

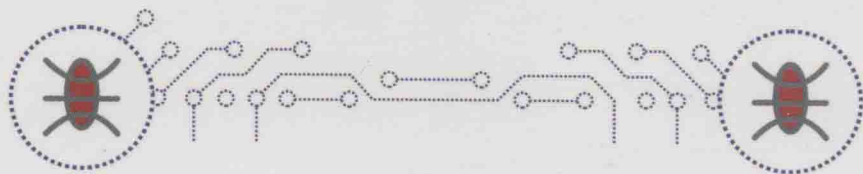
TURING

图灵原创



李建熠◎编著

# Web漏洞防护



中国工信出版集团



人民邮电出版社  
POSTS & TELECOM PRESS

TURING

图灵原创

# Web漏洞防护

李建熠◎编著



人民邮电出版社  
北京

## 图书在版编目 (C I P) 数据

Web漏洞防护 / 李建熠编著. — 北京 : 人民邮电出版社, 2019. 5

(图灵原创)

ISBN 978-7-115-51016-7

I. ①W… II. ①李… III. ①计算机网络—网络安全—研究 IV. ①TP393.08

中国版本图书馆CIP数据核字(2019)第054765号

## 内 容 提 要

本书以 OWASP Top 10 2017 中涉及的漏洞为基础, 系统阐述了常见的 Web 漏洞的防护方式。书中首先介绍了漏洞演示平台及一些常用的安全防护工具, 然后对 OWASP Top 10 2017 中涉及的漏洞防护方式及防护工具进行了说明, 接着介绍了如何通过 HTTP 响应头提升 Web 客户端自身对漏洞的防护能力, 最后讨论了在无法更改应用程序源码的情况下, 如何对应用进行外层的 WAF 防护。

本书适合关注于 Web 漏洞防护的任何读者。

---

◆ 编 著 李建熠

责任编辑 王军花

责任印制 周昇亮

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号

邮编 100164 电子邮件 315@ptpress.com.cn

网址 <http://www.ptpress.com.cn>

北京市艺辉印刷有限公司印刷

◆ 开本: 800×1000 1/16

印张: 18

字数: 431千字

2019年5月第1版

印数: 1-3 000册

2019年5月北京第1次印刷

---

定价: 79.00元

读者服务热线: (010)51095183转600 印装质量热线: (010)81055316

反盗版热线: (010)81055315

广告经营许可证: 京东工商广登字 20170147 号

作者

**李建熠**

毕业于北京邮电大学，  
安全从业者及爱好者，  
曾任职于美团点评，参  
与过美团点评安全从0到  
1的构建。



微信连接



回复“安全”查看相关书单



微博连接

关注@图灵教育 每日分享IT好书



QQ连接

图灵读者官方群I: 218139230

图灵读者官方群II: 164939616

**图灵社区**  
**iTuring.cn**

在线出版,电子书,《码农》杂志,图灵访谈

站在巨人的肩上  
**Standing on Shoulders of Giants**



[iTuring.cn](http://iTuring.cn)

# 前 言

## 概述

目前大多数公司发现并处理安全漏洞的方式如下：利用上线前的安全检测、安全扫描、应急响应中心（SRC）等各种机制发现业务中存在的安全漏洞，然后提交给业务方进行修复，最后安全部门会对相关的漏洞进行复检以确认修复情况。整个过程初看并不存在什么缺陷，但是在实际的执行过程中往往存在下面这些问题。

- ❑ 安全人员只给出漏洞详情，并未向业务方提供相关漏洞的具体修复方案，造成业务方无法对漏洞进行修补。
- ❑ 安全人员给出了修复方案，但是过于简略，如对输入进行过滤、添加 CSRF Token 等，造成业务方无法理解修复方案的具体含义，同时后期的开发人员也无法将其作为一种安全编码规范来遵守。
- ❑ 安全人员给出了详细的修复方案，但是未给出修复漏洞的工具，业务方需自行查找相关工具，这给他们带来了很大的麻烦，可能会造成漏洞修复不及时，同时他们选用的工具也可能存在缺陷或漏洞，会造成漏洞修复不完全或引入新的漏洞。

