**本章将分析并设计主要针对这两种漏洞的Web应用漏洞检测系统。**

3.1 系统总体需求分析

随着网络的普及率越来越高，网络安全已经成为一个不可忽视的问题。互联网中各式各样的站点都会遭受不同程度的攻击，导致网站的用户和运营商遭受不同程度的损失。目前，国内外都有一些用于检测Web应用漏洞的工具，但大部分都存在一定的缺点：漏报率和误报率高，扫描针对性差，测试脚本比较简单，检测正确不高。另外，国内有一些专门用于检测SQL注入和XSS漏洞的工具，但这些工具往往重视入侵功能，而忽视了检测功能，因此容易出现误报，检测不全。所以，一个使用简单、功能强大的Web应用漏洞扫描工具是很有需求的，尽可能多的检测出漏洞，以便尽早地修复目标站点存在的漏洞。

需求分析是软件开发的第一步，软件的设计与实现依赖于需求分析，直接影响到软件的功能情况。所以，首先必须做好良好的需求分析，这样在开发阶段就不必做过多的修改和重复性工作，提高软件的开发效率。

设计Web应用漏洞检测系统的主要目的是自动化地检测出Web应用程序中存在的漏洞，并且具有较高的检测效率和检测正确率，扫描结束后能够对结果生成报表，为开发人员和程序维护人员提供参考。

漏洞检测技术的本质是模拟黑客攻击的过程，向服务器发送包含攻击脚本的HTTP请求，然后分析响应信息中是否包含有漏洞特征的数据，如果包含特征数据，说明该请求存在漏洞，如果不包含就说明不存在漏洞。检测不同的漏洞需要采取不同的测试脚本量，测试脚本的丰富程度直接影响漏洞检测的准确率，但并不意味测试脚本越多越好，采取过多的攻击向量也会影响漏洞的检测效率。

本文设计的Web应用漏洞检测系统旨在完成以下需求：

(1) 系统的界面友好，操作简单。对于用户来说，操作越简单越好

(2) 能够检测出常见的Web应用程序的漏洞，包括SQL注入漏洞、XSS漏洞、目录遍历等漏洞，并具有较高的准确率。

(3) 用户可以选择默认配置来扫描，也可以根据实际需求配置相应的参数。

(4) 能够为扫描结果生成报表，以便于保存检测结果。

3.2 功能需求分析

根据实际需求，Web应用漏洞检测系统主要应该包括以下几个功能：信息收集、网络爬虫、漏洞扫描、报告生成、系统配置与扫描控制。系统的功能结构如图3-1所示。

图3-1 系统功能结构

Figure 3-1 Function Architecture of System

(1) 信息收集

信息收集的功能是获取目标服务器的基本信息，包括目标站点的操作系统类型、服务器的类型和版本、IP地址和端口号。按大类分，常见的服务器操作系统有windows和linux，他们都可以作为Web服务器，但是他们之间存在着很多的区别，在linux服务器中，通常有一些固定的文件和目录，例如：系统的敏感信息位于etc/passwd文件中，Web应用程序的目录往往位于var/www/中，而在windows系统中，Web应用程序的目录则没有固定的目录，由用户根据需求自定义。能够提供Web服务的服务器有很多，常见的有IIS、Apache、Nginx、Tomcat，不同类型的服务器是不一样的，也存在不同的漏洞，服务器类型相同版本不同之间也存在差别，不同版本的服务器存在不同的问题，这些信息对于漏洞检测都发挥着必不可少的作用。信息收集的用例图如图3-2所示。

图3-2 信息收集用例图

Figure 3-2 Use Case Diagram of Information Collection

(2) 网络爬虫

网络爬虫是漏洞扫描器中最重要的功能之一，漏洞的检测依赖于网络爬虫，因为漏洞的检测对象主要是通过网络爬虫所获取的URL和表单内容，为后续的漏洞检测提供服务。要求网络爬虫程序能够爬取整个站点的目录结构，下载必要的网页，并从页面中提取有效的链接和表单内容。由于有些链接是无效的，比如第三方站点的链接，对这些链接做检测是无意义的，并且无效的链接会影响爬虫效率，所以要求爬虫程序能够过滤无效的链接。另外，同一个链接或者表单可能在多个网页中出现，对它们进行重复检测是无意义的，所以需要爬虫程序能够有效的去重。为了控制爬虫深度，避免陷入一条爬行路径，陷入死循环，本系统采取宽度优先的爬虫策略。网络爬虫的用例图如图3-3所示。

