

doi:10.11830/ISSN.1000-5013.201605021



ASP.NET 的 SQL 注入攻击及防御

张学义, 钟志宏

(黔南民族师范学院 计算机与信息学院, 贵州 都匀 558000)

摘要: 研究 ASP.NET 网站的 SQL 注入攻击的成因、攻击距离,通过实例说明 ASP.NET 程序中的 SQL 注入攻击方式.研究表明:要防范 SQL 注入,最重要的就是要做好危险字符验证的程序、异常编码的处理,屏蔽错误信息,并从代码安全上进行阻止,如设置可靠的 SQL 参数、给数据进行加密处理、查验用户信息、设置存储路径和设置安全措施等,从而加强 ASP.NET 网站的性能和安全,避免 SQL 注入攻击的实现.

关键词: ASP.NET; SQL 注入攻击; 防御技术; 异常编码; 代码安全

中图分类号: TP 309.1 **文献标志码:** A **文章编号:** 1000-5013(2016)05-0633-04

SQL Injection Attacks and Defense Based on ASP.NET

ZHANG Xueyi, ZHONG Zhihong

(School of Computer and Information, Qiannan Normal University for Nationalities, Duyun 558000, China)

Abstract: This paper research the causes and attack distance of SQL injection attacks, and illustrates through examples the SQL injection attack methods in on the ASP.NET program. The research shows that hazard character verification and abnormalites codes handling are the most important to prevent SQL injection. By using keywords shield, SQL parameter optimization, data encryption, user information check, storage path setting, and implement safety measures can improve the security of ASP.NET website and effectively prevent SQL injection attacks.

Keywords: ASP.NET; SQL injection attack; defense technology; extraordinary code; code security

互联网技术的扩拓总是伴随着网络安全问题,以 ASP.NET 语言为基点 Web 应用程序的数量在不断提升,随之而来的网络安全破坏(攻击力)也更大了.很多网站程序员在编写 ASP.NET 网站程序时,并没有对用户输入的数据进行检验性测试,或者网站数据库自身存在一些比较大的安全漏洞,以及网站的防火墙并没有能够识别外部攻击. SQL 注入是目前 ASP 系统中存在最多的一种安全漏洞. SQL 注入攻击是指攻击者通过修改原有数据库的数据和参数,使系统识别不出非法入侵,这种找漏洞进入的入侵常被称为漏洞攻击,它会对程序数据库语句执行造成打击^[1]. 本文对 ASP.NET 网站的 SQL 注入攻击进行攻击,并提出相应的防范措施.

1 SQL 注入的原因及攻击远离

程序中,SQL 注入很大的原因在于编码和代码存在问题,其主要原因是在编写代码的过程中,对代码完善性的考虑欠周、简洁性及安全性的结果,从而导致脚本程序被入侵者破坏^[2]. 从入侵者方面出发,SQL 注入可以加强防御,所构成的防火墙够坚固,才可以避开防火墙的阻挡,这个操作十分容易,且能

够实现访问修改数据库的功能.

SQL 注入攻击产生的一个重要的环境,是输入键值打造的动态 SQL 语句经过 Web 应用程序进行数据库操作. 如果编写的代码要使用存储的这一过程,那么,这些存储过程将作为用户输入的输入内容传递,SQL 注入在这样的情况下也容易出现. 举一个常见的 SQL 注入攻击例子进行说明,所描述内容采用了用户输入字符串和拼接查询字符串进行 SQL 的查询^[3-4]:

```
var User Name;  
User Name=Raquest. form(“User Name”)  
var sql=“select * from Users where User Name=“+User Name+””;
```

使用者根据暗示内容填写一个用户名称,如填写“xiaoming”这几个字节时,输出的结果就是:SE-
LECT * FROM Users WHERE UserName=‘xiaoming’. 这时,脚本会执行下面查询:

```
SELECT * FROM Users WHERE User Name=‘xiaoming’;droptable Users--
```

其中:分号(;)为前面完成与后面查询的开端;“--”为对前面的内容进行解释,基本上理解前面的内容,字符后面的内容可以不计. 代码改正后,如果没有出现问题,服务器就会直接操作,而 SQL Server 操作本语句就会进行查看,并对 Users 进行清除的行为,这样一来,将会产生不可估量的问题.

