一种新的SQL注入防护方法的研究与实现

石聪聪 张 涛 余 勇 林为民
(中国电力科学研究院 南京 211106)

摘 要 当前Web应用安全问题日益严峻,而SQL注入是针对Web应用最为普遍的攻击手段之一。文中提出了一种新的SQL注入防护方法。该方法通过将静态模式匹配与动态特征过滤配合使用,避免单一方法存在的不足,从而达到良好的效果。该方法通过在安全环境下自动学习所有合法SQL语句,构建知识库;然后在实时工作环境下,利用模式匹配算法将SQL语句与知识库进行匹配,匹配成功则判定为合法SQL语句。对于匹配失败的SQL语句并不立即判定为非法,而是采用基于风险值的动态特征过滤算法进行深度特征检查,识别真正的非法SQL语句。基于本方法,设计并实现了一个原型系统。测试结果表明,该原型系统具有较好的性能优势,并能够很好地解决一般防注入方法带来的准确率与误报率之间的矛盾。

关键词 自学习,SQL语法树,模式匹配,特征过滤
中图法分类号 TP393.08 **文献标识码** A

New Approach for SQL-injecton Detection

SHI Cong-cong ZHANG Tao YU Yong LIN Wei-min
(China Electic Power Research Institute, Nanjing 211106, China)

Abstract Web application security is a serious isssue of information security, and SQL-injection is one of the most common attacks. This paper proposed an approach to counter SQL Injection which combines static mode-matching and dynamic feature-filtering. It learned automatically the structure feature of all legal SQL statements to construct knowledge library in safe environments, and then matched every SQL statement with knowledge library in real environments. If succeeded, this SQL statement was considered to be legitimate. If failed, it was not determined to be illegal immediately. We would take depth-feature check based on Value-at-Risk, and identitify the true illegal SQL statements. Experimental results prove that this proposed approach has good performance and perfect protection for SQL Injection.

Keywords Self-learning, SQL Syntax-tree, Pattern-marthing, Feature-filtering

1 引言

大部分Web应用系统需要与用户进行交互,从用户那里接受数据并进行处理,如果攻击者故意输入含有恶意构造的数据,则在系统执行处理时,这些数据中包含的恶意代码就会被系统或者其它客户端执行,从而对服务器或客户端造成损害,这就是所谓的SQL注入。由于SQL注入从正常的WWW端口访问,并且看起来和正常的Web页面访问没有什么区别,因此目前常见的防火墙检测不到SQL注入。不久之前,OWASP(Open Web Application Security Project 开源Web应用安全组织)公布了2010年10大Web攻击,而注入攻击被列为十大攻击之首。

SQL注入的产生经常是由于开发人员没有对用户输入部分做充分的安全检查,因此可以通过提高应用程序的代码质量来抵御SQL注入攻击。文献[1]列出了一些安全编程的指导原则。此外还可以对Web应用程序进行代码检测,提前发现可能存在的漏洞。文献[2]提出了一种黑盒测试方法。文献[3]提出一种针对JDBC程序的静态代码检测方法。但仅仅通过提高代码的质量或者减少代码漏洞并不能解决所有问题,黑客依然可以寻找新的攻击字符串来规避这些程序检测点。其他SQL防护手段还包括注入特征检查以及模式匹配等。SQL注入特征过滤是在网络服务前端增加一个应用级的防火墙,在接收到用户输入信息时进行集中检查,过滤掉恶意输入,例如Snort以及PHP中实现的"MagicQuotes"。相对而言,特征过滤实现较为简单,然而过滤程序通常采用特征字符串匹配算法,往往仅凭预先定义好的一系列正则表达式进行过滤。一方面如果进行大量的特征检查,性能下降较快,另一方面准确率以及误报率难以平衡。与特征过滤不同,模式匹配的方法并不是检查注入特征,而是将SQL语句与预定义的合法SQL语句进匹配,通过接收合法SQL语句来达到过滤恶意SQL语句的目的,比如文献[6,7]。在文献[6]中,作者通过静态分析应用程序代码,生成合法SQL语句模板,然后再将所有SQL语句与预定的模板进行匹配,识别其是否为应用程序自身产生的合法SQL语句。在文献[7]中,

