**SQL注入攻击检测与防范**

**2.1 SQL注入攻击检测**

面对各种复杂多样的SQL注入攻击技术, SQL注入攻击的检测与防范技术成为研究的重点。下面将介绍检测SQL注入攻击的各种方法。

2.1.1 黑盒测试

黑盒测试是通过工具利用网络爬虫技术抓取网站所有链接, 然后判断这些网页中是否存在SQL注入点, 用已知的SQL注入方法来尝试对Web应用进行SQL注入测试, 但对Web应用不做任何的破坏。

使用SQL注入工具输入需要测试的URL, 工具会自动地判断网站是否存在SQL注入漏洞, 同时在模拟注入攻击过程中, 通过增加字典的内容来优化攻击速度。渗透测试中的黑盒测试和普通的黑盒测试一样, 不需要考虑系统代码的实现过程, 只需测试系统是否会受到攻击。该方法的缺陷是测试的结果准确率取决于工具的好坏, 而且不能将产生漏洞的原因显示出来。

2.1.2 静态检测

静态检测就是分析Web应用程序的代码, 根据SQL注入漏洞形成的原理, 分析代码中是否存在SQL注入漏洞。Gould C等人[[2](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "b2-1671-1122-0-9-129)]提出的JDBC-Checker是一种静态代码检查技术, 它通过分析动态生成的SQL查询语句中的错误发现和预防SQL注入攻击。其实现过程分为两步：第一步创建描述特定SQL查询串在给定位置所有可能状态的有穷状态自动机模型, 利用该自动机解析Web应用程序生成的class文件, 计算过程间的控制流图; 第二步把SQL语言的类型系统当作一个上下文无关语言, 利用上下文无关语言的可达性分析方法对目标程序进行语义分析, 进而找出潜在的SQL注入漏洞。[图7](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "outline_anchor_15)是DVWA渗透测试中SQL注入测试的代码。从[图7](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "outline_anchor_15)中可以看出, 服务器接收用户请求没有做任何的过滤, [图8](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "outline_anchor_16)是对输入进行过滤的代码。

|  |  |
| --- | --- |
|  | * **[Figure Option](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html)** |
| [IMG_262](http://netinfo-security.org/article/2015/1671-1122-0-9-129/img_7.png) | **图7** 没有过滤测试代码 |

|  |  |
| --- | --- |
|  | * **[Figure Option](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html)** |
| [IMG_263](http://netinfo-security.org/article/2015/1671-1122-0-9-129/img_8.png) | **图8** 经过过滤的测试代码 |

2.1.3 动态检测

动态检测是Web应用程序根据用户的输入信息来动态构造SQL查询语句, 然后对查询语句进行分析, 判断其合法性, 阻止恶意的SQL查询语句在数据库上直接执行。常用的动态检测方法有以下3种[[3](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "b3-1671-1122-0-9-129)]：

1）分析合法请求的执行流。开发者需要依赖可靠字符的初始化, 但持久存储可靠字符可能会导致二阶攻击。

2）上下文敏感字符评判。不安全字符的初始化依赖于Web程序员, 而且移除不安全字符限制了应用程序的功能。

3）基于语法的分析树评判。攻击者通过输入标识符来操作输入字符, 从而对输入字符进行的语法约束产生影响。

2.1.4 静态与动态分析相结合

静态与动态代码分析相结合就是将静态代码分析验证与运行时的实时监控检查结合起来, Halfond W等人[[4](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "b4-1671-1122-0-9-129)]提出的AMNESIA技术就是该种技术。AMNESIA是一种基于模型的技术, 它的检测过程分为两部分：1）静态分析, AMNESIA使用静态分析方法建立不同的数据库来访问查询语句; 2）动态分析, AMNESIA会在查询语句被数据库执行之前, 将它们拦截下来, 检查是否违反静态阶段所建立的模型, 违反所建立模型的查询就被认为是SQL注入攻击而不会被执行[[4](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "b4-1671-1122-0-9-129)]。

AMNESIA的缺点是其精确性由静态分析时所建立的查询模型所决定, 某些类型的二义性代码或者查询开发技术不佳都可能导致这个步骤的准确性变差, 如果模型建立的不准确, 则会导致整个扫描检测结果失败。

2.1.5 基于污点跟踪技术

基于污点的跟踪技术是把用户输入当作恶意用户的可疑数据, 跟踪它在执行过程中的每一个动向, 进一步判断它是否会引发SQL注入攻击。例如, 文献[5]使用信息流分析的方法来检测与输入验证有关的错误, 用静态分析的方法来检测哪些污点数据流不满足敏感函数的前提条件。该技术需要满足两个假设：1）使用它的输入系统可以准确地表述出敏感函数的前提条件; 2）过滤器可以很准确地判断出通过的用户输入确实是没有被污染的数据。对于很多应用程序和函数来说, 这两个条件都很难达到。Nguyen-Tuong A等人[[6](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "b6-1671-1122-0-9-129)]提出修改一个PHP解释程序来跟踪每个字符准确的污染信息。这种技术能在未受信任的输入被用来创建某些类型的SQL令牌时, 使用敏感信息分析方法检测和拒绝查询。

