

实验 11 Sql注入攻击1

实验目的

通过实验，使学习者：

- 理解注入攻击的基本原理
- 掌握使用工具实施注入攻击的方法
- 理解Web系统防御注入攻击的对策

实验内容

- 1.尝试对owasp bwa v1.2中的webgoat 5.4中的Injection Flaws —— Numeric SQL Injection进行SQL注入攻击。
- 2.尝试对owasp bwa v1.2中的webgoat 5.4中的Injection Flaws —— Log spoofing进行注入攻击。
- 3.尝试对owasp bwa v1.2中的webgoat 5.4中的Injection Flaws —— xpath injection进行注入攻击。
- 4.尝试对owasp bwa v1.2中的webgoat 5.4中的Injection Flaws —— String SQL Injection进行注入攻击。
- 5.尝试对owasp bwa v1.2中的webgoat 5.4中的Injection Flaws —— LAB:SQL Injection。

实验过程

Numeric SQL Injection实验

- 1.启动 owasp bwa v1.2 虚拟机（启动后自动加载WebGoat 5.4 网站应用），查看其主机ip（下面以10.10.10.135为例）
- 2.启动 kali 2019虚拟机
- 3.在Kali 2019 虚拟机中打开设置了burpsuite proxy的浏览器firefox和Burp suite。
- 4.通过firefox浏览器访问 webgoat 5.4 应用，链接为：<http://10.10.10.135/WebGoat/attack>，认证用户名webgoat，密码为webgoat；然后点击左侧导航Injection Flaws——Numeric SQL Injection。

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws

✓ [Command Injection](#)

[Numeric SQL Injection](#) ✓

[Log Spoofing](#)

[XPath Injection](#)

[String SQL Injection](#)

[LAB: SQL Injection](#)

[Stage 1: String SQL Injection](#)

[Stage 2: Parameterized Query #1](#)

[Stage 3: Numeric SQL Injection](#)

[Stage 4: Parameterized Query #2](#)

[Modify Data with SQL Injection](#)

[Add Data with SQL Injection](#)

[Database Backdoors](#)

[Blind Numeric SQL Injection](#)

Solution Videos

[Restart this Lesson](#)

SQL injection attacks represent a serious threat to any database-driven site. The methods behind an attack are easy to learn and the damage caused can range from considerable to complete system compromise. Despite these risks, an incredible number of systems on the internet are susceptible to this form of attack.

Not only is it a threat easily instigated, it is also a threat that, with a little common-sense and forethought, can easily be prevented.

It is always good practice to sanitize all input data, especially data that will be used in OS command, scripts, and database queries, even if the threat of SQL injection has been prevented in some other manner.

General Goal(s):

The form below allows a user to view weather data. Try to inject an SQL string that results in all the weather data being displayed.

Select your local weather station:

SELECT * FROM weather_data WHERE station = 101

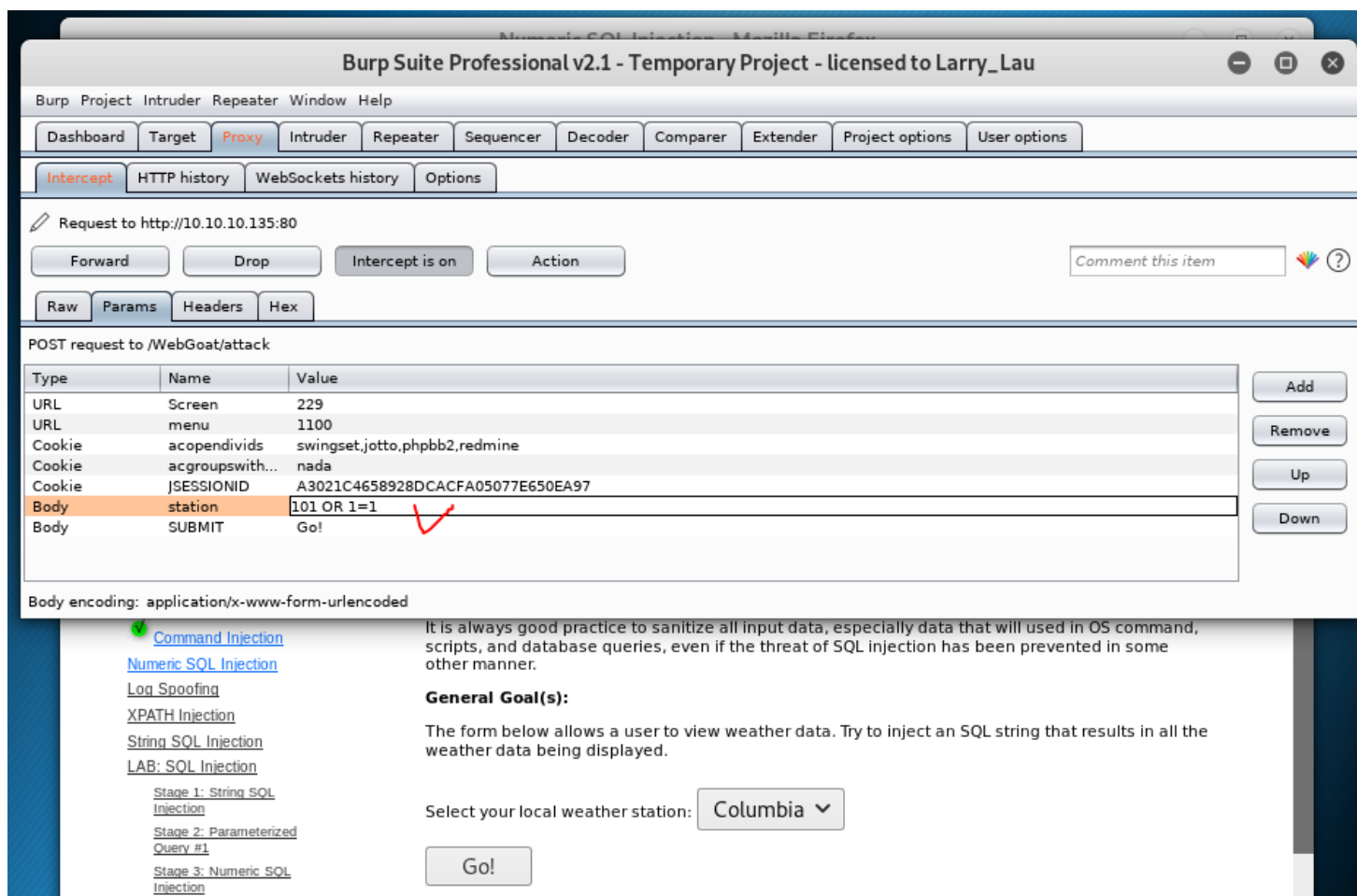
STATION	NAME	STATE	MIN_TEMP	MAX_TEMP
101	Columbia	MD	-10	102

OWASP Foundation | Project WebGoat | [Report Bug](#)

上图中可见，选择下拉选择框的地区，例如“Columbia”，然后点“Go！”，可见出现下方的天气信息。

5. 设置BurpSuite中的proxy-Interrupt is on，准备拦截访问请求；然后再次点击上一步中的“Go！”，在拦截到的请求参数中插入sql语句：

or 1=1



6.点击burp suite proxy中的“Forward”，然后在浏览器中会发现下图结果：

Injection Flaws

- [Command Injection](#)
- [Numeric SQL Injection](#)
- [Log Spoofing](#)
- [XPath Injection](#)
- [String SQL Injection](#)
- [LAB: SQL Injection](#)
 - [Stage 1: String SQL Injection](#)
 - [Stage 2: Parameterized Query #1](#)
 - [Stage 3: Numeric SQL Injection](#)
 - [Stage 4: Parameterized Query #2](#)
- [Modify Data with SQL Injection](#)
- [Add Data with SQL Injection](#)
- [Database Backdoors](#)
- [Blind Numeric SQL Injection](#)
- [Blind String SQL Injection](#)

Denial of Service
Insecure Communication
Insecure Configuration
Insecure Storage
Malicious Execution
Parameter Tampering
Session Management Flaws
Web Services

forethought, can easily be prevented.

It is always good practice to sanitize all input data, especially data that will be used in OS commands, scripts, and database queries, even if the threat of SQL injection has been prevented in some other manner.

General Goal(s):

The form below allows a user to view weather data. Try to inject an SQL string that results in all the weather data being displayed.

*** Congratulations. You have successfully completed this lesson.
* Bet you can't do it again! This lesson has detected your successful attack and has now switched to a defensive mode. Try again to attack a parameterized query.**

Select your local weather station:

Go!