石聪聪(1982—),男,硕士,工程师,主要研究方向为电力系统安全研究、软件安全,E-mail:shicongcong@gmail.com;张 涛(1976—),男,硕士,高级工程师,主要研究方向为电力系统安全防护研究、电力系统自动化;余 勇(1969—),男,博士,高级工程师,主要研究方向为电力系统安全防护;林为民(1965—),男,硕士,教授级高级工程师,主要研究方向为电力信息化及电力信息安全等。

作者根据开发人员制定预定的模式进行匹配。模式匹配的方法能够很好地解决 SQL 注入问题，而且效率较高。然而它的成功率依赖于模式建立的准确率，同时它的覆盖率也很难达到 100%，而且一定程度上可能需要修改应用程序。

本文提出了一种新的 SQL 注入防护方法。一方面无需人工干预能够自动学习所有合法 SQL 语句，构建合法 SQL 语句知识库；另一方面，将模式匹配方法与特征过滤方法混合使用，共同完成防护 SQL 注入的目的。

通过在学习阶段（安全环境下）对应用程序产生的所有 SQL 语句进行语法解析得到 SQL 语法树，再基于 SQL 语法树进行语法结构特征提取，构建知识库。然后在实时工作阶段（普通环境下）将所有 SQL 语句与知识库进行模式匹配，匹配成功即判定为合法 SQL 语句。由于知识库的覆盖率以及模式匹配的准确率难以达到 100%，对于匹配失败的 SQL 语句并不立即判定为非法 SQL 语句，而是进行深度 SQL 注入特征过滤，检查不通过才被判定为非法 SQL 语句。将两种方法配合使用，一方面能够很好缓解模式匹配可能带来的覆盖率以及匹配准确率不足的问题；另一方面可以解决特征过滤方法可能带来的高识别率和高误判率之间的矛盾；其次由于模式匹配效率高于特征过滤，相比单独使用特征过滤方法，本方法在性能上有较大提升。

本文所述的新的 SQL 注入防护方法主要关键点如下：

1）将模式匹配方法与特征过滤方法混合使用，避免了两种方法各自存在的缺陷；

2）基于 SQL 语法树进行模式特征提取更加准确，同时效率更高；

3）在模式匹配过程中，采用基于 SQL 语法树的匹配以及高效的 HASH 算法，提高匹配的准确率以及效率；

4）在特征过滤过程，采用基于注入风险值的计算方法，更加灵活。

本文第 2、第 3 节介绍该 SQL 注入防护的基本原理以及处理流程；第 4 节介绍具体原型系统的实现；第 5 节针对本 SQL 注入方法进行建模分析；第 6 节介绍在某些具体业务系统中对原型系统功能以及性能的测试情况；最后介绍本方法的一些不足，以及未来的改进措施。

2 原理介绍

对于特定应用系统，具有同构特征的 SQL 的语句数量是可控的，而且任何包含注入特征的 SQL 语句都将改变原有语句的语法结构，比如增加或者删除某个节点、子树结构变化等。因此如果某个 SQL 语句的语法结构特征与预期存在不一致的情况，则说明该 SQL 语句可能为注入 SQL 语句。本方法主要是基于上述前提，将 SQL 注入防护分为两个阶段：学习阶段和过滤阶段。学习阶段工作在安全环境下，学习上层应用程序产生的所有合法语句，提取语法结构特征，构建合法 SQL 语句的特征知识库。过滤阶段工作在现实的环境下，将实时产生 SQL 语句与学习阶段下建立的知识库进行模式匹配，匹配成功则为合法的 SQL 语句。匹配不成功的 SQL 语句，进行过滤特征检查，确定其是否包含注入特征的字符串，检查通过后才放过。

这种新的 SQL 注入防护方法能够很好地克服传统 SQL 注入防护方法可能导致存在大量的漏报以及误报情况，极大提高了 SQL 注入防护的准确率，同时通过采用高效模式匹配算法避免进行大量的深度内容过滤，有效改善了 SQL 过滤整体效率。该方法涉及主要原理如下。

