高技术通讯 2012 年 第22 卷 第11 期:1161 ~1168

doi:10.3772/j.issn.1002-0470.2012.11.009

模型驱动的 Web 应用 SQL 注入渗透测试①

田 伟② 许 静③ 杨巨峰 张 莹 刘 磊

(南开大学信息技术科学学院 天津 300071)
(南京大学计算机软件新技术重点实验室 南京 210093)

摘 要 针对结构化查询语言(SQL)注入渗透测试用例不充分造成测试漏报的问题,对基于形式化建模生成渗透测试用例问题进行了研究,提出了以下方法:将 SQL 注入漏洞渗透测试用例生成分为两步:第 1 步建立渗透测试用例的形式化模型,以用例模型更全面、有规律地描述当前各种 SQL 注入攻击的方法模式,指导生成更多种类的用例输入;第 2 步提出若干新的 SQL 注入漏洞渗透测试用例覆盖度准则,将用例模型实例化、生成覆盖更多样式的用例输入。实验表明,用上述方法生成的用例,优于当前其它研究中使用的随机枚举用例,可更有效地测出隐藏于 Web 应用不足防御措施之后的 SQL 注入漏洞,从而降低渗透测试结果的漏报。

关键词 Web,渗透测试,结构化查询语言(SQL)注入,攻击建模,安全漏洞,用例

0 引 言

结构化查询语言(structured query language,SQL)注入安全漏洞[1]对 Web 应用安全造成极大威胁,及时准确找出 Web 应用中 SQL 注入漏洞加以修补十分必要。渗透测试是检测 Web 应用安全漏洞存在性的有效手段,其原则是在真正攻击之前以模拟攻击找出软件安全漏洞[2,3],结果较可信并能反映漏洞利用情况,因此对其研究日益受到关注。

提高测试准确度,即减少测试结果的误报(false positive)和漏报(false negative),是渗透测试研究的根本目的。渗透测试有三个基本步骤,即信息收集、攻击生成和反应分析[4],因此为提高准确度,目前一些研究关注 SQL 注入渗透测试的信息收集和反应分析[4,5],通过提高输入点发现能力或改进分析反应等工作降低漏报或误报。攻击生成阶段的用例输入(攻击输入集合)也是影响测试准确度的重要因素。不充分的测试用例不能全面测试软件防御措施,难以触发隐藏于不充分防御措施后的安全漏洞,造成漏报而降低测试准确度。但目前对 SQL 注入渗透测试用例的研究尚显不足,主要存在两个问题:(1)渗透测试用例形式化建模问题。目前相关研究给出的 SQL 注入渗透测试用例多是随机枚举方式,难以揭示用例规律,且相当于用无限方式表达无限集合。因此,为表述用例规律性,据此明确测试中应使用什么用例,并能以有限规模方式描述这些用例,给出用例形式化模型十分必要。(2)渗透测试用例覆盖准则问题。目前研究尚未对渗透测试用例的充分性问题深入讨论,因此有必要引入其他领域软件测试研究中用例充分性评价思想:定义相应的渗透测试用例覆盖准则,作为衡量和指导渗透测试用例充分性的标准。因此,针对如何改进 SQL 注入渗透测试用例以减少测试漏报的问题,本文提出了模型驱动的 SQL 注入漏洞测试方法,该方法将渗透测试用例生成分为两步:第 1 步为 SQL 注入攻击建模,并根据攻击模型生成抽象的测试用例模型;第 2 步提出新的渗透测试用例覆盖准则,据此将第 1 步建立的用例模型实例化为可执行测试用例。实验表明,该方法可生成更充分的用例,有效测出隐藏于不充分黑名单过滤防御[6]后的 SQL 注入漏洞,降低测试的漏报。

① 863 计划(2009AA01Z152)和天津市自然科学基金重点(12JCZDJC20800)资助项目。
② 男,1982 年生,博士生;研究方向:软件测试,数据库安全;E-mail:tianwei8202@163.com
③ 通讯作者,E-mail:xujing@nankai.edu.cn
(收稿日期:2011-10-10)

1 基于安全目标模型的 SQL 注入建模

安全目标模型(security goal model,SGM)是一种新型的用来描述漏洞、安全特性、攻击或安全软件开发的模型[7]。测试用例一般定义为一个三元组 t = (Pre,In,Out),其中 Pre 为前置条件,In 为输入,Out 为期望输出,默认 Pre 永真并忽略。对于 SQL 注入安全漏洞的渗透测试用例,我们定义 In 为攻击输入集合、Out 定义为存在安全漏洞的反应特征。首先进行第 1 步的工作:对 SQL 注入攻击建模,以描述 SQL 注入 In 和 Out 的种类和规律。

