文章编号：2095-2163(2019)04-0177-03 中图分类号：TP311.1 文献标志码：A

Web 应用漏洞扫描工具的研究与设计

周康成[1,2]

（1 东华大学 计算机科学与技术学院，上海 201620；2 上海市计算机软件评测重点实验室，上海 201112）

摘 要：随着互联网的快速发展，网络安全显得格外重要。Web 应用漏洞扫描工具能够有效地检测出 Web 应用漏洞，防止黑客利用漏洞。但目前的 Web 应用漏洞扫描工具大多是单机应用，扫描效率不高。本文设计一种新的 Web 应用漏洞扫描系统架构，将以往的单机应用分解为多个模块，每个模块部署在不同的计算机上，利用多台计算机的计算能力来提高扫描效率。

关键词：Web 漏洞；系统架构；网络安全

Research and design of Web application vulnerability scanning tool

ZHOU Kangcheng[1,2]

（1 School of Computer Science and Technology, Donghua University, Shanghai 201620, China;
2 Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 201112, Cina）

【Abstract】 With the rapid development of Internet, network security is particularly important. Web application vulnerability scanning tool can effectively detect Web application vulnerabilities and prevent hackers from exploiting vulnerabilities. However, most of the current Web application vulnerability scanning tools are single-machine applications, and scanning efficiency is not high. This paper designs a new Web application vulnerability scanning system architecture, which decomposes the previous single-machine application into multiple modules. Each module is deployed on different computers, and the computing power of multiple computers is used to improve scanning efficiency.

【Key words】 Web vulnerability; system architecture; network security

0 引 言

Web 应用漏洞中最常见且危害较大的 2 类安全漏洞是 SQL 注入漏洞和 XSS 跨站脚本漏洞。目前的一些开源扫描工具主要是针对这 2 类安全漏洞进行安全扫描。在研究与设计 Web 应用漏洞扫描工具的方向上，许多学者已经做了大量研究，研究的主要目的就是为了解决 Web 应用漏洞扫描工具的扫描效率不高、漏洞误报率高和扫描覆盖不完全等问题。文献[1]考虑到了 HTML5 的一些新的特性，在系统设计上，对这些新特性提供了相应的支持。文献[2]通过在工具中集成 Webkit 来支持解析 JavaScript 脚本，解决网页动态渲染的问题，并利用分布式技术来提高扫描效率。文献[3]提出了基于爬虫和特征识别的漏洞检测模型，目的是为了解决注入点获取不准确等问题。文献[4]采用了 Fuzzing 测试技术，并且在系统实现上考虑到了反爬虫机制。本文在研究总结前人工作的基础上，提出了一种新的 Web 应用漏洞扫描系统架构，提高扫描效率。深入分析了各个模块之间的依赖关系，利用中间件将单机应用进行分解，将分解后的各个模块部署在不同的计算机上，利用多台计算机的计算能力来解决单机系统扫描效率不高的问题。

1 SQL 注入漏洞

SQL 注入漏洞是最常见 Web 应用漏洞之一，也是危害较大的一种 Web 应用漏洞。

1.1 产生原因

Web 应用分为前端和后端，后端数据通过前端展示给用户，后端接收前端发送过来的数据，进行处理、查询，然后将结果返回到前端。一般情况下，后端如果要获取相应的数据，就需要将用户输入的数据拼接成 SQL 语句，在数据库中执行。如果在这个过程中，前端提交了包含恶意数据的信息，后端对用户的输入信息不做任何检测或者检测不彻底，就将用户数据拼接到 SQL 语句中，那么就会造成 SQL 注入攻击。

1.2 检测方法

SQL 注入漏洞检测一般是采用黑盒测试的办法，通过构造各种各样的 HTTP 请求包，这些 HTTP 请求包中包含的参数在 SQL 语句中具有特殊意义，当后端接收到这类 HTTP 请求包，并作出响应，根据

作者简介：周康成(1994-)，男，硕士研究生，主要研究方向：Web 安全、网络安全。

收稿日期：2019-04-10

后端的响应数据来判断该 Web 站点是否存在 SQL 注入漏洞。一般的检测步骤如下:

