对客户端或服务器端进行。通过对 SQL 注入的一般过程及其流量特征分析，发现其在请求长度、连接数以及特征串等方面，与正常流量相比有较大区别,并据此提出了基于长度、连接频率和特征串的 LFF（length-frequency-feature）检测方法，首次从网络流量分析的角度检测 SQL 注入行为。实验结果表明，在模拟环境下，LFF 检测方法召回率在 95%以上，在真实环境下，该方法也取得较好的检测效果。

关键词：Web 安全；SQL 注入；网络流量；异常检测
中图分类号：TP393 **文献标识码：**A

Approach to detecting SQL injection behaviors in network environment

ZHAO Yu-fei[1], XIONG Gang[2], HE Long-tao[3], LI Zhou-jun[1]
(1. School of Computer Science, Beihang University, Beijing 100083, China;
2. Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China;
3. National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China)

Abstract: SQL injection attack is one of the main threats that many Web applications faced with. The traditional detection method depended on the clients or servers. Firstly the process of SQL injection attack was analyzed, and then the differences between attack traffic and normal traffic HTTP request length, HTTP connections and feature string were discovered. Based on the request length, request frequency and feature string, a new method, LFF (length-frequency-feature), was proposed to detect SQL injection behaviors from network traffic. The results of experiments indicated that in simulation environments the recall of LFF approach reach up to 95%, and in real network traffic the LFF approach also get a good detection result.

Key words: Web security, SQL injection, network traffic, outlier detection

1 引言

目前，大多数信息系统和商业应用都提供了基于数据库的 Web 服务，与此同时，针对 Web 服务中漏洞发起的攻击也越来越多。在 OWASP 发布的 2013 年 Web 安全漏洞 Top10 中[1]，由于操作难度低、危害巨大，SQL 注入漏洞名列第一，且其普遍程度呈现出逐年上升的趋势。

SQL 注入是一种利用应用程序数据库漏洞的注入技术。SQL 注入攻击针对的是使用后台数据库

收稿日期：2015-04-08；修回日期：2015-07-15
通信作者：熊刚，xionggang@iie.ac.cn
基金项目：国家高技术研究发展计划（“863”计划）基金资助项目（No.2015AA016004）；国家自然科学基金资助项目（No.61170189, No.61370126）；教育部博士点基金资助项目（No.20111102130003）；国家科技支撑计划基金资助项目（No.2012BAH46B02, No.2012BAH46B04）；中国科学院战略性先导科技专项课题基金资助项目（No.XDA06030200）
Foundation Items: The National High Technology Research and Development Program of China (863 Program) (No.2015AA016004), The National Natural Science Foundation of China (No.61170189, No.61370126), Ph.D. Programs Foundation of Ministry of Education of China (No.20111102130003), The National Key Technology R&D Program (No.2012BAH46B02, No.2012BAH46B04), The Strategic Priority Research Program of the Chinese Academy of Sciences (No.XDA06030200)

的交互式 Web 服务，主要通过网络应用层协议 HTTP 完成。SQL 注入攻击的目标一般不是 Web 服务器本身，而是为了非法获取用户信息、数据库中的敏感数据以及服务器数据信息[2~4]。由于 SQL 注入利用 SQL 语言的特性，理论上对 Oracle、MySQL、MS Server 等基于 SQL 语言的数据库都有效。通过对 SQL 注入行为的检测，可及时向被注入系统报警，有效保护企业及个人数据安全，同时可作为网络取证的证据，对网络攻击行为形成震慑作用。

不同于以往基于客户端或服务器端的检测方法，本文通过分析和研究 SQL 注入过程及其流量特征，提出一种在网络流量上检查 SQL 注入行为的方法，本文方法的主要贡献和创新点如下。

1) 从某 ISP 获取的 211 GB 原始网络流量中（经人工检测不含 SQL 注入），详细统计并分析了 HTTP 请求的 URI 长度、实体长度、HTTP 连接数以及请求特征串；并通过与 4 种常用 SQL 注入工具的注入流量相比较，发现注入流量在 HTTP 请求长度、连接频率以及请求特征串等方面，与正常流量相比有较大区别。这为从网络流量分析的角度来检测 SQL 注入行为奠定了理论基础。

2) 基于上述洞察，提出了 LFF（length- frequency-feature）检测方法。对于每条 HTTP 请求，首先分别检测其请求长度和连接频率；然后采用特征串匹配算法，对请求语句进行匹配；最后经投票决策，确定 HTTP 请求中是否有 SQL 注入。该方法首次从网络流量分析的角度来检测 SQL 注入行为。

3）通过大量实验，验证了 LFF 检测方法的有效性。首先，使用该方法对 2 种新的注入工具产生的 SQL 注入流量进行检测，检测的召回率在 95% 以上。然后，在真实网络环境下，对未经任何处理的 196 GB 真实网络流量进行检测，实验结果表明，在真实流量中检测到 SQL 注入行为，且该方法的检测召回率在 74%以上。这说明本文提出的方法能够有效地发现网络流量中的 SQL 注入行为。