2.1 语法树与特征提取

为了获取 SQL 语句的特征，需要对 SQL 语句进行解析，包括词法、语法分析，得到 SQL 语法树。然后遍历 SQL 语法树，进行语法裁剪，使之变为规范语法树，获取 SQL 语句的主干结构。裁剪过程负责将 SQL 语句中用户输入部分，比如数字以及字符串等，用通配符进行替换，同时将一些无用的节点删除。然后基于 SQL 规范语法树进行特征提取，提取特征包括：子树的个数 t_1、树的高度 t_2、节点个数 t_3、第一棵子树节点的个数 t_4、首节点的类型 t_5 等等，从而获取该 SQL 语句语法结构特征，即

$$\varphi(i)=\{t_1,t_2,t_3,\cdots,t_n\} \tag{1}$$

式中，n 为提取的特征个数，$t_1,t_2,t_3,\cdots,t_n$ 分别表示第 i 条 SQL 语句的 n 个特征值，$\varphi(i)$ 为第 i 条 SQL 语句的语法树特征向量。

如 select a from b where username='liming'and passwd=123456，通过语法解析、语法树的裁剪、用户输入替换（字符串输入用＃，数字用 MYM），得到图 1 所示的规范语法树。

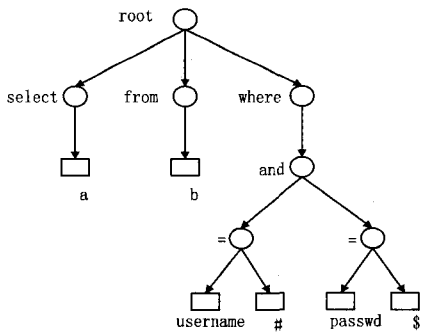

图 1 SQL 规范语法树

根据规范语法树提取模式特征，特征向量：$\varphi(i)=\{3,4,13,2,1,\cdots\}$（$i$ 表示学习的第 i 个语句）。

2.2 知识库与模式匹配

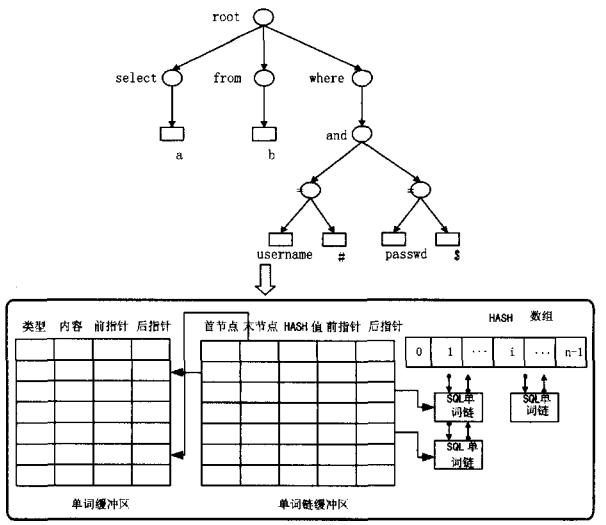

图 2 知识库组织

SQL 知识库包含学习阶段所有学习到的具有不同主干语法结构特征的 SQL 语句。同时知识库作为过滤阶段进行实时 SQL 过滤的基础。从实现的角度上来说，SQL 知识库是存储具有不同特征合法 SQL 语句的内存池。为了提高匹配的效率，知识库的内部存储采用多级别的索引进行组织，如图 2 所示。模式匹配主要利用 SQL 语句的语法结构特征即 $\varphi(i)$。利用 HASH 算法，对 $\varphi(i)$ 进行 HASH 计算，获取特征值 sql_value(i)，即 sql_value(i) = Hash($\varphi(i)$)。利用 sql_value(i) 做索引查找现有知识库。通过特征值的定位，找到具有相同特征值的 SQL 语句之后，再进行各个节点的逐个匹配，最终确定是否与知识库匹配。在学习过程中，将与知识库不匹配的 SQL 语句加入知识库。

