# 基于协议混合变形的Web安全模糊测试与效用评估方法

涂 玲[1] 马 跃[2] 程 诚[3] 周彦晖[1]
(西南大学计算机与信息科学学院 重庆 400715)[1]
(重庆大学软件学院 重庆 401331)[2] (海南医学院 海口 571199)[3]

**摘 要** 在Web应用安全模糊测试中,存在测试用例覆盖率低、测试效用无法得到有效验证及漏洞检测结果无法得到有效评估等问题。提出了协议变形和动态特征并行混合的测试用例生成方法,建立了按典型漏洞分类的输入特征组合规则和协议变形规则,并形成了基于污染传播策略漏洞响应数据分析和有效性验证的方法。实验表明所提方法增大了测试用例的多样性以及提高了覆盖率,降低了在网站过滤环境复杂情况下的漏洞检测的漏报率和误报率。

**关键词** 安全测试,协议变形,污染传播策略,测试有效性
**中图法分类号** TP309 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.05.025

## Hybrid Protocol Deformation Based Web Security Fuzzy Testing and Utility Evaluation Approach

TU Ling[1] MA Yue[2] CHENG Cheng[3] ZHOU Yan-hui[1]
(College of Computer and Information Science, Southwest University, Chongqing 400715, China)[1]
(School of Software Engineering, Chongqing University, Chongqing 401331, China)[2]
(Hainan Medical University, Haikou 571199, China)[3]

**Abstract** In the Web application security fuzzy testing, there are some problems such as low coverage of test cases, ineffective verification of test cases utilities and lack of quantitative evaluation of vulnerability detection results. In this paper, we proposed a method of generating dynamic features combination and protocol deformation test cases for typical Web security vulnerabilities. The rules of input feature combination and protocol deformation rules are devised, and the algorithm based on pollution propagation strategy and effectiveness validation method are established. Experiments show that the proposed method enhances the diversity and coverage of test cases, and reduces the false negative rate and false positive rate of vulnerability detection in the complex situation of web filtering environment.

**Keywords** Security testing, Protocol deformation, Pollution propagation strategy, Testing efffectiveness

随着Web应用的广泛使用,Web安全性测试在软件测试领域中广受关注。渗透测试、模糊测试技术作为软件测试研究中一种新的思路和方法,近年来已经开始被广泛地应用到Web应用测试领域中。

Web安全测试总体上可以分为静态和动态两种。静态分析技术是指在Web应用程序不运行的情况下对Web应用程序的代码进行安全性评审[1-3],以发掘Web应用程序中可能会存在的安全缺陷。Balzarotti D等人提出了基于污染流分析的静态分析方法[4]。动态检测技术首先对运行中的Web应用程序进行测试[5-6],然后根据运行结果推测Web应用程序中可能会存在的各种问题。

在实际测试中,大多数测试方法结合了静态分析和动态检测[7]的优点,这种结合方法可在一定程度上提高漏洞识别率,并有效降低误报率。潘古斌等提出了一种代码静态分析及净化单元动态检测的XSS漏洞检测方法,该方法描述了XSS漏洞所对应的源规则、净化规则和接收规则,以及净化单元动态检测算法[8]。Win W等人[9]提出了一个结合了动态分析和静态分析的SQL注入检测框架,通过和原有要执行的SQL语句进行比较分析来决定是否存在注入漏洞。此外,文献[16]着重研究了现有动态监测方法的不足,提出了一种避免所有攻击向量遍历的检测方法。

近期大多数研究集中于模型和流程环节的改进。在测试用例生成环节,Wang J等人[10]使用增广攻击树的用例模型来规范SQL注入用例的生成过程,但测试用例集合的充分性仍不能保证。文献[11]在Wang J和田伟的研究基础上,将两种模型进行有效结合,提出了一个基于有向图的SQL注入测试用例集成生成模型。文献[12]从测试用例语句代码功能方面入手,提出了一种基于模板的动态组合生成测试用例的方法,进一步提升了测试用例生成的多样性和覆盖率,但该方法未考虑Web应用的过滤机制。基于异常响应分析环节,蒋华