2　相关工作

马小婷等[5]全面分析了 SQL 注入检测常用的原理和技术。大体而言，现有的 SQL 注入检测分为 2 类。

第一类是服务器端检测技术。Halfond[6,7]和 Shar 等[8]将注入语句的静态特征与后台运行时的动态特征相结合，通过尽可能少的前期学习，实现 SQL 注入检测；Shahriar 等[9]采用静态代码分析，根据代码中数据库相关的 API 调用找出数据库查询语句，再计算出表、列、值的离散程度，从而判断是否有 SQL 注入发生。此外，还通过机器学习方法[10]、基于规约的方法[11]以及污点跟踪技术[12]检测 SQL 注入。总体来看，服务器端可以检测全部的数据库查询请求，但需要预先分析所有正常的 SQL 查询语句，还需要通过源码分析出注入点，前期工作量大，且检测结果依赖于关键词的提取，准确率有待提高。

第二类是客户端 SQL 注入漏洞检测。一般是在无法得到网站源码的情况下进行，只能通过监控 Web 应用程序的行为判断是否有 SQL 注入漏洞。Huang 等[13]提出了基于错误注入和行为监控的 Web 应用程序安全分析方法，将软件测试技术应用到 Web 应用程序漏洞检测；Stefan 等[14]设计了 SecuBat，通过爬虫模块、攻击模块和分析模块的联合使用判断该页面是否存在 SQL 注入漏洞；APPELT[15]设计了一种高效的 SQL 注入工具，采用多种绕过技术，以检测网页中是否存在 SQL 注入漏洞。客户端检测的优点在于可以忽略不同编程语言的特性，但其最大的缺点就是容易漏检，难以覆盖所有检测点。

在网络流量检测方面，王苏南[16]优化了高速复杂网络环境下异常流量检测方法，提出了无监督的网络异常溯源算法；Zhang 等[17]采用改进的最邻近算法，利用机器学习进行网络流量检测和分类，但该方法处理速度较慢；周爱平等[18]系统总结了高速网络环境中的网络测量方法，并比较了这些方法的优劣，总体而言，这些方法都有较大局限性；王鹏等[19]提出了粒度自适应的多径流量分割算法，能较好地区分不同类型的流量。

3　SQL 注入过程及其特征分析

一次完整的 SQL 注入过程可分为 3 个阶段：寻找注入点、确定数据库类型、数据窃取。这 3 个阶段相互独立又紧密联系。

本文对从某 ISP 获取的 211 GB 原始网络流量进行分析（经人工检测不含 SQL 注入）。然后在服务器上搭建了有 SQL 注入漏洞的网站，通过 4 种常用注入工具向该网站实施 SQL 注入攻击，并获取到不同工具的注入流量，同时也对 4 种软件的注入流量进行了统计分析。这 4 种软件分别是：啊 D2.32、

明小子 4.3、pangolin4.0[20]、sqlmap1.0[21]。下面具体分析每一个阶段的特征和统计结果。

3.1 寻找注入点

这一阶段需要寻找有注入漏洞的网页。一般的方法是通过爬虫遍历网站所有网页，对网页所提交的参数进行更改，根据服务器返回的信息，判断是否存在注入漏洞。这一过程中常用的攻击方式包括注释符攻击（添加单引号）、重言式攻击（添加“ and 1=1-- ”）和其他 SQL 语言(添加 drop 或 delete)。由于需要遍历网站及参数，客户端会多次连接服务器，故在单位时间内会产生大量连接。

由上述分析可知，对于四元组<源 IP，源端口，目的 IP，目的端口>，在客户端频繁连接服务器时，除源端口，其他值不会发生改变。所以，可以通过三元组<源 IP，目的 IP，目的端口>标记和区分主机发起的 HTTP 请求。本文希望通过检测单位时间内 HTTP 连接数（即连接频率），找出网络流量中的 SQL 注入行为。

在 211 GB 原始流量中，HTTP 请求次数为 712 521 次，共汇聚成 34 509 条三元组。表 1 列出了总连接数前五的三元组，相应的 IP 已做匿名化处理。

从表 1 可以看出，每个三元组的平均连接次数差异很大，这是网络状况等多方面的因素造成的。经统计，所有三元组的平均连接次数为 4.87 次/秒。

4 种注入工具 HTTP 连接数的统计结果如表 2 所示。其中，啊 D2.32 和 sqlmap1.0 注入成功，另外 2 种工具失败。

在表 2 中，啊 D2.32 和 sqlmap1.0 的平均连接次数较大，而其他 2 种工具的平均连接次数较少。这是因为明小子 4.3 和 pangolin4.0 注入失败，这 2 种工具都设置了较大的超时重传时间。

比较表 2 和表 1，SQL 注入工具的连接频率会高于正常网络流量中 HTTP 连接频率的平均值。因此，计算单位时间内 HTTP 请求的三元组次数，可以作为 SQL 注入检测的一个标准。虽然正常网络中平均 HTTP 连接频率为 4.87 次/秒，但大部分的连接频率远低于这个值，所以，本文将 HTTP 连接频率的阈值设置为 3 次/秒。检测时，若某三元组 1 s 内的连接数大于 3 次，则认为连接频率过大，可能存在 SQL 注入行为，需要进一步的检测。

3.2 确定数据库类型

这一阶段对数据库中的表做 select 操作，根据数据库返回信息判断数据库类型。这个过程中需要向 URI（或 HTTP 实体）拼接 SQL 查询语句，故 URI 长度（或 HTTP 实体长度）与正常网络流量相比会发生很大变化。