图3-3 网络爬虫用例图

Figure 3-3 Use Case Diagram of Crawler

(3) 漏洞检测

漏洞检测功能是Web应用漏洞检测系统最本质的功能，希望通过扫描检测出目标站点存在的Web应用漏洞，本系统涉及的漏洞主要包括SQL漏洞、XSS漏洞。系统会根据不同类型的漏洞，调用相应的漏洞检测模块。检测的基本原理是模拟黑客攻击的过程，向服务器发送包含特定测试脚本的HTTP请求，然后通过分析响应包的信息来判断其是否存在漏洞。

图3-4 漏洞检测的用例图

Figure 3-4 Use Case Diagram of Vulnerability Detection

另外，要求漏洞检测过程中能够尽可能多的发现更多的漏洞的，并且具有较低的误报率。

(4) 报告生成

报告生成的功能主要是把漏洞结果保存到本地，这样做的好处是有方面：一方面是是可以保存扫描结果以便于日后分析使用，另一方面是可以将扫描结果发送给相关的开发人员或者维护人员，便于他们及时修复漏洞。报告中需要包括以下基本信息：检测时间、服务器信息、漏洞数量统计、漏洞详情。报告的用例图如图3-5所示。

图3-5 报告的用例图

Figure 3-5 Use Case Diagram of Report

(5) 系统配置

对于大多数用户来说，系统的默认配置就能满足他们的要求。为了满足不同用户的个性化需求，用户可以对一些参数进行设置，主要包括：扫描速度、漏洞检测模块、爬虫深度。另外，用户启动扫描任务后，可以根据实际需求暂停或者结束扫描任务。系统配置的用例图如图3-6所示。

图3-6 系统配置的用例图

Figure 3-6 Use Case Diagram of System Configuration

3.3 性能需求分析

本系统除了要满足以上的功能需求外，还需满足一定的性能需求。本文的性能需求主要从以下四个方面入手：扫描速度、检测正确率、可扩展性和系统稳定性。

(1) 检测速度

除了有良好的功能需求外，漏洞检测速度也是Web应用漏洞检测系统应该考虑的，因为在有些情况下，目标站点的内容很多，上面有大量的链接和表单，如果检测速度很慢，那将耗费大量的时间，所以要求扫描器具有较快的扫描速度。而扫描速度又主要取决于爬虫的速度和漏洞检测的速度，因此，爬虫策略的设计显得尤为重要，要求系统具有较好的爬虫策略和是漏洞检测方法。

(2) 检测正确率

开发Web应用漏洞检测系统的目的是尽可能多的检测出目标站点中存在的漏洞，所以要求Web应用漏洞检测系统具有较高的检测正确率，而误报和漏洞情况较少，主要依赖于丰富的测试脚本和完善的漏洞库。

(3) 可扩展性

不管对是什么软件，都是需要不断更新和维护的，不断改进和完善不足的地方，修复存在的问题，并根据实际情况添加新的功能。Web应用的漏洞也在逐渐变化，要求系统能够添加新的漏洞插件才能检测出最新的漏洞，所以系统要有一定的可扩展性，方便升级。

(4) 系统稳定性

良好的稳定性能够使系统平稳的运行，如果运行过程中总是出现奔溃的情况，那给用户的体验是很糟糕的，所以要求系统具有较好的稳定性，使系统能够稳定、可靠地运行，在扫描任务的时候不会出现系统奔溃或出错的情况。

3.4 系统总体架构设计

根据以上的需求分析，结合现有的国内外Web应用漏洞的检测工具，以下的章节将对Web应用漏洞检测系统进行详细的设计。

系统的总体架构如图3-7所示。Web应用漏洞系统主要由调度模块控制各个模块，使得各个模块能良好的运行。

图3-7 系统总体架构

Figure 3-7The Overall Architecture of System

Web应用漏洞检测系统是根据系统的配置文件对目标站点进行漏洞扫描，主控模块控制各个子模块，各模块之间协同完成漏洞检测功能。系统开始工作后，首先读取配置文件的内容，然后对目标站点进行爬虫和漏洞扫描，并能够生成扫描报告。其具体的工作流程如图3-8所示。