目前，国内鲜有图书针对安全漏洞修复方案及修复工具进行详细介绍，网上相关文章大多只是简单介绍，缺少全面且详尽的讲解。

本书通过归纳国内外 Web 安全防护的相关文章（特别是与 OWASP 相关的文章），总结工作中的一些经验，对漏洞防护的方法进行说明。开发人员可以将这些防护方法作为日常编码的一种规范来遵从，以减少同类漏洞的出现。为了使漏洞修复更加便利、快捷，书中将会介绍漏洞防护的相关工具及其使用方法。同时，考虑到开发人员对于安全漏洞及其原理不甚了解，本书将会结合相关实例进行具体的说明，方便读者对安全漏洞有相对直观的认识。

关于 Web 安全防护，我们不能奢求通过某种手段防护所有的漏洞，而需要采取多种防护手段才能产生良好的防护效果。本书共介绍了 3 种防护手段。

- ❑ 在能够更改程序源码的情况下，对问题代码进行修改，从而对漏洞进行防护。这是防护 Web 漏洞的主要方式，也是本书介绍的重点，对应本书的第 2 章到第 13 章。

- 提升 Web 客户端自身防护漏洞的能力，以缓解潜在的漏洞危害。这种方式主要通过 HTTP 响应头进行设置，对应本书的第 14 章。
- 通过客户端与服务器之间的通信数据，分析出潜在的恶意流量并及时拦截，防止 Web 漏洞的产生。这种方式主要通过 WAF 进行防护，对应本书的第 15 章。

安全漏洞的种类繁多，一本书不可能囊括所有的漏洞。本书对漏洞的普遍性和严重性这两个因素进行考量，涵盖的范围主要是 OWASP Top 10 2017 中所涉及的漏洞。同时，考虑到目前大多数企业都将 Java 作为主要的开发语言，因此在漏洞说明及工具介绍上都将以 Java 为主，但是漏洞防护的方法是通用的，可以将它们应用到其他语言上。

本书涉及的内容比较庞杂，且本人在对某些知识的认识上存在一定的局限性，书中的内容难免会有些不足，甚至错误之处。如果大家在阅读中发现不足或错误之处，欢迎通过邮件方式（我的邮箱：[balckarbiter@gmail.com](mailto:balckarbiter@gmail.com)）向本人指正。

## 本书结构

全书共包含 15 章。

第 1 章主要介绍了漏洞演示平台及书中将多次用到的安全防护工具，同时简要介绍了 OWASP Top 10 的内容。

第 2 章到第 13 章是本书的重点部分，详细介绍 OWASP Top 10 2017 中涉及的漏洞防护方式及防护工具，读者可以通过目录进一步了解其内容。

第 14 章主要介绍了如何通过 HTTP 响应头提升 Web 客户端自身对漏洞的防护能力。

第 15 章主要介绍了在无法更改应用程序源码的情况下，如何对应用进行外层的 WAF 防护。

## 致谢

感谢 OWASP 基金会项目与技术总监 Harold L. Blankenship 先生对 OWASP 相关文章引用的授权，本书最后将会详细列出所引用的文章，以示感谢。