HTTP 协议主要有 GET 和 POST 2 种交互方式，不同的方式检测 HTTP 请求的不同部分。GET 方式检测 URI 长度， POST 方式检测 HTTP 实体长度。因为这一阶段中 URI 长度（或 HTTP 实体）变化明显，本文试图通过检测 URI 长度（或 HTTP 实体长度），在发生 SQL 注入时找出异常的 HTTP 请求。

3.2.1 URI 长度统计

从 211 GB 原始流量中，共提取到 712 516 条 URI 长度大于零的 HTTP 请求，对每条请求计算

表 1 211 GB 原始流量中总连接数前五的三元组

源 IP	目的 IP	目的端口	HTTP 连接数	持续时间/s	每秒平均连接次数
197.237.*.*	202.113.*.*	80	28 456	12 372	2.3
118.228.*.*	23.5.*.*	80	18 093	25 848	0.7
115.132.*.*	202.113.*.*	80	11 312	6 285	1.8
199.244.*.*	143.90.*.*	80	9 487	11 029	0.86
111.118.*.*	119.9.*.*	80	8 475	1 022	8.3

表 2 4 种注入工具注入流量的 HTTP 连接数

软件	SQL 注入次数	持续时间/s	每秒平均注入次数	峰值
啊 D2.32	381	46	8.3	20
明小子 4.3	166	81	2	3
pangolin4.0	361	171	2.1	10
sqlmap1.0	500	79	6.3	13

URI 长度并进行统计。结果显示，正常网络流量中，平均 URI 长度为 78.3 byte，加权平均 URI 长度为 103.1 byte，其中长度小于 100 byte 的 URI 占到总量的 80.69%。URI 统计结果如图 1 所示，长度小于 100 byte 的 URI 长度频率分布直方图如图 2 所示。4 种 SQL 注入工具的注入流量中 URI 长度频数分布直方图如图 3 所示。

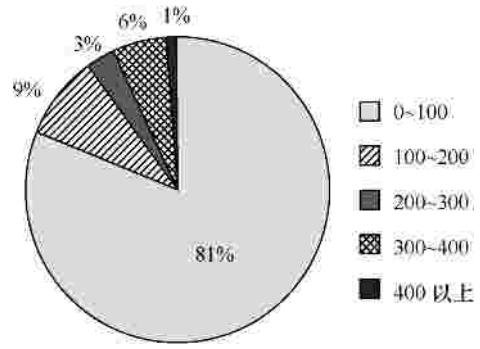

图 1　URI 长度统计

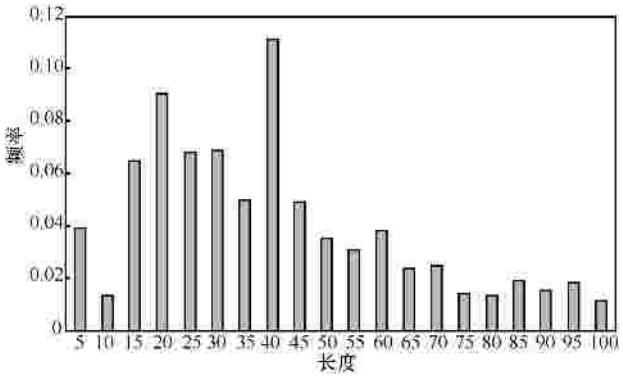

图 2　URI 长度小于 100 byte 的频率分布直方图

表 3 比较了正常流量和注入流量 URI 长度的相关统计值。表 3 表明，有 3 种工具的 URI 平均值大于正常流量平均值；但 4 种工具的加权平均值均小于正常流量的加权平均值；4 种注入工具 URI 长度的众数都大于正常流量众数；正常流量 URI 长度的标准差是 5 种中最大的，说明该数据离散程度高，具有较强的代表性。由图 1 可知，80%以上的 URI 长度小于 100 byte，但由于其余 20%的 URI 长度较大，导致正常流量 URI 加权平均值偏大。从表 2 中可以发现，明小子 4.3HTTP 请求个数偏少，导致其 URI 统计值偏低，但从图 3(b)中可以看出，仍有大量的 URI 长度集中在 80~90 byte 的范围内。综上所述，可以认为当 HTTP 请求的 URI 长度超过某个阈值时，该请求可能包含异常行为，需要进行进一步的判断。