SELECT * FROM weather_data WHERE station = 101 OR 1=1

STATION	NAME	STATE	MIN_TEMP	MAX_TEMP
101	Columbia	MD	-10	102
102	Seattle	WA	-15	90
103	New York	NY	-10	110
104	Houston	TX	20	120
10001	Camp David	MD	-10	100
11001	Ice Station Zebra	NA	-60	30

注入成功。

Log Spoofing实验


- 1.启动 owasp bwa v1.2 虚拟机（启动后自动加载WebGoat 5.4 网站应用），查看其主机ip（下面以10.10.10.135为例）
- 2.通过任意浏览器访问 webgoat 5.4 应用，链接为：<http://10.10.10.135/WebGoat/attack>，认证用户名webgoat，密码为webgoat；然后点击左侧导航Injection Flaws—Log Spoofing。

The screenshot shows the OWASP WebGoat v5.4 application interface. At the top, there is a language selector set to 'English' and a 'Log Spoofing' title bar. Below the title bar, there are navigation links: 'Show Params', 'Show Cookies', and 'Lesson Plan'. The main content area is divided into two columns. The left column contains a list of topics: Introduction, General, Access Control Flaws, AJAX Security, Authentication Flaws, Buffer Overflows, Code Quality, Concurrency, Cross-Site Scripting (XSS), Improper Error Handling, and Injection Flaws. Under 'Injection Flaws', there are links for 'Command Injection', 'Numeric SQL Injection', 'Log Spoofing' (highlighted with a green checkmark), 'XPath Injection', 'String SQL Injection', and 'LAB: SQL Injection'. The right column is titled 'Solution Videos' and contains a 'Restart this Lesson' link. Below this, there is a list of instructions: '* The grey area below represents what is going to be logged in the web server's log file.', '* Your goal is to make it like a username "admin" has succeeded into logging in.', and '* Elevate your attack by adding a script to the log file.'. A login form is present with fields for 'User Name' (containing 'admin') and 'Password', and a 'Login' button. Below the login form, a grey box displays the message 'Login failed for username: admin' with a red checkmark next to it. At the bottom right, there is a credit line: 'Created by Sherif Koussa SoftwareSecured'.

说明：图中灰色部分代表了服务器日志中的记录。日志是网站安全维护时要审查的重点，写入虚假日志会扰乱、迷惑维护人员开展工作。本题的目标是在输入框中键入含特殊字符的字符串，使日志文件（灰色部分）出现“Login Succeeded for username: admin”的记录。

- 3.在“user name：” 输入框后键入如下内容：

```
whor%0d%0aLogin Succeeded for username: admin
```

**OWASP WebGoat v5.4**

Show ParamsShow CookiesLesson Plan

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws

✔

[Command Injection](#)

✔

[Numeric SQL Injection](#)

✔

[Log Spoofing](#)

[XPath Injection](#)

[String SQL Injection](#)

[LAB: SQL Injection](#)

Stage 1: String SQL Injection

Stage 2: Parameterized Query

Solution Videos

✧ The grey area below represents what is going to be logged in the web server's log file.

✧ Your goal is to make it like a username "admin" has succeeded into logging in.

✧ Elevate your attack by adding a script to the log file.

* Congratulations. You have successfully completed this lesson.

User Name : whor%0d%0aLogin Succeed

Password :

Login

Login failed for username: whor

Login Succeeded for username: admin

Created by Sherif Koussa **SoftwareSecured**

OWASP Foundation | Project WebGoat | Report Bug

注入成功。

这个例子仅是对日志欺骗的一个简单示例，实际的日志欺骗会复杂得多。

XPATH注入攻击

1.启动 owasp bwa v1.2 虚拟机（启动后自动加载WebGoat 5.4 网站应用），查看其主机ip（下面以 10.10.10.135为例）

2.通过任意浏览器访问 webgoat 5.4 应用，链接为：<http://10.10.10.135/WebGoat/attack>，认证用户名 webgoat，密码为webgoat；然后点击左侧导航Injection Flaws——XPath Injection。

阅读页面提示可知，网站提供了一个工资查询表单，键入用户名和口令会输出工资。如下图所示：

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws

- ✓ [Command Injection](#)
 - ✓ [Numeric SQL Injection](#)
 - ✓ [Log Spoofing](#)
 - [XPATH Injection](#) ✓
 - [String SQL Injection](#)
 - [LAB: SQL Injection](#)
- Stage 1: String SQL

Solution Videos

[Restart this Lesson](#)

The form below allows employees to see all their personal data including their salaries. Your account is Mike/test123. Your goal is to try to see other employees data as well.

Welcome to WebGoat employee intranet

Please confirm your username and password before viewing your profile.

*Required Fields

*User Name:

Mike

*Password:

Submit

Username	Account No.	Salary
Mike	11123	468100

Created by Sherif Koussa **SoftwareSecured** ✓

如何查看其他人或所有人的信息呢？可以尝试注入攻击。一般情况下工资和员工信息存在数据库中，但这个例子中的信息存放在xml中。

3.在user name后的输入框中键入下面内容：

```
Mike' or '1'='1' or 'a'='a
```

4.在password输入框输入任意字符，例如“123”，之后提交，有下列结果：

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws

- ✓ [Command Injection](#)
- ✓ [Numeric SQL Injection](#)
- ✓ [Log Spoofing](#)
- ✓ [XPATH Injection](#)
- [String SQL Injection](#)
- [LAB: SQL Injection](#)
 - [Stage 1: String SQL Injection](#)
 - [Stage 2: Parameterized Query #1](#)
 - [Stage 3: Numeric SQL Injection](#)

Solution Videos

[Restart this Lesson](#)

The form below allows employees to see all their personal data including their salaries. Your account is Mike/test123. Your goal is to try to see other employees data as well.