2 实例分析

2.1 基于 ASP.NET+SQL Server 的网站攻击实例解析

采用后台登陆窗口的 SQL 注入后台窗口代码^[5]:

```
string sqlstr=“select * from administrators  
where name=“+name+”’and password=“+psaa-word+””’;  
Sql Command comm=new SqlCommand(strsql,conn);  
OleDbDataReader dr=comm. Execute Reader();  
if (! dr. Read())  
Response. Write (“<script> alert (用户名或其对应的密码不正确)  
</script>”);  
else  
Response. Write (“<script> alert (成功登录)</script>”);
```

其中:conn 为数据库连接的对象;pass-word 为密码;name 为使用者昵称或名字. 若以 administrator 作为操作管理的名字,pass-word 设置为 654321,则数据库接收到来自 ASP.NET 应用的 SQL 语句为:

```
select * from administrators where name=‘ad-ministrator’ and pasword=‘654321’
```

由此可以在数据库中找到一条符合的数据,能直接登上账号. 如果用户输入的用户名为‘haha’OR
3<4--,密码为 456 ,那么,ASP.NET 应用发送给数据库服务器的 SQL 语句为

```
select * from administrators where name=‘haha’OR 3<4 --‘and password’=‘456’
```

前面说到“--”是解释内容的,那么,3<4 恒为真,OR 运算中两个数据只要一个是对的,另外一个也是对的. 按照这样的方式,前面的 SQL 语句 where 以后的条件成立,那么 dr. read()返回的结果和 administrators 表就是正确的,意味着可以登录. 这种方式是登录账户及密码都是直接操作的一种手段. 另外,也有一些不同的操作方法,但是其原理都相同,只需要令 dr. read()返回“真”的结果就可以.

2.2 数据更新时的 SQL 注入

首先,假设与后台数据库内部存在表 users,其结构如表 1 所示. 表 1 中:用户名用 name 表示;密码用 pass-
word 表示;用户等级用 level 表示,1 表示管理员,2 表示正常用户. 当普通用户更新密码时,代码^[6]如下:

:

```
strin strsql=“update users set pass-word=“+password+”’where name=“+name+””’;
```

表 1 用户信息表

Tab. 1 User information table

字段名称	字段含义	数据类型	是否为主键
Name	用户名	char 型	是
Password	密码	char 型	—
Level	用户等级	char 型	—

```
OleDbCommand comm=new OleDbCommand(strsql,conn);  
comm. Execute Non Query();  
:
```

在 conn 处输入数据库的连接对象,其中“password”是数据库的新密码,这个密码在未操作的情况下反馈到数据库并进行动作后,若登录账户 xiaogang 填写的密码是‘789’,level=‘9’,转换成 SQL 语句并发送到数据库为:update users set password=‘789’,level=‘9’ where name=‘xiaogang’.这样的话,原来普通用户 xiaogang 就变更成管理员。

3 ASP.NET 网站 SQL 注入防范

3.1 SQL 注入的防范措施

针对 SQL 的实现经过、实现理论和加入攻击的特性制定相应的抵御方法,从代码安全上进行阻止,如设置可靠的 SQL 参数、给数据进行加密处理、查验用户信息、设置存储路径和设置安全措施。ASP.NET 网站通过上述抵御方法,可以有效地加强其性能和安全,避免 SQL 注入攻击的实现^[7-10]。

1) 测试输入内容. 程序员测试输入内容的数据大小及类型,强制执行输入内容的限制. 会产生异常通常是因为输入了不符合要求的内容。

2) 测试字符串变量的内容. 只输入允许测试的数据后,才能对字符串变量进行检测. 通常不被允许检测的数据,有以 0 和 1 构成的二进制字符串,“:”(查询分隔符),“/ * ... */”和“—”等注释符号,“” (字符串分隔符号),以及由“\”开始的转义序列字符。