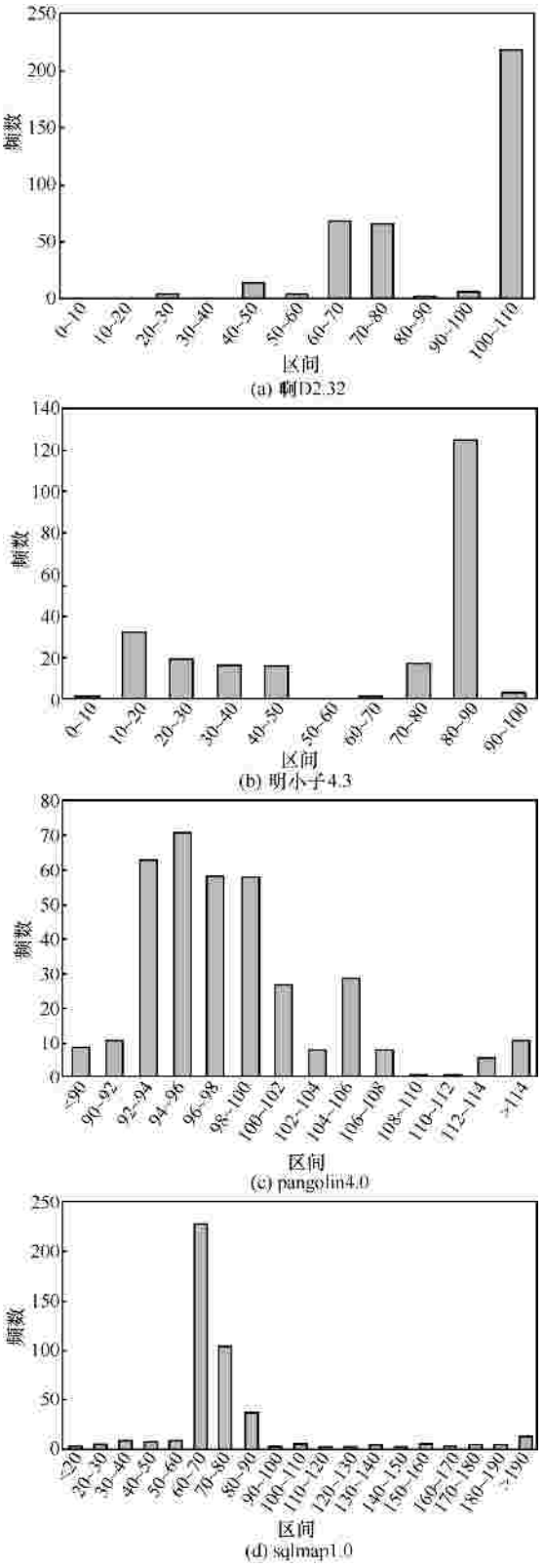

图 3　4 种工具注入流量 URI 长度频数分布直方图

由图 2 可知，大部分 HTTP 请求的 URI 长度在 5~70 byte 之间，故本文设置 URI 长度阈值为 70 byte，当长度大于 70 时，认为该请求存在异常。

表 3 URI 长度统计值

软件	URI 长度平均值/byte	加权平均值/byte	最小值/byte	最大值/byte	众数/byte	标准差	URI 总个数
211 GB 原始流量	78.3	103.1	1	1 442	37	106.4	712 516
啊 D2.32	86.6	93.7	1	106	101	18.67	383
明小子 4.3	62.5	68.0	1	95	86	28.39	230
pangolin4.0	102.8	99.0	29	617	96	43.44	357
sqlmap1.0	79.8	80.0	19	413	67	42.78	443

3.2.2 HTTP 实体长度统计

首先对 211 GB 原始流量中 POST 请求的实体长度进行统计，结果如图 4 所示，横坐标表示以前一个坐标为下限、当前坐标为上限的区间。

在 211GB 原始流量中共提取到 190 938 条 POST 请求，其实体长度分布呈现出“两头大，中间小”的情况，这是因为通常用 POST 方式向服务器输入数据，比如提交 HTML 表单。浏览网页时，登录、评论等行为都会产生 POST 请求；向网站服务器传输图片等文件时，采用 POST 方式会有较大载荷，这样就造成图 3 的情况。由于并不是所有注入工具都支持 POST 方式，本文挑选支持该方式的 sqlmap1.0，对该工具 HTTP 请求实体长度也进行了统计，结果如图 5 所示。

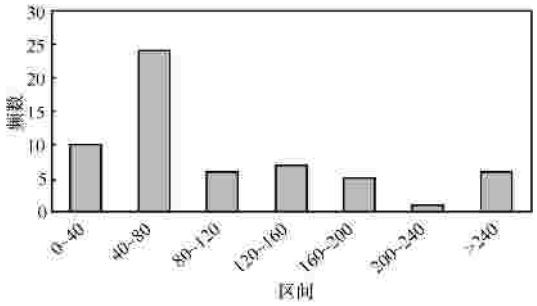

图 5 sqlmap1.0 POST 请求实体长度频数分布直方图

图 5 和图 4 的对比结果如表 4 所示。可以看出，正常流量的实体长度分布呈现出两极化趋势，POST 注入方式并没有在实体长度上表现出明显的特征。将图 5 与图 3(d)比较，不论是 POST 还是 GET，其请求长度值都集中在 60~80 byte 的范围；但 POST