根据 SQL 注入攻击特点,当前对其攻击建模研究主要采用攻击树建模[8,9],而研究表明攻击树建模方式自身存在一些缺点,如攻击行为和结果都用节点表示,容易造成混乱;树分支和节点易产生冗余和重复等。

作为对攻击树模型的改进,本文应用这种新的 SGM 为一阶 SQL 注入攻击[2,3]建立了模型。限于篇幅,本文不对一阶 SQL 注入手段收集分析和 SGM 建模规则等进行详述,相关内容可参考文献[2,5,9]。

安全目标模型描述角度是安全相关的行为目标,所以基于攻击者直接攻击目标对 SQL 注入攻击建模。图 1 模型以根节点表示实现 SQL 注入攻击的总目标,进而自底向上描述、按照攻击直接目标,将实现这个总目标攻击分为三类子目标:窃取系统信息、绕过认证和注入运行恶意命令,模型向上进一步描述了实现这三种子目标各自所需的攻击子目标,直至最上端的攻击注入和探查注入点攻击子目标。

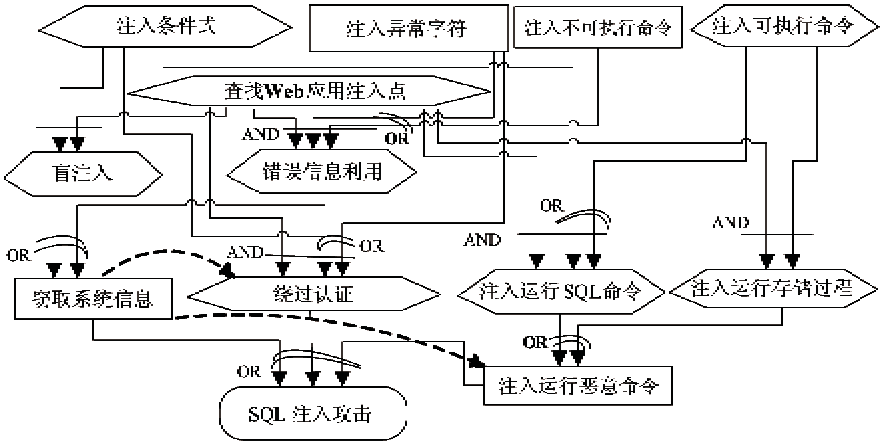


图 1 SQL 注入攻击 SGM 图

图 2 – 图 4 是对图 1 的展开详细描述:将“窃取系统信息”子目标展开描述为图 2 中两个子模型:(a)错误信息利用子目标模型;(b)盲注入子目标模型。同样为图 1 中“注入运行恶意命令”子目标细节建模:分为图 3 中(a)注入运行 SQL 命令攻击和(b)注入运行存储过程攻击两个模型;图 4 展开描述图 1 中“绕过认证”攻击子目标。

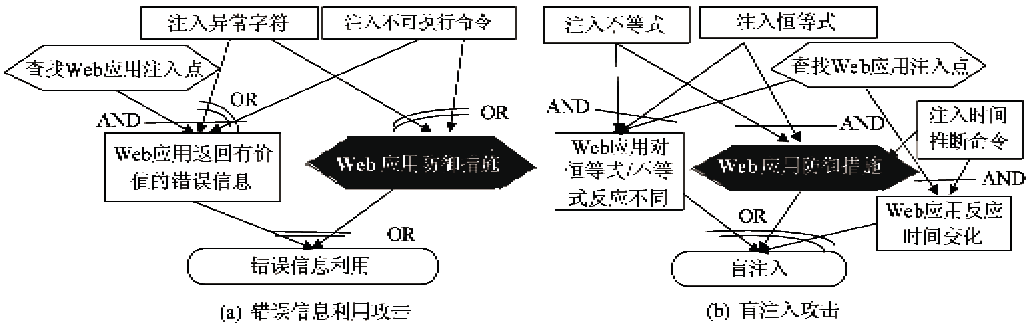
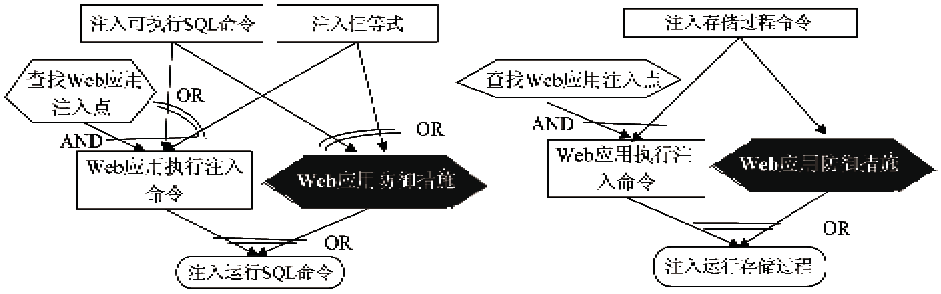