*** Congratulations. You have successfully completed this lesson.**

Welcome to WebGoat employee intranet

Please confirm your username and password before viewing your profile.

*Required Fields

*User Name:

anyone' or 1=1 or '1'='1

*Password:

...

Submit

Username	Account No.	Salary
Mike	11123	468100
John	63458	559833
Sarah	23363	84000

Created by Sherif Koussa **SoftwareSecured**

OWASP Foundation | Project WebGoat | Report Bug

注入成功。

说明：

开发人员构造的xpath查询语句大致为：

```
//user[name/text()='Mike' and password/text()='test123']
```

注入后查询语句改为：

```
//user[name/text()='Mike' or '1'='1' or 'a'='a' and password/text()='123']
```

String SQL Injection

- 启动 owasp bwa v1.2 虚拟机（启动后自动加载WebGoat 5.4 网站应用），查看其主机ip（下面以 10.10.10.135为例）
- 通过任意浏览器访问 webgoat 5.4 应用，链接为：<http://10.10.10.135/WebGoat/attack>，认证用户名 **webgoat**，密码为**webgoat**；然后点击左侧导航Injection Flaws——String SQL Injection。

[Introduction](#)
[General](#)
[Access Control Flaws](#)
[AJAX Security](#)
[Authentication Flaws](#)
[Buffer Overflows](#)
[Code Quality](#)
[Concurrency](#)
[Cross-Site Scripting \(XSS\)](#)
[Improper Error Handling](#)
[Injection Flaws](#)

✔ [Command Injection](#)
✔ [Numeric SQL Injection](#)
✔ [Log Spoofing](#)
[XPath Injection](#)
[String SQL Injection](#) ✔
[LAB: SQL Injection](#)
 [Stage 1: String SQL Injection](#)
 [Stage 2: Parameterized Query #1](#)
 [Stage 3: Numeric](#)

Solution Videos

[Restart this Lesson](#)

SQL injection attacks represent a serious threat to any database-driven site. The methods behind an attack are easy to learn and the damage caused can range from considerable to complete system compromise. Despite these risks, an incredible number of systems on the internet are susceptible to this form of attack.

Not only is it a threat easily instigated, it is also a threat that, with a little common-sense and forethought, can easily be prevented.

It is always good practice to sanitize all input data, especially data that will be used in OS command, scripts, and database queries, even if the threat of SQL injection has been prevented in some other manner.

General Goal(s):

The form below allows a user to view their credit card numbers. Try to inject an SQL string that results in all the credit card numbers being displayed. Try the user name of 'Smith'.

Enter your last name:

```
SELECT * FROM user_data WHERE last_name = 'Your Name'
```

No results matched. Try Again.

[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

3. 尝试进行下列sql注入，然后点击“Go！”

Your Name' or 1=1 or '1'='1

General Goal(s):

The form below allows a user to view their credit card numbers. Try to inject an SQL string that results in all the credit card numbers being displayed. Try the user name of 'Smith'.

* Congratulations. You have successfully completed this lesson.
* Now that you have successfully performed an SQL injection, try the same type of attack on a parameterized query. Restart the lesson if you wish to return to the injectable query.

Enter your last name:

```
SELECT * FROM user_data WHERE last_name = 'Your Name' or 1=1 or '1'='1'
```

USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
101	Joe	Snow	987654321	VISA		0
101	Joe	Snow	2234200065411	MC		0
102	John	Smith	2435600002222	MC		0
102	John	Smith	4352209902222	AMEX		0
103	Jane	Plane	123456789	MC		0
103	Jane	Plane	333498703333	AMEX		0
10312	Jolly	Hershey	176896789	MC		0
10312	Jolly	Hershey	333300003333	AMEX		0
10323	Grumpy	youaretheweakestlink	673834489	MC		0
10323	Grumpy	youaretheweakestlink	33413003333	AMEX		0
15603	Peter	Sand	123609789	MC		0
15603	Peter	Sand	338893453333	AMEX		0
15613	Joesph	Something	33843453533	AMEX		0