(1)提取 URL 链接中的参数信息。

(2)将参数值替换为精心设计的具有特殊意义的字符串。

(3)利用修改后的参数值构造 HTTP 请求数据包,并发送请求至后端。

(4)接收后端返回的响应数据,并分析响应,判断是否存在漏洞。

(5)重复步骤(1)。

2 XSS 漏洞

2.1 产生原因与分类

XSS 跨站脚本漏洞根据不同的产生原因,主要分为 3 类:基于 DOM 的跨站脚本漏洞、反射型 XSS 跨站脚本漏洞和存储型 XSS 漏洞。

(1)基于 DOM 的跨站脚本漏洞。是一种基于 DOM 文档对象模型的漏洞,恶意的 JavaScript 脚本在浏览器中被执行,获取和修改 DOM 中的数据。

(2)反射型 XSS 跨站脚本漏洞。后端对用户提交的数据没有做过滤操作或过滤不完全,就直接将数据提交给其它用户,而这些数据中包含了恶意的 JavaScript 脚本,从而产生反射型 XSS 漏洞。

(3)存储型 XSS 漏洞。这种类型的 XSS 漏洞危害高于前两者,而且更加难以被发现,后端对用户提交的数据没有做过滤或过滤不完全,就将这些数据存储在数据库中,当其它用户访问站点,获取这些数据时,就会造成存储型 XSS 漏洞。

2.2 检测方法

XSS 漏洞检测方法类似于 SQL 注入漏洞检测方法,均是采用黑盒测试方法,通过构造 HTTP 请求包,发送到后端,分析后端返回的响应数据来判断是否存在 XSS 漏洞。XSS 漏洞检测的一般步骤如下:

(1)提取页面和 URL 链接中的注入点。

(2)将参数值替换为精心构造的字符串,这些字符串中包含 JavaScript 脚本。

(3)构造 HTTP 请求包,并发送至后端。

(4)接收后端的响应数据,根据响应数据判断是否存在 XSS 跨站脚本漏洞。

(5)重复步骤(1)。

3 扫描系统架构设计与分析

Web 应用漏洞扫描系统一般分为爬虫阶段和检测阶段。通过网络爬虫获取目标站点所有的 URL 链接,并分析提取出 URL 链接和 Web 页面中的注入点,然后在检测阶段中,利用精心构造好的字符串去替换参数值,构造 HTTP 请求包,提交给后端,根据后端返回的响应数据分析判断是否存在 Web 漏洞。在深入分析了这 2 个阶段之后发现,爬虫阶段与检测阶段具有较弱的依赖关系,检测阶段只需要给出待检测的 URL 链接即可,爬虫阶段也不依赖于检测阶段,而目前已有的 Web 应用漏洞扫描工具将爬虫模块与检测模块耦合在一个单体应用中,单机系统的计算能力有限,无法大幅度提高系统扫描效率。

针对上面提出的问题,本文设计实现了一种新的 Web 应用漏洞扫描系统,该系统利用爬虫模块与检测模块不具有强依赖关系,采用消息中间件将爬虫模块与检测模块分离,爬虫模块与检测模块可分别部署在不同的计算机上,爬虫模块负责爬取站点所有的 URL 链接,并将可能存在注入点的 URL 链接的相关信息提交到消息中间件中;检测模块开启一个线程监听消息中间件,当消息中间件中存在待扫描的 URL 链接时,会主动从消息中间件中拉取 URL 链接的相关信息,并放入本地的阻塞队列中。作为一个扫描任务,阻塞队列中任务由线程池中的线程去完成。系统整体的架构如图 1 所示。

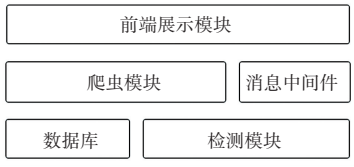

图 1 系统架构图

Fig. 1 System architecture diagram

整个系统采用 B/S 架构,浏览器端用于接收用户输入,提交扫描任务,并实时向用户展示最新的扫描情况。后端主要分为爬虫模块、消息中间件以及检测模块,可分别部署在 3 台不同的计算机上。

