计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(10):219−225 [doi: 10.15888/j.cnki.csa.007079]

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

基于 Mycat 的拟态数据库中间件研究①

曹国栋, 倪　明, 喻卫东, 王　灿

(华东计算技术研究所, 上海 201808)
通讯作者: 曹国栋, E-mail: guodong-cao@qq.com

摘　要: 数据库系统的安全不仅依赖于数据库本身的安全, 还受网络环境和操作系统的安全性影响, 拟态防御是邬江兴院士首次提出的基于动态异构冗余体系架构, 协同实现数据库所依赖环境, 使数据库安全由被动防御变为主动防御. 本文通过 Mycat 作为数据库中间件, 通过对数据库的读写分离, 集群的高可用, 分布式事务处理, 对指纹化 SQL 识别, 以减轻单一数据库的数据存取和处理压力. Mycat 将数据库模块变为异构动态冗余模式, 实现数据库拟态化, 使数据高效, 快速, 安全的存取和切分.

关键词: 拟态防御; Mycat; 高集群; 高可用; 分布式处理; 指纹 SQL

引用格式: 曹国栋,倪明,喻卫东,王灿.基于 Mycat 的拟态数据库中间件研究.计算机系统应用,2019,28(10):219–225. http://www.c-s-a.org.cn/1003-3254/7079.html

Research on Mimic Data Processing Based on Mycat Middleware

CAO Guo-Dong, NI Ming, YU Wei-Dong, WANG Can

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

Abstract: The security of the database system depends not only on the security of the database itself, but also on the security of the network environment and the operating system. The mimicry defense is proposed firstly by Academician WU Jiang-Xing based on the dynamic heterogeneous redundancy architecture, and the database is dependent on the environment. Security has changed from passive defense to active defense. This study uses Mycat as a database middleware, through the separated read and write of the database, the cluster's high availability, distributed transaction processing, fingerprinting SQL identification, to alleviate the data access and processing pressure of a single database. Mycat turns the database module into a heterogeneous dynamic redundancy mode, which enables database mimicking, enabling efficient, fast, and secure access and segmentation of data.

Key words: cyberspace minic defense; Mycat; high cluster; high availability; distributed processing; fingerprint SQL

现在社会, 信息技术飞速发展, 伴随着数以亿计的数据产生, 数据库作为整个系统中信息输入和输出的重要组件, 对于其保护和处理, 日益成为人们关注的焦点. 据统计仅 2018 年数据泄露事件高达 945 次, 导致的信息泄露数量达到 45 亿条之多, 信息量陡增 133 个百分点.

2018 年 5 月, 在南京举办的“强网杯”拟态防御挑战赛, 在国内外 22 支顶尖战队的高强度攻击下, 拟态防御设备成功封堵了所有攻击, 即使开放管理员权限, 在拟态防御设备中随意注入后门, 也没有任何战队突破拟态防御, 为拟态防御进行全方位、高强度的安全检验. 拟态防御[1,2](Cyberspace Mimic Defense, CMD)是在功能等价的条件下提供可控的执行环境的跳变和迁移, 使攻击者对目标环境的难以掌握[3,4]. 在拟态环境中, 海量数据存储与访问是系统设计与使用的瓶颈问题, 利用开源的分布式存储数据库中间件 Mycat, 通过对数据进行水平切分, 将不同的表映射到不同的数据

① 收稿时间: 2019-03-14; 修改时间: 2019-04-04; 采用时间: 2019-04-10; csa 在线出版时间: 2019-10-15

库中, 通过集群管理, 事务分布式处理[5], 实现数据库容量的扩充和数据库结构的冗余, 加之其对 SQL 语句有拦截和分析的作用, 依据指纹特征对执行体指纹化 SQL 指令进行特征化处理, 发现并剔除攻击者注入的非法指令. Mycat 对数据库返回的数据进行表决, 判断异常数据库, 并对出现故障的数据库进行还原保护, 实现数据库的拟态化.