感谢带我走上安全之路并给予我指导的陈伟先生、李晖老师、祁麟先生和张文师兄，感谢在写作过程中给予我支持与帮助的朋友。

感谢王军花和武芮欣两位女士，是她们的努力才使本书最终与广大读者见面，她们的专业意见给了我极大的帮助。

感谢我的父母、妻子和兄长，是他们的支持与陪伴，才使我最终完成本书。



# 目 录

第 1 章 使用工具介绍 .....	1	第 3 章 其他注入防护 .....	34
1.1 WebGoat .....	1	3.1 命令注入防护 .....	34
1.2 ESAPI .....	5	3.2 XML 注入防护 .....	34
1.3 Apache Shiro .....	8	3.3 XPATH 注入防护 .....	38
1.3.1 Apache Shiro 的特征 .....	8	3.4 LDAP 注入防护 .....	39
1.3.2 Apache Shiro 的核心概念 .....	9	3.5 JPA 注入防护 .....	40
1.3.3 与 Spring 集成 .....	12	3.6 小结 .....	43
1.4 Spring Security .....	15	第 4 章 认证防护 .....	44
1.5 OWASP Top 10 .....	17	4.1 认证缺陷 .....	44
第 2 章 SQL 注入防护 .....	19	4.2 认证防护 .....	44
2.1 SQL 注入介绍 .....	19	4.2.1 用户名及密码设置 .....	45
2.2 SQL 注入分类 .....	20	4.2.2 忘记密码 .....	46
2.2.1 按参数类型分类 .....	20	4.2.3 凭证存储 .....	47
2.2.2 按注入位置分类 .....	20	4.2.4 密码失窃 .....	48
2.2.3 按结果反馈分类 .....	20	4.2.5 其他安全防护 .....	49
2.2.4 其他类型 .....	21	4.3 会话管理安全 .....	50
2.3 实例讲解 .....	21	4.3.1 会话 ID 的属性 .....	51
2.3.1 字符型注入 .....	22	4.3.2 会话管理的实现 .....	52
2.3.2 数字型注入 .....	22	4.3.3 Cookie .....	53
2.3.3 联合查询注入及堆查询注入 .....	23	4.3.4 会话 ID 的注意事项 .....	54
2.3.4 盲注入 .....	24	4.3.5 会话过期 .....	55
2.4 检测 SQL 注入 .....	25	4.3.6 会话管理及其他客户端防御 .....	57
2.5 防护方案 .....	26	4.3.7 会话攻击检测 .....	58
2.5.1 漏洞实例 .....	27	4.3.8 会话管理的 WAF 保护 .....	59
2.5.2 预编译与参数绑定 .....	28	4.4 防护工具 .....	59
2.5.3 白名单验证 .....	29	4.4.1 Argon2 密码散列 .....	59
2.5.4 输入编码 .....	30	4.4.2 Apache Shiro 认证 .....	63
2.5.5 MyBatis 安全使用 .....	32	4.4.3 Apache Shiro 会话管理 .....	65
2.6 小结 .....	33	4.5 小结 .....	68

第5章 数据泄露防护	69	第8章 安全配置	129
5.1 传输层安全防护	69	第9章 XSS防护	131
5.1.1 SSL/TLS 注意事项	70	9.1 XSS分类	131
5.1.2 其他注意事项	75	9.1.1 反射型 XSS	132
5.1.3 传输层安全检测工具	75	9.1.2 DOM型 XSS	134
5.2 数据加密存储	77	9.1.3 存储型 XSS	136
5.2.1 密码学简史	78	9.1.4 其他分类	137
5.2.2 加密模式及填充模式	83	9.2 检测 XSS	138
5.2.3 杂凑函数及数据完整性保护	88	9.3 XSS防护方法	139
5.2.4 加解密使用规范	90	9.3.1 反射型 XSS 和存储型 XSS 的 防护	140
5.3 安全数据共享	104	9.3.2 DOM型 XSS 防护	143
5.3.1 数据仓库的构建	104	9.4 防护工具	144
5.3.2 数据仓库的保护	105	9.4.1 OWASP Java Encoder	144
5.3.3 数据仓库的管理	106	9.4.2 OWASP Java HTML Sanitizer	149
5.4 小结	106	9.4.3 AngularJS SCE	158
第6章 XXE防护	107	9.4.4 ESAPI4JS	160
6.1 XML介绍	107	9.4.5 jQuery Encoder	164
6.2 XXE攻击方式及实例介绍	109	9.5 小结	167
6.2.1 内部XXE实例	110	第10章 反序列化漏洞防护	168
6.2.2 外部XXE实例	111	10.1 Java的序列化与反序列化	168
6.3 检测XXE	112	10.1.1 序列化	168
6.4 XXE防护	113	10.1.2 反序列化	169
6.4.1 DOM	113	10.1.3 自定义序列化与反序列化	170
6.4.2 SAX	116	10.1.4 Java反序列化漏洞	171
6.4.3 其他	117	10.1.5 其他反序列化漏洞	175
6.5 小结	117	10.2 检测反序列化漏洞	178
第7章 访问控制防护	118	10.3 反序列化漏洞的防护	179
7.1 访问控制的分类	118	10.4 防护工具	180
7.2 常见问题	119	10.4.1 自定义工具	180
7.2.1 不安全对象的直接引用	119	10.4.2 SerialKiller	181
7.2.2 功能级访问控制缺失	120	10.4.3 contra-r00	183
7.2.3 跨域资源共享的错误配置	121	10.5 小结	185
7.3 工具防护	123	第11章 组件缺陷的检测	186
7.3.1 Apache Shiro 访问控制	123	11.1 潜在缺陷	186
7.3.2 ESAPI 随机化对象引用	126	11.2 检测缺陷组件	186
7.3.3 Spring Security CORS 配置	127	11.2.1 Retire.js	187
7.4 小结	128		