2.3 风险值与特征过滤

非法 SQL 语句一般存在某些注入特征，比如包含空字符串、'or '，'union'，注释符、恒等等。然而并不是包含其中任何一个字符串的 SQL 语句都是非法 SQL 语句，因此特征过滤不可避免存在误报率。为了提高特征过滤的准确率并降低误报率，本文采用一种动态风险值计算方法，给每个注入特征字符串赋予一个风险值，同时还可以根据应用程序以及后台数据库类型，定义一些特定的注入特征，比如敏感表、系统文件、数据库管理命令等，并赋予风险值。然后定义 SQL 注入的阈值(Threshold)，通过深度 SQL 语句的特征检查后，只有 SQL 语句中所有注入特征的风险值累积大于预先定义的阈值，才判定为非法 SQL 语句。计算方式如下：

$$R=\sum_{i=1}^{n} r_i * k \tag{2}$$

式中，R 为对应 SQL 语句的累计风险值，n 为预先定义的最大注入特征规则的个数，r_i 为第 i 注入特征的风险值，k 为该注入字符串出现的次数。当 SQL 语句累计风险值 R 大于阈值 Threshod 时，该语句被判定为非法 SQL 语句。通过动态调整各种注入特征的风险值以及阈值，这种特征过滤方法更加灵活，准确性更好。