1 拟态介绍

拟态防御 (CMD) 是中国工程院邬江兴院士在 2013 年的提出的关于网络空间安全防御的创新性理论, 网络空间拟态防御是基于一种主动防御和被动防御相结合的网络安全防御架构, 克服了以往计算机系统漏洞, 后门或者病毒的时间不确定性, 危害未知性, 资源破坏性, 为将来的网络安全防御提供普适创新的理论和方法指导[6, 7].

1.1 拟态防御应用

在网络基础设施中, 一些高校和科研院所已提出了拟态防御中异构性的构造方法, 文献[8]提出了利用软硬件多样性实现多源异构化处理方法; 文献[9,10]提出了利用软件多样化编译对单一软件进行异构化处理的方法,也随之研制出了包括拟态 Web 服务器[11,12]、拟态防御路由器[13,14]拟态 DNS 服务器[15]等应用设备.

1.2 拟态防御原理

拟态防御的基本原理是: 在不依赖未知攻击特征信息的前提下, 通过多个等价等功能的不同体系结构的执行体, 利用动态异构冗余架构 (DynamicHeterogeneous Redundant Architecture, DHRA)[2], DHRA 模型如图 1 所示, 实现运行环境、网络、数据、软件等结构的主动切换或快速转移, 代表性技术如表 1, 使攻击者难以判断目标对象的运行环境和机制, 提高攻击者在时间维度和空间维度的攻击成本和难度[8].

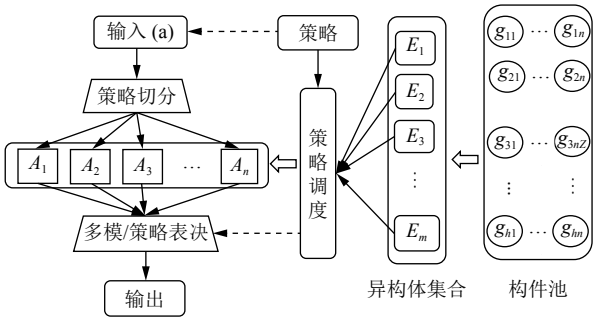

图 1 DHRA 模型

表 1 动态性技术分类

序号	类别	代表性技术
1	动态网络	IP 地址跳变端口跳变
2	动态平台	基于编译多样性技术的 N 变体系统
3	动态运行环境	地址空间随机化, 指令集随机化
4	动态软件	软件实现多样性
5	动态数据	数据与程序随机化加密存储

在高可靠的非相似余度“容错”模式下, 执行体通过可重组, 可重构, 可重建和可重定义等动态构造方法, 实现异构性, 动态性和冗余判断. 动态异构冗余模型 (DHRA) 由输入, 输入代理, 执行体集 ($A_1, A_2, A_3, \cdots, A_n$), 多模/策略表决器, 输出组成. 根据系统输入, 异构元素池中选择异构元素, 组成 m 个异构构件, 通过系统的策略调度, 形成 n 个异构执行体, 经过对执行体产生的结果多模判决, 决定系统输出[2].

2 拟态数据库中间件

数据库中间应用于 Web 服务器和数据服务器之间, 对两者之间交互的数据库指令进行拦截. 拟态数据库中间件要实现数据库的切分和扩容, 集群管理, 事务分布式处理, 与此同时, 要实现对 SQL 语言进行特征处理, 识别和执行 SQL 指纹语句, 剔除攻击者注入的非法指令, 并对出现故障的数据库进行还原保护.

基于此现状, 本研究提出利用 Mycat 在拟态环境中作为数据库的中间件, 借助于动态冗余异构模型设计, 利用拟态安全的环境为依托, 简化拟态环境中数据的切分, 存取和同步, 调用对应 SQL 指纹指令对前端指纹 SQL 执行进行去指纹化.

