计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(9):251−257 [doi: 10.15888/j.cnki.csa.007070]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于异构冗余的拟态数据库模型设计与测试①

赵琳娜, 倪　明, 喻卫东

(华东计算技术研究所, 上海 201808)
通讯作者: 赵琳娜, E-mail: 851985418@qq.com

摘　要: 数据库作为信息系统核心组件, 存放着大量重要数据信息, 易受到危害最大的 SQL 注入攻击. 传统数据库防御手段需要攻击行为的特征等先验知识才能实施有效防御, 具有静态、透明、缺乏多样性等缺陷. 本文在此背景下, 以拟态防御动态异构冗余原理为基础, 使用保留字拟态化模块、指纹过滤模块、拟态化中间件模块实现 SQL 注入指令的指纹化、去指纹化、相似性判决, 提出具有内生安全性的拟态数据库模型, 并使用渗透测试演练系统 DVWA 中的 SQL 注入模块对该模型进行安全性测试, 验证了拟态数据库模型的可用性和安全性.

关键词: Web 安全; SQL 注入; 拟态防御; 动态异构冗余; 数据库

引用格式: 赵琳娜,倪明,喻卫东.基于异构冗余的拟态数据库模型设计与测试.计算机系统应用,2019,28(9):251–257. http://www.c-s-a.org.cn/1003-3254/7070.html

Design and Test of Mimetic Database Model Based on Heterogeneous Redundancy

ZHAO Lin-Na, NI Ming, YU Wei-Dong

(East China Institute of Computing Technology, Shanghai 201808, China)

Abstract: As the core component of the information system, the database stores a large amount of important data information and is vulnerable to the most harmful SQL injection attacks. Traditional database defense methods require prior knowledge such as the characteristics of attack behavior to implement effective defense, and have the defects of static, transparent, and lack of diversity. In this context, based on the dynamic heterogeneous redundancy principle of mimicry defense, the reserved word mimicry module, fingerprint filtering module and mimetic middleware module are used to realize fingerprinting, de-fingerprinting and similarity judgment of SQL injection instructions. A mimetic database model with endogenous security is proposed, and the model is tested using the SQL injection module in the penetration test rehearsal system DVWA to verify the availability and security.

Key words: Web security; SQL injection; micmic defense; dynamic heterogeneous redundancy; database

随着组织机构开展的互联网业务日益增多, 由于大量已知和未知的漏洞及后门的存在, 面临的安全隐患不容小觑. 数据库作为 Web 应用的核心组成部分, 存有网站动态资源和用户信息, 最易受到攻击. 当数据库被恶意攻击成功后, 会导致用户信息泄露、内容被篡改等不良后果. 除此之外, 攻击者利用漏洞完成一次数据攻击所需要的时间极短, 给管理员及时发现数据库入侵并补救的工作带来了极大困难.

传统数据库的防御手段是基于威胁特征感知的精确防御, 需要攻击行为的特征等先验知识才能有效实施防御, 具有静态、透明、缺乏多样性等缺陷. 拟态防御动态异构冗余架构利用漏洞的平台相关性, 在期望功能等价条件下, 拟态界内的执行体具有处理结构的相异性和冗余性, 对“已知的未知”风险或“未知的未知”

① 基金项目: 国家重点研发计划 (2016YFB0800100)
Foundation item: National Key Research and Development Program of China (2016YFB0800100)
收稿时间: 2019-03-06; 修改时间: 2019-04-02; 采用时间: 2019-04-04; csa 在线出版时间: 2019-09-05

威胁具有内生防御效果[1].

本文依托拟态防御的动态异构冗余原理, 提出拟态数据库模型, 通过保留字拟态化模块、指纹过滤模块和拟态化中间件模块分别实现注入指令的异构化、去指纹化和基于相似性判决的选择性执行, 具有内生安全性.