3 详细流程

3.1 学习阶段流程

学习阶段主要是在安全的环境下学习应用程序产生的所有合法 SQL 语句，构建知识库，详细流程如图 3 所示。

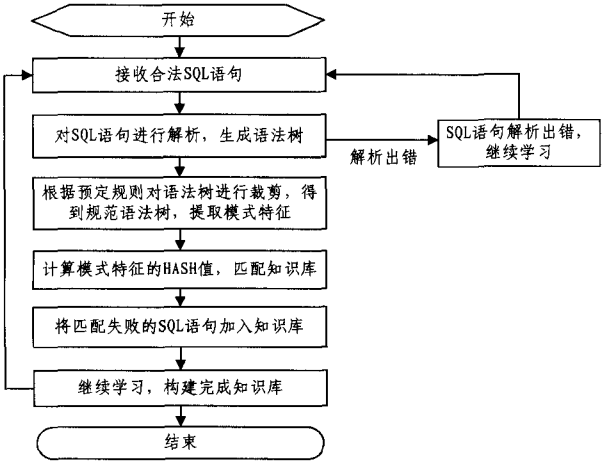

图 3 学习流程

3.2 过滤阶段流程

过滤阶段主要工作在现实环境中，实现对非法 SQL 语句的过滤，详细流程如图 4 所示。

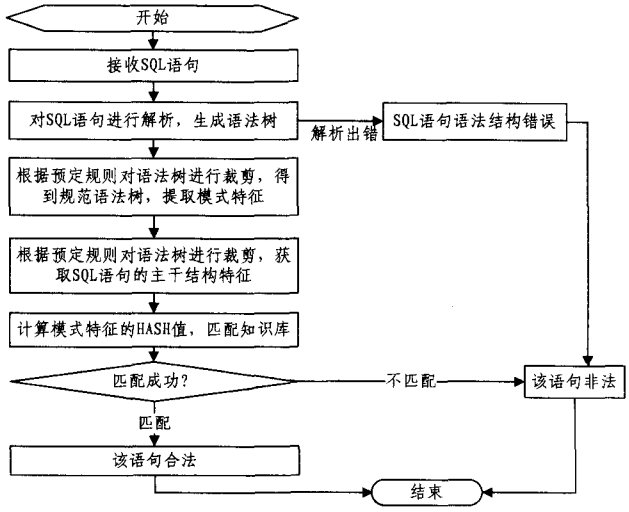

图 4 过滤流程

4 设计与实现

本节介绍这种新的 SQL 注入防护方法原型系统的设计与实现，模块图如图 5 所示。

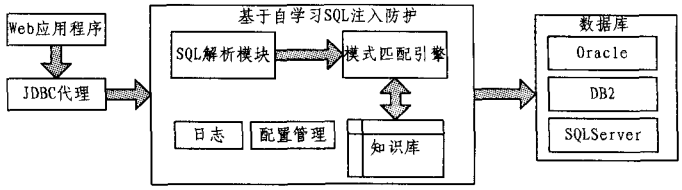

图 5 原型系统模块组成图

SQL 解析模块：负责解析 SQL 语句，包括词法解析、语法解析、语义解析。词法解析负责对存储一段连续缓冲区的原始 SQL 语句进行分解，得到一个个独立的单词，并将其组织成为单词链表形式。语法解析负责根据特定数据库的 SQL 语法对词法解析后的单词链进行语法结构分析，构造出 SQL 语法树。语义分析负责根据 SQL 语法树分析该 SQL 语句的语义特征。

模式匹配引擎：负责将解析后的 SQL 语法树按照预定义的策略进行裁剪，包括替换用户输入部分，删除无用的节点等，然后提取特征，初始化特征向量，再利用 HASH 算法计算特征值。以特征值为索引，与知识库中的 SQL 语法树进行匹配。如果匹配通过则成功退出，如果匹配失败，则说明知识库内不存在相同的 SQL 语法树。学习阶段，模式匹配引擎负责将新的 SQL 语法树加入知识库；过滤阶段，模式匹配引擎将该 SQL 语句提交给 SQL 过滤模块进行注入特征检查。

知识库：负责存储学习阶段获取的具有不同主干特征的 SQL 语法树。为提高存储与删除的效率，采用静态数组链表形式进行存储，内部存储采用多级别的索引进行组织。

SQL 特征过滤模块：根据预先定义的注入特征表，对 SQL 语句进行详细的特征检查，计算 SQL 语句的风险值。

将风险值与预先定义的阈值进行比较，判定其是否为非法SQL语句。

配置管理模块：负责接收本地管理员或者远程管理客户端指令，动态更新本地策略，以及实时切换系统工作模式（学习阶段、过滤阶段）。

日志模块：负责接收其他单元的报警信息，记录日志，同时将报警信息发送至远程管理客户端。

5 建模分析

为了验证该SQL注入防护方法，我们采用简单的建模方法。假设通过充分的学习，知识库对于合法SQL语句的覆盖率为 $f(0\%<f<100\%)$。基于风险值的特征过滤算法对于非法语句识别的准确率为 $a(0\%<a<100\%)$，对于合法语句产生的误报率为 $b(0\%<b<100\%)$。定义通过采用新的注入防护方法后准确率为 A，误报率为 B。

假设所有SQL语句为 T 条，其中 T_1 为合法SQL语句，T_2 为非法SQL语句。

对于合法SQL语句，经过知识库的匹配后，通过的合法SQL语句为 T_1*f，剩下的 $T-T_1*f$ 被判定为可疑SQL语句，其中 T_1-T_1*f 为合法SQL语句，另外 T_2 为非法SQL语句。经过特征过滤模块后，其中合法SQL语句检查通过的个数为：$(T_1-T_1*f)*(1-b)$；非法SQL语句检查不通过的个数为：T_2*a。

这样，整个模块的准确率及误报率分别为：

$$A=\frac{T_2*a}{T_2}=a \tag{3}$$

$$B=\frac{(T_1-T_1*f)*b}{T_1}=(1-f)b \tag{4}$$

由此可见，将模式匹配方法与特征过滤方法混合使用并不能直接提升整体模块对于非法SQL语句检查的准确率，不过大幅降低了合法SQL语句的误报率，而且覆盖率 f 越高，误报率降幅更大。为了提高整个模块注入防护能力，可以通过增加注入特征的检查力度来提高准确率，使其依然能保持较低的误报率，很好地避免准确率与误报率之间的矛盾。

6 评估

在对原型系统的评估实验中，为提高测试准确性，我们选取两个不同的业务系统进行测试，搭建仿真业务环境，采用与实际在线运行性能相同的服务器、网络设备并进行相同配置，通过在Web服务器和数据库服务器之间设置代理的形式建立防御系统，考察原型系统的整体性能以及功能。