2.1.6 随机指令集

SQL rand即随机指令集, 开发人员设置密钥, 服务器会将SQL语句中的某些关键字进行随机化, 用户输入的内容构造成SQL语句之后, 其中的关键字会被其他内容代替。一般情况下, 攻击者不会使用其他内容代替SQL语句中的关键字, 并且也不知道随机的指令密钥。因此, 当恶意用户输入SQL语句时, 程序不会对其中的内容进行随机化, 这样就会导致服务器出错, 不会执行SQL注入语句。这种方法通过密钥来改变指令, 因此必须保证密钥不泄露, 否则还会出现注入。此外, 这种方法对资源消耗比较大, 它需要为数据库设置一个代理。

**2.2 SQL注入攻击防范**

目前SQL注入攻击的防范技术可以分为两个方面。一是编写代码时要注意用户输入验证、使用参数化语句、输出编码、规范化方法等; 二是在Web应用平台的配置上要注意保护数据库以及其他注意事项。

2.2.1 用户输入验证

用户输入验证就是验证用户输入是否合法, 是否会对系统造成攻击行为, 包括白名单验证和黑名单验证[[7](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "b7-1671-1122-0-9-129)]。

1）白名单验证

白名单验证就是只接受白名单当中的用户输入, 用户请求时, 程序会对用户的输入与白名单中的标准进行判断, 如果不符合就会拒绝用户输入。白名单是开发人员自定义的, 可以是某个变量允许的值, 也可以是利用正则匹配用户的输入。

输入信息比较复杂时, 白名单可能会把正确的输入拒绝。因此, 对用户输入要求比较少的情况下才使用白名单, 对于复杂的用户输入不单独使用白名单, 可以采用与其他的防范技术相结合的方式防范攻击。

2）黑名单验证

黑名单是拒绝包含黑名单中内容的用户输入。黑名单的效率没有白名单高, 因为存在威胁的字符远远要比不危险的字符多, 黑名单中的内容越多, 那么用户输入匹配的时间就越长。但是在对用户输入要求比较多的情况下可以使用黑名单。通常不单独使用黑名单来防止SQL注入, 面对不同的情况可以选择使用黑名单。如[图9](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "outline_anchor_18)所示, 当用户输入在黑名单中的关键字时, 就会提出警告。

|  |  |
| --- | --- |
|  | * **[Figure Option](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html)** |
| [IMG_264](http://netinfo-security.org/article/2015/1671-1122-0-9-129/img_9.png) | **图9** 黑名单测试代码 |

2.2.2 使用参数化语句

Web应用和数据库的开发越来越简单, 开发人员可以使用参数代替用户的输入来对数据库进行访问, 但是, 使用参数化语句不能做到万无一失, 恶意用户可以通过抓包来修改参数。另外, 在与数据库交互的地方, 恶意用户可将恶意代码插入到HTML中, 这样恶意代码就会被存储进数据库。当用户访问Web应用程序的其他位置时, 形成SQL注入。

2.2.3 规范化

恶意用户一般绕过验证的方法是通过抓包将需要执行的信息先编码, 然后发送到应用程序。[表1](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "outline_anchor_19)列出了单引号转换为各种编码类型的结果。

|  |  |
| --- | --- |
| IMG_265 | **表1** 单引号不同编码表示表 |

在实际的环境中, [表1](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "outline_anchor_19)的编码方法可能不会解释成单引号, 因为编码属于应用服务器中或者Web服务器层, 我们很难遇测这些编码过的字符在系统中怎样解码。

以下是两种规范化的方法[[8](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "b8-1671-1122-0-9-129)]：

1）拒绝所有存在隐患或者不符合规范的输入。当用户的输入含有异常字符或者其中有字符编码时, 服务器就会拒绝接受这些字符。这种方法既是黑名单又是白名单, 使用这样的方法可以做到不接受经过编码的字符, 但也有可能把正确的输入也拒绝了。

2）如果方法1）比较难实现, 可以使用白名单。但是这种方法的原理是递归查看是否有编码, 也就是如果用户输入中有编码, 程序会解码, 然后再去判断是否有编码, 如果有再去解码, 直到没有编码为止, 这样太耗费资源。要想简单的实现这种方法, 可以对字符串只进行一次解码, 解码之后如果还存在字符就直接拒绝。