1 传统数据库防御手段

根据世界著名的 Web 安全与数据库安全研究组织 OWASP 发布的十大最关键的 Web 应用安全风险 (OWASP Top 10), 与数据库安全密切相关的 SQL 注入攻击 (SQL Injection) 在 2013 年和 2017 年均被列为威胁最严重的攻击方式[2]. 攻击者通过利用客户端任何能够提交信息与数据库进行交互的地方作为注入点, 提交重构的 SQL 命令, 欺骗数据库执行, 并针对服务器端返回的结果进行分析, 窃取数据库内的关键信息、恶意篡改数据和控制服务器.

基于 SQL 注入攻击的原理, 当前防御方法大多从代码层面和平台层面入手[3], 代码层面的防御方法主要是在需要提交数据的页面设置规范的 SQL 语句格式, 对用户提交的信息进行过滤, 保证用户提交的信息不会对 SQL 查询语句造成歧义, 如参数化查询、参数过滤、存储过程等; 平台层面的防御方法主要是利用 URL 重置技术, 将含有注入点的网址屏蔽起来, 如 URL 重定向技术等. 除此以外, 还配合使用设置有限的访问权限、预编译、加强日常监督与检测等手段.

1.1 参数化查询

参数化查询 (parameterized query) 指在设计与数据库链接并访问数据时, 在需要填入数值或数据的地方, 采用参数传递值, 避免直接赋值. 数据库系统把参数值应用到查询计划中, 执行查询结果, 若改变参数值再执行, 则得到不同的查询结果, 但 SQL 语句不需要重构提交, 也不需要再次生成查询计划, 参数值的变化影响不了查询计划[4].

使用参数化查询技术, 数据库服务器不会将参数的内容视为 SQL 指令的组成部分来处理, 在一定程度上避免了数据库编译运行参数中含有恶意指令的情况.

1.2 参数过滤

参数过滤分为对参数类型或长度的限制和对危险字符的过滤/转义.

针对特定的、功能单一的参数提交接口, 尽可能详细地限制允许用户输入的参数类型及长度. 当参数类型必须包含字符或长度无法直接控制时, 通过在后台代码中设置黑/白名单等过滤规则, 将参数中的非法字符转义为合法字符提交或不予执行, 对参数中的敏感信息进行检测及过滤/转义, 避免非法字符被系统重构为查询语句导致 SQL 注入.

1.3 存储过程

存储过程存储在数据库系统内部, 是一组完成特定功能的 SQL 语句集. Web 应用程序通过调用存储过程事先构建好的 SQL 查询语句代码来执行相应操作.

存储过程在使用时起到 3 点安全作用: 首先, 存储过程在执行前要进行预编译, 编译出错不予执行; 其次, 对于数据的授权访问是基于存储过程而不是直接访问基本表, 攻击者无法探测到 Select 语句; 最后, 存储过程可以指定和验证用户提交的参数类型[5].

1.4 URL 重置

URL 重置是将提交至服务器的 Web 程序截取下来, 自动将请求重新定位到其他 URL 地址的过程. 目前可以通过两种方式实现 URL 重置 (以 ISAPI 筛选器为例)[3]:

1) 使用 ISAPI 筛选器在 IIS Web 服务器级别实现 URL 重写;

2) 使用 HTTP 模块或 HTTP 处理程序在 ASP.NET 环境实现 URL 重写.

微软的 IIS 中包含有 ISAPI_Rwrite 的 URL 重定向组件, 提供基于正则表达式的网址重写引擎, 对并不存在的 URL 请求重写[6]. 通过对 Rewrite 组件的应用, 实现对含有非法参数的 URL 进行重定向.

在已知现有 SQL 注入攻击方法的特征下进行针对性防御的传统防御手段仍然存在缺陷, 如参数化查询及参数过滤在某些情况下应用场景单一, 并且在过滤规则设置不完善的情况下仍可以被攻击者绕过进而成功实现 SQL 注入; 存储过程并不支持所有的数据库平台, 如果存储过程中执行的命令是拼接字符串, 则仍然存在被注入攻击的隐患; URL 重置技术在利用 ISAPI 筛选器实现时对平台有一定的局限性, 只适用于 Windows 系列操作系统的服务器.