2.1 中间件简介

Mycat 是用于解决传统关系型数据库大数据存储不足而设计的开源分布式数据存储中间件[16], 使用 NIO 重构的网络模块, 优化缓冲内核, 增强聚合等基本特性, 兼容 Oralce、PostgreSQL 等多种数据库, 实现跨语言, 跨平台, 跨数据库的通用中间件, 并提供和原生数据库一致的命令访问的支持, 可实现集群管理, 自动扩容, 智能优化的功能[5], Mycat 架构如图 2. Mycat 作为中间件, 其功能更好的对数据库实现拟态防御中动态异构冗余模型 (DHRA) 构造, 使数据的访问和处理更安全, 高效.

2.2 中间件在拟态中集群管理

由于在拟态环境中需要多个执行体, 为保证执行体执行的数据统一和多模裁决, 所有执行体共用一个

数据库集群, 这就要求数据库能够同时容纳并处理大量数据, 并且对指纹 SQL 语言有切分判决能力. Mycat 作为数据库的中间件能在拟态环境中搭建以 Mysql 为底层节点的分布式数据库系统[17], 系统通过 Mysql 的通信协议[18]与用户以及底层数据库通信, 实现负载均衡、指纹 SQL 语句重写、读写分离、多台数据库并行处理以及结果集合并等功能[5], 又因其是开源程序, Mycat 对整个集群能透明地访问和管理, 集群管理如图 3, 这在拟态环境中, 集群管理为数据的安全访问和处理提供了更可靠高效的保障.

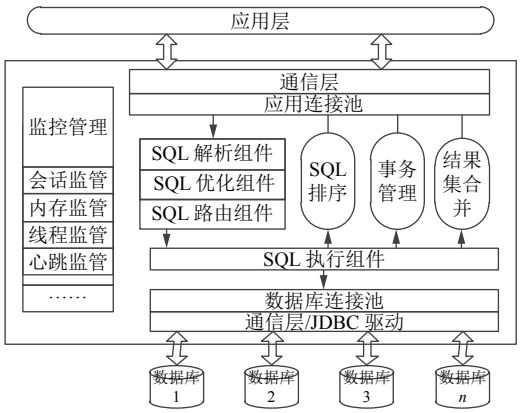

图 2 Mycat 架构

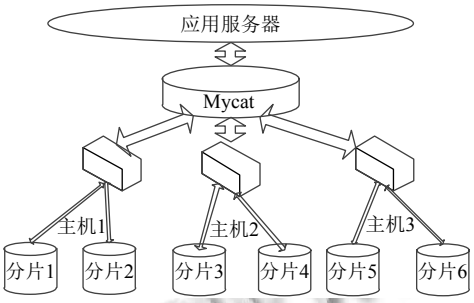

图 3 Mycat 集群管理

2.2.1 数据库中间件的切分

在拟态环境中, 多个异构执行体访问同一数据库, 这就对数据库存储和运算能力有一定的要求. 作为执行体和数据库的中间件, Mycat 对执行体发送的 SQL 语句进行拦截分析, 通过逻辑表中的取模分片算法、分片枚举、Hash 分片等特定算法, 将数据的存储和读取, 由一个数据库分散到多个数据库中, 减少了单个数据库存储和运算压力, 实现数据分布式存储[5], Mycat 分布式数据存储构架如图 4.