图3-8 Web漏洞检测过程

Figure 3-8 The Detection Process of Web Vulnerability

Web应用漏洞扫描系统的具体执行过程为：

第一步：启动系统。系统初始化，读取配置文件的内容。

第二步：网络爬虫。网络爬虫是系统的重要组成部分，爬取待测试的URL以及表单，为后续的漏洞检测提供检测对象。

第三步：漏洞检测。构造漏洞探测的HTTP请求，根据响应信息判断目标检测对象是否存在漏洞，本文主要涉及SQL注入漏洞和XSS漏洞。

第四步：生成漏洞报告。以文档的形式导出目标站点的漏洞扫描结果。

3.5 系统功能模块设计

根据系统的总体架构设计，可以将系统划分为六个功能模块：配置文件模块、调度模块、网络爬虫模块、SQL注入模块、XSS模块、报告生成模块，以下对它们进行详细的设计。

3.5.1 配置文件模块

配置文件主要用于设置系统的一些参数，主要包括系统扫描的线程数、网络爬虫的最大深度、请求超时的时间、扫描时候需要用到的漏洞模块以及一些文件的目录，这些基本信息都保存在Configure.XML文件中。其他的配置文件还包括LoginAuth.XML、SQLPayload.XML和XSSPayload.XML等。LoginAuth.XML文件主要用于保存登录认证信息，因为有些情况需要登录目标站点后才能访问，这时用户先要成功登录目标站点，服务器端认证通过后会给系统返回一个会话标志cookies，保存在LoginAuth.XML文件中，后续的网络爬虫和漏洞检测都需要带上这个cookies。SQLPayload.XML用于保存检测SQL注入漏洞的测试脚本。XSSPayload.XML用于保存检测XSS漏洞的测试脚本。

3.5.2 调度模块

调度模块除了参与系统初始化的工作外，其主要任务是调度扫描模块，包括网络爬虫模块、各个漏洞检测模块以及扫描报告生成模块。很多情况下，我们会直接在代码里写死了各个模块的调用顺序，先后分别执行各个模块。这种方法存在一个明显的缺点，如果需要添加或者删除一个扫描模块，就要修改源代码，维护起来很不方便。

本文的方法是通过配置文件来控制扫描模块的执行顺序，配置文件中有一个默认的顺序，用户可以通过系统设置来改变该配置文件的内容。调度模块会根据配置文件，动态地调控扫描模块的执行顺序，如果需要添加或者删除一个扫描模块，只需要修改配置文件即可，而不需要修改源代码。

3.5.3 网络爬虫模块

网络爬虫的目的是爬取目标站点的整个目录、有效的超链接和表单内容，本系统只要求爬取本站点内的链接和表单，因为漏洞检测只针对目标站点，其他站点的链接则不予检测。

本文采用宽度优先的爬虫策略，这样可以避免陷入死循环，另外，系统也可以控制爬虫的深度。在提取URL和表单的过程中，需要根据一定的策略过滤掉不满足要求的URL以及表单，主要是去除非目标站点的、无效的、重复的URL和表单内容。

网络爬虫的主要过程为：

(1) 获取用户输入的URL，通过正则表达式检查URL是否符合规范：URL是否以http开头或者https开头等。如果URL不符合规范，提示用户URL的格式不正确。

(2) 如果URL是合法的，保存并加入爬虫队列。

(3) 判断爬虫队列是否为空，如果爬虫队列为空，则爬虫过程结束。

(4) 从爬虫队列中取出队首的URL，向服务器发送请求并把该网页从服务器下载到本地。

(5) 解析下载回来的HTML页面，首先判断该页面中是否有表单信息，如果有表单并且不重复，提取表单内容并保存。在HTML页面中，包含URL链接的标签有<a>、<iframe>、<script>、<form>等，从这些标签中提取有效的URL。

(6) 对提取的URL进行标准化处理。因为提取的URL可能是相对路径，所以需要这些URL补充完整，使其成为绝对路径。

(7) 过滤并去重URL。网络爬虫的主要目标是站内的链接，但是爬虫过程中也采集其他站点的链接，需要过滤掉这些站外的链接，根据主机地址进行过滤。另外，需要对站内的URL进行去重处理。

(8) 为了避免形成死循环，重复地爬虫某一URL，所以在将URL加入爬虫队列之前，先要判断获取的URL是否已经爬虫过，如果还未曾爬虫，则回到步骤(2)，如果已经爬虫过，直接丢弃。