2 拟态数据库模型

现有数据库防御手段是在已知攻击特征等先验知识的前提条件下才能实施有效防御, 但在安全领域仍然存在大量未知风险. 针对现有数据库防御手段的静态性和确定性, 本文基于拟态防御的动态异构冗余原

理, 提出具有内生安全性的拟态数据库模型. 该模型通过保留字拟态化模块完成 SQL 保留字指纹特征的差异化描述, 调用指纹过滤模块对 SQL 注入指令进行去指纹处理, 使用拟态化中间件对来自不同执行体的 SQL 注入指令进行在线相似性判决.

2.1 拟态数据库模型结构

拟态数据库模型在传统数据库系统的基础上, 新增保留字拟态化模块、指纹过滤模块和拟态化中间件模块对客户端输入的合法或非法的 SQL 指令进行判决及处理. 结构如图 1 所示.

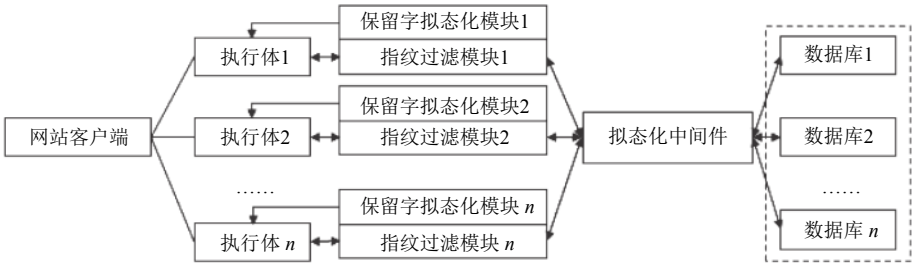

图 1 拟态数据库模型结构

由图 1 可知, 在拟态数据库模型中, 保留字拟态化模块先对运行在各执行体上的 Web 应用源码进行 SQL 保留字的指纹差异化描述; 当用户从网站客户端发起合法的 SQL 任务时, SQL 保留字经过指纹过滤模块的去指纹化处理后传入拟态化中间件模块进行表决, 底层数据库执行经过表决的 SQL 语句后返回执行结果; 当用户从网站客户端发起恶意 SQL 任务时, 由于未经过保留字拟态化模块的指纹差异化描述, 所以传入的未带有指纹特征的 SQL 指令不会被执行;若攻击者获取到少数执行体的中保留字的指纹特征, 伪造经过指纹化处理的 SQL 指令, 发起恶意 SQL 任务, 此时不管是从网站客户端还是从受攻击的执行体上传 SQL 任务, 在经过拟态化中间件时, 由于和其他执行体上的 SQL 保留字的指纹特征不同, 因此伪造的 SQL 指令只会加入到受攻击执行体的对应判决队列, 而此时其他执行体的对应判决队列为空, 根据少数服从多数的判决原则, 伪造的恶意 SQL 语句并不会通过相似性判决继续向下执行.

2.2 保留字拟态化模块

保留字是有特定语义的单词或字符串, 是数据库识别指令的方法, 在客户端通过 SQL 语言操作数据库时, 所使用的 SQL 指令即属于 SQL 保留字.

在未使用保留字拟态化模块时, 运行在各执行体上 Web 应用中的数据库访问 SQL 保留字是同构的. 保留字拟态化模块基于拟态防御中各执行体异构冗余的原理, 通过对运行在各执行体上 Web 应用中的 SQL 保留字进行指纹特征差异化描述, 实现 Web 应用在各拟态执行体上 SQL 保留字异构化, 防止恶意注入的 SQL 保留字被后续模块执行. SQL 保留字拟态化模块的工作原理如图 2 所示.