6.1 性能

由于该SQL注入防护系统需要对SQL语句作为额外处理，因此一定程度上会增加用户访问数据库的时间。根据具体工作流程，我们将对SQL语句的解析以及与知识库匹配作为第一阶段，将对SQL语句进行深度特征过滤作为第二阶段。测试对象选取长度分别为128、256、512、1024、2048字节的SQL查询语句，循环100000次，统计处理数据时间。测试结果如图6所示。

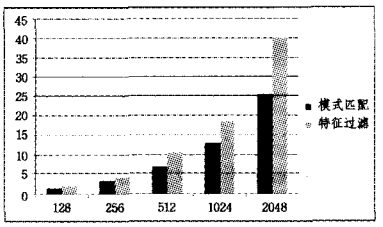

图6 性能分析图

根据测试结果，系统对SQL语句处理的时间与SQL语句长度直接相关，同时模式匹配相比深度特征过滤，处理时间有较大优势。经过验证，对于常规数据库查询（查询10行数据），循环100000次，统计时间约为130s。因此无论模式匹配还是特征过滤所消耗的时间，相比常规数据库访问时间，比重较小。而且由于大部分SQL语句只通过模式匹配而不经过特征过滤，因此整体处理性能较高。

6.2 功能

功能测试主要是测试原型系统对非法SQL语句过滤的功能。我们将原型系统部署在业务系统运行的真实环境下，通过一段时间学习，业务系统一知识库学习到的合法SQL语句为1265条，业务系统二学习到的SQL语句为469条。

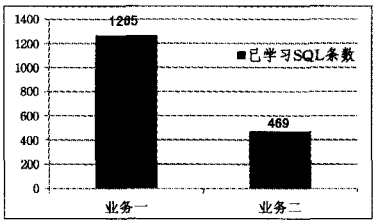

图7 已学习SQL语句

然后构造不同方式的SQL注入语句进行攻击，攻击语句为200条。同时，混合1500合法SQL语句。测试结果表明，业务系统一：识别攻击语句201条，合法语句1499条；业务系统二：识别攻击语句208条，合法语句1492条。业务系统一对于非法SQL语句的识别准确率为100%，对于合法SQL语句的误报率为0.07%。业务系统二对于非法SQL语句的识别准确率为100%，对于合法SQL语句的误报率为0.53%。

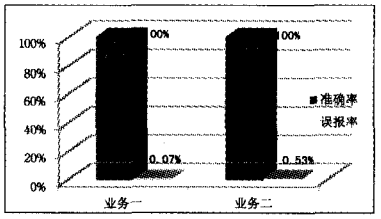

图8 准确率与误报率

经验证，利用这种方法在保证对所有非法语句全部识别的基础上，误报率也能保证处于可控范围。

结束语 本文提出了一种新的SQL注入方法。本方法将静态模式匹配方法与动态特征过滤方法混合使用，可以很好地解决采用单一SQL注入方法存在的缺陷，并且其易于实现，适合各种典型部署的Web应用，同时具有较好的可扩展性。然而，随着业务系统的复杂度提高，同时不同程序开发人

员的编程习惯不同，某些应用程序可能会随着用户输入不同产生可变的 SQL 语句，这给本方法带来很大挑战，一方面导致知识库无法完全学习到所有合法 SQL 语句，另一方面由于可变语句的存在，应用程序产生的 SQL 语句的数量不可控，因此大量可变语句的出现将导致知识库巨大。虽然可以通过配合特征过滤方法来解决大部分的可变 SQL 语句，但是依然无法从根本上解决。我们也注意到可变语句的产生在很大程度上有其特有的共性，比如相关的语句结构具有很大的相似性，变化的只是 SQL 语句的局部。因此，在未来的研究工作中，我们考虑采用基于 SQL 语法树的相似度进行模式匹配，实现对可变 SQL 语句的处理。

参 考 文 献

[1] Patel N, Mohammed F, Soni S. SQL Injection Attacks: Techniques and Protection Mechanisms[J]. International Journal on Computer Science and Engineering, 2011, 3(1)

[2] Das D, Sharma U, Bhattacharyya D K. An Approach to Detection of SQL Injection Attack Based on Dynamic Query Matching [J]. 2010 International Journal of Computer Applications, 2010, 1(25)