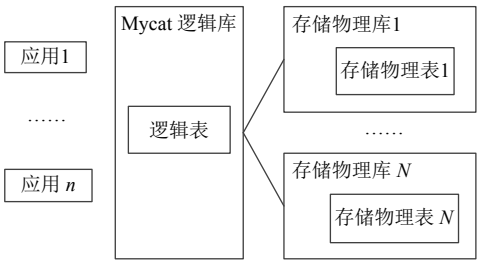

图 4 Mycat 分布式数据存储构架

2.2.2 数据库中间件高可用性

在拟态环境中, 多个执行体对数据库进行频繁的数据存储和读取, Mycat 作为执行体和数据库连接点, 其本身的高可用性涉及所连接数据库乃至整个拟态系统的高可用性. 数据库经过切分后, 在各个节点的数据库独立运行前提下, Mycat 使用主从复制高的可用配置, 将 dataHost 中的 writeNode 配置为主节点, readNode 配置为从节点, 在逻辑表中, 可配置多个 readNode 和 writeNode, 实现多写多读. 在 Mycat 正常运行时, 其内部对 dataHost 中的全部 readHost 和 WriteHost 节点定期发起心跳检测[5], 将全部的 DML SQL 发送给第一个 writeNode, 当第一个 writeNode 所在的节点出现宕机, 默认 3 次心跳检测失败后, Mycat 将自动切换到下一个可用的 writeNode, 并执行 DML SQL 语句. 由于 Mycat 是无状态中间件, 在拟态环境下, 用 HAProxy 等负载均衡软件部署为集群模式[1], Mycat 高可用架构如图 5.

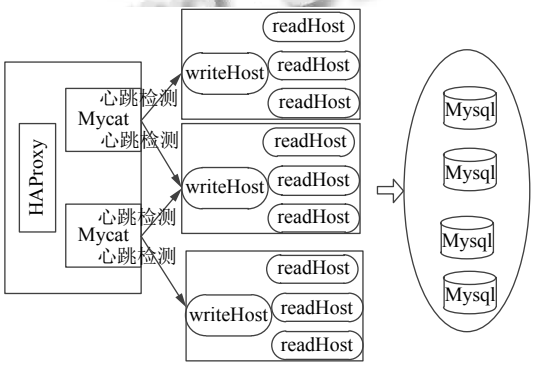

图 5 Mycat 高可用架构

2.2.3 数据库中间件分布式处理

事务处理由一组操作构成, 其具有原子性 (atomicity)、隔离性 (isolation)、持久性 (durability) 和一致性 (consistency)4 个特性[5], 在拟态环境中, 我们希望通过中间件的分布式处理[17](Distributed Transaction Processing, DTP) 将所有 SQL 语句正确执行. 分布式事

务处理 (DTP) 是在一个或多个数据库完成 SQL 语句执行过程的集合, 其关键是知道事务在数据库中任何地方所做的动作, 通过事务准备, 提交, 反馈, 产生统一的结果, 如果某个步骤发生错误, 就需要回滚到上一步已经完成的操作. X/Open 定义了分布式事务处理模型, 其由应用程序 (ApplicationProgram, AP)、资源管理器 (ResourceManagement, RM)、通信资源管理器 (Communication Resource Management, CRM)、事务管理器 (Transaction Management, TM) 四部分组成[5]. AP 可以和 TM 以及 RM 通信, TM 和 RM 互相之间可以通信, TM 和 RM 通过 XA 接口进行双向通信, TM 通知 RM 提交事务或者回滚到上一事务正确执行完的点, RM 把提交结果通知给 TM.

在拟态环境中, 在准备阶段, Mycat 中事务管理器通知节点数据库准备分支事务, 节点数据库准备结果; 在提交阶段, 事务管理器通知节点数据库提交分支事务, 节点数据库将结果提交给 Mycat,正常提交执行过程如图 6. 当在第一阶段出现某一个数据读取和改写失败, 第二阶段就回滚到第一阶段已经预提交成功的数据, 提交失败执行过程如图 7.