爬虫模块的详细设计如图3-9所示：

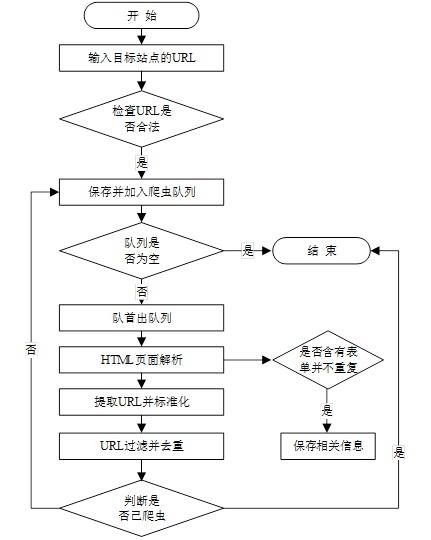
爬图3-9 爬虫模块的设计

Figure 3-9 The Design of Crawler Module

另外，爬取完所有的URL之后，还要对URL进行相似度判断。爬虫模块是漏洞检测模块的基础，为其提供待检测的链接和表单内容。为了避免重复检测，提高漏洞检测效率，需要对相似的URL和相同的表单内容进行相似度去重处理，之后的漏洞检测过程主要针对去重后的URL和表单。比如以下两个URL不需要重复检测：

1) http://www.xxxx.com/news.php?id=23

2) http://www.xxxx.com/news.php?id=33

以上两个链接只有参数的值不相同，就认为是相似的URL，只需要对其中之一进行检测，不需要重复检测它们。

爬虫结束后，就正式进入漏洞检测的流程，这里主要包括SQL注入漏洞和XSS漏洞检测，首先设计SQL注入模块。

3.5.4 SQL注入模块

SQL注入模块的主要功能是对爬虫模块中获取URL或者Form表单进行检测，判断其是否存在注入漏洞。需要注意的是：有的URL和表单中可能存在多个参数，这时候需要逐一对他们进行测试。

SQL注入模块可以分为两个子模块：注入点检测模块和注入点分析模块。SQL注入模块主要包括两个子模块，如图3-10所示：

图3-10 SQL注入模块

Figure 3-10 SQL Injection Module

(1) 注入点检测模块

注入点检测模块的主要功能是判断目标URL或者表单是否存在SQL注入。检测过程中必须覆盖URL和表单中的所有参数，因为有的URL和表单中包含不止一个参数，这种情况需要依次对所有的参数进行检测。另外，有的Web应用程序本身包含有一定的防护程序或者架设了防护设备，实际上是存在漏洞的，为了能够绕过防护程序，提高漏洞的检出率，所采用的payload（检测过程中所用到的测试脚本）是经过多种编码等混淆处理过的，参考2.3.3节的攻击技巧，测试脚本库的设计对于检测漏洞是很关键。

SQL注入检测的原理是将URL或者表单中参数的值设为特定的测试脚本，然后向服务器发送请求，根据响应页面的信息来判断目标URL或者表单是否存在SQL注入漏洞。常用的检测方法是先在带参数的值后面添加一个单引号，看响应页面是否正常，如果响应正常，继续采用and 1=1、and 1=2的方法，如果还是没有出现异常情况，则需要利用特定的测试数据进一步测试，直到使用完所有的测试脚本。

检测SQL注入漏洞的过程如图3-11所示，本文以URL的检测为例，表单的测试过程类似，只是HTTP请求方法可能是GET请求，也可能是POST请求。

图3-11 SQL注入模块设计

Figure 3-11 The Design of SQL Injection Module

1) 从爬虫过程中获取的URL列表中读取一个URL，判断该URL中是否包含参数，如果不包含则不需要检测，如果包含，则需要进一步测试。因为SQL注入漏洞存在需要和数据库交互的参数中。

2) 获取该URL的所有参数，需要为所有的参数测试一遍。

3) 选择测试脚本，构造相应的HTTP请求。

4) 依次为每个参数构造测试语句并发送请求。分别为每个参数构造其值为测试脚本的HTTP请求，然后向服务器端发送该请求。URL中的所有参数都需要测试一遍。

5) 获取服务器的响应信息，根据响应信息来判断该是否存在SQL注入漏洞。

(2) 注入点分析模块

注入点分析模块的主要功能如图3-12所示。

图3-12 注入点分析模块

Figure 3-12 The Module of Injection Point Analysis

1) 确认URL参数的类型。判断目标参数是数字型还是字符型，并判断是否是搜索型注入。在检测过程中，不同的参数类型所采取的payload是不一样的，因为构造的SQL语句是必须是完整的。