到稿日期:2017-02-04 返修日期:2017-04-27 本文受国家科技支撑计划项目/课题(2015BAK41B00/2015BAK41B01)资助。
涂 玲(1978—),女,硕士,实验师,主要研究方向为可信计算、智能教育,E-mail:tuling@swu.edu.cn;马 跃(1981—),硕士,讲师,主要研究方向为视觉传媒、计算机应用、图像处理,E-mail:mayue@cqu.edu.cn(通信作者);程 诚(1990—),硕士,主要研究方向为软件测试;周彦晖(1972—),男,硕士,副教授,主要研究方向为深度学习、大数据。

等人提出了一种动态污点分析与 HTTP 请求分析处理相结合的方法，该方法通过标记应用程序中的敏感数据，跟踪标记数据的流向进行污点分析[13]。Duchene F[14]采用的是一种数据流控制逆向分析。Rawat S 等人[15]采用的是模型推断和渐进模糊技术相结合的跨站攻击检测方式，摒弃了基于数据的传统模糊方法，采用模型推断技术记录程序输入状态信息的转变过程。

现有的模糊测试工具也不断增多，应用逐渐广泛，例如 APPScan，SQLmap[17]，FindBugs[18]等。

目前 Web 安全模糊测试方法仍存在不足，包括测试用例覆盖率偏低，忽略了测试效率问题，误报和漏报率较高；由于模糊测试方法的随机性，导致测试结果的可解释性差，测试验证需要手工完成。

## 1 动态规则和协议变形的混合测试用例生成方法

针对测试用例生成的问题，协议变形和动态特征并行混合方法将已有的 XSS 攻击和 SQL 注入测试用例进行拆分并分类处理，同时将结果存放在各自的功能模板库中，随后通过模板参数设定将其自动组合，从而生成大量的、多变化的、随机的测试用例。

### 1.1 协议混合变形的基本方法流程

结合动态生成测试用例的优点以及协议变形的有效性，协议混合变形方法分为两个部分：网站过滤协议分析和测试用例变形生成方法。

基于动态生成和协议变形的测试用例生成方法的思想为并发进行动态生成和协议变形，在随机生成大量测试用例的同时对测试用例进行协议变形处理，有效生成针对 Web 应用安全性检测的最优测试用例集合，如图 1 所示。

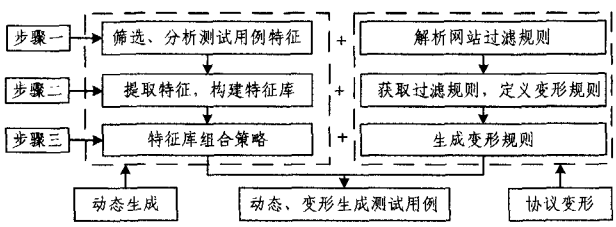


图 1 基于动态生成和协议变形的测试用例生成方法

### 1.2 过滤与变形的混合规则

目前网站常见的过滤规则的思路为针对特殊字符和特殊关键字进行过滤操作。为了对过滤机制进行测试，从而获取 Web 应用过滤机制，下面定义网站的过滤规则。

**定义 1**(过滤规则) 设测试用例代码集合 $C$ 的取值是有限的，即一定能找到一种方法 $F$ 使得 $C$ 中的用例代码被过滤。将对任何测试用例代码进行过滤的规则集合定义为 $F$，将利用集合 $F$ 中规则所能过滤掉的测试用例代码集合定义为 $Z$，可得：

$$F=\{f_1,f_2,\cdots,f_i,\cdots,f_m\} \tag{1}$$

$$Z=\{z_1,z_2,\cdots,z_j,\cdots,z_n\} \tag{2}$$

其中，$0<i<m$，$0<j<n$，$m,n\neq\infty$。

对于任何测试用例代码 $C$，如果有 $C_f\odot Z$，此类代码就会被过滤掉。即：

$$F(C_f)\rightarrow C_f^*\notin Z \tag{3}$$

针对目前网站的过滤机制进行研究，发现主要存在删除、编码、替换 3 种方式。常见的过滤方法如表 1 所列。

表 1 常见的过滤方法

| 过滤方式 | 输入字符 | 输出字符 |
|---|---|---|
| 删除 | Document. write | . write |
| 编码 | < | &.lt |
| 变换 | Document | xDocument |