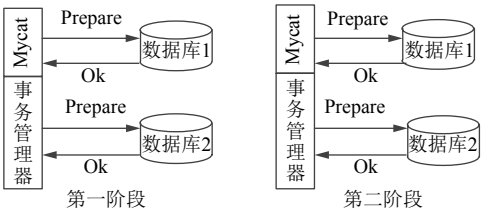

图 6 提交成功

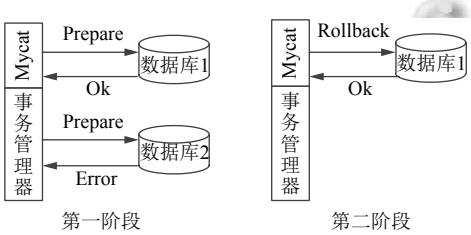

图 7 提交失败

2.2.4 数据库中间件对指纹 SQL 处理

现在网络数据窃取成功, 主要是攻击者了解并熟悉被攻击系统的语言, 进而注入代码攻击成功, 在拟态环境中, SQL 脚本采用基于指纹化的数据库指令异构[2], 通过对正常的执行语句加入指纹, 让攻击者不了解指纹化的执行语言, 使其注入的指令是无法正确执行. 当指纹 SQL 语句通过 Mycat 时, 数据库首先判断请求的地址, 若请求来自网站端, 将请求指令加到执行体对性的队列中, 然后进行指纹 SQL 语言过滤, 指纹 SQL 处理结构如图 8, 利用多模表决机制, 将 SQL 语句进行一致性表决, 表决一致, 则执行正常 SQL 语句, 表决不一致, 则进行异常处理.

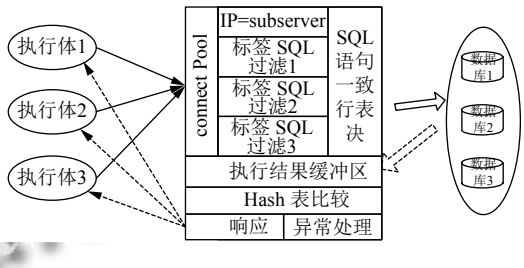

图 8 指纹 SQL 处理结构图

数据库返回的结果保存到缓冲区中, 相同的数据会被映射到 Hash 表中的相同位置, 所有结果返回完整后, 对数据库返回内容的完整性进行 Hash 比较, 比较通过后, 数据返回给执行体, 若比较不一致, 把出错信息发送给拟态系统的反馈调度服务器进行决策调度.

Hash 表中数据信息进行比较伪代码如下所示: 数据库的返回结果 R_i 包括多个数据包, 形式为: $R_i = \{P_1, P_2, \cdots, P_n\}$

For i in 1..n:

P_i = get_resp()

Put_to_push()//将数据库返回的结果放入 Hash 表中

If get(P_i) > n/2+1//相同结果超过一半时

Send(P_i) //返回结果

Remove(P_i)//删除已经对比完成的数据

2.2.5 数据库多模表决机制

在拟态环境中, 用动态冗余架构特性改变传统防御环境中的相似性, 确定性和静态性; 利用矢量空间多模裁决机制, 如图 9, 形成非协同条件下, 多元动态目标协同一致攻击难度以实现“面防御”功能. 在数据库环境中, 目标对象外部 SQL 服务请求依据策略分发给各个数据库, 数据库的输出矢量经过 Hash 表比较进行裁决输出.

多模表决器按照规则对 n 个功能相同但相互独立的数据库的输出进行表决操作, 保证系统正确输出, 本文采用 n 取 k 表决模型, 当 k 个数据库正常读取时, 便认为数据库正常运行, 当 k 个及以上数据库返回的数据异常时, 才会输出错误非正常数据, 在异构冗余的拟

态环境中, 不同的数据库运行环境不同, 攻击注入方式不同, 无论从时间和技术上, 对攻击者都是极大的消耗, 由此数据库的安全性和可靠性得以提高.

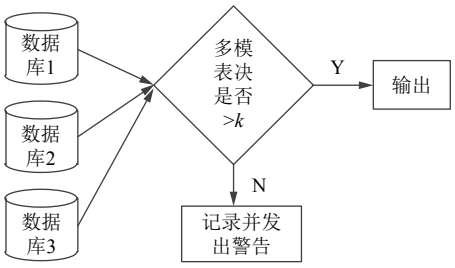

图 9 拟态表决器模型

3 实验与分析

为了验证 Mycat 在拟态环境中能够高效保障数据的读取和安全性, 我们在开源的企业办公系统然之协同中, 使用三模异构冗余模型进行实验测试, 然之协同的优势如下:

(1) 功能齐全, 能满足日常的办公需求, 使用范围广, 人数无限制.

