FLAG、ID、SUB 知道内容在报文中的偏移量. ID 和但这个偏移量会根据数据部分. 其中 DATA FLAG 一般填充 0 库服务器、客户端版本不同而变化, ID 还会受到协议版本号、SUB ID 和数据部分可能会重复出现,类型、长度等影响. 如图 2 因此需要根据报文中所示. ID 和 SUB ID 可以唯一关键词来确定后续数据部分的具体类型偏移量.

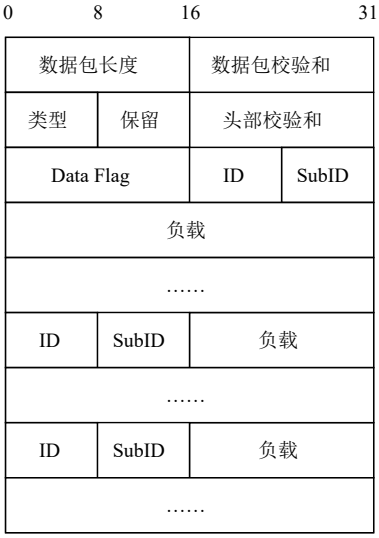

图 2 TNS 协议 Data 类型数据包格式

3 TNS 请求解析方案

本节将从样本获取、人工初步分析两方面介绍 TNS 请求解析方案的设计依据.

3.1 样本获取

根据对市场上常用的 TNS 协议版本、数据库服务器、客户端的调研, 以及在某电信集团数据中心机房采集到的数据流量显示, 常用的 TNS 协议版本包括 308、310、312、313、314、315. 为了保证通用性, 本文解析支持 TNS308 到 31 的 5 全版本. 在数据库服务器方面配置了 Windows 和 Linux 下的 Oracle8i、9i、10g、11g、12c, 而客户端采用了 java 连接 (包括 classes12、ojdbc14、ojdbc5、ojdbc6、ojdbc7 的 JAR 包)、Navicat(主版本号包括 10、11、12)、PL/SQL Developer(主版本号包括 6、7、8、9、10、11)、Oracle SQL Developer(主版本号包括 2、3、4) 和 PowerDesigner, 使用到的动态连接库包括 instant client 10.2、11.2 和 12.2 的 oci.dll.

3.2 人工初步分析

对于采集到的数据包, 根据请求者的 IP 和数据包中 ASCII 码是否包含连续的英文字母, 将请求报文单独提取了出来. 统计了请求报文数据类型的 ID 和 subID, 并结合前面一章的分析得到了请求报文的一个简单结构, 如图 3 所示. 初步分析揭示了 TNS 协议解析的两个主要问题.

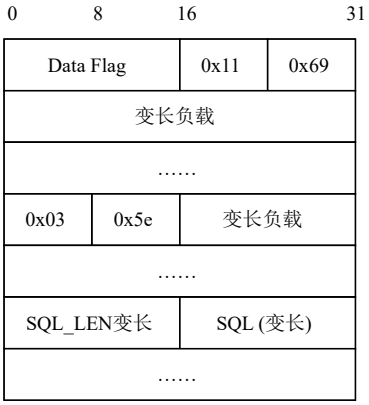

图 3 请求报文格式

第一个问题是协议中数据的位置和格式不固定. 对于提取协议请求内容来说, 重要的是知道内容在报文中的偏移量. 但这个偏移量会根据数据库服务器、客户端版本不同而变化, 还会受到协议版本号、数据类型、长度等影响. 因此需要根据报文中关键词来确定偏移量.

第二个问题是协议解析涉及的范围广, 数据量大, 人工分析的方式耗时耗力, 并且无法精确定位偏移量. 需要一个系统的分析方法. 而诸多协议逆向的方法[7–9]在不依赖先验知识的情况下对协议格式进行分析, 对包含负载的报文格式解析效果并不好.

本文通过数据挖掘的方法, 提取基于位置的关键词, 建立关键词与偏移量的关系. 依次对不同 TNS 协议的版本获取偏移量规则, 以提高工作效率. 之所以用协议版本作为样本划分, 是因为相对于操作系统、服务器、客户端版本, 相同协议版本的报文显示出更强的规律性.