图 2 SQL 注入攻击中“窃取系统信息”SGM 图



(a) 注入运行SQL命令攻击　　(b) 注入运行存储过程攻击

图 3　SQL 注入攻击中"注入运行恶意命令"SGM 图

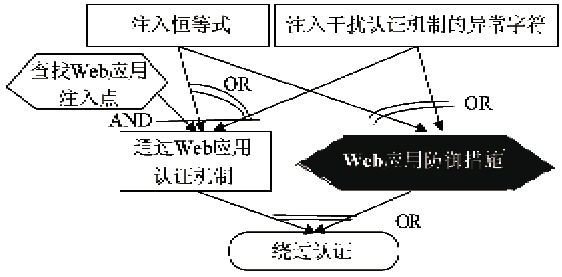


图 4　SQL 注入攻击中"绕过认证攻击"SGM 图

模型中自上而下每条路径代表一种攻击子目标的过程:最上端描述了攻击输入,中部子目标节点描述 Web 对攻击输入的反应,即 Web 有何种反应成为 SQL 注入漏洞;反作用节点(黑色)表示攻击被 Web 应用防御拦截造成不成功的路径。将图 2 至图 4 中自上而下实现每种攻击子目标的成功攻击路径定义为攻击模式(暂不考虑注入点因素):

定义 1　攻击模式(attack scheme)为三元组 < OBJ,INP,OUT > ,其中 OBJ 为攻击目标,INP 为攻击输入,OUT 为 Web 应用存在安全漏洞的反应。本文研究重点是其中的用例输入(INP),即形式化描述 attack pattern library[3],以下对其抽象建模进行研究。

2　SQL 注入渗透测试用例建模

对 SQL 注入攻击输入进行形式化描述:定义注入攻击输入算子 ∇。∇_i 代表某类攻击输入集合,如 ∇_{SQL} 代表 SQL 注入攻击输入总体集合。结合图 1 – 图 4 模型描述,表 1 给出了各 SQL 注入攻击输入算子的定义。

将算子结合起来形成算子表达式,以表述实现某种攻击目标所需的攻击输入集合及其规律。定义 ‖ 为算子的 OR 操作,&& 为算子的 AND 操作。即 $\nabla_1 \| \nabla_2 = \{x | x \in \nabla_1 \vee x \in \nabla_2\}$,对应 SGM 中的 OR 操作符,表示对于实现某攻击目标,∇_1 和 ∇_2 所代表的两种攻击输入可选其一(或都选);$\nabla_1 \&\& \nabla_2 = \{x, y | x \in \nabla_1 \wedge y \in \nabla_2 \vee x \in \nabla_2 \wedge y \in \nabla_1\}$,对应 SGM 的 AND 操作符,表示 ∇_1 和 ∇_2 所代表的两种攻击输入应同时一起使用。且 $\nabla_1 \| \nabla_1 = \nabla_1$。

表 1　SQL 注入攻击输入算子

算子	定义
∇_{DS}	异常字符集合
∇_{LG}	选取干扰认证机制的异常字符集合
∇_{DC}	不可执行命令集合
∇_{CON}	条件式集合
∇_{IE}	选取恒等式集合
∇_{NE}	选取不等式集合
∇_{TI}	利用条件式构造时间推断命令集合
∇_{COMS}	可执行命令集合
∇_{SQLC}	选取可执行 SQL 命令
∇_{SP}	选取可执行存储过程命令
∇_{AND}	以 AND 关系注入条件式
∇_{OR}	以 OR 关系注入条件式
∇_{disg}	对注入进行伪装

定义算子的复合运算为 ·,算子 ∇_1 与算子 ∇_2 进行复合运算记为 $\nabla_1 \cdot \nabla_2$,表示 ∇_1 处理算子 ∇_2 代表的注入参数,生成新的或复合的参数形式。

上述算子之间运算符的优先级定义:复合运算符 · 的运算顺序为从右至左,即 $\nabla_1 \cdot \nabla_2 \cdot \nabla_3$ 意为 $\nabla_1 \cdot (\nabla_2 \cdot \nabla_3)$。&& 操作符优先级高于 || 操作符,即 $\nabla_1 \| \nabla_2 \&\& \nabla_3$ 等价于 $\nabla_1 \| (\nabla_2 \&\& \nabla_3)$。复合运算符 · 的优先级高于 && 和 || 操作符,括号的优先级最高。基于上述的算子定义和图 2 – 图 4 的 SGM 描述等,将定义 1 的攻击模式具体化。

表 2 中模式集合一方面以形式化符号描述攻击输入(INP),以有限符号表述无限攻击输入,克服随机枚举用例表达方式无限性和无规律性;另一方面,分类详述各种 SQL 注入攻击模式,可指导生成包括