11.2.2	OWASP Dependency Check	190	14.2.2	在线检测工具	229
11.2.3	Sonatype AHC	193	14.2.3	插件检测工具	230
11.3	小结	196	14.3	安全响应头设置建议	231
<b>第 12 章</b>	<b>跨站点请求伪造防护</b>	<b>197</b>	14.3.1	知名网站实例	231
12.1	CSRF 分类	197	14.3.2	设置建议	233
12.1.1	GET 型 CSRF	197	14.4	配置安全响应头	233
12.1.2	POST 型 CSRF	198	14.4.1	Spring Security 统一配置	233
12.1.3	CSRF 实例	198	14.4.2	http_hardening 配置安全 响应头	237
12.1.4	CSRF 结合 XSS	200	14.4.3	服务器配置文件配置安全 响应头	238
12.2	检测 CSRF	202	14.5	小结	238
12.3	CSRF 防护	202	<b>第 15 章</b>	<b>WAF 防护</b>	<b>239</b>
12.3.1	不完全的防护方式	203	15.1	ModSecurity	239
12.3.2	正确的防护方式	204	15.1.1	编译与导入	240
12.4	防护工具	209	15.1.2	配置 ModSecurity	241
12.4.1	自定义防护工具	210	15.1.3	ModSecurity 测试	244
12.4.2	Spring Security 防护 CSRF	215	15.2	规则解析	245
12.4.3	前后端分离	216	15.2.1	指令	246
12.5	小结	217	15.2.2	处理阶段	247
<b>第 13 章</b>	<b>输入验证</b>	<b>218</b>	15.2.3	变量	247
13.1	输入验证的方式	218	15.2.4	转换函数	249
13.2	ESAPI 输入验证	218	15.2.5	行为	250
<b>第 14 章</b>	<b>HTTP 安全响应头</b>	<b>222</b>	15.2.6	操作符	253
14.1	安全响应头介绍	222	15.3	OWASP ModSecurity CRS	255
14.1.1	HSTS	222	15.3.1	CRS 导入	255
14.1.2	HPKP	223	15.3.2	CRS 规则文件	257
14.1.3	X-Frame-Options	223	15.4	防护测试	259
14.1.4	X-XSS-Protection	224	15.4.1	DVWA 环境搭建	259
14.1.5	X-Content-Type-Options	224	15.4.2	SQL 注入测试	261
14.1.6	Content-Security-Policy	224	15.4.3	命令注入测试	264
14.1.7	Referrer-Policy	226	15.4.4	XSS 测试	267
14.1.8	Expect-CT	226	15.4.5	文件包含测试	272
14.1.9	X-Permitted-Cross-Domain- Policies	226	15.4.6	文件上传测试	274
14.1.10	Cache-Control	228	15.5	小结	277
14.2	HTTP 安全头检测	228	<b>参考文献</b>	<b>278</b>	
14.2.1	命令行检测工具	228			

# 使用工具介绍



本章首先介绍了漏洞演示工具 WebGoat 的安装及使用，然后讨论了书中将多次使用的安全工具 ESAPI、Apache Shiro 和 Spring Security，最后简述了 OWASP Top 10 的相关内容和本书结构。