4 关联挖掘获取偏移量

本节采用关联挖掘的思想, 确定字段值与偏移量的关系. 关联挖掘最早应用于顾客交易数据库中项集的关联规则[10], 随后关联规则的挖掘被广泛研究. 在关联挖掘中, 首先获得频繁集, 再从频繁集中通过预设的置信度获取关联规则. 在确定关键词与偏移量关系时, 可以先找到特定偏移量的频繁关键词, 再从中获取规则.

4.1 关联挖掘的数据格式

假设第k个报文偏移量为s, 则它的 SQL 前面存在s个字节. 将第i个字节的偏移量与值组为 2 元组, 记为

$$a_i^k = (loc_i^k, v_i^k) \tag{1}$$

其中, loc_i^k为第i个字节的偏移, v_i^k为第i个字节的值. 由于第i个字节的偏移量已知为i, 则有

$$a_i^k = (i, v_i^k) \tag{2}$$

并且有

$$a_i^k = a_j^m \Leftrightarrow i = j, v_i^k = v_j^m \tag{3}$$

也就是说仅当偏移量和值都相同时, 才认为两个报文上的某个字节是相同的. 记F为字节与偏移量的映射关系, 则有

$$F(a_1^k, a_2^k, \cdots, a_i^k, \cdots, a_s^k) = s \tag{4}$$

由于这个映射关系和某个具体的报文无关, 因此可以略去k, 即

$$F(a_1, a_2, \cdots, a_i, \cdots, a_s) = s \tag{5}$$

报文中对结构有关键影响的字节只占少数. 也就是说, 只需部分关键字节的值, 就可以确定报文偏移量. 转化为数学语言, 即

$$\exists r = [x_1, x_2, \cdots, x_i, \cdots, x_k] \tag{6}$$

满足

$$x_i \in \{a_1, a_2, \cdots, a_i, \cdots, a_s\}, i = 1, 2, \cdots, s \tag{7}$$

使得

$$r \to s \tag{8}$$

式 (8) 是一个关联规则的形式. 在关联挖掘中, 以a_i和s的取值范围作为项集, 将数据格式定为

$$msg = [a_1, a_2, \cdots, a_i, \cdots, a_s, s] \tag{9}$$

并从样本中获取数据集, 使用 Apriori 算法, 可以挖掘出形如式 (8) 的关联规则. 由于式 (8) 是针对偏移量的关联规则, 可以将样本数据先按照偏移量的不同划分为若干个组, 对每个组分别求频繁集. 每个组内偏移量是定值, 因此偏移量s一定在频繁集中. 在求关联规则时, 再将各组频繁集汇总. 依据偏移量的不同进行划分, 可以有效的降低计算量.

4.2 样本数据提取

样本数据提取指的是从初始网络数据包中提取出$a_1, a_2, \cdots, a_i, \cdots, a_s$和偏移$s$. 本文将 SQL 语句限制为 select、insert、create、drop、delete 等单词开头的常用语句. 然后采用字符串匹配的方式, 即可确定 SQL 语句的起始位置, 以及从开头到 SQL 的每一个字节, 从而自动而快速地生成样本数据集.

4.3 算法描述

关联挖掘采用的是 Apriori 算法[11], 具体描述如下:

算法 1. 关联挖掘偏移量算法

输入: 某个 TNS 协议版本的数据包样本集D

输出: 该 TNS 协议版本的字段值与偏移量的关联规则

1) 根据偏移量的长度将样本集D划分为 M 个小样本集D_i(i=1,2,...,n), 每个小样本集D_i中的样本具有相同的偏移量, 不同小样本集中的样本具有不同偏移量.

2) 对于每个D_i, 查找所有偏移和值都不变的字节, 将其组合为固定集S_i, 并从D_i中移除.

3) 对于每个D_i, 进行关联挖掘, 得到频繁集L_i.

4) 将S_i重新添加回频繁集L_i

5) 将所有频繁集L_i汇总, 得到大频繁集L. 设置置信度为 0.95, 从L中获取关联规则.

上述算法将置信度设置得接近于 1 是因为, 对于一个特定的字段组合, 其对应的偏移量也是通常固定的. 第 5) 步中, 仅搜索和偏移量相关的规则.

在挖掘的过程中发现, 小样本集D_i中数据报文存在着不少字节, 它们的偏移和值是完全固定的, 所以一定会出现在频繁集中. 但是这部分字节并不会在计算时被忽略, 反而是出现在每一次频繁集的迭代中. 为了减少计算量, 第 2) 步将这些偏移和值固定的字节在挖掘之前从样本集中移除, 在第 4) 步挖掘结束后重新添入频繁集.