不同攻击种类的用例,以下将表 2 中 INP 模型作为用例模型,对各算子式实例化生成用例集,使用例包含各类攻击输入。此工作实现前述第 1 步根据攻击模型生成抽象测试用例模型。

表 2 SQL 注入各攻击子目标攻击模式集合 M

攻击模式	攻击目标(OBJ)	攻击输入(INP)	安全漏洞反应(OUT)
m_1	错误信息利用	$\nabla_{DS}\int \| \nabla_{DC} \| \nabla_{disg} \cdot (\nabla_{DS} \| \nabla_{DC})$	Web 应用返回特定错误信息
m_2	盲注入	$\nabla_{TI} \cdot \nabla_{CON} \| \nabla_{AND} \cdot \nabla_{IE} \cdot \nabla_{CON}\&\& \nabla_{AND} \cdot \nabla_{NE} \cdot \nabla_{CON} \| \nabla_{disg} \cdot (\nabla_{TI} \cdot \nabla_{CON} \| \nabla_{AND} \cdot \nabla_{IE} \cdot \nabla_{CON}\&\& \nabla_{AND} \cdot \nabla_{NE} \cdot \nabla_{CON})$	Web 应用对恒等式/不等式反应状态不同或反应时间变化
m_3	注入运行 SQL 命令	$\nabla_{SQLC} \cdot \nabla_{COMS} \| \nabla_{OR} \cdot \nabla_{IE} \cdot \nabla_{CON} \| \nabla_{disg} \cdot (\nabla_{SQLC} \cdot \nabla_{COMS} \| \nabla_{OR} \cdot \nabla_{IE} \cdot \nabla_{CON})$	Web 应用执行注入的命令
m_4	注入运行存储过程	$\nabla_{SP} \cdot \nabla_{COMS} \| \nabla_{disg} \cdot (\nabla_{SP})$	Web 应用执行注入的命令
m_5	绕过认证	$\nabla_{OR} \cdot \nabla_{IE} \cdot \nabla_{CON} \| \nabla_{LG} \cdot \nabla_{DS} \| \nabla_{disg} \cdot (\nabla_{OR} \cdot \nabla_{IE} \cdot \nabla_{CON} \| \nabla_{LG} \cdot \nabla_{DS})$	通过 Web 应用认证机制

定义 2 对于 Web 应用某输入点 p、攻击模式 m_i,若将 m_i. INP 模型的某实例化用例 inp 输入 p,使 Web 应用出现 m_i. OUT 所描述的反应,则称 p 对 inp 是可注入的,记为 $inp \propto p$。

根据以上定义给出对 Web 应用进行 SQL 注入安全漏洞渗透测试算法:

算法 1. Web 应用 SQL 注入渗透测试算法

输入:受测 Web 应用,攻击模式集合 M,Web 中输入点与 M 的对应关系 S;

输出:受测 Web 系统中存在 SQL 注入安全漏洞的输入点及可注入用例信息集合 VIP.

① VIP = Φ;
② 爬行/分析受测 Web 系统,从中查找 S 中所属可输入点集合 P;
③ 根据 Web 应用实际情况和覆盖准则实例化攻击模式集合 M 中 INP 模型,生成实例化用例集合 INP';
④ Foreach 注入点 $p_i \in P$;
⑤ {根据 S 确定 p_i 相对应的攻击模式集合 $M_i \subseteq M$;
⑥ Foreach 攻击模式 $m_j \in M_i$
⑦ {Foreach 实例化用例 $inp \in m_j$. INP'
⑧ { if ($inp \propto p$) //根据定义 2
⑨ {将 p 和 inp 的信息存入 VIP;} //endif
⑩ } // end of Foreach 实例化用例 inp
⑪ } // end of Foreach 攻击模式 m_j
⑫ } // end of Foreach 注入点 p_i
⑬ return VIP;

本研究重点是模型支持的用例生成,对基于模型的注入点规律生成问题研究从略。为简化讨论,在此仅考虑 Web 应用中带参数 URL 与登录表单[2,4]两类输入点作为测试对象。根据测试经验设定上述对应关系 S:

表 3 Web 应用输入点类型与 M 的对应关系 S

输入点类型	攻击模式
带参数 URL	m_1. m_2. m_3. m_4.
登录认证 FORM	m_1. m_3. m_4. m_5

3 形式化用例模型的实例化

本部分进行上述第 2 步工作:将测试用例模型,根据 Web 应用实际情况和一定覆盖准则,实例化为可执行的测试用例。其中 Web 应用实际情况包括输入点格式要求(如字符型、数值型)、数据库类型等。而覆盖准则的作用是衡量对模型实例化用例是否已经充分。以下重点对后者进行论述。