## 1.1 WebGoat

WebGoat 是由 OWASP 组织使用 Java 语言研发的一款 Web 应用，用于演示 Web 应用中常见的漏洞。该项目是开源的，可以从 GitHub 上获取其最新源码 (<https://github.com/WebGoat/WebGoat>)。WebGoat 中的漏洞类型与最新的 OWASP Top 10 保持一致，如最新版本的 WebGoat 8.0 与近期发布的 OWASP Top 10 2017 保持一致，是我们用于漏洞演示及学习的重要工具。注意不要将 WebGoat 部署到企业的生产环境中，因为攻击者可以利用上面的漏洞对生产环境造成损害。

本次对漏洞实例的展示使用最新版的 WebGoat 8.0，目前共有 11 个漏洞课程，包括 SQL 注入攻击、XXE (XML External Entity Attack, XML 外部实体攻击)、认证缺陷 (认证绕过)、XSS (Cross-Site Scripting, 跨站脚本) 攻击、访问控制缺陷 [包括不安全对象直接引用 (IDOR) 和功能级访问控制缺失]、不安全的登录、跨站请求伪造 (CSRF)、组件缺陷及一些其他课程。除主体课程外，WebGoat 8.0 还提供了一个辅助工具 WebWolf，用于协助学习相关的课程。它主要包括 3 个功能：文件上传、邮件服务和请求消息记录。

下面我们简要介绍如何搭建 WebGoat 环境。这可以使用多种方式，如 Docker、Jar 包和源码，其中最简单的方式就是直接使用 Jar 包，具体的搭建步骤如下所示。

(1) 下载最新版的 Jar 包 (地址: <https://github.com/WebGoat/WebGoat/releases>)，这里需要用到 webgoat-server.jar 和 webwolf.jar。

(2) 使用命令 `java -jar webgoat-server.jar` 运行 WebGoat 主体课程，访问 `http://localhost:8080/WebGoat/login` 进行注册，然后使用注册的用户名和密码登录，即可开始学习课程，WebGoat 运行截图如图 1-1 所示。从运行截图中可以看出，左侧为课程的导航栏，右侧为课程的具体内容，此处不展开说明，第 2 章至第 13 章将详细介绍课程内容。



图 1-1 WebGoat 运行截图

(3) 使用命令 `java -jar webwolf.jar` 运行 WebWolf，访问 `http://localhost:8081/login` 进行注册并使用已注册的用户名及密码登录，即可使用 WebWolf 相关功能。WebWolf 的运行截图如图 1-2 所示。

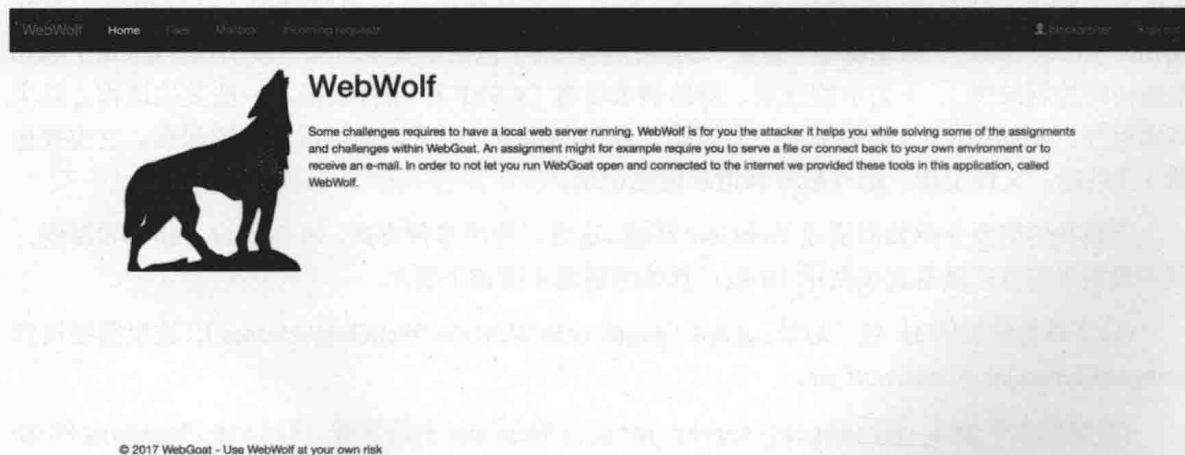


图 1-2 WebWolf 运行截图

WebWolf共包含3个功能,第一个功能为文件上传。WebWolf可以作为一个文件服务器使用,通过所上传文件最右侧的 link 超链接能够获取文件的存储地址,如图 1-3 所示。