综合SQL 注入

- 1.启动 owasp bwa v1.2 虚拟机（启动后自动加载WebGoat 5.4 网站应用），查看其主机ip（下面以 10.10.10.135为例）
- 2.通过任意浏览器访问 webgoat 5.4 应用，链接为：<http://10.10.10.135/WebGoat/attack>，认证用户名 webgoat，密码为webgoat；然后点击左侧导航Injection Flaws——LAB:SQL Injection。
- 3.尝试完成Stage 1。利用sql注入实现旁路认证。先试用表单，以Neville和任意密码login。可见“login failed”。



[Introduction](#)
[General](#)
[Access Control Flaws](#)
[AJAX Security](#)
[Authentication Flaws](#)
[Buffer Overflows](#)
[Code Quality](#)
[Concurrency](#)
[Cross-Site Scripting \(XSS\)](#)
[Improper Error Handling](#)
[Injection Flaws](#)

✓ [Command Injection](#)
✓ [Numeric SQL Injection](#)

✓ [Log Spoofing](#)

[XPath Injection](#)

✓ [String SQL Injection](#)

[LAB: SQL Injection](#)

[Stage 1: String SQL Injection](#)

[Stage 2: Parameterized Query #1](#)

[Stage 3: Numeric SQL Injection](#)

[Stage 4: Parameterized Query #2](#)

[Modify Data with SQL Injection](#)

[Add Data with SQL Injection](#)

[Database Backdoors](#)

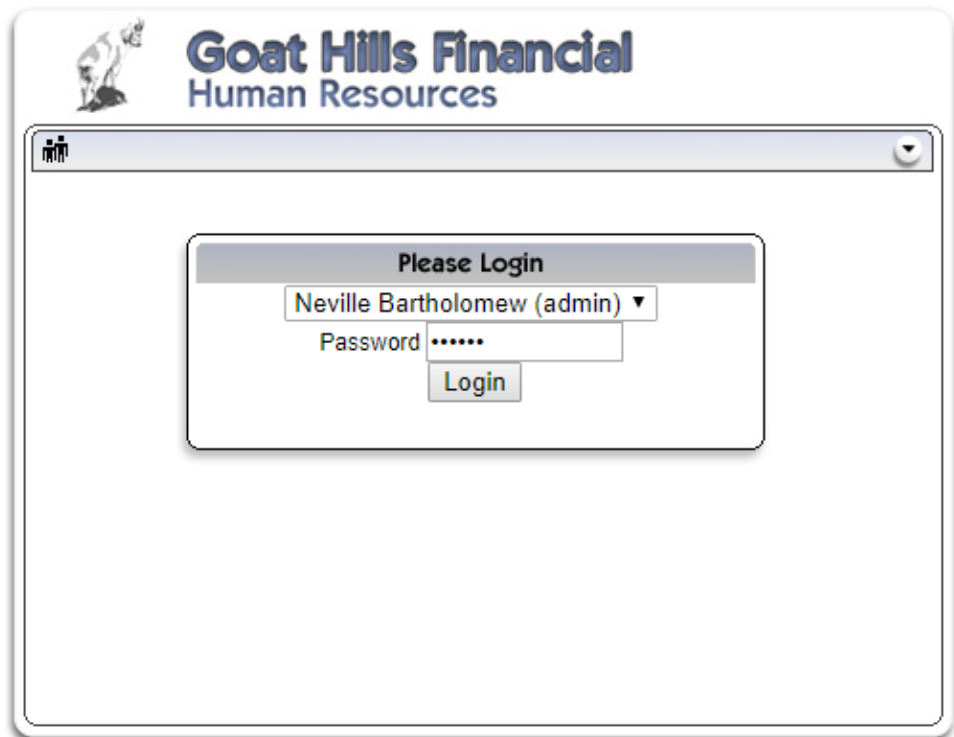
Solution Videos

[Restart this Lesson](#)

Stage 1

Stage 1: Use String SQL Injection to bypass authentication. Use SQL injection to log in as the boss ('Neville') without using the correct password. Verify that Neville's profile can be viewed and that all functions are available (including Search, Create, and Delete).