本研究选择对目前 Web 应用的黑名单过滤防御[6]进行测试,为测试其是否可以防御各式 SQL 注入攻击输入(用例),对渗透测试用例覆盖度定义原则是:尽量全面覆盖多样式的攻击输入。为此基于测试用例的用户输入覆盖准则和等价类划分(分域测试)的思想,定义若干新的实例化用例覆盖准则。

定义 3 命令动词覆盖准则:$\forall mv(mv \in MV \rightarrow \exists tc \in TC \wedge < tc, mv >)$: MV 为测试选用的受测 Web 应用可执行命令动词集合;TC 为渗透测试实例化用例集合;< tc, mv > 表示用例 tc 的构造中使用了命令动词 mv 且未使用其它动词。

此准则主要用于可执行命令类 ∇_{COMS} 的实例化,构造含各种动词的受测 Web 应用可执行命令语句,以测试 Web 应用是否全面防御了由各种动词构造的可执行命令注入。

定义 4 关系谓词覆盖准则:$\forall op(op \in OP \rightarrow$

$\exists tc \in TC \wedge < tc, op >$):OP 为测试选用的 SQL 语法条件式关系谓词集合;<tc,op>用例 tc 的构造中使用了关系谓词 op 且未使用其它关系谓词。

此准则用于条件式类∇_{CON}实例化:使实例化条件式用例集合包含各种 SQL 语句条件式关系谓词,如算数比较运算符 <,<=,<>,=或集合包含运算符 IN,Between 等,以测试 Web 对不同形态条件式注入的防御能力。

文献[3]描述了大数量非法数据集中选取有限子集作为测试用例的方法:非法字符覆盖比例(illegal coverage ratio, ICR)和随机覆盖比例(random coverage ratio, RCR)。在此对∇_{DS}实例化用例覆盖准则采用 RCR 法。并以∇_{DC}实例化用例代表∇_{DC}进行测试:若受测 Web 应用对随机的任意∇_{DS}不出现错误信息导致的信息泄露,则说明其已经正确配置异常输入处理机制;若受测 Web 应用对∇_{DS}产生了可利用错误信息,则说明∇_{DC}输入也会导致相同问题,故暂不对∇_{DC}进行实例化。对命令动词数量较多的$\nabla_{SP} \cdot \nabla_{COMS}$也采用 RCR 法,随机选择若干存储过程命令动词构造实例化用例,随机用例数量视测试规模而定。

∇_{AND} ∇_{OR}实例化是以 and 或 or 关键字连接构造的条件式。对∇_{disg}实例化覆盖准则要求其包括测试中选定的各种对攻击输入伪装手段,即对于由$\nabla_{disg} \cdot$TC 生成的 ∇_i:

定义 5　伪装手段覆盖准则:$\forall dg(dg \in DG \rightarrow \exists tc \in TC \wedge < tc, dg >)$:DG 为伪装手段集合;<tc,dg>表示用例 tc 使用且只使用 dg 方式进行伪装并加入了 ∇_i 中。

4　实验结果及其比较

本研究关注渗透测试用例充分性问题,即用例集突破不充分黑名单防御机制测试到隐藏的 SQL 注入安全漏洞的能力。所以需要既可知 SQL 注入安全漏洞位置和数量,又具有不充分黑名单防御机制的 Web 应用为测试对象,为此搭建如下的实验平台。

4.1　受测 Web 应用

开发了两个 Web 应用作为测试的目标系统:一个 JSP 网站和一个 ASP 网站,代码行数分别约为 5500 和 15000;其功能和结构都体现真实 Web 应用并可实际运行,以使对其渗透测试可作为对实际 Web 应用测试的代表。应用文献[6]中提出的预设分级防御评价测试工具效果的思想,在两个 Web 应用与后台数据库通道中(表 6、表 7 中①~⑪),对 SQL 注入攻击设定两个级别的防御措施(表 4):Level 0 对应的是普通的未设防造成的 SQL 注入漏洞;Level 1 黑名单过滤关键字是不充分的(即只在过滤黑名单中列举一部分可能的攻击输入关键字或符号,另一些攻击输入未被过滤),这相当于在网站添加了"不充分防御措施后的 SQL 注入漏洞"。以此满足上述实验环境要求。

表 4　目标 Web 应用防御分级

防御级别	防御措施
level 0	不采用防御措施
level 1	黑名单过滤

4.2　测试系统和实例化用例设定

根据算法 1 开发了对受测 Web 应用安全漏洞自动渗透测试系统:采用相关研究通用的"爬行—注入—分析"检测方式[1,2]:以爬虫模块遍历受测 Web 应用得到其所有页面集合(表示设定本测试系统对 4.1 中预设的 Web 漏洞存在位置未知,使后文公平评价不同工具测试总时间),提取带参数形式的 URL 及登录认证 FORM 作为输入点(实现算法 1 的②)。测试用例注入模块负责将各攻击模式实例化用例注入相应输入点;漏洞判断模块根据 Web 反应判断各输入点是否存在漏洞(实现算法 1 的④-⑬,具体方法从略)。