图 1-3 文件上传

第二个功能为模拟邮件服务器。WebGoat 能够向名为 `user@random` 的邮箱发送邮件,该邮件服务器会收到发送的相关邮件,并在页面中展示,如图 1-4 所示。

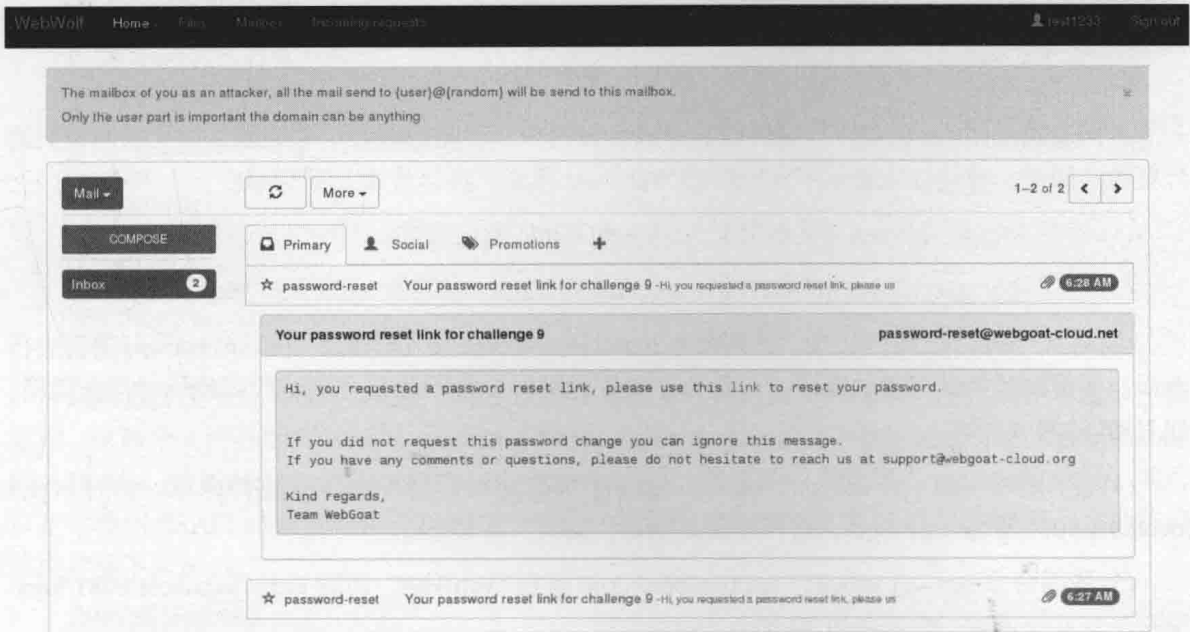


图 1-4 邮件服务器

第三个功能为请求记录。向 URL 地址 `http://127.0.0.1:8081/WebWolf/landing/*` 发起请求，该页面将会记录请求与响应消息，并在页面下方展示，如图 1-5 所示。