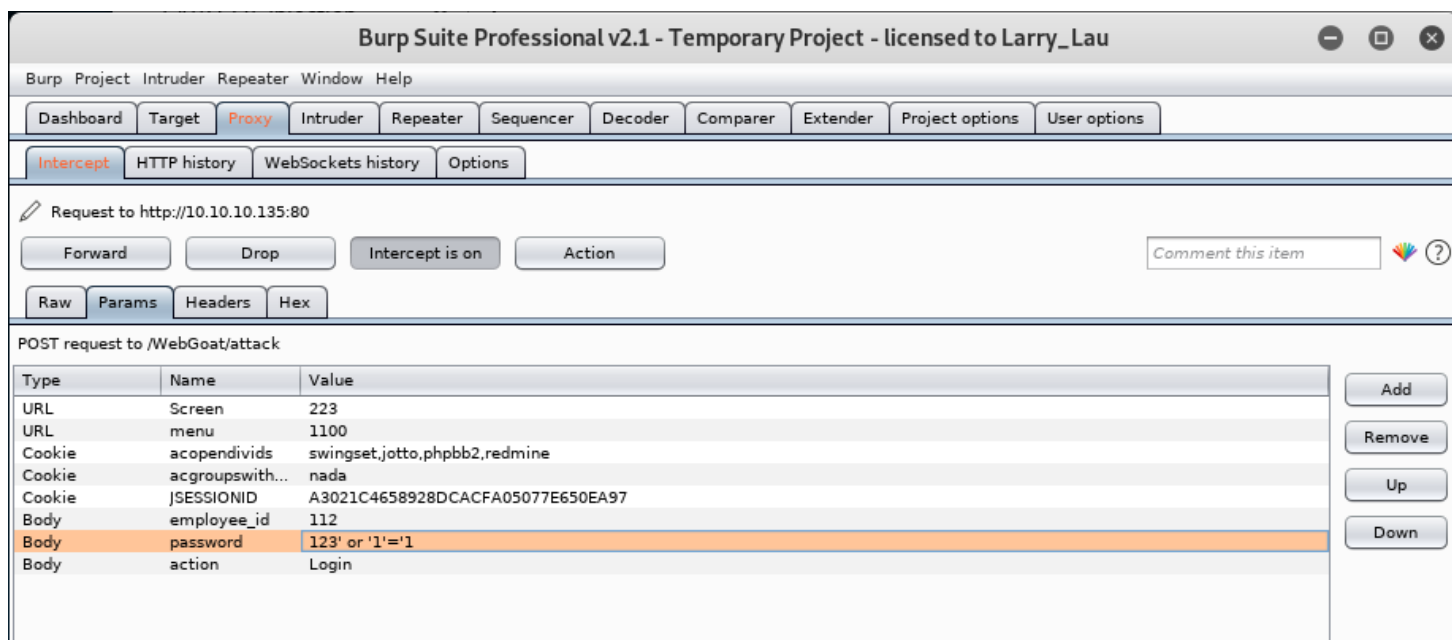
* **Login failed**



4.启动Kali 2019 虚拟机中打开设置了burpsuite proxy的浏览器firefox和Burp suite。通过firefox浏览器访问相关链接，并用burp suite拦截请求，在password参数值处进行sql注入。