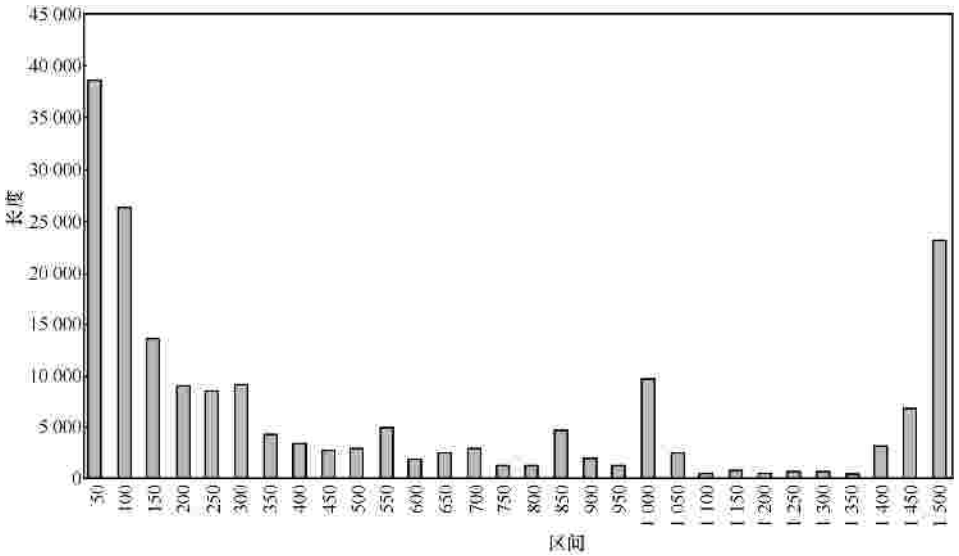

图 4 211 GB 原始流量 POST 请求实体长度统计

表 4 实体长度统计值

软件	平均值	加权平均值	最小值	最大值	众数	标准差	总个数
原始流量	505.6	532.9	1	1 484	1 460	531.4	190 938
sqlmap1.0	109.6	119.0	18	470	18	95.28	59

请求的总次数要远小于 GET 方式的总次数，说明 POST 注入有更强的隐蔽性。因此对于这种类型的注入需要采取其他的检测方法。

3.3 数据窃取阶段

确定数据库类型后，一般会根据字典猜测表名和列名，再通过二分查找等方法确定列中数据个数和内容。由于需要猜测内容，注入时会向 URI（或 HTTP 实体）拼接 SQL 查询语句，拼接的语句中含有大量 SQL 函数。可以将 SQL 查询语句或 SQL 函数作为特征串进行匹配和检测。

特征串检测适用于 SQL 注入的全过程，尤其适用于这一阶段。因为 SQL 注入攻击会利用常见数据库信息进行注入。分析 4 种工具的注入流量，本文归纳出 SQL 注入语句的如下特征。

1) 利用 SQL 语言注释符截断原有语句，通过“and”或“or”添加查询条件。

2) 含有 select、union、from 和 where 等 SQL 语言关键词。

3) 含有 SQL 函数，包括 chr、exists、ascw、count 等。

4) 注入语句使用字母大小写替换等方式绕过检测。

可以发现，特征 1）和特征 2）具有很强的局限性，在正常的 HTTP 请求中，也会有相似关键词，若只以特征 1）和特征 2）作为特征串，会导致误报率较高。特征 3）具有一定特异性，其缺点是需要大量 SQL 函数作为匹配的集合，由于 SQL 函数众多，不可能全部包括。特征 4）导致特征串集合规模成指数增长，故必须采用其他处理方法。

通过上述分析，本文认为用于检测的特征串应具备以下特点：特征串不是简单的 SQL 关键词或函数名，而是 SQL 语句片段；特征串不仅包括 SQL 语言，还应包括常见数据库信息；特征串应包括注入工具所使用的字典；特征串尽可能多的覆盖注入语句的绕过方式。为了构造符合上述特点的特征串集合，本文采用 2 种方法，分别在下面详细介绍。

3.3.1 基于序列对比的 Needle-Wunsch 算法

Needle-Wunsch 算法最早用于生物序列分析，适用于整体相似性度较高的 2 个序列。本文采用改进的 Needle-Wunsch 算法进行逆向匹配，从而找出注入语句的最长公共子串。对表 2 中 1 408 条 SQL 注入语句两两互求最长公共子串，共得到 161 820 条不同的子串。去除没有意义的部分，根据重复次数排序，结果如表 5 所示。

可以看出，得到的公共子串不是分散的关键词，而是基本完整的 SQL 语句，有利于进一步匹配；同时，公共子串中含有 admin 等字典内容，有利于 SQL 注入语句的查找。但该算法得到的结果对字母大小写敏感，故本文又采用正则表达式方法，作为对绕过方式的检验。

3.3.2 特征串的正则表示

虽然通过 NW 算法，解决了特征串的选取问题，但仍可能通过字母大小写替换等方式绕过特征串检测。为了解决这个问题，以更加完备地描述 SQL 注入语句的特征，本文对上述获取的特征串使用正则表达式进行表示。表 6 给出一些常见注入语句及其正则表示。

用于检测的特征串集合由 3 部分组成，第 1 部分由 NW 算法得到的公共子串的正则表示组成；第 2 部分是常用数据库信息，包括数据库系统表等；

表 5　最长公共子串重复次数前五

最长公共子串	重复次数
%20and%20(select%20count(*)%20from%20admin)%20and%201	48 426
%20and%20(select%20Count(1)%20from%20[admin]%20where%201=1)%20between%2010%20and%2010	16 290
%20and%20(select%20%20from%20admin)	10 425
%20AND%20SELECT%20	10 122
%20AND%20EXISTS%28SELECT%20%20FROM%20%29	9 030