2) 确认注入方式。确定注入方式是报错注入、布尔盲注、时间盲注还是联合查询注入。在后续的漏洞利用过程中，针对不同的注入方式所采取的payload是不一样的。

3) 确认目标站点的数据库的类型，判断其是SQL Server、Access、Mysql还是Oracle。不同的数据库之间语法结构是类似的，但是它们之间也存在一定的区别，比如：各数据库之间的系统函数是不同的；SQL Server允许多语句查询，而Access则不允许；各数据库的数据库引擎也是不一样；SQL Server没有limit语法而Mysql有。在漏洞利用过程中，针对不同的数据库需要采用不同的payload。

通过检测SQL注入漏洞的过程也能确定一部分的注入点信息，注入点分析模块需要对未确认的信息进行进一步的分析，包括目标站点所使用数据库类型等信息。

3.5.5 XSS模块

和SQL注入模块一样，XSS模块的检测对象也是爬虫模块中获取的URL以及Form表单，包括GET请求和POST请求中的参数，同样也需要对所有的参数进行检测。

XSS漏洞检测的基本原理是构造相应的XSS测试数据包，然后向服务器发送该请求数据包，根据服务器的响应信息来判断是否存在XSS漏洞，如果在响应信息中找到相应的测试脚本，就认为存在XSS漏洞，反之则不存在。

为了使XSS漏洞的检测结果尽量准确，所采用的测试脚本库的越全越好，需要几十条。但是如果目标检测点不存在XSS漏洞，测试过程中会依次尝试测试脚本库中的所有测试脚本，每次测试都会与服务器进行一次交互。测试完所有的测试脚本后，都没有在响应信息中找到相应的特征信息，就认为待测试点不存在XSS漏洞。

由于存在XSS漏洞的检测点所占的比例是少数的，会造成与服务器的交互次数会过多，影响系统的检测效率。本文的方法是先用合法字符串的测试脚本进行探测，然后再用攻击测试脚本进行攻击测试。合法的字符串是指一段不带非法字符的字符串，应用程序一般不会对其进行任何的过滤处理，比如“xsstest”。该字符串满足两个条件：一是不包含用于构造XSS攻击测试脚本的字符以及字符串，如：“<”、“>”、“script”等；另一个是字符串长度不宜过长，一般不超过10个字符，以防止被应用程序截断。用合法测试脚本进行测试后，如果响应信息中包含相应的合法字符串，需要用测试脚本库中的测试脚本进行进一步的测试。反之，如果响应信息中不包含相应的合法字符串，则认为待测试点不存在XSS漏洞，不需要用测试脚本进行进一步的测试。这样设计的原因是有的目标应用程序会过滤测试脚本中的特殊字符，使得响应页面中不会包含预期的特征字符串。但是目标应用程序不会过滤合法的字符串，所以如果提交的合法字符串不会出现在响应页面中，则提交的攻击测试脚本也一定不会出现在响应信息中。因此，这种方法可以减少与服务器的交互次数，提高检测效率。

XSS漏洞的检测流程如图3-13所示，其具体的步骤为：

(1) 从XSS测试列表中读取一条记录，作为检测对象。

(2) 使用合法的测试脚本构造测试请求。也就是使用合法的测试脚本（即合法的字符串，如：xsstest）替换待测试URL或者表单中的参数，然后向服务器发送该请求。

(3) 获取服务器的响应信息，分析其响应的内容中是否包含字符串“xsstest”，如果不包含该字符串，说明待测试点不存在XSS漏洞，结束测试。反之，需要继续执行步骤(4)。

(4) 依次从测试脚本库中选择测试脚本。

(5) 构造测试语句并发送请求。用上一步骤中的测试脚本替换待测试的参数，然后发送该请求。

(6) 判断该响应信息中是否包含相应的测试脚本字符串，如果包含，说明该测试点存在XSS漏洞。反之，如果测试完所有的测试脚本，都没有在响应页面中找到相应的测试脚本，就认为该测试点不存在XSS漏洞。

图3-13 XSS模块设计

Figure 3-13 The Design of XSS Module

3.5.6 报告生成模块

报告生成模块主要用于生成扫描报告。报告内容包括漏洞整体情况的统计以及漏洞详情两部分。漏洞整体情况主要用于统计系统存在哪些漏洞以及其数量，漏洞详情则是用于描述漏洞的详情，在什么位置存在何种漏洞。