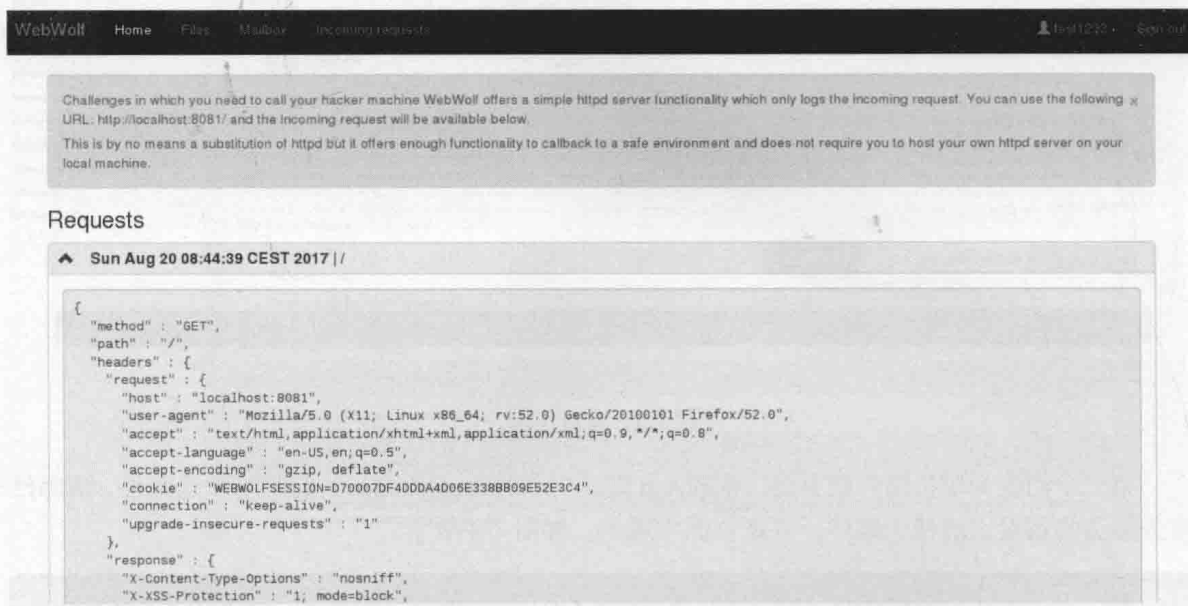


图 1-5 请求记录

虽然直接使用 Jar 包进行环境搭建的方式比较简单，但是也存在一些缺陷，如无法对源码进行更改。因此，我建议大家使用源码搭建 WebGoat 环境，具体步骤如下所示。

(1) 确保 Java 8、Maven（版本大于 3.2.1）和 Git 环境已安装。

(2) 使用命令 `git clone https://github.com/WebGoat/WebGoat.git` 下载 WebGoat 源码。

(3) 进入 WebGoat 源码目录，使用命令 `mvn clean compile install -DskipTests` 对项目进行编译。尽量跳过测试代码的编译，一是为了节省时间，二是因为某些测试代码编译时存在错误，如 XXE 的测试代码。

(4) 使用命令 `mvn -pl webgoat-server spring-boot:run` 运行 WebGoat 主体课程。访问 `http://localhost:8080/WebGoat/login`，进行注册并登录，即可开始课程的学习。

(5) 使用命令 `mvn -pl webwolf spring-boot:run` 运行 WebWolf，访问 `http://localhost:8081/login` 进行注册并登录后，即可使用 WebWolf 相关功能。

---

**注意** WebGoat 8.0 没有默认用户，我们需要注册自己的用户，然后完成登录。

---



## 1.2 ESAPI

ESAPI 的全称为 The OWASP Enterprise Security API，是一款免费、开源的 Web 应用程序安全控制包。它为常见的 Web 安全漏洞提供了防护方案，并为构建 Web 应用安全防御体系提供了重要的参考。ESAPI 除了提供 Java 安全包的实现外，还提供了 .NET、ASP、PHP、Python 和 JavaScript 等安全包的实现，因此使用它可以解决多种语言的安全问题。

ESAPI 的使用比较简单，只需要导入依赖的 log4j 等相应的 Jar 包就可以开始使用了。下面展示使用 Maven 完成相关依赖及 ESAPI 的导入示例，其中 ESAPI 是通过本地 Jar 包导入的，Jar 包的地址通过 `<systemPath>` 标签来定义：

```
<dependency>
  <groupId>org.owasp.esapi</groupId>
  <artifactId>esapi</artifactId>
  <version>2.1.0</version>
  <scope>system</scope>
  <systemPath>${basedir}/lib/esapi-2.1.0.jar</systemPath>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>1.7.25</version>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-log4j12</artifactId>
  <version>1.7.12</version>
</dependency>
<dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>1.2.17</version>
</dependency>
```