5 实验分析

本节首先用前述的挖掘算法, 获取挖掘结果, 并对其含义做解释. 然后在实际系统工作环境下, 使用挖掘得到的规则对请求报文做解析. 同时对规则的准确度做评价.

实验环境采用 3.1 节描述的所有客户端和服务器, 分别在 Windows 和 Linux 环境下, 两两连接, 运行 SQL 脚本采集数据. 将采集数据按照 TNS 协议版本进行划分, 再分别使用关联挖掘的方法获取偏移量的规则.

5.1 最大最小规则

挖掘结果显示, 对于同一个偏移量, 得到规则不止一个. 记偏移量为s的集合为

$$R^s = \{r | r \to s\} \tag{10}$$

定义最小规则$r_{\min} \to s$为:

$$\begin{cases} \exists r_{\min} \in R^s \\ \forall r \in R^s, r \not\subset r_{\min} \end{cases} \tag{11}$$

定义最小规则$r_{\max} \to s$为

$$\begin{cases} \exists r_{\max} \in R^s \\ \forall r \in R^s, r_{\max} \not\subset r \end{cases} \tag{12}$$

若$r_{\min} \subseteq r_{\max}$, 则$r_{\min}$和$r_{\max}$构成一对最大最小规则. 很明显, 除非$R$为空集, $r_{\min}$和$r_{\max}$总是存在. 并且, 对于一个偏移量s, 可能存在不止一对最大最小规则.

在规则集中, 最大最小规则有很多用处. 最小规则因为包含的字节数量最小, 没有冗余, 适合直接用于请求报文 SQL 的提取. 而最大规则由于包含了更多的信息量, 适合对报文结构作进一步的分析. 对 TNS315 协议, 035e 报文头部到 SQL 的一对最大最小规则如图 4(a) 和 4(b) 所示. 图中最大规则因为篇幅原因, 省去了字节值前面的 0x.

{(2,0*x*02), (5,0*x*00), (7,0*x*01), (14,0*x*00),
(16,0*x*02), (19,0*x*00), (35,0*x*00)}→ 40

(a) 偏移量为40的最小规则

{(2,02), (5,00), (6,01), (7,01), (9,01), (10,01), (11,0*d*),
(12,00), (13,00), (14,00), (15,00), (16,02), (17,7*f*),
(18,*f*8), (19,00), (20,00),..., (35,00),...}→ 40

(b) 偏移量为40的最大规则

图 4 偏移量为 40 的最小规则及最大规则

5.2 细化协议格式

通过对各种最大规则的分析对比, 得到了请求报文的一个更详细的格式. 图 5 中是 TNS315 版本请求报文的部分格式. 图中的 Magic1、Magic2、Magic3 为定长字段. 图中的变长字段, 如 SQL_LEN、VarD1、VarD2 等, 均满足 len+data 的结构. Len 为 1 个字节, data 长度为 len 个字节. 图中的 VarS1 为变长字段, 一般为 2 个、4 个或 8 个字节的填充字段. 实验结果说明, 数据挖掘方法对获取变长结构的关联字段和报文中的定值字段非常有效.

5.3 规则解析效果测试

本文使用最小规则来解析并提取报文中 SQL 语句. 图 6 显示了各个 TNS 协议版本挖掘出的最小规则的数量. TNS 310 版本支持的服务器和客户端版本最多, 其格式最复杂, 挖掘出的规则数量也最多. 而 TNS 314 版本的格式则相对比较简单.

本文根据挖掘出的最小规则, 进行 Oracle 报文解析的实际测试. 通过上海某电信集团机房的专业数据库操作人员为期 2 个月的采集和测试, 初期的测试通过率仅为 71%, 如图 7 所示. 发现问题包括: 部分报文结构的缺失、部分变长字段发现了新结构. 主要原因在于样本集不够全面, 无法覆盖所有报文格式. 本文通过使用异常样例扩充样本集, 重新挖掘来完善规则, 提高测试通过率.