2.2.4 实时监控保护措施

实时监控保护就是时刻对用户访问Web网站时发送的请求进行监控, 这样可以做到最大限度地阻止用户进行SQL注入攻击。

实时监控保护措施防范SQL注入攻击比其他方法更有效。虽然实时监控是在用户进行攻击时进行检测, 但相比白盒测试这种在攻击之前进行防范的的方法来说, 测试耗费的资源更少。而且白盒测试不能检测或者阻止未知的SQL注入, 实时监控可以阻止大多数的SQL注入。

在管理员安装实时监控系统时还要注意, 有的防护系统对于整个服务器的硬件资源要求比较高, 而且对资源的消耗也比较高。另外有的监控系统对于管理人员的技术要求比较高, 要花费大量的时间去配置监控系统, 使得系统运行起来万无一失。

Web应用防火墙WAF就是用来防范Web攻击的监控系统, 并且不需要改变基础设施配置。另外基于WAF的设备不会占用太多的系统资源[[9](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "b9-1671-1122-0-9-129)]。

2.2.5 保护数据库

当攻击者找到Web应用的SQL注入漏洞后一般有两种目的：一是获取数据库中的信息, 也就是想从数据库中获取一些敏感信息, 可以是管理员的用户名、密码, 也可是一些个人信息等; 二是攻击者想通过SQL注入漏洞提升自己的权限控制这台服务器, 从而可以进一步对内网渗透, 获取其他机器的控制权限。

控制未授权的用户访问应用程序的数据可以通过以下几个技术来限制访问：严格对各个用户的权限控制、编写代码时使用的用户权限必须另外添加、对数据库中敏感信息进行加密。此外, 还可以通过低权限用户对数据库服务器进行保护, 管理员也要注意及时更新数据库的版本和取消公共权限等。

2.2.6 其他部署注意事项

除了对服务器错误处理、数据库安全配置等, 还要对服务器所在的网络进行安全部署, 如路由器、交换机等。通过安全的配置来巩固整个系统的安全, 从而减少SQL注入攻击, 这些技术可以用在系统底层来防范SQL注入攻击[[10](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "b10-1671-1122-0-9-129)]。IIS服务器中是否开启这些Web服务扩展决定了服务器的安全程度。

**2.3 新型SQL注入防范技术**

SQL注入攻击发展至今, 其方法越来越多, 另外当前Web应用开发比较容易, 这也导致了Web应用的漏洞比较多。虽然一些开发人员做了防范措施, 但是要防范所有的SQL注入攻击几乎是不可能的, 所以本文提出一种改进的SQL注入的防范方法[[11](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "b11-1671-1122-0-9-129)]。

该方法分为三步, 全面防范SQL注入攻击。首先, 利用正则表达式和业务逻辑对用户输入进行合法性验证; 然后设置代理过滤器, 使用随机指令集将SQL查询语句进行关键字随机化; 最后在服务器上使用污点追踪技术对数据跟踪, 并及时作出响应[[12](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "b12-1671-1122-0-9-129), [13](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "b13-1671-1122-0-9-129)]。

虽然说大多管理员会使用黑名单或者正则表达式来防范SQL注入攻击, 但这仍会存在对于过滤过的用户输入完全相信等问题, 只有使用完整的三步防范技术才能做到万无一失。使用三步防范技术优点可以概括如下。

1）由于没有对用户的输入进行严格的验证, 新的防范技术使用正则表达式和业务逻辑是比较有效的验证方法。大多数开发人员对于恶意用户的输入只有过滤, 并没有后续的防范。恶意用户一旦绕过第一层的防范, 没有后续的检测, 还是可以提取数据库中的信息的。如果使用代理服务器SQL查询语句中的关键字随机化, 而恶意用户不知道随机的哪些关键字, 随机化的密钥也不知道, 所以恶意用户构造的SQL查询语句不会被执行。

2）通过密钥来随机编码动态构造的SQL语句中的关键字, 解码时就需要随机编码的密钥, 密钥是由管理员保管的, 所以在前两步的防范下SQL注入发生的几率很小, 但是并不能完全排除被攻击的可能性。所以最后要使用污点跟踪技术对用户的行为进行跟踪, 根据SQL注入的模型进行匹配来判断是否存在SQL注入漏洞[[14](http://netinfo-security.org/article/2015/1671-1122-0-9-129.html" \l "b14-1671-1122-0-9-129)]。

上述的防范技术将三种有效防范SQL注入攻击的方法结合在一起, 相辅相成, 改善了目前单一的防范SQL注入攻击方法的局面, 也可以防范绝大多数的SQL注入攻击, 增加了Web应用的安全性。