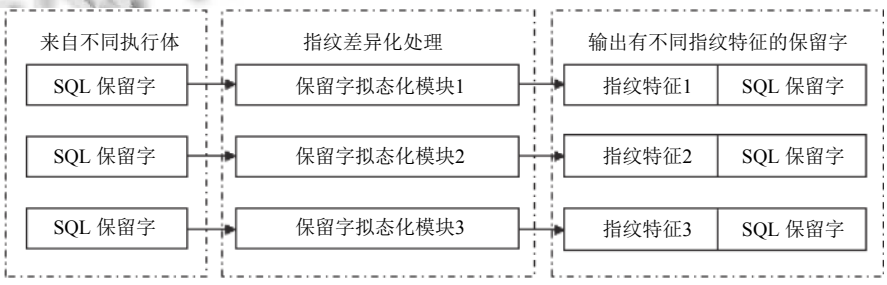

图 2 保留字拟态化模块工作原理

2.3 指纹过滤模块

保留字拟态化模块输出的来自不同执行体的数据库访问 SQL 保留字带有不同的指纹特征, 并不是可以被正常识别并执行的 SQL 指令, 为保证指令能够在后

续拟态化模块中的可用性, 引入指纹过滤模块.

指纹过滤模块在拟态化中间件对来自各执行体的SQL 指令进行相似性判决前被调用, 将经过指纹差异化处理的 Web 应用的数据库访问 SQL 保留字进行去指纹化, 使异构化的 SQL 保留字可被后续拟态化模块识别并进一步执行. 指纹过滤模块的工作原理如图 3 所示.

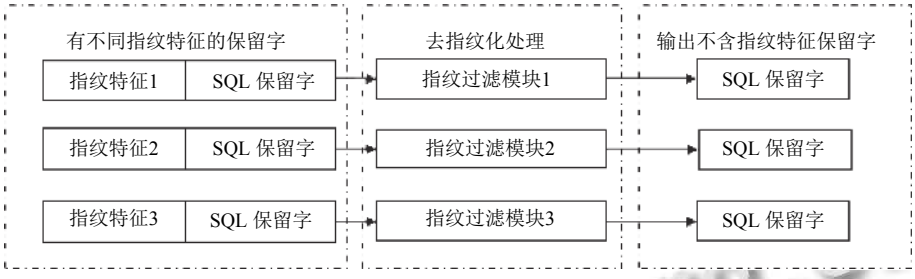

图 3 指纹过滤模块工作原理

2.4 拟态化中间件模块

数据库中间件[7]处于数据库和用户应用之间, 用于屏蔽异构数据库的底层细节问题. 当用户向 Web 服务器发出 SQL 请求时, 通过数据库中间件对 SQL 请求进行特定的分析, 根据分析结果连接数据库, 将 SQL 请求转发给数据库服务器. 数据库服务器执行 SQL 语句后, 将查询结果通过数据库中间件, 传回给用户. 以 MyCAT 为例, 传统数据库中间件结构[8]如图 4 所示.