根据第 3 节的论述,对各攻击模式算子式实例化设定如表 5 所示(实现算法 1 的③,目前人工生成用例再装入用例注入模块使用)。

表 5　实例化渗透测试用例设定

参试算子	覆盖准则	实例化用例设定
∇_{DS}	RCR 覆盖	选取 5 个随机字符/字符串
$\nabla_{SQLC} \cdot \nabla_{COMS}$	命令动词覆盖	MV={select, create, insert, update, delete, drop, alter}每个动词构造 1 个用例
$\nabla_{SP} \cdot \nabla_{COMS}$	命令动词覆盖	MV={随机选择 3 个存储过程动词} 各动词生成 1 个用例
$\nabla_{TI} \cdot \nabla_{CON}$	命令动词覆盖	MV={waitfor, benchmark, sleep} 各动词生成 1 个用例
$\nabla_{IE} \cdot \nabla_{CON}$, $\nabla_{NE} \cdot \nabla_{CON}$	关系谓词覆盖	OP={<>, <, >, <=, >=, between, IN, like}各谓词生成 1 个用例
∇_{disg}	伪装手段覆盖	DG={Unicode 编码, ASCII 编码, 大小写混合法}

根据以上准则,对表 2 中各攻击模式的实例化用例输入示例如下(伪装形式从略):

m_1:'@ ^ * $;# admin-- @ @ vesion;-- having 1 =1-- / * * /

m_2:;waitfor delay 0:0:10'--

and 7 > =56 and 6 > =6 and 's'>'q'and 'u'>'z' and f'IN ('d',f') and 'w'IN ('d',f') ……

m_3:;create table temp (id nvarchar(255), num1 nvarchar(255));--

;insert into temp values(666', 'attacker')--

;select * from temp;--

;update temp set num1 = tester';--

;drop table temp;-- ……

m_4:';exec master..xp_cmdshell 'ping 127.0.0.1 -- ……

m_5:or 6 < >9; or 'something'LIKE 'some%'--

admin-- / * * / ……

依据定义 3 – 定义 5 的准则和表 5 的设定,最终生成约 103 个测试用例(包括用例的伪装和非伪装形式)。

4.3 测试结果比较

目前相关研究中[4,5]多采用一些 Web 漏洞测试工具作为效果比较,在此选择较知名的 Acunetix 6.5 和 IBM Rational AppScan 7.7[1]一起对目标 Web 应用的 SQL 注入漏洞进行渗透测试。本文以 tool A 和 tool B 指代两种工具(无对应顺序)。Tool A 和 Tool B 是当前渗透测试研究中,普遍采用的随机枚举用例方式的代表:很多相关研究[5,10]的攻击输入集合即是来源于对这些测试工具的总结。本文应用算法 1 的渗透测试工具以 My tool 表示。

一些研究中以测试出的可注入输入点数量为评判依据,但此数量并非具有绝对意义:Web 应用多个输入点往往会对应内部同一个 SQL 语句(如若干带参数 URL 传递参数给同一个 SQL 语句),在此情况下找到其中一个注入点和找到几个注入点指向的是同一个后台数据库交互通道(表 8),对于定位和修补漏洞的作用差别不大[10]。所以本文对各方法测试出的存在漏洞的注入点集合,按各注入点所对应的交互通道进行分类统计,即找到某通道所含的一个注入点即视为可测出本通道。表 6 至表 7 中以每页面代表一个与后台数据库交互通道,各工具测出的注入点类型分布如表所示(√表示可测出此处漏洞):

表 6 对 ASP 目标 Web 应用的 SQL 注入安全漏洞渗透测试结果

防御级别	存在 SQL 注入安全漏洞的 Web 页	注入点类型	My tool	tool A	tool B
level 0	① http://192.168.111.222/chat/useradmin.asp	登录认证 FORM	√	√	√
	② http://192.168.111.222/hzp/sub.asp	带参数 URL	√	√	√
level 1	③ http://192.168.111.222/hzp/login.asp	登录认证 FORM	√	√	√
	④ http://192.168.111.222/hzp/comment.asp	带参数 URL	√	√	
	⑤ http://192.168.111.222/bbs/admin/admincheck.asp	登录认证 FORM	√		
	⑥ http://192.168.111.222/hzp/vpro.asp	带参数 URL	√		

