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.

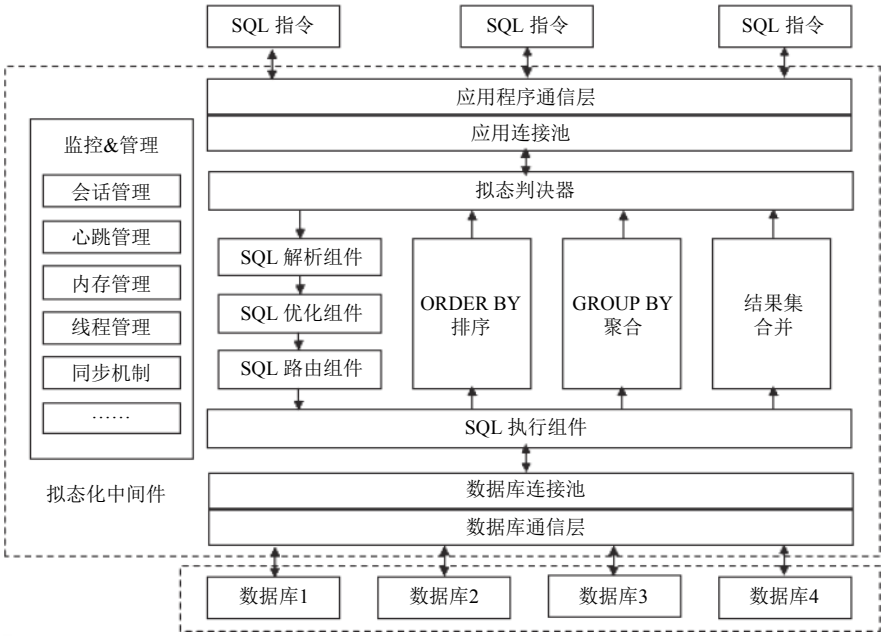



图 5 拟态化中间件结构

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, $\varepsilon_k(vul)=\dfrac{N'}{N}$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, $\alpha_i$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$\dfrac{n+1}{2}$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.

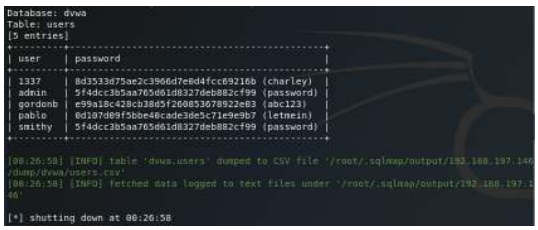



图 6 DVWA 低安全级别数据库 SQL 注入结果

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.

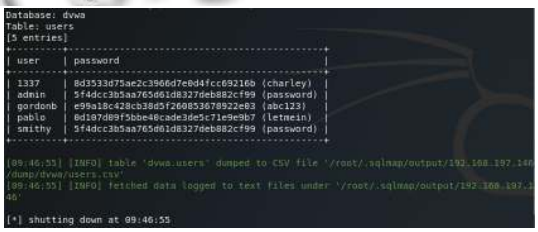



图 7 DVWA 高安全级别数据库 SQL 注入结果

(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.

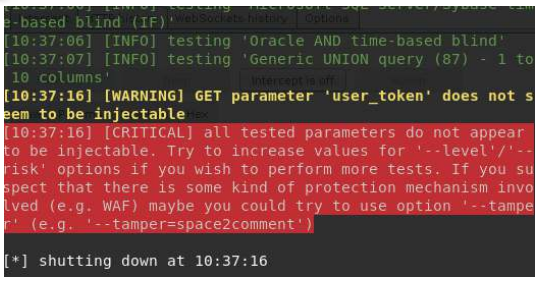
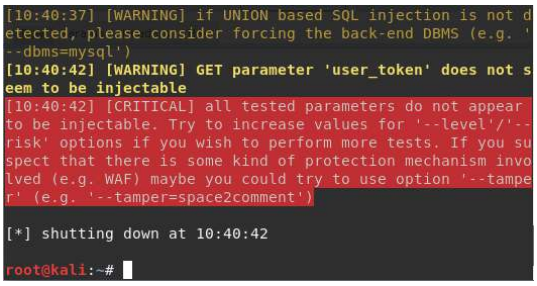



图 8 低级 SQL 注入结果

(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.



```
[10:40:37] [WARNING] if UNION based SQL injection is not d
etected, please consider forcing the back-end DBMS (e.g. '
--dbms=mysql')
[10:40:42] [WARNING] GET parameter 'user_token' does not s
eem to be injectable
[10:40:42] [CRITICAL] all tested parameters do not appear
to be injectable. Try to increase values for '--level'/'--
risk' options if you wish to perform more tests. If you su
spect that there is some kind of protection mechanism invo
lved (e.g. WAF) maybe you could try to use option '--tampe
r' (e.g. '--tamper=space2comment')

[*] shutting down at 10:40:42

root@kali:~#
```

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.

```
root@kali:~# python SQL.py
输入目标域名（例：http://127.0.0.1/）
http://192.168.197.146/
DVWA登录账号
admin
DVWA登录密码
password
[*] INFO Testing :http://192.168.197.146//vulnerabilities/sqli/?id=11%27+
[*] INFO Testing :http://192.168.197.146//vulnerabilities/sqli/?id=11%27+
[*] INFO Testing :http://192.168.197.146//vulnerabilities/sqli/?id=11%27+
[*] INFO Testing :http://192.168.197.146//vulnerabilities/sqli/?id=11%27+
[*] INFO Testing :http://192.168.197.146//vulnerabilities/sqli/?id=11%27+
[*] INFO Testing :http://192.168.197.146//vulnerabilities/sqli/?id=11%27+
[*] INFO Testing :http://192.168.197.146//vulnerabilities/sqli/?id=11%27+
[*] INFO Testing :http://192.168.197.146//vulnerabilities/sqli/?id=11%27+
[*] INFO Testing :http://192.168.197.146//vulnerabilities/sqli/?id=11%27+
[-] INFO 获取字段长度失败......
[*] TESTING DATABASE LEN .......
[-] INFO 获取数据库信息失败.......
[-] INFO 注入测试失败......
[-] INFO 程序结束！
```

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.