|  |  |
| --- | --- |
| Juiceshop Walkthrough Diploma in CSF  Year 3 Apr 2022 (Semester 5) | Week 2 |
| Practical |
| Kill Chatbot | |

**OBJECTIVES**

* To extract sensitive information about Juiceshop from public websites
* OWASP vulnerabilities: Using components with known vulnerabilities and Vulnerable and Outdated components (A09:2017, A06:2021)

**Challenge – Kill Chatbot**

* To permanently disable the support chatbot so that it can no longer answer customer queries
* Kill Chatbot - Difficulty: 5/6 stars

**Step 1**

Firstly, we will have to login to an account to access the chatbot. We will employ the same SQL injection method as in the Cross-Site Imaging challenge, where we add the string ’ or 1=1-- into the email field and anything else into the password field. This allows us to login to an admin account. To get to the chatbot, the hamburger menu on the top left of the site can be clicked, followed by Support Chat.

Graphical user interface, website

Description automatically generated

This will open the chatbot interface, with the chatbot asking for a name.

Graphical user interface, text, application, chat or text message

Description automatically generated

**Step 2**

The additional hints for this challenge state that understanding how the chatbot works would help us in completing it, hinting that we need to find the source code for this chatbot. Luckily, this can be found very easily on the GitHub page for Juice Shop at <https://github.com/juice-shop/juicy-chat-bot>.

By inspecting the index.js file for this chatbot, we find this line that is highlighted which allows the chatbot to set the username of the user. We can also notice that there is an apparent lack of sanitization for the input taken in by the chatbot, which we can exploit.

Graphical user interface, text, application

Description automatically generated

**Step 3**

A few lines below this, we can notice the function that is used to create a response to the user’s queries, which relies on an outside function named process. We can make use of this knowledge in combination with the knowledge that there is no input sanitization to our advantage to solve this challenge.

Graphical user interface, text, application, email

Description automatically generated

**Step 4**

By simply coming up with a username to give to the chatbot such as admin"); process=null; users.addUser("1337", "test when it asks for our name, the line users.addUser("token", "name") would when executed lead to these 3 lines being executed instead

users.addUser("token", "admin");

process = null;

users.addUser("1337", "test")

The first line will give a genuine answer to the chatbot on what our name is, in this case admin is our username. The second line will allow us to reset the process function to a null variable, which will lead to an error when the respond function attempts to call the process function. This will lead to an error every time the user tries to query the bot. Lastly, the third line is just there to ensure that the second line is able to be executed as there is still a “) after the input that we give, which would result in an error if we just gave the input of the first two lines.

Graphical user interface, application

Description automatically generated

After submitting this string, all it takes is another query of any string from us to break the chatbot and render it unable to answer queries. With this, we have successfully finished this challenge.

**~ End of Challenge ~**