3) 采用存储过程的方式. 当对数据库进行程序访问时,设置存储路径必须通过参数进行录入,并且尽可能存储路径的形式访问数据库. 在设置 T-SQL 程序代码的存储路径时,要尽可能使用静态 SQL,而非动态 SQL. 当使用动态 SQL 时,输入的参数不能是具有关键字、列名和表名的 SQL 语句或者是其中的一部分,必须是能够建立动态 SQL 的数值. 在访问过程中,严禁直接接入 SQL 后 exec 执行,而应该通过 p_executesql 和它的形参参数进行访问. 当使用静态 SQL 时,可以自动侦测输入代表安全代码的安全字符后,并且自动进行防御攻击. 长度检查特性和类型验证特性存在于 SQL Server 的 Parameters 集合中. 在使用 Parameters 集合的过程中,长度检查特性和类型验证特性会将可操作代码视为文字,并对不属于该范围的数据进行报警处理。

4) 加强安全性. 安全种类的 SQL 参数在使用程序的拼接语句的过程中,能够有效提高其安全稳定性. 因为一部分因素的影响,导致在访问数据库时,程序不可以按照设置的存储路径进行. 所以,通过安全种类的 SQL 参数,才可以对具有拼接 SQL 语句的程序进行访问。

5) 对敏感数据进行加密存储. 用户对敏感数据进行保存时,实施加密用户输入的数据并检测输入数据的内容的安全性. 以存储在数据库内的数据为基础对比加密后的敏感数据,就会发现和之前已经保存过的数据不同,经过用户加密后的敏感数据能够更加有效地制止 SQL 的注入攻击. System. WEB. Security. FormsAuthentication 加密方式在 ASP. NET 集成环境中,更易于加密数据。

6) 禁止将服务器端错误消息返回给页面浏览者. 程序设计人员可以利用已经在程序内设置的数据为基础,在操作 WEB 程序产生问题时发现起因,并表明严禁数据库中的数据外泄,指向系统显示错误界面. 比如,对 ASP. Net 中的配置文件(WEB. config)进行设置:<custom Errors mode=“Remote Only” default Redirect=“errors. htm”/>. 设置完成后,将一致显示“errors. htm”的自定义的用户界面,而不再显示如源文件存储位置、源错误位置等具体的原因。

7) 安全部署网站系统. a) 数据库服务应部署在专用的物理服务器上,而 Web 应用程序则不同,其必须要在多重的物理服务器上进行部署. 因此,可以利用交换设备的访问控制机制或者防火墙,有效地抵制存储于数据库服务器内的数据包传输至互联网的行为. 在现实数据访问中,既可以通过将 Web 程序与数据库数据分区存放进行实现,也可以通过分别设置 Web 程序和数据库存储进行实现. b) NTFS 格式应作为服务器分区的唯一格式,以最小权限为基本原则,合理配置 Web 程序权限和数据库数据权限. c) WEB 应用程序访问数据库时,应使用具有指定权限的数据库账号. d) 使网络环境更加可靠、安全和 Web 应用程序更加安全健康. 应用层安全的基础是物理层和网络层的安全,所以在保障网站本身安

全可靠的同时,要注重设备检测由程序输入的不安全字符和防火墙的合理化,以及定期对服务器的执行系统产生的补丁进行修补.

3.2 SQL 注入的防范策略

1) 严格筛选用户录入的数据和上交的参数. 在对 Web 页面进行设计时,对数值型参数,要对其是否包含非法字符进行判断,要严格筛选含有分号、双引号、单引号和逗号等标点符号的字符型参数;当一同出现多个字符串,如 select,delete,*,from,union 等,也不能消除警惕,应以用户录入参数的长短为依据,检查是否为合法程序,如果不是合法代码,就对其进行错误警示.