为了快速找到测试用例输入时 Web 应用过滤的内容，可采用生僻的字符组合对特殊字符进行前后标记，从而获取 Web 应用对特殊字符的过滤情况，例如使用 ccllllwjp〈 〉ccllll-wjp 标记〈 〉。

测试用例的混淆反过滤技术集多种等价变形方法于一体，包括编码、大小写转换等，故针对每一类测试用例集合，都应对该集合中的部分或全部测试用例采用不同的反过滤方法进行等价变形，以绕过不同的防御策略。

**定义 2**(变形规则) 定义对任何过滤掉的测试用例代码 $C_f$ 进行的变形规则集合为 $Y$，$Y$ 为一系列对测试用例代码进行转换和重置的规则集合，所产生的测试用例变形能成功绕过过滤规则 $F$，将利用 $Y$ 中方法变形的测试用例集合定义为 $W$，$W$ 即是可绕过过滤机制的测试用例代码库。即：

$$Y=\{y_1,y_2,\cdots,y_i,\cdots,y_m\} \tag{4}$$

$$W=\{w_1,w_2,\cdots,w_j,\cdots,w_n\} \tag{5}$$

其中，$0<i<m$，$0<j<n$，$m,n\neq\infty$。

若有 $C_f\odot Z$，变形规则将其转换变为可绕过过滤机制的测试用例，即：

$$Y(C_f)\rightarrow Y_f^*\in W \tag{6}$$

假设过滤规则与变形规则一一对应，如过滤规则 $f_1$ 对应变形规则 $y_1$，则有：

$$W=(f_1\cap y_1)\cup(f_2\cap y_2)\cup\cdots\cup(f_m\cap y_m) \tag{7}$$

**定义 3**(变形混合规则) 变形规则 $Y$ 中存在 $m$ 种变形规则，变形规则可单独使用，也可混合使用。变形混合规则即从变形规则 $Y$ 中选择 1 个或多个规则同时使用，定义变形规则 $Y$ 混合使用时的规则集为 $X$，则有：

$$X_{i+1}=Y_i(X_i),Y\in\{Y_i,X_{i+1}\} \tag{8}$$

已知测试用例集合 $C$(即通过动态特征规则生成的用例库)、过滤规则集 $F$ 以及变形规则集 $Y$，在进行 XSS 或 SQL 注入漏洞挖掘时，根据过滤规则 $F$ 找到适当的变形规则 $Y_a$，变形规则集的作用在于利用一切可能的方法对测试用例进行转换和重置，使之达到绕过过滤的目的。以下列步骤完成对测试用例的变形：

第一步 遍历 $C$，取出 $C$ 中的每一个值 $C_i$；

第二步 对每一个 $C_i$ 使用恰当的变形规则中的方法 $Y_a$ 进行转换，即 $Y_a(C_i)$，以遍历完 $C$ 和 $Y_a$ 的值为变形终止。

### 1.3 Web 典型漏洞的协议混合变形分析

下面说明上述方法在典型 Web 安全漏洞的具体应用。我们在文献[19]中已经部分阐述了针对跨站脚本(XSS)漏洞的协议混合变形测试生成过程，这里主要阐述针对 SQL 注入漏洞的协议混合变形的分析和方法应用。

针对 SQL 注入测试用例的网站过滤规则和变形规则与

XSS 攻击测试用例的方法类似。通过本文 1.2 节所示的网站过滤协议分析之后，得到网站过滤机制，Web 应用通常会过滤 SQL 注入测试用例的一些常见关键字和关键字符。常见关键字为 and，exec，insert，select，delete，update，count，chr，mid，master，truncate，char，declare 等；常见关键字符为'，*，%等。