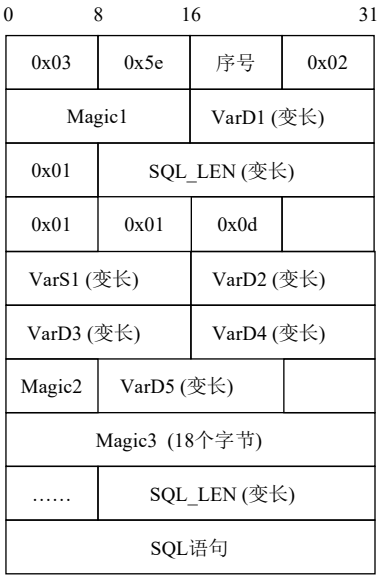

图 5 更详细的报文格式

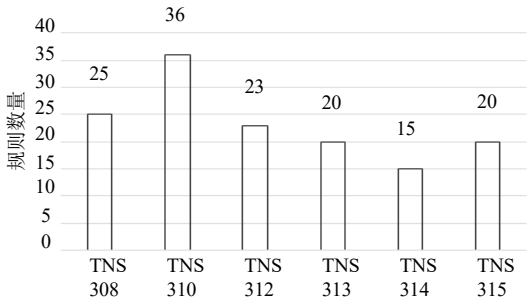

图 6 不同 TNS 协议版本的规则数量

在这之后, 本文一边实地运行解析系统, 一边更新解析系统的规则集. 采集到的数据包如果解析异常, 就会被添加进样本集中, 重新进行挖掘. 挖掘出的新规则会被添加进正在运行的解析系统规则集中. 到目前为止, 解析系统已运作 9 个月. 除了开头 2 个月有个别异常, 之后均能正常解析.

6 结论与展望

本文针对 Oracle 数据库 TNS 协议的请求报文, 提出了一个解决方案, 适用于多种常用操作系统、服务器、客户端和协议版本. 采用数据挖掘的方法来处理字节数多、意义不明的报文, 获取对报文结构有重要

影响的字段. 由于初期样本采集覆盖范围不够广, 挖掘的结果对于样本中出现频率少的报文类型并不友好. 后期校正使用解析过程中的异常样例扩充样本集, 反复挖掘, 提高对所有类型报文的适用性. 实地采集数据进行解析, 可以有效提取出请求报文中的 SQL 语句, 数据挖掘的方法可以有效地从大量数据中提取出规则, 省时省力. 下一步计划采用现有方法继续研究响应报文中的内容.

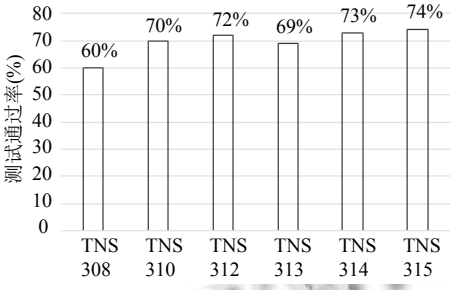

图 7 不同 TNS 协议版本的测试通过率

参考文献

1 Verizon Inc. 2016 Data Breach Investigations Report. http://www.verizonenterprise.com/resources/reports/rp_DBIR_2016_Report_en_xg.pdf. [2016-12-01].

2 安华金和. 2016 年数据库漏洞安全威胁报告. http://www.dbsec.cn/service/pdf/Database-vulnerability-security-threat-report-2016.pdf. [2016-12-01].

3 DB Engines. DB Engines Ranking. https://db-engines.com/en/ranking. [2017-5-27].

4 权元文. 基于 TNS 的 Oracle 数据库安全增强系统设计与实现. 电脑编程技巧与维护, 2011, (20): 142–144.

5 徐有为. 基于 TNS 协议的 Oracle 审计网关系统的设计与实现[硕士学位论文]. 北京: 中国科学院大学, 2014.

6 王召. 基于数据库审计系统 TNS 协议解析的研究与实现[硕士学位论文]. 北京: 华北电力大学, 2015.

7 Beddoe M. The protocol informatics project. http://www.4tphi.net/~awalters/PI/PI.html.

8 Cui WD, Kannan J, Wang HJ. Discoverer: Automatic protocol description generation from network traces. Proceedings of the 16th USENIX Security Symposium. 2007.

9 Wang YP, Li XJ, Meng J, *et al*. Biprominer: Automatic mining of binary protocol features. 2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies. Gwangju, South Korea. 2011. 179–184. [doi: 10.1109/PDCAT.2011.25]

10 Agrawal R, Imieliński T, Swami A. Mining association rules between sets of items in large databases. Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. 1993. 207–216. [doi: 10.1145/170036.170072]

11 Agrawal R, Srikant R. Fast algorithms for mining association rules. Proceedings of the 20th VLDB Conference. Santiago, Chile. 1994. 487–499.