表 7 对 JSP 目标 Web 应用的 SQL 注入安全漏洞渗透测试结果

防御级别	存在 SQL 注入安全漏洞的 Web 页	注入点类型	My tool	tool A	tool B
level 0	⑦http://192.168.111.222:8080/mynews/admin/login1.jsp	登录认证 FORM	√	√	√
	⑧http://192.168.111.222:8080/mynews/ViewNews1.jsp	带参数 URL	√	√	√
level 1	⑨http://192.168.111.222:8080/mynews/admin/login2.jsp	登录认证 FORM	√		√
	⑩http://192.168.111.222:8080/mynews/ViewNews2.jsp	带参数 URL	√	√	
	⑪ http://192.168.111.222:8080/mynews/ViewNews.jsp	带参数 URL	√		

表 8 各存在漏洞的 Web 页包含的输入点数量

存在漏洞的 Web 页	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪
输入点数量	2	6	2	47	2	47	2	19	2	18	9

两个目标 Web 应用设计上未采用造成自动爬行页面工具解析困难的技术[2],故而三种测试工具都可以自动探查到表 6、表 7 中①~⑪输入点,所以测试结果差异不涉及各自发现注入点的能力问题。

在此情况下,对于未设防的输入点(level 0),三种工具都可以测出较全漏洞(①②⑦⑧);而在不充分防御措施的情况下(level 1),tool A 和 B 存在漏报,本文方法可测试出隐藏在不足防御之后的 SQL 注入安全漏洞。原因在于:

(1)本文采用攻击模型指导下的渗透测试用例生成,图 2 - 图 4 将 SQL 细化描述为 5 种攻击子目标(模式),可全面考虑其攻击输入作为用例,而 tool A 和 B 用例主要基于对实际已知攻击输入的随机罗列,用例盲目性强,难以保证较全面考虑各类攻击输入;

(2)本文提出用例生成方法可覆盖更全面的用例样式(表 9)。测试结果表示 tool A 和 B 在不充分黑名单防御机制阻挡下,其简单用例被过滤拦截,而认为此输入点已进行了防护而不存在安全漏洞,产生了漏报。本文依据新定义的覆盖准则生成用例,使测试用例可覆盖更多样式(如不同的条件式、命令样式等),以尝试突破 Web 应用简单防御措施的拦截和过滤,从而发现简单防御机制之后的 SQL 注入漏洞,避免漏报。

三种方法对两个目标 Web 应用测试均未产生误报。

应用本文所提出的渗透测试用例覆盖准则,对各测试方法用例评价对比如表 9 所示。

表 9 各测试方法用例覆盖度评价

准则	tool A	tool B	My tool
命令动词覆盖	Select,exec master	未覆盖	7 个 SQL 命令动词、3 个存储过程动词
关系谓词覆盖	=	=	< >, <, >, < =, > =, between, LIKE ,IN
伪装手段覆盖	未覆盖	unicode 编码	大小写混合,unicode 编码,ASCII 编码

各工具对两个 Web 应用测试所用时间(包括 3 种工具从爬行—注入—分析到测试完成,不包括 My tool 人工实例化用例的时间)和各自用例规模(攻击输入个数)如表 10 所示。

表 10 各方法测试时间总消耗

	JSP(min)	ASP(h)	用例规模(个)
My tool	49	5.8	103
tool A	39	4.3	45
tool B	14	2.3	32

时间统计结果印证了文献[1]的研究结论:测试工具测试所需时间与其发送接收的数据量没有必然的联系或比例关系。本方法更充分的用例未引起不可实现的时间消耗,所用时间属于相对可接受范围,证实其可行性。

5 本研究的特点分析

当前对渗透测试的研究,除一些专事测试效果评估的研究[1,2]外,多分别围绕信息收集—攻击生成—反应分析这三个渗透测试的基本关键步骤展开,且对攻击生成阶段的研究一般关注如何使既有攻击输入集合(用例)深入到达软件内关键位置[3,4,10],而未对攻击输入集合本身的规律和充分性进行研究。如对 SQL 注入漏洞测试研究[5]引用的攻击输入集合中,重言式攻击输入皆为"1 = 1"的形式,若 Web 应用仅过滤了等号构成的条件式,以此集合检测不到成功的攻击,但不表示此 Web 应用也可防御其它形式的重言式注入(如 49 > 45)。如此等等,说明当前研究未充分考虑如何以多种类、多形态的攻击输入测试 Web 应用防御机制发现防御机制的安全漏洞的问题。因此,本文的创新在于针对攻击输入集合(用例)本身的规律和覆盖度问题,研究如何生成更充分的渗透测试用例以全面测试 Web 安全防御机制,探查到隐藏在不足防御之后的安全漏洞,从而使测试更加面向受测 Web 应用往往采用一定防御的实际情况。

