Identifying SSTI

Before exploiting an SSTI vulnerability, it is essential to successfully confirm that the vulnerability is present. Furthermore, we need to identify the template engine the target web application uses, as the exploitation process highly depends on the concrete template engine in use. That is because each template engine uses a slightly different syntax and supports different functions we can use for exploitation purposes.

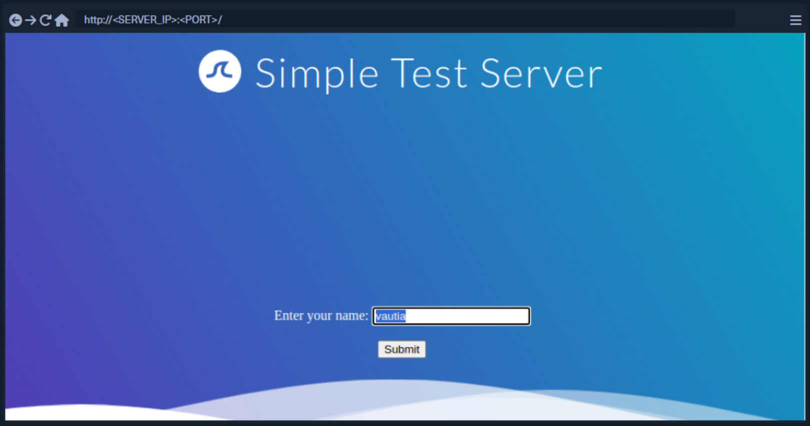
Confirming SSTI

The process of identifying an SSTI vulnerability is similar to the process of identifying any other injection vulnerability, such as SQL injection. The most effective way is to inject special characters with semantic meaning in template engines and observe the web application's behavior. As such, the following test string is commonly used to provoke an error message in a web application vulnerable to SSTI, as it consists of all special characters that have a particular semantic purpose in popular template engines:

```
${{<%[%'"}}%\.
```

Since the above test string should almost certainly violate the template syntax, it should result in an error if the web application is vulnerable to SSTI. This behavior is similar to how injecting a single quote (`'`) into a web application vulnerable to SQL injection can break an SQL query's syntax and thus result in an SQL error.

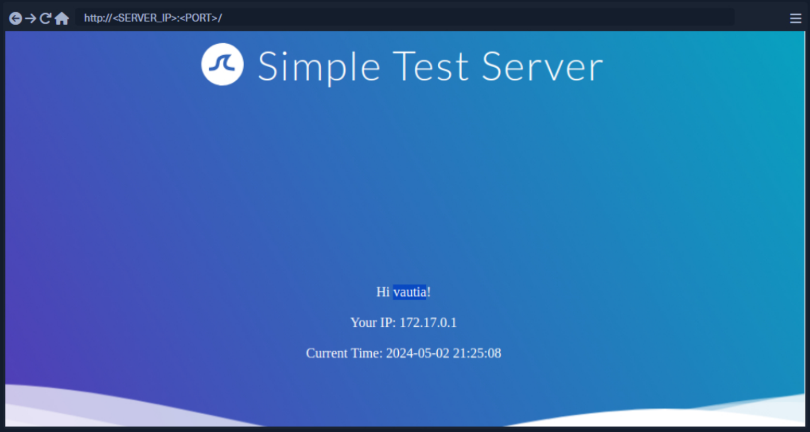
As a practical example, let us look at our sample web application. We can insert a name, which is then reflected on the following page:


http://<SERVER_IP>:<PORT>/

Simple Test Server

Enter your name: vautia

Submit


http://<SERVER_IP>:<PORT>/

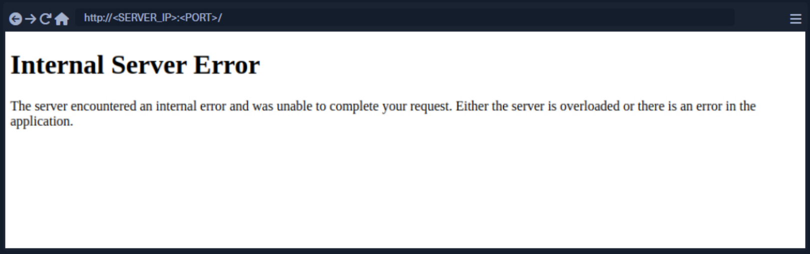
Simple Test Server

Hi vautia!

Your IP: 172.17.0.1

Current Time: 2024-05-02 21:25:08

To test for an SSTI vulnerability, we can inject the above test string. This results in the following response from the web application:


http://<SERVER_IP>:<PORT>/

Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

As we can see, the web application throws an error. While this does not confirm that the web application is vulnerable to SSTI, it should increase our suspicion that the parameter might be vulnerable.