2) 设置数据库服务器的权限. 在 Web 界面连接数据库的时候,尽可能不使用超级管理员身份. 一般情况下,不允许 Web 页面干涉系统的存储方式和系统表的读取方式,即使是户表,对权限设置也要慎重考虑,对只需要读操作权限的用户,不给予插入、更新等权限.

3) 将不重要的交互式提交表格页面进行关闭或删除. 在编写代码的过程中,程序员屏蔽掉代码层内常见的危险字符,这样就可以阻止或者屏蔽一些简单的网站注入攻击.

4) 作为网站管理员,要及时打补丁并强化数据. 应定期、及时地通过相关设施和器具检查 Web 页面收到的攻击,实施监测数据库运行情况,禁止一切无用的功能和服务.

4 结 论

对于 SQL 注入漏洞的检测条件是有限制的,检测效果重点在于代码覆盖率. 通常情况下,会采用黑盒测试技术进行客户端 SQL 注入漏洞检测技术,当然检测效果也是取决于所建立检测漏洞模型是否准确,同时,也需要原代码给予一定的支持,如服务器端源代码的静态检测技术和动态检测技术,及两种检测技术相结合共同检测. 但是污点跟踪技术是在发现 SQL 注入漏洞之后,而且也只是会终止进程或是发出报警,无法对抗利用漏洞发起的拒绝服务攻击,而综合检测技术无法追踪漏洞的位置和起因等,它只是提供安全部署的框架.

ASP.NET 网站开发的信息系统被入侵,是因为代码存在的问题被入侵者发现. 因此,当程序员在编写程序时,首先,检测对客户端提交的变量参数和字符变量参数;然后,根据以下 6 点防御 SQL 注入的攻击. 1) 通过类安全的参数代码机制打造动态 SQL 语句. 2) 简短单表输入和查阅字符会降低有害代码强行 SQL 命令的频率. 3) 检测填写的权限问题,保证填写的数据是可用的,同时,通过客户端和服务器的双向验证实现更严格的访问控制. 4) 代码设置前先做评定,脆弱敏感的信息加上密码后再放置到数据库中,对比填写的密码是否和数据库一致,没有针对性意义的数据没事,有针对性的则要进行防范. 5) 对返回数据进行检测,超过的记录都按照出错来进行处理. 6) 操作者的权限放置到最基本要求.

参考文献:

[1] 马凯,蔡皖东,姚烨. Web 2.0 环境下 SQL 注入漏洞注入点提取方法[J]. 计算机技术与发展,2013,23(3):121-124,128.

[2] 丁允超,范小花. SQL 注入攻击原理及其防范措施[J]. 重庆科技学院学报(自然科学版),2012,14(5):136-139.

[3] 石聪聪,张涛,余勇,等. 一种新的 SQL 注入防护方法的研究与实现[J]. 计算机科学,2012(增刊 1):60-64.

[4] 王伟平,李昌,段桂华. 基于正则表示的 SQL 注入过滤模块设计[J]. 计算机工程,2011,37(5):158-160.

[5] 周益宏,陈建勋. 浅析基于 ASP.NET 的网站 SQL 注入攻击及防范措施[J]. 计算机安全,2010(6):93-95.

[6] 王云,郭外萍,陈承欢. Web 项目中的 SQL 注入问题研究与防范方法[J]. 计算机工程与设计,2010,31(5):976-978.

[7] 周琰. SQL 注入检测方法的研究与实现[D]. 西安:西北大学,2011:16-20.

[8] 竺霞芳. 双层防御 SQL 注入攻击的方法[D]. 武汉:华中科技大学,2011:32-38.

[9] 刘合叶. 多功能 SQL 注入检测系统的实现及攻击防范方法研究[D]. 北京:北京交通大学,2009:48-52.

[10] 陈柏生,吴可沾,杨育辉. 互联网用户安全登录平台设计[J]. 华侨大学学报(自然科学版),2011,32(6):638-640.

(责任编辑:黄晓楠 英文审校:吴逢铁)