此外,本文方法在攻击模型指导下生成多种类、有规律的用例[11],不需要分析改变 Web 应用源代码[12],且生成用例范围不只局限于重言式攻击[13]。

6 结 论

提高测试准确度是渗透测试领域研究的根本目的,本文研究从改进用例的角度提高渗透测试准确度的问题。研究表明,通过对 SQL 注入和渗透测试用例适当建模,及提出渗透测试用例覆盖准则,可指导生成更有条理的用例(表 2)、优化用例集的结构(表 9),改进的用例集相比当前其它研究中采用的随机枚举的用例,可更有效测试出 Web 应用不充分防御后的 SQL 注入安全漏洞,从而降低测试漏报、提高测试准确度。下一步可将本文用例改进的研究与对渗透测试信息收集、攻击生成和反应分析三个阶段的相关研究成果相结合,共同降低测试的漏报和误报,提高渗透测试准确度。

参考文献

[1] Bau J, Bursztein E, Gupta D,et al. State of the art: automated black-box web application vulnerability testing. In: Proceedings of the 2010 IEEE Symposium on Security and Privacy, Oakland, USA, 2010. 332-345

[2] Doup'e A, Cova M, Vigna Gi. Why johnny can't pentest: an analysis of black-box web vulnerability scanners. In: Proceedings of the 7th GI International Conference on Detection of Intrusions and Malware and Vulnerability Assessment, Bonn, Germany, 2010. 111-131

[3] Antunes J, Neves N, Correia M, et al. Vulnerability discovery with attack injection. *IEEE Transactions on Software Engineering*, 2010,36(3):357-369

[4] Halford W, Choudhary S, Orso A. Improving penetration testing through static and dynamic analysis. In: Proceedings of the 2nd IEEE International Conference on Software Testing, Verification and Validation, West Sussex, UK, 2011. 195-214

[5] Antunes N, Laranjeiro N, Vieira M, et al. Effective detection of SQL/XPath injection vulnerabilities in web services. In: Proceedings of the IEEE International Conference on Services Computing, Bangalore, India, 2009. 260-267

[6] Fong E, Gaucher R, Okun V, et al. Building a test suite for web application scanners. In: Proceedings of Annual Hawaii International Conference on System Sciences, Hawaii, USA, 2008. 479-486

[7] Byers D, Shahmehri N. Unified modeling of attacks, vulnerabilities and security activities. In: Proceedings of 2010 ICSE Workshop on Software Engineering for Secure Systems, New York, USA, 2010. 36-42

[8] Wang J, Phan R C, John N, et al. Augmented attack tree modeling of SQL injection attacks. In: Proceedings of the 2nd IEEE International Conference on Information Management and Engineering, Chengdu, China, 2010. 182-186

[9] Marback A, Do H, He K, et al. Security test generation using threat trees. In: Proceedings of ICSE Workshop on Automation of Software Test, Vancouver, Canada, 2009. 62-69

[10] Kiezun A, Guo P, Jayaraman K, et al. Automatic creation of SQL injection and Cross-Site Scripting attacks. In: Proceedings of the 31st International Conference on Software Engineering, Vancouver, Canada, 2009. 199-209

[11] Li N, Xie T, Jin M Z, et al. Perturbation-based user-input-validation testing of web applications. *The Journal of Systems and Software*, 2010, 83(7): 2263-2274

[12] Shahriar H, Zulkernine M. MUSIC: Mutation-based SQL injection vulnerability checking. In: Proceedings of the 8th International Conference on Quality Software, Oxford, UK, 2008. 77-86

[13] Ruse M, Sarkar T, Basu S. Analysis and detection of SQL injection vulnerabilities via automatic test case generation of programs. In: Proceedings of 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet, Seoul, Korea, 2010. 31-37

Model-driven penetration test of the SQL injection in Web applications

Tian Wei, Xu Jing, Yang Jufeng, Zhang Ying, Liu Lei

(College of Information Technical Science, Nankai University, Tianjin 300071)

(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)

Abstract

To resolve the problem of how to generate adequate test cases to reduce the false negative in penetration testing for the SQL (structured query language) injection vulnerability, this paper proposes a novel model-driven penetration test case generation method. This method divides the penetration test case generation for the SQL injection vulnerability into two steps: 1) Building the model of penetration test case, which reveals the regularity of current SQL injection attacks to expound what test case should be used and describes them in a formal way; and 2) Instantiating the penetration test case model according to a series of coverage criteria proposed in the study to generate the test case covering more attack patterns. The experiment shows that compared with randomly enumerated test cases used in the current related work, the test cases generated by the proposed method can more effectively find the SQL injection vulnerability hidden behind the inadequate defense mechanism, which reduces the false negative and improves the test accuracy.

Key words: Web, penetration testing, structured query language (SQL) injection, attack model, vulnerability, test case