(2) 私有化部署, 安全性高.

(3) 开源产品, 能修改代码满足个性化需求.

硬件配置如表 2.

表 2

类别	描述
操作系统	Windows 7 专业版 64 位 SP1
处理器	英特尔 Core i7-6700 @ 3.40 GHz 四核
内存	32 GB(三星 DDR4 2400 MHz)
主硬盘	希捷 ST1000DM003-1SB102
转速	7200 转/分

3.1 环境配置

在拟态环境中, 3 个异构执行体运行的系统分别为 windows7(IP:192.168.126.133), Ubuntu (IP: 192.168.126.134) 和 centos (IP: 192.168.126.135), Mycat 安装在 IP 地址为 192.168.126.141, 并建立“ranzhi”数据库.

(1) Schema.xml 配置

Schema.xml 作为 Mycat 最重要的配置文件之一, 管理着逻辑库, 分片规则, 节点主机等信息, 关键代码如下:

```
<table name="crash_trade" primaryKey="ID" type="global" dataNode="dn1,dn2" />
<dataNode name="dn1" dataHost="localhost1" database="db1" />
<dataHost name="localhost1" maxCon="1000" minCon="10" balance="0" writeType="0" dbType="Mysql" dbDriver="native" switchType="1" slaveThreshold="100">
<writeHost host="hostM1" url="192.168.126.141:3306" user="root"
password="******">
<readHost host="hostS2" url="192.168.126.141:3306" user="root" password="*******" />
<readHost host="hostS2" url="192.168.126.142:3306" user="root" password="*******" />
</writeHost>
```

(2) server.xml 配置

server.xml 配置 Mycat 的系统服务信息,服务端口为 8066. 关键代码如下:

```
<user name="root">
<property name="password">*****</property>
<property name="schemas">ranzhi</property>
```

在各自的系统中, 安装然之协同, 其所连立的数据库设置为 Mycat 所在的物理地址和 Mycat 的端口, 如图 10.

图 10 执行体 Mycat 连接

3.2 实验结果分析

(1) 指纹 SQL 识别

通过 Mycat, 指纹 SQL 能够在数据库中正常执行, 如图 11, 无指纹的 SQL 语言, 不能运行, 如图 12.

```
| name  | char(20) | YES  |     | NULL    |       |
| sec   | char(20) | YES  |     | NULL    |       |
| age   | int(11)  | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
3 rows in set (0.01 sec)

mysql> abcINSERT INTO users(name, sec, age)values('张
Query OK, 1 row affected (0.01 sec)
```

图 11 指纹 SQL 执行结果

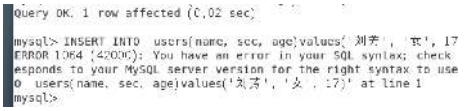

图 12 无指纹 SQL 执行结果

(2) 数据库注入测试

在一般的数据库中, 进行数据库注入攻击测试, 数据库返回数据, 如图 13, 图 14 所示.

当使用本文中拟态环境中的数据库, 进行相同的数据库注入攻击, 未能返回数据, 攻击失败, 如图 15 所示.

图 13 数据库注入过程

图 14 数据库注入结果

图 15 数据库注入失败

数据库安全测试如表 3 所示.

经上述实验, 在拟态环境中, 数据库成功防御数据库注入攻击, 保障了数据库安全.