表 6　常见注入语句的正则表示

注入语句	正则表示	说明
’or 1=1	(’\s+)(or|and)\s+[[:alnum:]]+\s*=\s*[[:alnum:]]+\s*(--)?	构造重言式注入语句
;and (select count(*) from admin)>0	;?select(\(|%28)\s*count(\(|%28)*\(\)|%29)\s+from\s+[^)]+\s*(\)|%29)\s*(>|%3E)?(=|%3D)?\s*\d+	猜测数据库名

第 3 部分是注入工具使用的猜测字典。

综上所述，通过寻找注入点、确定数据库类型以及数据窃取，构成了一次完整的 SQL 注入攻击。本文分析了每个阶段的流量特征，并据此对每个阶段都提出了一种检测方法。为了综合利用这些方法，本文提出了基于长度、连接频率和特征串的 LFF 检测方法。

4 基于 LFF 的 SQL 注入检测方法

4.1 检测方法描述

针对 SQL 注入不同阶段的特征，以及前文得出的相关参数的阈值，本文提出了 LFF (length-frequency-feature)检测方法，其步骤如下。

1) 判断 HTTP 请求是 GET 还是 POST。若是 GET 方式，提取 URI 后进入步骤 2)，若是 POST 方式，提取实体后进入步骤 3)。

2) 对 URI 进行长度检测，若长度大于 70 byte，则发出警告后进入步骤 3)，否则直接进入步骤 3)。

3) 提取当前 HTTP 请求的三元组，判断其是否存在于三元组表中；若存在，相应三元组连接数加 1，若不存在，则创建该三元组；对各组进行连接频率检测，若连接频率大于 3 次/秒，则发出警告后进入步骤 4)，否则直接进入步骤 4)。

4) 对步骤 1）提取出的结果进行特征串匹配。匹配的方法是找出特征串集合中每一项在该结果中首次出现的位置，若返回值不为空，则表明匹配成功，匹配检测发出警告后进入步骤 5)，否则直接进入步骤 5)。

5) 对步骤 2）到步骤 4）中的检测结果进行投票决策。若有 2 种及以上的检测方法发出警告，则判定当前 HTTP 请求存在 SQL 注入行为；若只有一种检测发出警告，则记录该请求，由人工进一步判断；若都没有发出警告，则认为当前 HTTP 请求不是 SQL 注入语句，进入步骤 6)。

6) 获取下一条 HTTP 请求，重复步骤 1）至步骤 5)。

LFF 检测方法流程如图 6 所示。

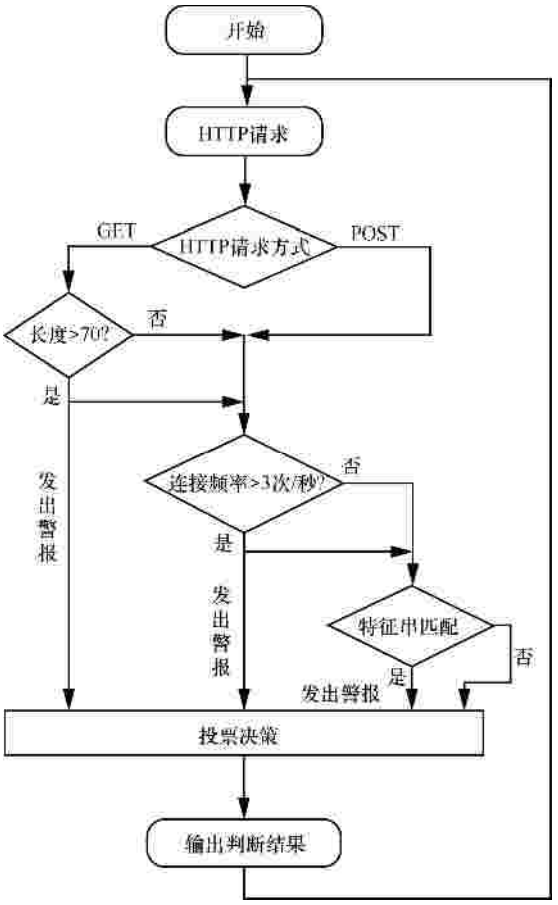

图 6 LFF 检测方法流程

4.2 方法有效性验证

为了验证 LFF 检测方法的有效性，本文使用 2 个新的注入工具 Havij1.7pro 和 safe3，对搭建在服务器上的网站进行 SQL 注入攻击，并获取注入时的流量。其中，Havij1.7pro 注入失败，safe3 注入成功。连同之前的 4 个注入工具，对这 6 个注入工具的检测结果如下。

4.2.1 连接频率检测验证

根据 3.1 节所述，将连接频率阈值设置为 3 次/秒，对超过阈值的 HTTP 请求发出警报。6 个注入工具的检测结果如表 7 所示，连接频率检测并没有

表 7 HTTP 连接数检测验证

指标	啊 D2.32	明小子 4.3	pangolin4.0	sqlmap1.0	Havij1.7pro	safe3
HTTP 连接数	385	230	362	507	527	650
SQL 注入次数	381	166	361	500	523	649
报警次数	381	114	308	500	0	624
检测召回率	100%	68.67%	85.31%	100%	0	96.15%

检测到 Havij1.7pro 的注入行为。这是因为该工具注入失败，其等待时间设置为 0.5 s，当服务器没有返回信息时，该工具每隔 0.5 s 会再进行一次尝试，所以连接数较低。但对于其他注入工具，该方法有较好的检测结果。