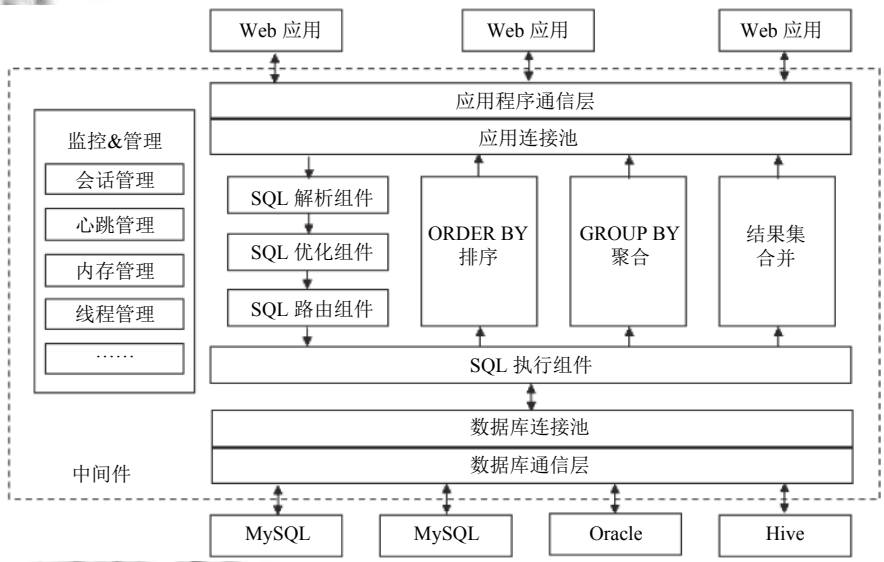

图 4 传统数据库中间件结构

根据图 4 可知传统数据库中间件的核心由监管模块、执行模块组成, 执行模块包括 Web 应用端及数据库端的通信协议、连接池以及 SQL 处理部分, 其中 SQL 处理部分包括 SQL 解析组件、SQL 优化组件、SQL 路由组件和 SQL 执行组件.

当 Web 应用通过通信协议向数据库端发起 SQL 任务时, 在监管模块的协调下, SQL 解析组件首先对 SQL 语句进行语法分析, SQL 优化组件将 SQL 语句转译为底层数据库可识别的语言, SQL 路由组件确定对应的数据库, 最后由 SQL 执行组件生成执行计划, 保证分发到底层数据库的 SQL 语句能正确执行[9]. 数据库中间件对数据库返回的执行结果进行排序、合并等处理后, 将最终结果返回给 Web 应用.

拟态化中间件模块在传统数据库中间件的基础上, 增加了在线拟态判决器和同步机制.

在线拟态判决器是对已经过指纹过滤的来自不同执行体的 SQL 指令根据少数服从多数的原则进行相似性判决, 选择相似程度最高的 SQL 指令进行执行;

同步机制[10]通过交换每个成员中所有已更新的记录和对象, 实现副本集成员的同步; 用于维护分布式环境下各个节点数据库数据的一致性. 拟态判决器的结构如图 5 所示.

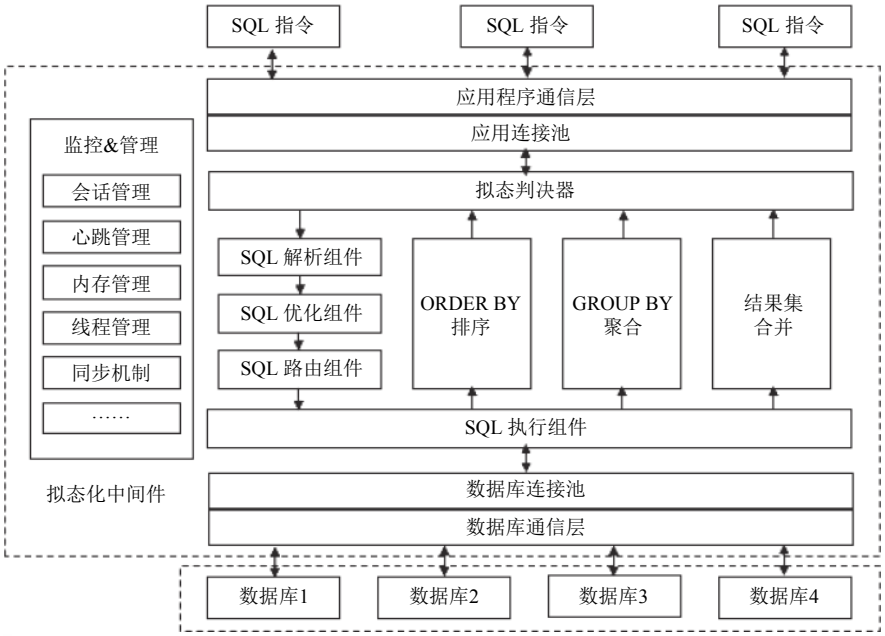

图 5 拟态化中间件结构

由图 5 知, 当部署在拟态执行体上的 Web 应用发起 SQL 任务时, 拟态化中间件根据各执行体的连接信息, 在监管模块的协调下, 调用对应的指纹过滤模块对异构的 SQL 保留字进行去指纹化; 随后将来自不同执行体的 SQL 指令加入对应的判决队列中, 并使用拟态判决器在线进行相似性判决; 若判决来自不同执行体的 SQL 指令一致, 则该 SQL 指令将生成执行计划发送到底层数据库进行执行; 若判决来自不同执行体的 SQL 指令不完全一致, 则按照少数服从多数的原则, 执行多数执行体相同的 SQL 指令; 若判决来自不同执行体的 SQL 指令完全不一致, 则均不执行且返回报错信息. 在成功执行 SQL 指令后, 将各执行体 SQL 指令保存在对应缓冲区中的执行结果进行完整性验证, 在同步机制的作用下对出现故障的数据表信息进行恢复, 最终返回验证合法的结果.