注入内容可以如下：

```
123' or 1=1 or '1'='1
```



5.注入完成后，点击浏览器页面上的“login”，可以看到第一阶段成功。

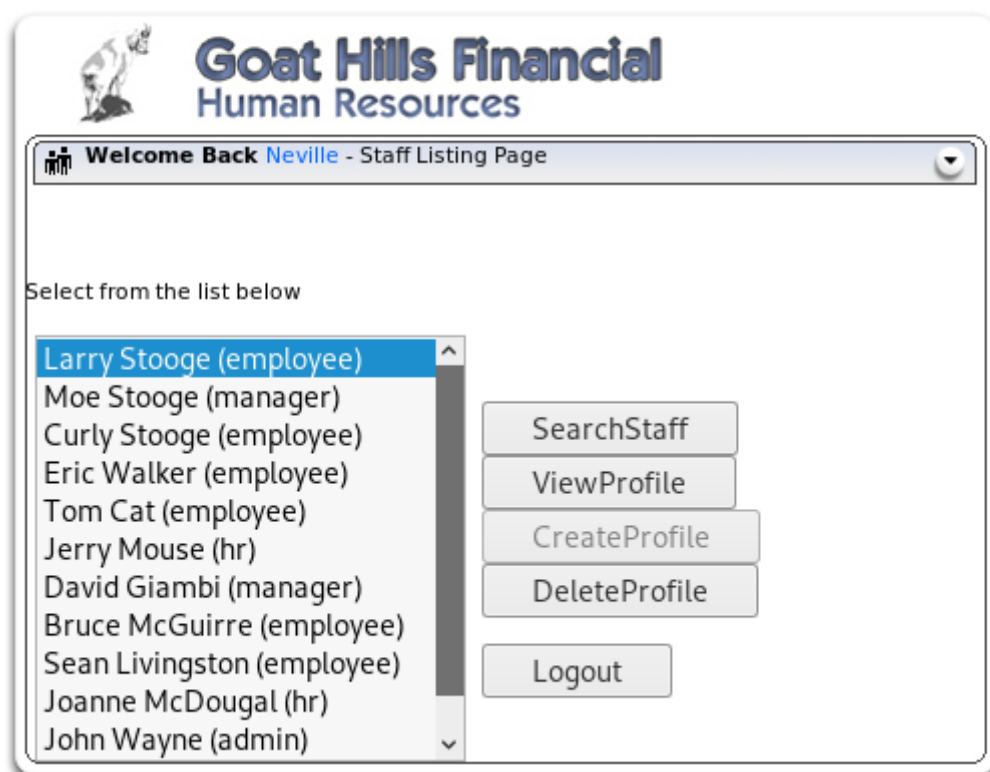
Stage 2

Stage 2: Block SQL Injection using a Parameterized Query.

THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT

Implement a fix to block SQL injection into the fields in question on the Login page. Repeat stage 1. Verify that the attack is no longer effective.

- * You have completed Stage 1: String SQL Injection.
- * Welcome to Stage 2: Parameterized Query #1



6.第二阶段，使用参数化查询方式修改源代码，使其能够防御一般的sql注入攻击。

过程略

7.第三阶段，首先要在密码输入处进行注入，以Larry用户身份登录。

注入内容为：

```
123' or '1'='1
```

LAB: SQL Injection

OWASP WebGoat v5.4

Show Params Show Cookies Lesson Plan

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws

Command Injection

Numeric SQL Injection

Log Spoofing

XPATH Injection

String SQL Injection

LAB: SQL Injection

Stage 1: String SQL Injection

Stage 2: Parameterized Query #1

Stage 3: Numeric SQL Injection

Stage 4: Parameterized Query #2

Modify Data with SQL Injection

Solution Videos

Stage 3

Stage 3: Execute SQL Injection to bypass authorization.
As regular employee 'Larry', use SQL injection into a parameter of the View function (from the List Staff page) to view the profile of the boss ('Neville').

* Login failed

Restart this Lesson

Goat Hills Financial
Human Resources

Please Login

Larry Stoooge (employee)

Password

Login

Burp Suite Professional v2.1 - Temporary Project - licensed to Larry_Lau

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

Intercept HTTP history WebSockets history Options

Request to http://10.10.10.135:80

Forward Drop Intercept is on Action

Raw Params Headers Hex

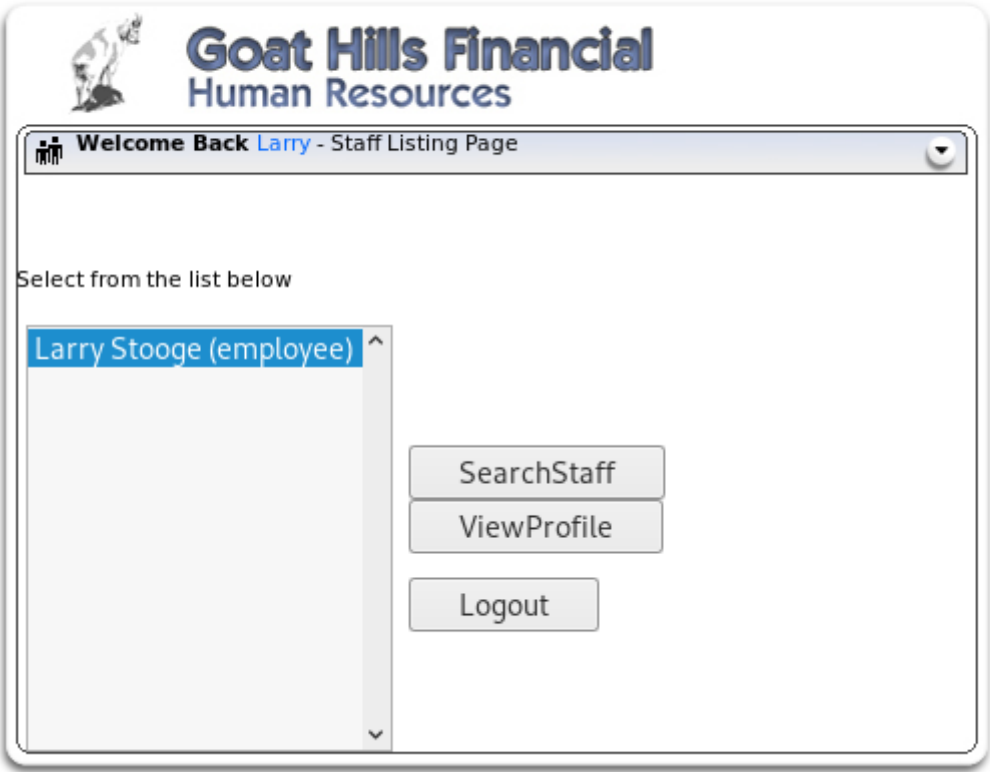
POST request to /WebGoat/attack

Type	Name	Value
URL	Screen	223
URL	menu	1100
Cookie	acopendivids	swingset,jotto,phpbb2,redmine
Cookie	acgroupswith...	nada
Cookie	JSESSIONID	A3021C4658928DCACFA05077E650EA97
Body	employee_id	101
Body	password	123' or 'a'='a
Body	action	Login