4.2.2　长度检测验证

根据 3.2 节所述，将请求长度阈值设置为 70 byte，对超过阈值的 HTTP 请求发出警报。结果表明，长度检测产生警报的 HTTP 请求都是 SQL 注入语句，误检率为 0。6 个注入工具的检测结果如表 8 所示。除工具 sqlmap1.0，该检测方法对其他工具都有较好的检测效果。通过对注入工具的分析，认为长度检测结果与注入工具的设计有关，sqlmap1.0 产生的注入语句长度较短，故检测效果较差。但从总体效果看，该检测已经起到一定的作用。为了提高检测召回率，可以适当减少阈值。

4.2.3　特征串检测验证

根据 3.3 中提取出的特征串，对 6 种工具的注入行为进行检测，检测结果如表 9 所示。可以看出，6 种注入工具检测召回率都在 95%以上，有很好的检测结果。但仍有部分注入语句没有检测出。这是因为还有一些语句采用了其他的绕过手段，比如，对注入点的参数进行算术运算；或是对注入语句进行 ASCII 编码。这些问题需要在未来的工作中解决。

4.2.4　投票决策验证

上面各节只是对各检测方法的验证，本节将对投票决策的结果进行验证和分析。根据表 7~表 9 的结果，横向来看，各检测模块都有一定的局限性，不能只根据一种检测方法的结果就认为存在 SQL 注入行为；纵向来看，每种工具都至少被 2 种方法检测出，即各检测方法之间具有互补性。因此，投票决策模块的原则是：若有 2 种及以上的检测方法发出警报，就认为是 SQL 注入行为；若只有一种检测发出警报，则记录相关信息，由人工做进一步判断；若都没有发出警报，则认为是正常 HTTP 请求。

本文对投票决策的结果进行人工验证，结果如表 10 所示。结果表明，对各注入工具的召回率在 95%以上。表 10 与表 9 相比，检测正确有所上升。这是因为，虽然有些语句的特征串并没有发现，但仍然能通过单位时间连接次数和请求长度发现其异常。

表 8　长度检测模块验证结果

指标	啊 D2.32	明小子 4.3	pangolin4.0	sqlmap1.0	Havij1.7pro	safe3
HTTP 连接数	385	230	362	507	527	650
SQL 注入次数	381	166	361	500	523	649
长度模块报警次数	292	145	360	188	468	640
召回率	76.64%	87.35%	99.72%	37.6%	89.48%	98.61%

表 9　6 种注入工具特征串检测结果

指标	啊 D2.32	明小子 4.3	pangolin4.0	sqlmap1.0	Havij1.7pro	safe3
HTTP 连接数	385	230	362	507	527	650
SQL 注入次数	381	166	361	500	523	649
特征匹配报警次数	367	166	361	499	522	648
召回率	96.33%	100%	100%	99.80%	99.80%	99.85%
误判率	3.67%	0	0	0.2%	0.2%	0.15%

表 10　LFF 投票决策结果验证

指标	啊 D2.32	明小子 4.3	pangolin4.0	sqlmap1.0	Havij1.7pro	safe3
HTTP 连接数	385	230	362	507	527	650
SQL 注入次数	381	166	361	500	523	649
LFF 判定注入次数	372	166	361	499	522	648
召回率	97.64%	100%	100%	99.8%	99.80%	99.85%
误判率	2.36%	0	0	0.2%	0.2%	0.15

5 真实网络环境下的实验验证

从相同 ISP 又获得 196GB 原始网络流量，这些流量事先未经过任何处理和标记，利用 LLF 检测方法对其进行检测，作为真实网络环境下的验证，并希望从中发现 SQL 注入行为。检测结果如表 11 所示。

表 11 196 GB 真实流量检测结果

SQL 注入次数	LFF 检出次数	召回率	错误率	误报率
3 033	2 286	74.77%	0	0

LFF 检测完成后，由人工对 196 GB 流量进行标记和验证。其中，共有 3 033 条注入语句，LFF 方法检测出 2 268 条注入语句，召回率为 74.77%，误报率为 0。部分注入语句未被发现的原因是：一些语句采用了 ASCII 编码等方式进行绕过；检测所用的特征串集合没有覆盖所有的特征串。结果表明，该方法没有产生误报，没有将搜索引擎产生的相似流量标记为 SQL 注入流量。

经过核查，3 033 条注入语句中，有 90%以上来自同一 IP 地址，该 IP 地址为 218.30.117.72，属于北京市某公司数据中心，可能为租用云服务器实施 SQL 注入攻击。

6 结束语

本文从网络流量检测的角度，提出了针对 SQL 注入的 LFF 检测方法。LFF 检测包括长度检测、连接频率检测和特征串检测。3 种检测方法以 SQL 注入各阶段特征为依据，并以大量正常流量作为分析基础，其结果具有一定的普遍性和代表性。在模拟环境下，该方法检测召回率在 95%以上；在真实网络环境中，检测召回率在 74%以上，能够较准确地发现网络中的 SQL 注入行为。未来的工作包括 2 个方面：1) 深入分析 SQL 注入的其他特征，以检测出有绕过形式的注入语句；2) 拓展该方法的应用范围，检测 XSS 等其他类型的攻击。

参考文献：

[1] OWASP 2013 top 10 risks[EB/OL]. https://www.owasp.org/index.php/Top_10_2013-Top_10, 2015-3-12.