[3] Howard M, LeBlanc D. Writing Secure Code (Second edition) [M]. Microsoft Press, Redmond, Washington, 2003

[4] Huang Y, Huang S, Lin T, et al. Web Application Security Assessment by Fault Injection and Behavior Monitoring[C]// Proceedings of the 11th International World Wide Web Conference (WWW 03). May 2003

[5] Gould C, Su Z, Devanbu P. JDBC Checker: A Static Analysis Tool for SQL/JDBC Applications[C]// Proceedings of the 26th International Conference on Software Engineering (ICSE 04). Formal Demos, 2004: 697-698

[6] Maor O, Shulman A. Sql injection signatures evasion[EB/OL]. http://www.imperva.com/application_defense_center/white_papers/sql_inje%ction_signatures_evasion.html, White paper, 2004-04

[7] Ashcraft K, Engler D. Using programmer-written compiler extensions to catch security holes[C]// Proceedings of the IEEE Symposium on Security and Privacy. May 2002: 143-159

[8] Buehrer G, Weide B, Sivilotti P. Using Parse Tree Validation to Prevent SQL Injection Attacks[C]// Proceedings of the 5th International Workshop on Software Engineering and Middleware. Lisbon, Portugal, September 2005

[9] Su Z, Wassermann G. The Essence of Command Injection Attacks in Web Applications[C]// Proceedings of the 33rd Symposium on Principles of Programming Languages. Charleston, South Carolina, January 2006

[10] 周敬利，等. 一种新的反 SQL 注入策略的研究与实现[J]. 计算机科学，2006，133(111)

[11] 余静，等. 基于 SQL 注入的渗透性测试技术研究[J]. 计算机工程与设计，2007，28(15)

[12] 陈小兵，等. SQL 注入攻击及其防范检测技术研究. 计算机工程与设计，2007，43(11)

(上接第 56 页)

动窗口技术的标量乘法算法进行了改进。由表 4 和图 1 可知，所获得的改进算法的效率有明显提高，并降低了空间消耗，能有效提升 ECC 的实现效率。

参 考 文 献

[1] Noroozi E, Kadivar J, Shafiee S H. Energy analysis for wireless sensor networks[C]// IEEE International Conference on Mechanical and Electronics Engineering (ICMEE 2010). IEEE, 2010(2): 382-386

[2] Longa P, Gebotys C. Setting speed records with the (fractional) multibase non-adjacent form method for efficient elliptic curve scalar multiplication[C]// Department of Electrical and Computer Engineering University of Waterloo, Canada, 2009

[3] Shah P G, Huang X, Sharma D. Algorithm Based on One's Complement for Fast Scalar Multiplication in ECC for Wireless Sensor Network[C]// IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA). 2010: 571-576

[4] Okeya K, Schmidt-Samoa K, Spahn C, et al. Signed Binary Representations Revisited[C]// Advances in Cryptology- CRYPTO 2004(LNCS3152). New York: Springer-Verlag, 2004: 123-139

[5] Huang X, Shah P G, Sharma D. Minimizing Hamming Weight Based on 1's Complement of Binary Numbers Over $GF(2^m)$[C]// The 12th International Conference on Advanced Communication Technology (ICACT). Gangwondo, Korea, Feb. 2010, 2: 1226-1230

[6] Mahdavi R, Saiadian A. Efficient Scalar Multiplications for Elliptic Curve Cryptosystems Using Mixed Coordinates Strategy and Direct Computations[C]// Cryptology and Network Security (LNCS6467). 2010: 184-198

[7] Hankerson D, Menezes A, Vanstone S. Guide to elliptic curve cryptography[M]. Springer-Verlag Professional Computing Series, 2004

[8] Sakai Y, Sakurai K. Efficient Scalar Multiplications on Elliptic Curves with Direct Computations of Several Doublings[J]. IEICE Transaction on Fundamentals, 2001, E84(A): 120-129

[9] Okeya K, Schmidt-Samoa K, Spahn C, et al. Signed binary representations revisited[C]// Andances in Cryptology- CRYPTO' 04, LNCS 3152. 2004: 123-139