拟态防御动态异构冗余 (Dynamic Heterogeneous Redundancy, DHR) 架构的抗攻击性分析[11]中, 若拟态界的执行体集中有$n(n=2k+1,k=0,1,2\cdots)$个异构执行体, 针对特定漏洞的攻击成功率如下:

$$P_{vul}=q_{vul}\cdot\varepsilon_k(vul)\cdot I(vul) \tag{1}$$

针对随机选择的漏洞子集组合攻击成功率如下:

$$P_S=\sum_{i=1}^{N}\alpha_i\cdot q_{vul_i}\cdot\varepsilon_k(vul_i)\cdot I(vul_i) \tag{2}$$

其中, q_{vul_i}是攻击者利用漏洞vul_i对含vul_i的执行体进行攻击的成功率, 与攻击者能力密切相关, $\varepsilon_k(vul)=\frac{N'}{N}$是漏洞$vul$的$k$阶输出一致率, 即对漏洞$vul$, 在$N$次攻击中有$N'$次其$k$个输出相同且与正常输出不一致, α_i是攻击者利用漏洞vul_i攻击时的选择权重, $I(vul)=\begin{cases}1, & t_{vul}\leqslant T\\ 0, & t_{vul}>T\end{cases}$为系统动态变化周期 T 和攻击时间t_{vul}的综合考虑参数. 由于在拟态防御环境下, 系统动态变化周期 T 较短, 使得的$I(vul)=0$概率增大; 系统内各执行体异构性高, 因此k阶输出一致率$\varepsilon_k(vul)$低, 且指纹特征在 Web 应用源代码中的 SQL 保留字中, 大大增加了攻击者获取到$\frac{n+1}{2}$个执行体中指纹特征的难度, 因此攻击者难以获得足够多的指纹特征伪造恶意指令绕过拟态化中间件中的相似性判决.

3 模型测试

SQL 注入攻击作为主流的数据库攻击方式, 主要

集中在利用网站服务端口接收用户输入的功能[12], 将构造的 SQL 语句传入数据库服务器, 欺骗其执行开发者规定外的恶意任务.

DVWA 是一款内置不同安全级别漏洞模块的渗透测试演练系统, 安全级别越低, 漏洞利用成功率也越高. 现使用其 SQL 注入模块来测试拟态数据库模型的基本防御功能.

3.1 测试过程

为单独测试拟态数据库模型的基本防御功能, 简化测试模型, 准备一组由 3 个同构执行体组成的 Web 服务器和一台数据库服务器, 安装相同的版本的 DVWA;在使用拟态数据库的情况下, 对部署在三台 Web 服务器上 DVWA 中的 SQL 保留字进行指纹异构化处理, 在数据库服务器中使用指纹过滤模块和拟态化中间件模块.

测试过程如下:

(1) 在使用传统数据库的情况下, 先后用 sqlmap、pangolin 等 SQL 注入测试工具分别对 DVWA 中低安全级别的 SQL 注入测试模块和高安全级别的 SQL 注入测试模块实施 SQL 注入测试;

(2) 在使用拟态数据库模型的情况下, 使用注入 DVWA 低安全级别的传统数据库使用的 SQL 注入方法 (后文称低级 SQL 注入), 针对 DVWA 低安全级别的 SQL 注入测试模块实施注入;

(3) 若注入失败, 则使用注入 DVWA 高安全级别的传统数据库使用的 SQL 注入方法 (后文称高级 SQL 注入), 针对使用拟态数据库模型的 DVWA 低安全级别的 SQL 注入测试模块实施注入;

(4) 在传统 SQL 注入方法失败的情况下, 构造含有某一执行体指纹特征的 SQL 注入语句, 再次进行 SQL 注入测试.

