TP Class Assignment - 06/May/2019

Each group must answer the questions of the following exercises in the Github area of their group until the end of 20/May/2019. For each day of delay, 0.15 points will be deducted from the grade of this assignment.

Note that these exercises should be done in the virtual machine provided.

WebGoat installation instructions, for those who do not use the available virtual machine:

In this lesson we will use a WebGoat Docker image. WebGoat is a deliberately insecure Web application maintained by OWASP, which is intended to be a complement to Web application security classes, and therefore contains common application failures.

You can install Docker directly on your computer's operating system.

If you want to install docker on your computer follow the instructions at https://store.docker.com/search?type=edition&offering=community for your operating system.

If your computer's operating system is the same as the virtual machine, follow the instructions below to install docker:

1. `sudo apt-get update`
2. `sudo apt-get install apt-transport-https ca-certificates curl gnupg2 software-properties-common`
3. `curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -`
4. `sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable"`
5. `sudo apt-get update`
6. `sudo apt-get install docker-ce`

Note: Whenever in the exercises below you are asked to use WebGoat, you must perform the following steps:

1. Ensure that you have already installed docker on your computer or virtual machine.
2. Obtain version 7.1 of WebGoat, through the command `sudo docker pull webgoat/webgoat-7.1`
3. Boot WebGoat with the `sudo docker run -p 8080:8080 -t webgoat/webgoat-7.1` command.

Note that WebGoat takes 30 - 50 seconds to start up, having started when a line appears on the screen containing the following information `INFO: Starting ProtocolHandler ["http-bio-8080"]`

5. In the virtual machine Firefox access http://localhost:8080/WebGoat.

At the end, to stop docker and WebGoat, you should make the following commands:

- `sudo docker ps` to get the Container ID
- `sudo docker kill <Container ID>` where `<Container ID>` is the value indicated in the previous command.

Exercises

1. *Injection*

Experience 1.1

Go to https://free.codebashing.com/courses/java and follow the *SQL Injection* example.

Experience 1.2

Go to https://free.codebashing.com/courses/dotnet and follow the *XXE Injection* example.

Question 1.1 - *String SQL Injection*

In WebGoat try to solve the exercise *Injection Flaws -> String SQL Injection*:

1. The goal is to use *SQL Injection* so that all credit cards are displayed. Experiment with some names and see how they behave.
2. Run the *SQL Injection* attack - try using a tautology.

In all exercises try first to solve without help. If you get to a point where you can not move forward, use "Show Hints" for help. You should only use "Show Solution" in extreme situations.

Question 1.2 - *Numeric SQL Injection*

In WebGoat try to solve the exercise *Injection Flaws -> Numeric SQL Injection*:

1. The goal is to use *SQL Injection* so that all weather data is displayed. Try some places and see how it behaves.
2. Run the *SQL Injection* attack - try using a tautology. The problem is that apparently we have no way to change the query ...
3. You may be able to use the Firefox development tools (upper right corner of the browser and select the "Developer" button, followed by the "Inspector" tool). But how do we modify the information that is sent when we click the "Go!" Button?
4. Maybe you can start by looking for "Columbia" in the code. Then change the HTML to perform the tautology.

In all exercises try first to solve without help. If you get to a point where you can not move forward, use "Show Hints" for help. You should only use "Show Solution" in extreme situations.

Question 1.3 - *Database Backdoors*

In WebGoat try to solve the exercise *Injection Flaws -> Database Backdoors*, one step at a time.

- Step 1

1. The goal is to use *SQL Injection* to run more than one SQL command for an account with ID 101. Try out how the application behaves with this ID.   2. Try to exploit the vulnerability to change the salary to a higher value.

Remember that SQL UPDATE has the following syntax:

UPDATE table_name SET column1 = value, column2 = value2, ... WHERE some_column = some_value

- Step 2

3. The goal is relatively simple. Carry out the attack with the indicated SQL command to change the salary to a higher value.

In all exercises try first to solve without help. If you get to a point where you can not move forward, use "Show Hints" for help. You should only use "Show Solution" in extreme situations.

2. XSS

Experience 2.1

Goto https://free.codebashing.com/courses/nodejs and follow the *Persistent (Stored) XSS* example.

Question 2.1 - *Reflected XSS*

In WebGoat try to solve the *Cross-Site Scripting (XSS) -> Reflected XSS Attacks* exercise:

1. The purpose is to use the purchase form to reflect the user input. How can you do it?
2. This form is very similar to what you can find on several websites. It has a list of products, prices and quantities. You can change the quantity and see the price changed. At the end, the user must provide the credit card and the CCV to pay for the selected goods.
3. By trial and error, try combination of letters and digits in the various input fields to see what type of validations the server makes. If you find any field with potential for Reflected XSS, try the following script: `<script>alert(" SSA !!! ")</ script>`

In all exercises try first to solve without help. If you get to a point where you can not move forward, use "Show Hints" for help. You should only use "Show Solution" in extreme situations.

Experience 2.2 - *Stored XSS*

In WebGoat try to solve the *Cross-Site Scripting (XSS) -> Stored XSS Attacks* exercise:

1. The goal is to leave a message on your favorite site that triggers an action when another user reads your message. How can you do it?
2. Perform some tests on the message form to verify the type of validation and filtering of the input that is made.
3. If you feel you have potential for Stored XSS, try the following script: `<script language="javascript" type="text/javascript">alert(document.cookie);</script>` which will display cookies in a Popup.
4. Check if it worked.
5. How could an attacker benefit from this vulnerability?

In all exercises try first to solve without help. If you get to a point where you can not move forward, use "Show Hints" for help. You should only use "Show Solution" in extreme situations.

3. Broken Authentication

Question 3.1 - *Forgot Password*

In WebGoat try to solve the exercise *Authentication flaws -> Forgot Password*.

Authentication mechanisms usually have the ability to retrieve the password by answering a personal question. However, it is often simple to guess the answer to the question!

1. The WebGoat application allows users to recover their password if they can answer a question about their favorite color. Try the user "webgoat" and color "red" and see what happens.
2. Now try to attack another user. What users normally exist on a system? Of all these users, what account would you like to compromise? Try it.
3. How can you get the password for this account? Maybe you can try to hit the color by brute force ...

In all exercises try first to solve without help. If you get to a point where you can not move forward, use "Show Hints" for help. You should only use "Show Solution" in extreme situations.