**定义 4**（SQL 注入测试用例特征）　对于任何 SQL 攻击的源代码 $C_{sql}$，其代码特征 $T_{sql}=(H,B,M)$，包含 3 个特征参数：闭合特征 $H$、逻辑特征 $B$、注释特征 $M$。

SQL 注入测试用例特征库如图 2 所示。

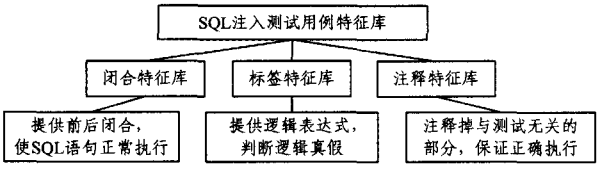


图 2　SQL 注入测试用例特征库

以 3 个 SQL 攻击测试用例为例，将测试用例特征提取至所属特征库，如表 2 所列。

表 2　SQL 测试用例特征提取表

| 编号 | 测试用例示例 | 闭合标签库 | 标签特征库 | 注释特征库 |
|---|---|---|---|---|
| 1 | 1'or1=1−− | 1' | or1=1−− | −− |
| 2 | 1'and 2>3−− | 1' | and 2>3 | −− |
| 3 | abc" or'bbc' between' abc 'and'cbc'/* | abc" | or'bbc' between 'abc'and'cbc' | /* |

SQL 注入测试用例特征库可设计为 3 个部分，分别为特征编号 $N$、特征 $F$、参数类型 $T$，其中特征编号和参数类型可无限增加。SQL 测试用例特征库设计如表 3 所列。

表 3　SQL 测试用例特征库设计表

| 特征库设计 | 闭合特征库 | 逻辑特征库 | 注释特征库 |
|---|---|---|---|
| 特征编号 $N$ | $N_h$ | $N_b$ | $N_m$ |
| 特征 $F$ | $F_h$ | $F_b$ | $F_m$ |
| 参数类型 $T$ | $T_h$ | $T_b$ | $T_m$ |
| 对应特征函数 $f(x)$ | $f(h)$ | $f(b)$ | $f(m)$ |

针对 Web 过滤 SQL 注入关键字符和关键字的情况，SQL 测试用例可以通过变形绕过 Web 过滤以实现 SQL 注入。SQL 注入测试用例的常见变形方法如表 4 所列。

表 4　SQL 注入测试用例的变形方法

| 变形方法 | 描述 | 示例 |
|---|---|---|
| 大小写混合 | 针对常见的大小写字符的过滤方式 | AnD 代替 AND 和 and。 |
| UNICODE 编码 | 对字符进行 unicode 编码。 | 输入"and%201%3D1"与输入"and 1=1"等价 |
| 添加注释 | 1. 添加注释来冒充空格<br>2. 添加注释来分割敏感字 | 1. select/* aa */username，其中/* aa */来冒充空格<br>2. SEL/* aa */ECT 注释分割 |
| ASCII 码 | 使用 ASCII 码改变测试用例中部分或全部字符 | A −>chr(65) |
| 空格法 | 使用空格添加，用 Tab 代替空格等 | |
| 上下文无关文法 | 过滤机制不会递归过滤 | 若网站过滤 and，可将其变形为 Aandnandd，当过滤结束后，剩下的字符正好组成一个新的 and。 |

本文以逻辑特征库为例，如表 5 所列。所有特征使用解析算法 $A_{sql}$，参数解析后进行内容替换。

表 5　逻辑特征库

| $N_b$ | $F_b$ | $T_b$ | | | | |
|---|---|---|---|---|---|---|
| | | {1} | {2} | {3} | {4} | … |
| 1 | or{1}={1}{2} | 数字 | 注释特征库 | | | … |
| 2 | or{1}+{2}>={1}{3} | 数字 | 数字 | 注释特征库 | | … |
| … | … | … | … | … | … | … |

SQL 注入测试用例函数生成算法与 XSS 攻击测试用例生成方法类似：

选择闭合特征库第 1 条特征{1}"{2}，其中参数{1}是数字，参数{2}为逻辑式特征库。逻辑式特征库第 1 特征如表 5 所列，信合注释特征库第 1 条特征"−−"生成 SQL 测试用例。