3.2 测试结果

以使用 sqlmap 为例, 测试结果及说明如下:

(1) 使用传统数据库时对 DVWA 低安全级别 SQL 注入模块进行 SQL 注入的结果.

如图 6 所示, 在不使用拟态数据库模型的情况下, 当 Web 应用的安全性较差时, 攻击者可以轻松实现 SQL 注入.

(2) 使用传统数据库时对 DVWA 高安全级别 SQL 注入模块进行 SQL 注入的结果.

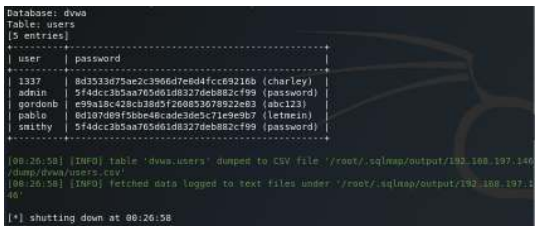

图 6 DVWA 低安全级别数据库 SQL 注入结果

如图 7 所示, 在不使用拟态数据库模型的情况下, 尽管 Web 应用的安全性较高, 但攻击者也可以通过各种绕过手段, 使用 sqlmap 等工具成功实施 SQL 注入攻击.

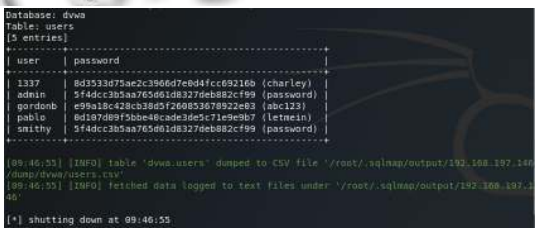

图 7 DVWA 高安全级别数据库 SQL 注入结果

(3) 使用拟态数据库模型时对 DVWA 低安全级别的 SQL 注入测试模块进行低级 SQL 注入的结果.

如图 8 所示, 在使用拟态数据库模型的情况下, 对 DVWA 低安全级别的 SQL 注入测试模块进行低级 SQL 注入, 由于低级 SQL 注入手段通过构造 SQL 语句进行注入, 并未带有执行体指纹信息, 拟态数据库模型并不会将其当做 SQL 指令进行执行, 因此拟态数据库模型可以防御低级 SQL 注入.

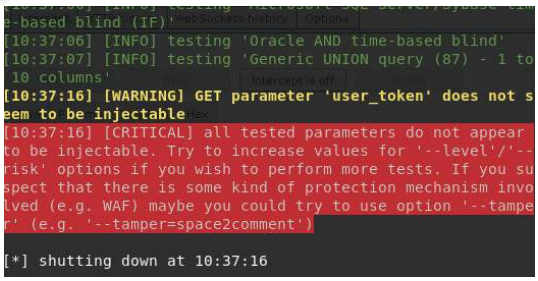

图 8 低级 SQL 注入结果

(4) 使用拟态数据库模型时对 DVWA 低安全级别的 SQL 注入测试模块进行高级 SQL 注入的结果.

如图 9 所示, 在使用拟态数据库模型的情况下, 对 DVWA 低安全级别的 SQL 注入测试模块进行高级 SQL 注入, 尽管高级 SQL 注入可通过绕过手段成功将

SQL 指令注入传统数据库欺骗其执行, 但由于最终拟态数据库模型接收到的 SQL 语句中并未带有指纹特征, 因此并不会将 SQL 注入语句作为指令进行执行, 说明拟态数据库模型可以防御高级 SQL 注入.