3.1 爬虫模块设计与实现

爬虫模块的主要功能是负责爬取 Web 站点所有的 URL 链接,并将可能存在注入点的 URL 链接提交至消息中间件中。其部分核心代码如下。

```
SimplePageProcessor processor=new SimeplePage
Processor();
Spider spider=Spider.create(processor);
spider.run();
public void send(String rootUrl,String url){
if(hasInjectionPoint(url)){
    rabbitTemplate.convertAndSend(rootUrl,url);
```

```
    }
    }
```

3.2　检测模块的设计与实现

本文设计的 Web 应用漏洞扫描系统主要对 SQL 注入漏洞和 XSS 跨站脚本漏洞进行扫描。

检测模块会开启一个线程监听消息中间件，当消息中间件中存在待扫描的 URL 时，就会主动从消息中间件中获取该 URL 及其相关信息，并封装成 DetectTask 对象，DetectTask 对象实现了 Runnable 接口，最后将该对象提交的线程池的阻塞队列中，由线程池中的线程去执行检测功能，部分核心代码如下。

```
public void run(){
while(true){
    String message =rabbitTemplate.get(QUEUE
                    _NAME);
    DetectTask task=new DetectTask(message);
    service.excute(task);
}
}
```

4　系统测试

本文设计的系统测试方案分为 2 大部分，第一部分为功能测试，主要目的是为了验证各个模块能够完成预期的功能；第二部分为性能测试，测试系统在模块分离前后的扫描时间。

4.1　功能测试

4.1.1　前端展示模块

前端展示模块的预期效果：并能正常将用户输入的扫描任务提交至后端，能实时展示后端扫描情况。

测试结果：前端成功将任务提交至后端，并实时展示后端扫描情况，测试通过。

4.1.2　爬虫模块

爬虫模块的预期效果：能够对目前站点的 URL 链接进行爬取；能够与消息中间件建立通信，并提交 URL 链接及相关信息至消息中间件。

测试结果：能够正常执行爬虫功能，并且将待扫描的 URL 链接成功提交至消息中间件中，测试通过。

4.1.3　检测模块

检测模块的预期效果：能够检测出部分 SQL 注入漏洞和 XSS 跨站脚本漏洞；能够与消息中间件建立通信，从而获取待扫描的 URL 链接。

测试结果：能够从消息中间件中获取待扫描的 URL 链接，并完成检测功能，测试通过。

4.2　性能测试

主要测试模块分离对系统扫描时间的影响。对同一 Web 站点，分别采用单体应用与模块分离后的系统进行扫描。

测试机器选择 2 个 4G 内存，第八代 I5 处理器，消息中间件 RabbitMQ 部署在 2G 内存，单核 CPU 上。

为了进行测试，在本地部署了一个 Web 站点，用于扫描。测试结果见表 1。

表 1　测试结果

Tab. 1　Test result

	扫描时间/s
系统分离前	362
系统分离后	218

5　结束语

本文研究了 SQL 注入漏洞和 XSS 漏洞的产生原因和检测方法，分析了目前 Web 应用漏洞扫描工具存在的不足，针对存在的问题，设计了一种新的 Web 应用漏洞扫描系统。通过消息中间件将单体应用中的 2 个模块进行分离，利用多台计算机的计算能力提高扫描效率，突破单体扫描系统的局限性。

参考文献

[1] 吴柳. 基于 HTML5 的跨站脚本攻击检测技术研究与实现[D]. 北京：北京邮电大学，2016.

[2] 孙晓飞. Web 应用漏洞分析与检测的研究[D]. 北京：北京邮电大学，2016.

[3] 姬硕. 基于爬虫与分布式技术的 Web 应用漏洞扫描工具的研究与设计[D]. 北京：北京邮电大学，2016.

[4] 路艳华. Web 应用漏洞检测系统研究与设计[D]. 西安：西安电子科技大学，2014.