SQL 注入测试用例函数生成算法如下：

1）生成闭合特征库函数 $f(h)$：$f(h)$(1，数字，逻辑特征库，$A_{sql}$)。

2）将逻辑特征库替换为 $f(b)$：$f(h)$(1，数字，$f(b)$(1，数字，注释模板库，$A_{sql}$)，$A_{sql}$)。

3）将上述注释特征库替换为 $f(m)$：$f(h)$(1，数字，$f(b)$(1，数字，$f(m)$(1，$A_{sql}$)，$A_{sql}$)，$A_{sql}$)。

以 Web 应用针对书写字符的过滤方式为例，过滤规则示例如表 6 所列。

表 6　字符书写方式过滤规则描述表

| 过滤规则 $F$ | 描述 |
|---|---|
| $f_{21}$ | 对所有的小写字符或大写字符测试用例输入进行过滤 |
| $f_{22}$ | 对关键字进行过滤 |

每个过滤规则对应相应的反过滤的变形规则，即出现 1 对 1，即 1 对 $n$ 的对应关系。针对上述字符书写方式过滤规则对应的变形规则如表 7 所列。

表 7　字符书写方式变形规则描述表

| 变形规则 $Y$ | 描述 |
|---|---|
| $y_{21}$ | 混合输入大小写字符，以绕过 $f_{21}$，$f_{22}$ |
| $y_{22}$ | 进行关键字符的 ASCII 码编写，以绕过 $f_{21}$，$f_{22}$ |
| $y_{23}$ | 进行关键字的上下文无关法的变形，以绕过 $f_{21}$，$f_{22}$ |

由表 7 和表 8 可得 $f_{2x}$ 和 $y_{2x}$ 的关系为：

$$W=(f_{21}\cap y_{21})\cup(f_{21}\cap y_{22})\cup(f_{21}\cap y_{23})\cup(f_{22}\cap y_{21})\cup(f_{22}\cap y_{22})\cup(f_{22}\cap y_{23}) \quad (9)$$

## 2　效用评估和分析

污染传播策略研究[20]围绕污染数据解决 3 个问题：如何标记污染数据源，如何记录污染数据传播过程，如何检测污染数据。其基本流程为：标记→跟踪和记录→分析和检测。

在污染传播策略的基础上，结合 SQL 注入和 XSS 的攻击特点，以模糊测试之后得出的 Web 应用存在的漏洞注入点数据为污染数据，通过污点传播过程后分析漏洞信息验证其是否存在，从而有效减少误报情况，方便 Web 开发人员对异常的寻找和修复。

### 2.1　基于污染传播策略的漏洞数据分析

将污染传播策略三步骤与 Web 应用漏洞检测相结合，建

立基于污染传播策略的漏洞检测方法的流程如下：

1)污染传播策略包括以下行为：污染数据输入、污染数据传播、污染数据净化、污染数据到达接收点及其他部分。形式化表示如下：

$F:G\rightarrow\{\text{input},\text{spread},\text{clean},\text{accept},\text{common}\}$

2)污染传播策略包括以下规则：输入规则、传播规则、净化规则、接收规则。其形式化表示如下。

输入规则$(IR)=\{I_j\mid j\in[1,m]\}$：定义污染数据到 Web 应用的注入点位置。

传播规则$(SR)=\{S_j\mid j\in[1,m]\}$：定义污染数据在 Web 应用中如何进行传播。

净化规则$(CR)=\{C_j\mid j\in[1,m]\}$：定义 Web 应用将用户输入进行净化以防止污染传播的方法。

接收规则$(AR)=\{A_j\mid j\in[1,m]\}$：定义污染数据到达 Web 应用的漏洞接收点。

3)对于上述污染传播行为和污染传播规则之间的对应关系可定义为：

$$R:G\rightarrow IR\cup SR\cup CR\cup AR\begin{cases}I_j\in IR, & \text{if } F(G)=input\\ S_j\in SR, & \text{if } F(G)=spread\\ C_j\in CR, & \text{if } F(G)=clean\\ A_j\in AR, & \text{if } F(G)=accept\\ \text{others}\end{cases}$$