Add Remove Up Down

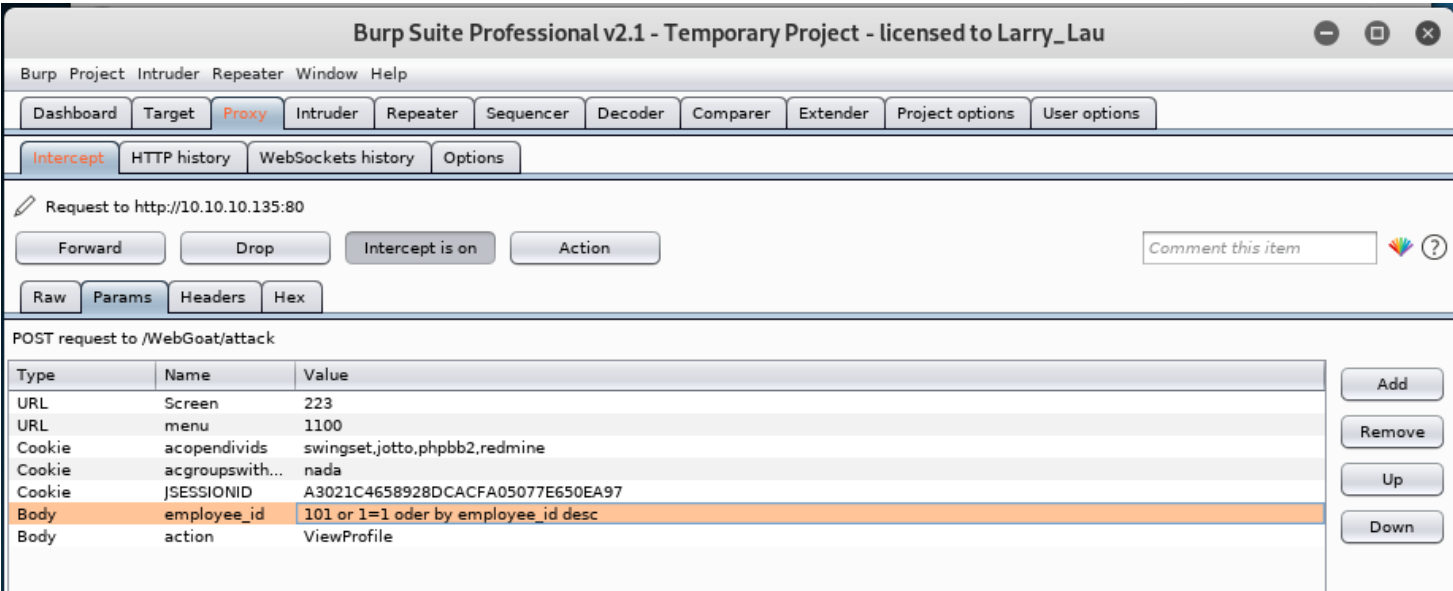
8.然后在选择点击 View Profile ，并使用burp进行拦截。

Stage 3
Stage 3: Execute SQL Injection to bypass authorization.
As regular employee 'Larry', use SQL injection into a parameter of the View function (from the List Staff page) to view the profile of the boss ('Neville').




9.在 employee_id 参数处注入下面内容：

101 or 1=1 order by employee_id desc




Implement a fix to block SQL injection into the relevant parameter. Repeat stage 3. Verify that access to Neville's profile is properly blocked.

- * You have completed Stage 3: Numeric SQL Injection.
- * Welcome to Stage 4: Parameterized Query #2



Goat Hills Financial
Human Resources

 **Welcome Back** [Larry](#)

First Name:	Neville ✓	Last Name:	Bartholomew
Street:	1 Corporate Headquarters	City/State:	San Jose, CA
Phone:	408-587-0024	Start Date:	3012000
SSN:	111-111-1111	Salary:	450000
Credit Card:	4803389267684109	Credit Card Limit:	300000
Comments:		Manager:	112
Disciplinary Explanation:		Disciplinary Action Dates:	112005

ListStaff

EditProfile

Logout

注入成功。

实验结论