[2] MCDONALD, S. SQL Injection: modes of attack, defense, and why it matters[EB/OL]. http://www.governmentsecurity.org/articles/SQLInjectionModesofAttackDefence and Why It Matters. php, 2015-3-11.

[3] ORSO A, HALFOND W G J, VIEGAS J. A classification of SQL injection attacks and countermeasures[C]//The International Symposium on Secure Software Engineering. c2006.

[4] APPELT D, NGUYEN D C, BRIAND L. Behind an application irewall, are we safe from SQL injection attacks[C]//IEEE International Conference on Software Testing, Verification and Validation (ICST). c2015:1-10.

[5] 马小婷,胡国平,李舟军.SQL 注入漏洞检测与防御技术研究[J].计算机安全,2010, (11):18-24.
MA X T, HU G P, LI Z J. Research on detection and prevention technologies for SQL injection vulnerability[J]. Computer Security, 2010, (11):18-24.

[6] HALFOND W G J, ORSO A. AMNESIA: analysis and monitorin for NEutralizing SQL-injection attacks[C]//20th IEEE/ACM International Conference on Automated Software Engineering. ACM, c2005: 174-183.

[7] HALFOND W G J, ORSO A. Detection and prevention of SQL injection attacks[J]. Malware Detection, 2006, (27): 85-109.

[8] SHAR L K, TAN H B K, BRIAND L C. Mining SQL injection cross site scripting vulnerabilities using hybrid program analysis[C]//2013 International Conference on Software Engineering. IEEE Press, c2013:642-651.

[9] SHAHRIAR H,NORTH S,CHEN W C. Early detection of SQL injection attacks[J]. International Journal of Network Security & Its Applications, 2013, 5(4):53-65.

[10] VALEUR F,MUTZ D,VIGNA G. A learning-based approach to the detection of SQL attacks[M]. Detection of Intrusions and Malware, and Vulnerability Assessment. Springer Berlin Heidelberg, 2005: 123-140.

[11] KEMALIS K,TZOURAMANIS T.SQL-IDS: a specification-based approach for SQL-injection detection[C]//2008 ACM Symposium on Applied Computing. ACM, c2008:2153-2158.

[12] 陆开奎.基于动态污点分析的漏洞攻击检测技术研究与实现[D].成都: 电子科技大学,2013.
LU K K. The Research and realization of dynamic taint analysis based security attack detection technology[D]. Chengdu: University of Electronic Science and Technology of China, 2013.

[13] HUANG Y W,HUANG S K,TSAI C H. Web application security assessment by fault injection and behavior monitoring[C]//WWW'03 International Conference on World Wide Web. c2003:148-159.

[14] KALS S, KIRDA E, KRUEGEL C, et al. SecuBat: a Web vulnerability scanner[C]//International Conference on World Wide Web. c2006: 247-256.

[15] APPELT D, NGUYEN C D, BRIAND L C, et al. Automated testing for SQL injection vulnerabilities: an input mutation approach[C]//In-

ternational Symposium on Software Testing & Analysis, c2014: 259-269.

[16] 王苏南. 高速复杂网络环境下异常流量检测技术研究[D]. 郑州: 解放军信息工程大学,2012.
WANG S N. Research on anomaly detection technology in high-speed complex network environment[D]. Zhengzhou: PLA Information Engineering University,2012.

[17] ZHANG J, XIANG Y, WANG Y, et al. Network traffic classification using correlation information[J]. IEEE Transactions on Parallel & Distributed Systems, 2013, 24(1):104 - 117.

[18] 周爱平, 程光, 郭晓军. 高速网络流量测量方法[J]. 软件学报,2014, 25(1):135-153.
ZHOU A P, CHENG G, GUO X J. High-speed network traffic measurement method[J]. Journal of Software, 2014, 25(1):135-153.

[19] 王鹏, 兰巨龙, 陈庶樵. 粒度自适应的多径流量分割算法[J]. 通信学报, 2015, 36(1):211-217.
WANG P, LAN J L, CHEN S Q. Multipath traffic splitting algorithm based on adaptive granularity[J].Journal on Communicatio , 2015, 36(1):211-217.

[20] Pangolin-SQLinjection tools [EB/OL]. http://nosec.org/cn/productservice/pangolin, 2014-12-22.

[21] Sqlmap-Automatic SQL injection and databasetakeover tool[EB/OL]. http://sqlmap.org/, 2015-3-5.

作者简介：

赵宇飞（1990-），男，山西太原人，北京航空航天大学博士生，主要研究方向为网络安全。

熊刚（1977-），男，湖北汉川人，博士，中国科学院信息工程研究所正高级工程师、博士生导师，主要研究方向为网络测量、信息对抗、信息安全等。

贺龙涛（1974-），男，贵州遵义人，博士，国家计算机网络应急技术处理协调中心正高级工程师、博士生导师，主要研究方向为信息安全。

李舟军（1963-），男，湖南湘乡人，博士，北京航空航天大学教授、博士生导师，主要研究方向为网络与信息安全、数据挖掘与人工智能。