4)根据上述关系，根据污染传播策略对 Web 漏洞信息进行分析的方法为：

$\exists\, path=\{g_1,g_2,\cdots,g_j,g_m\}$

$F(g_1)=input$

$F(g_m)=accept$

$\forall_{1<j<m}j:spread(g_j,g_{j+1})$

$\forall_{1<j<m}j:F(g_j)\neq clean$

上述表示当存在一条从污染数据注入点 $g_1$ 到 Web 应用漏洞接收点 $g_m$ 的路径，其中任意 $g_{j+1}$ 均是通过 $g_j$ 调用 *spread* 得到的，且该污染数据在到达接收点前未被进行净化操作，可得该 Web 应用的漏洞信息分析结果为真实存在安全漏洞。

### 2.2 数据分析和验证步骤

完整的测试验证流程包括污染数据的标记、传播和检测。

步骤 1 标记污染数据。根据模糊测试后的漏洞检测结果，保存测试用例和注入点，将外部输入数据标记为污染数据。

步骤 2 记录污染数据。记录污染数据的传播过程，即根据定义的污染传播规则，对目的数据进行污染和对污染数据进行净化，并记录下污染传播的相关路径。

步骤 3 检测与分析污染数据。若污染数据抵达接收点，则需检测污染数据是否改变了 SQL 语句的语法结构或者是否构成了要输出的 Html 语句，并判断是否存在污染数据。若存在，则该 Web 应用真实存在 SQL 注入或 XSS 漏洞。

在具体的污染数据跟踪和记录中，若发现 SQL 语句中或 Html 语句中改变了 SQL 语句的语法结构和构成了要输出的 Html 语句，则确定该处为 SQL 注入或 XSS 漏洞攻击点，调用污点路径追踪算法得到污点数据来源及在程序中的传播路径。漏洞验证时，当存在一条从污染数据入口到污染操作的路径，且这个数据在使用之前没有被净化或没有被完全净化，那么该 Web 应用就存在漏洞。反之，会发现有些数据已经被净化，说明该 Web 应用并不存在此类异常。

在图 3 所示的污点传播实例中，共有 5 条路径：

1)$IR\rightarrow1\rightarrow3$；2)$IR\rightarrow1\rightarrow2\rightarrow AR$；3)$IR\rightarrow4\rightarrow5$；

4)$IR\rightarrow4\rightarrow6$；5)$IR\rightarrow4\rightarrow7\rightarrow AR$

*IR* 为污染数据，图 3 所示路径只有 2)、5)到达了 Web 漏洞数据的接收点 *AR*，说明经历过 1+2，4+7 操作，污染数据仍可以到达注入点，即该漏洞是真实存在的，漏洞数据没有出现误报，且其传播轨迹为 $IR\rightarrow1\rightarrow2\rightarrow AR$，$IR\rightarrow4\rightarrow7\rightarrow AR$，而没有到达接收点的污染数据则在污染传播过程中已被净化为非污染数据。

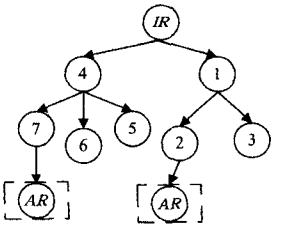


图 3 Web 应用漏洞数据分析过程简图

## 3 实验结果

本文选取来源于 GitHub 中的 3 个 Web 应用程序为实验数据，如表 8 所列。

表 8 Web 应用信息表

| Web 应用程序 | XSS 漏洞个数 | SQL 注入漏洞个数 |
|---|---|---|
| Repair Station | 86 | 105 |
| Bookstore | 30 | 24 |
| Office Talk | 5 | 6 |

本文算法实现基于比较通用的开源工具 SQLmap 用于检测 SQL 注入攻击，Xenotix 用于检测 XSS 攻击。分别使用已经生成好的改进工具 SQLmap3，Xentiox3，以及现有检测工具 FindBugs 对表 8 中的 Web 应用进行扫描分析，记录漏洞检测结果。