如果直接使用这个程序，会因为缺少配置文件报错，所以需要导入两个配置文件才能够正常运行。这两个文件分别为 ESAPI.properties 和 validation.properties，下载地址为 <https://github.com/ESAPI/esapi-java-legacy/tree/develop/configuration/esapi>。将这两个文件放入 resources 目录，就可以正常使用 ESAPI 的相关功能了，这两个文件中某些内容的含义将在第 13 章中说明，文件的具体内容如下所示：

```
ESAPI.properties
ESAPI.printProperties=true
ESAPI.AccessControl=org.owasp.esapi.reference.DefaultAccessController
ESAPI.Authenticator=org.owasp.esapi.reference.FileBasedAuthenticator
ESAPI.Encoder=org.owasp.esapi.reference.DefaultEncoder
ESAPI.Encryptor=org.owasp.esapi.reference.crypto.JavaEncryptor
ESAPI.Executor=org.owasp.esapi.reference.DefaultExecutor
ESAPI.HTTPUtilities=org.owasp.esapi.reference.DefaultHTTPUtilities
ESAPI.IntrusionDetector=org.owasp.esapi.reference.DefaultIntrusionDetector
```



```
ESAPI.Logger=org.owasp.esapi.reference.Log4JLogFactory
ESAPI.Randomizer=org.owasp.esapi.reference.DefaultRandomizer
ESAPI.Validator=org.owasp.esapi.reference.DefaultValidator

# 认证配置
Authenticator.AllowedLoginAttempts=3
Authenticator.MaxOldPasswordHashes=13
Authenticator.UsernameParameterName=username
Authenticator.PasswordParameterName=password
# 记住 token 的持续时间 (以天为单位)
Authenticator.RememberTokenDuration=14
# 会话超时 (以分钟为单位)
Authenticator.IdleTimeoutDuration=20
Authenticator.AbsoluteTimeoutDuration=120

# 编码器配置
Encoder.AllowMultipleEncoding=false
Encoder.AllowMixedEncoding=false
Encoder.DefaultCodecList=HTMLEntityCodec,PercentCodec,JavaScriptCodec

# 加解密配置
Encryptor.MasterKey=tzftzf56ftw
Encryptor.MasterSalt=123456ztrewq
Encryptor.PreferredJCEProvider=
Encryptor.EncryptionAlgorithm=AES
Encryptor.CipherTransformation=AES/CBC/PKCS5Padding
Encryptor.cipher_modes.combined_modes=GCM,CCM,IAPM,EAX,OCB,CWC
Encryptor.cipher_modes.additional_allowed=CBC
Encryptor.EncryptionKeyLength=128
Encryptor.ChooseIVMethod=random
Encryptor.fixedIV=0x000102030405060708090a0b0c0d0e0f
Encryptor.CipherText.useMAC=true
Encryptor.PlainText.override=true
Encryptor.HashAlgorithm=SHA-512
Encryptor.HashIterations=1024
Encryptor.DigitalSignatureAlgorithm=SHA1withDSA
Encryptor.DigitalSignatureKeyLength=1024
Encryptor.RandomAlgorithm=SHA1PRNG
Encryptor.CharacterEncoding=UTF-8
Encryptor.KDF.PRF=HmacSHA256

# HTTP 工具配置
HttpUtilities.UploadDir=C:\\ESAPI\\testUpload
HttpUtilities.UploadTempDir=C:\\temp
# 设置 Cookie 的属性
HttpUtilities.ForceHttpOnlySession=false
HttpUtilities.ForceSecureSession=false
HttpUtilities.ForceHttpOnlyCookies=true
HttpUtilities.ForceSecureCookies=true
# HTTP 头的最大值
HttpUtilities.MaxHeaderSize=4096

# 文件上传配置
HttpUtilities.ApprovedUploadExtensions=.zip,.pdf,.doc,.docx,.ppt,.pptx,.tar,.gz,.tgz,.rar,.war,
```