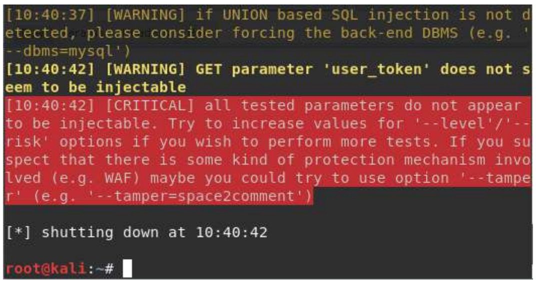

图 9　高级 SQL 注入结果

(5) 构造含某一执行体指纹特征的 SQL 注入语句进行 SQL 注入的结果

如图 10 所示, 在使用拟态数据库模型的情况下, 使用含某一执行体指纹特征的 SQL 注入语句对 DVWA 低安全级别的 SQL 注入测试模块进行 SQL 注入失败, 因为拟态数据库模型只接收到了带有一个执行体指纹特征的 SQL 指令, 在拟态化中间件模块进行 SQL 指令的相似性判决时, 另外两个执行体对应的判决队列为空, SQL 指令不会继续向下执行. 因此在少量指纹泄露的情况下, 拟态数据库模型可以防御构造指纹特征的 SQL 注入.

图 10　含指纹特征的 SQL 语句注入结果

4　结论与展望

针对传统数据库防御手段的静态性和确定性, 本文基于拟态防御的动态异构冗余原理提出应用于拟态系统中的拟态数据库模型. 模型的安全性测试结果证明该模型在执行体同构环境下的可用性和安全性, 若应用于执行体异构的拟态系统中, 则系统数据库的安全性更会大大增加. 但在测试过程中发现, 引入拟态数据库模型, 由于存在去指纹化处理及相似性判决步骤, 模型的时间性能有所降低, 影响了数据库查询的效率.

今后将进一步探索能有效提高拟态数据库模型判决速率切实可行的实现方法, 并在现有拟态数据库模型的基础上, 对保留字指纹特征的短期内动态变化以及注入攻击触发指纹特征变化的可能性进行更深一步的研究.

参考文献

1 邬江兴. "网络安全再平衡战略"之抓手: 拟态防御. 中国信息安全, 2018, (6): 46–50. [doi: 10.3969/j.issn.1674-7844.2018.06.037]
2 谭军. OWASP 发布十大 Web 应用安全风险. 计算机与网络, 2017, 43(23): 52–53. [doi: 10.3969/j.issn.1008-1739.2017.23.060]
3 李晓龙. 基于 SQL 注入攻击的三种防御技术. 湖北文理学院学报, 2013, 34(5): 18–21.
4 隋励丽, 张恒博. 基于参数查询防止 SQL 注入攻击的方法. 大连民族学院学报, 2012, 14(5): 495–497. [doi: 10.3969/j.issn.1009-315X.2012.05.020]
5 童晓冬. 用存储过程防范 SQL Injection 的技术分析. 商场现代化, 2010, (12): 11. [doi: 10.3969/j.issn.1006-3102.2010.12.007]
6 吴贵山. SQL 注入攻击防御策略的研究. 计算机与网络, 2012, 38(9): 70–73. [doi: 10.3969/j.issn.1008-1739.2012.09.071]
7 黄伟婷. 数据库中间件技术及其应用. 漳州师范学院学报(自然科学版), 2006, 18(2): 25–30. [doi: 10.3969/j.issn.1008-7826.2006.02.006]
8 叶炜. 分布式数据库中间件中的查询优化[硕士学位论文]. 上海: 东华大学, 2016.
9 漆绍洋. 基于 Mysql 的分布式访问中间件中 sql 处理模块的设计与实现[硕士学位论文]. 南京: 南京大学, 2016.
10 苏昆仑, 张铮, 仝青, 等. 一种数据库离线表决与同步方案的设计和实现. 信息工程大学学报, 2018, 19(1): 114–117. [doi: 10.3969/j.issn.1671-0673.2018.01.023]
11 邬江兴. 网络空间拟态防御导论. 北京: 科学出版社, 2017. 283–295.
12 张卓. SQL 注入攻击技术及防范措施研究[硕士学位论文]. 上海: 上海交通大学, 2007.