根据以上方案，可得各工具检测 XSS 漏洞和 SQL 漏洞的结果，该结果以误报情况进行展示，如表 9 和表 10 所列。

表 9 SQL 注入漏洞误报结果

| Web 应用 | SQL 注入漏洞个数 | FindBugs 误报率/% | SQLmap 3 误报率/% |
|---|---|---|---|
| Repair Station | 105 | 8.6 | 4.8 |
| Bookstore | 24 | 12.5 | 4.2 |
| Office Talk | 6 | 0 | 0 |

表 10 XSS 注入漏洞误报结果

| Web 应用 | XSS 注入漏洞个数 | FindBugs 误报率/% | Xenotix3 误报率/% |
|---|---|---|---|
| Repair Station | 86 | 12.8 | 6.9 |
| Bookstore | 30 | 13.3 | 6.7 |
| Office Talk | 5 | 0 | 0 |

实验表明，本文提出的方法的误报率要低于 FindBugs，在测试用例的生成上更具有针对性和较好的覆盖率。

**结束语** 本文提出了一种新的模糊测试用例生成方法，该方法将测试模板规则的动态生成和网站过滤协议分析变形

相结合，寻找攻击测试用例特征组合优化的快速搜索方法，以在不同过滤机制下能够快速组成最有效的攻击测试用例，提高了测试用例生成的智能化、多样性以及针对性，能够减少 Web 安全检测过程中的漏报情况。针对漏洞分析过程中存在的漏洞响应无法得到有效验证的问题，本文提出了一种基于污染传播策略的漏洞数据分析方法，经过污染检测分析和验证，减少测试的误报率。本文通过对比实验证明该方法可行且有效。

## 参 考 文 献


[1] LUO Y X. Static Code Analysis and Defense for Software Security Defects [D]. Bejing: Institute of Software, Chinese Academy of Sciences, 2007. (in Chinese)
罗宇翔. 面向软件安全缺陷的静态代码分析及防御[D]. 北京：中国科学院软件研究所，2007.

[2] KULENOVIC M, DONKO D. A survey of static code analysis methods for security vulnerabilities detection[C]// International Convention on Information and Communication Technology, Electronics and Microelectronics. 2014: 1381-1386.

[3] WASSERMANN G, SU Z. Static detection of cross-site scripting vulnerabilities[C]// ACM/IEEE 30th International Conference on Software Engineering, 2008 (ICSE'08). IEEE, 2008: 171-180.

[4] BALZAROTTI D, COVA M, FELMETSGER V, et al. Saner: Composing static and dynamic analysis to validate sanitization in web applications[C]// IEEE Symposium on Security and Privacy, 2008(SP 2008). IEEE, 2008: 387-401.

[5] PIETRASZEK T, BERGHE C V. Defending against injection attacks through context-sensitive string evaluation[C]// Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2005: 124-145.

[6] HALFOND W G J, ORSO A, MANOLIOS P. WASP: Protecting Web applications using positive tainting and syntax-aware evaluation[J]. IEEE Transactions on Software Engineering, 2008, 34(1): 65-81.

[7] BALZAROTTI D, COVA M, FELMETSGER V, et al. Saner: Composing static and dynamic analysis to validate sanitization in web applications[C]// IEEE Symposium on Security and Privacy, 2008(SP 2008). IEEE, 2008: 387-401.

[8] PAN G B, ZHOU Y H. Finding XSS Vulnerabilities Based on Static Analysis and Dynamic Testing [J]. Computer Science, 2012, 39(B06): 51-53. (in Chinese)
潘古兵，周彦晖. 基于静态分析和动态检测的 XSS 漏洞发现[J]. 计算机科学，2012，39(B06)：51-53.

[9] WIN W, HTUN H H. A simple and efficient framework for detection of sql injection attack[J]. IJCCER, 2013, 1(2): 26-30.

[10] WANG J, PHAN R C W, WHITELY J N, et al. Augmented attack tree modeling of SQL injection attacks[C]// 2010 The 2nd IEEE International Conference on Information Management and Engineering (ICIME). IEEE, 2010: 182-186.