表 3 数据库安全测试结果

防御内容	防御类别	防御预期结果	防御机理			
			异构	冗余	动态	清洗
数据库安全测试	指纹 SQL 指令执行	信息碎片随机化传输模块中在集群数据库中进行数据节点随机化传输读取	√	√	√	
	数据库高可用性	随机化传输模块中数据库单一节点出现故障, 不影响节点正常进行存取,		√	√	
	数据库分布式处理	多个数据库能够保持数据同步	√		√	√
	SQL 注入测试	执行体间 SQL 指纹不一致, 攻击无法在全部执行体中成功执行	√	√	√	√
	SQL 指令指纹破坏测试	SQL 指令动态随机切换, 被破坏执行体被清洗还原	√		√	√

4 结论与展望

通过在拟态环境中使用 Mycat 作为数据库访问的中间件, 实现了数据库的集群管理, 高可用性, 以及对指纹 SQL 语言的处理, 极大的提高拟态防御中对数据安全的保障和数据存储, 读取的高效性. 在未来的信息时代, 网络空间的数据存取和保护越来越得到重视, 随着社会科技的发展, 会有更多的数据库中间件出现, 其功能和架构模式也会不断的更新和成熟, 数据库中间件在拟态环境中会发挥更大的作用.

参考文献

1 邬江兴. 网络空间拟态防御导论-上册. 北京: 科学出版社, 2017.

2 邬江兴. 网络空间拟态防御导论-下册. 北京: 科学出版社, 2017.

3 莫建平, 应凌云, 苏璞睿, 等. 恶意代码动态分析中的反虚拟化问题研究. 计算机系统应用, 2018, 27(12): 1–8. [doi: 10.15888/j.cnki.csa.006683]

4 蔡桂林, 王宝生, 王天佐, 等. 移动目标防御技术研究进展. 计算机研究与发展, 2016, 53(5): 968–987. [doi: 10.7544/issn1000-1239.2016.20150225]

5 Mycat 权威指南. http://download.csdn.net/detail/ming_311/9067463, [2015-08-31].

6 邬江兴. 拟态计算与拟态安全防御的原意和愿景. 电信科学, 2014, 30(7): 2–7. [doi: 10.3969/j.issn.1000-0801.2014.07.001]

7 邬江兴. 网络空间拟态安全防御. 保密科学技术, 2014, (10): 4–9, 1.

8 仝青, 张铮, 邬江兴. 基于软硬件多样性的主动防御技术. 信息安全学报, 2017, 2(1): 1–12.

9 Zhang YJ, Pang JM. A new compile-time obfuscation

scheme for software protection. Proceedings of 2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. Chengdu, China. 2016. 1–5.

10 张峰, 李亚伟. SFMEA 安全性分析技术在软件安全性测试中的应用. 计算机系统应用, 2019, 28(2): 158–163. [doi: 10.15888/j.cnki.csa.006779]

11 仝青, 张铮, 张为华, 等. 拟态防御 Web 服务器设计与实现. 软件学报, 2017, 28(4): 883–897. [doi: 10.13328/j.cnki.jos.005192]

12 张铮, 马博林, 邬江兴. Web 服务器拟态防御原理验证系统测试与分析. 信息安全学报, 2017, 2(1): 13–28.

13 马海龙, 伊鹏, 江逸茗, 等. 基于动态异构冗余机制的路由器拟态防御体系结构. 信息安全学报, 2017, 2(1): 29–42.

14 马海龙, 江逸茗, 白冰, 等. 路由器拟态防御能力测试与分析. 信息安全学报, 2017, 2(1): 43–53.

15 王禛鹏, 扈红超, 程国振. 一种基于拟态安全防御的 DNS 框架设计. 电子学报, 2017, 45(11): 2705–2714. [doi: 10.3969/j.issn.0372-2112.2017.11.018]

16 项凯. 面向海量高并发数据库中间件的研究与应用[硕士学位论文]. 上海: 上海交通大学, 2015.

17 赵艳, 李钧. 异构数据源分布式事务处理研究. 计算机工程, 2009, 35(4): 69–71. [doi: 10.3969/j.issn.1007-130X.2009.04.021]

18 安延文. 数据库审计系统中 MySQL 协议的研究与解析[硕士学位论文]. 北京: 华北电力大学 (北京), 2016.