[11] WEI C T. Research on Key Technology of SQL Injection and XSS Attack Automated Detection[D]. Beijing: Beijing University of Posts and Telecommunications, 2015. (in Chinese)
韦存堂. SQL 注入与 XSS 攻击自动化检测关键技术研究[D]. 北京：北京邮电大学，2015.

[12] LI Z, XU X, LIAO L J, et al. Using Templates Combination to Generate Testing Vectors Dynamically in Detecting Web Applications Vulnerabilities[J]. Application Research of Computers, 2015, 32(10): 3004-3008. (in Chinese)
李政，许欣，廖乐健，等. 使用模板组合动态生成测试用例的 Web 应用漏洞发掘方法[J]. 计算机应用研究，2015，32(10)：3004-3008.

[13] JIANG H, XU Z Y, WANG X. XSS Attack Defense Method Based on Behavior [J]. Computer Engineering and Design, 2014, 35(6): 1911-1914. (in Chinese)
蒋华，徐中原，王鑫. 基于行为的 XSS 攻击防范方法[J]. 计算机工程与设计，2014，35(6)：1911-1914.

[14] DUCHENE F, RAWAT S, RICHIER J L, et al. LigRE: Reverse-engineering of control and data flow models for black-box XSS detection[C]// 2013 20th Working Conference on Reverse Engineering (WCRE). IEEE, 2013: 252-261.

[15] DUCHENE F, GROZ R, RAWAT S, et al. XSS vulnerability detection using model inference assisted evolutionary fuzzing[C]// SECTEST 2012-3rd International Workshop on Security Testing (affiliated with ICST). IEEE Computer Society, 2012: 815-817.

[16] CAO L B, CAO T J. Research on Cross-site Scripting Vulnerability Detection Method Based on Dynamic Testing [J]. Computer Application and Software, 2015, 32(8): 272-275. (in Chinese)
曹黎波，曹天杰. 基于动态测试的 XSS 漏洞检测方法研究[J]. 计算机应用与软件，2015，32(8)：272-275.

[17] WANG Q, BAI M. Research about Using Tool of SqlMap GET injection and Principle Analyzing on Linux Platform [J]. Computer Security, 2013(6): 74-76. (in Chinese)
王琦，白淼. 渗透工具 SqlMap GET 注入使用及原理分析[J]. 计算机安全，2013(6)：73-76.

[18] LV Z Y, HUANG S, HUI Z W. Improvement of Defect Detection Mode for Function Return Value Based on FindBugs[J]. Journal of PLA University of Science and Technology (Nature Science Edition), 2015, 16(6): 518-523. (in Chinese)
吕增援，黄松，惠战伟. 基于 FindBugs 的函数返回值缺陷检测模式的改进[J]. 解放军理工大学学报(自然科学版)，2015，16(6)：518-523.

[19] CHENG C, ZHOU Y H. Finding XSS Vulnerabilities Based on Fuzzing Test and Genetic Algorithm [J]. Computer Science, 2016, 43(6A): 328-333. (in Chinese)
程诚，周彦晖. 基于模糊测试和遗传算法的 XSS 漏洞挖掘[J]. 计算机科学，2016，43(6A)：328-333.

[20] TANG H P, HUANG S G, ZHANG L. Detection Algorithm for Leak Detection in Pollution Propagation Analysis [J]. Journal of Chinese Computer System, 2010(11): 2227-2230. (in Chinese)
唐和平，黄曙光，张亮. 污染传播分析的漏洞利用检测算法[J]. 小型微型计算机系统，2010(11)：2227-2230.

[21] LIU L C, FAN W J. From the Viewpoint of Software Software Process to Approach the Reusable Requirement Analysis-Passingly Review Analysis Between CMM and ISO9000[J]. Journal of Chongqing University of Technology(Natural Science), 2012, 26(1): 53-60. (in Chinese)
刘兆存，范玮佳. 软件过程中可复用需求分析[J]. 重庆理工大学学报(自然科学版)，2012，26(1)：53-60.