Identifying the Template Engine

To enable the successful exploitation of an SSTI vulnerability, we first need to determine the template engine used by the web application. We

Cheat Sheet

? Go to Questions

Table of Contents

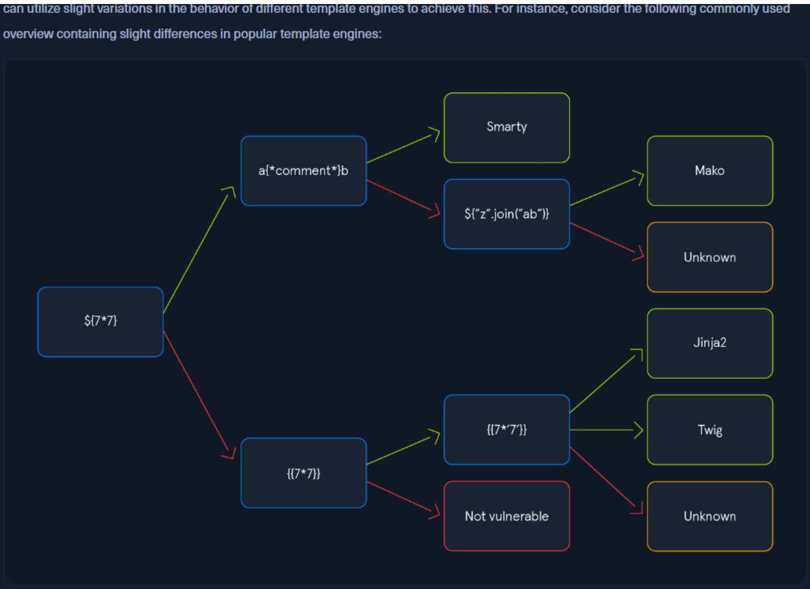
My Workstation

OFFLINE

Start Instance

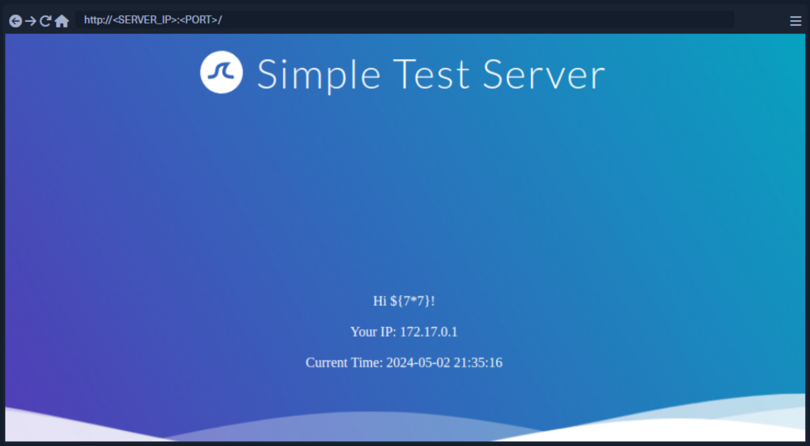
∞ / 1 spawns left

can utilize slight variations in the behavior of different template engines to achieve this. For instance, consider the following commonly used overview containing slight differences in popular template engines:



We will start by injecting the payload `${7*7}` and follow the diagram from left to right, depending on the result of the injection. Suppose the injection resulted in a successful execution of the injected payload. In that case, we follow the green arrow; otherwise, we follow the red arrow until we arrive at a resulting template engine.

Injecting the payload `${7*7}` into our sample web application results in the following behavior:

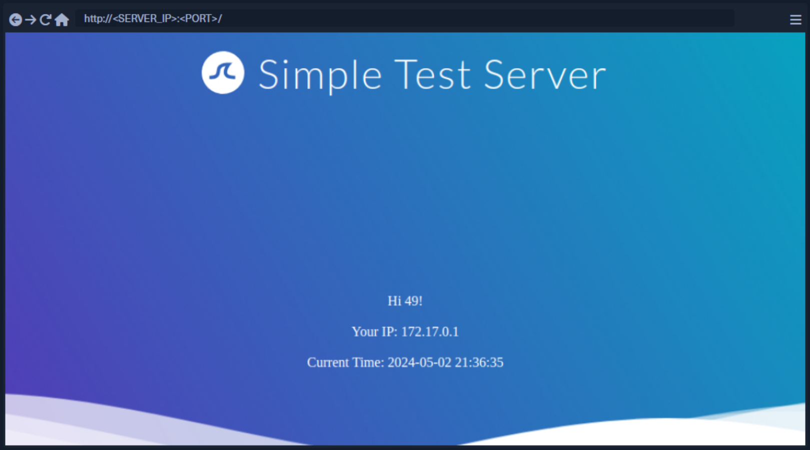


Hi ${7*7}!

Your IP: 172.17.0.1

Current Time: 2024-05-02 21:35:16

Since the injected payload was not executed, we follow the red arrow and now inject the payload `{{7*7}}`:



Hi 49!

Your IP: 172.17.0.1

Current Time: 2024-05-02 21:36:35

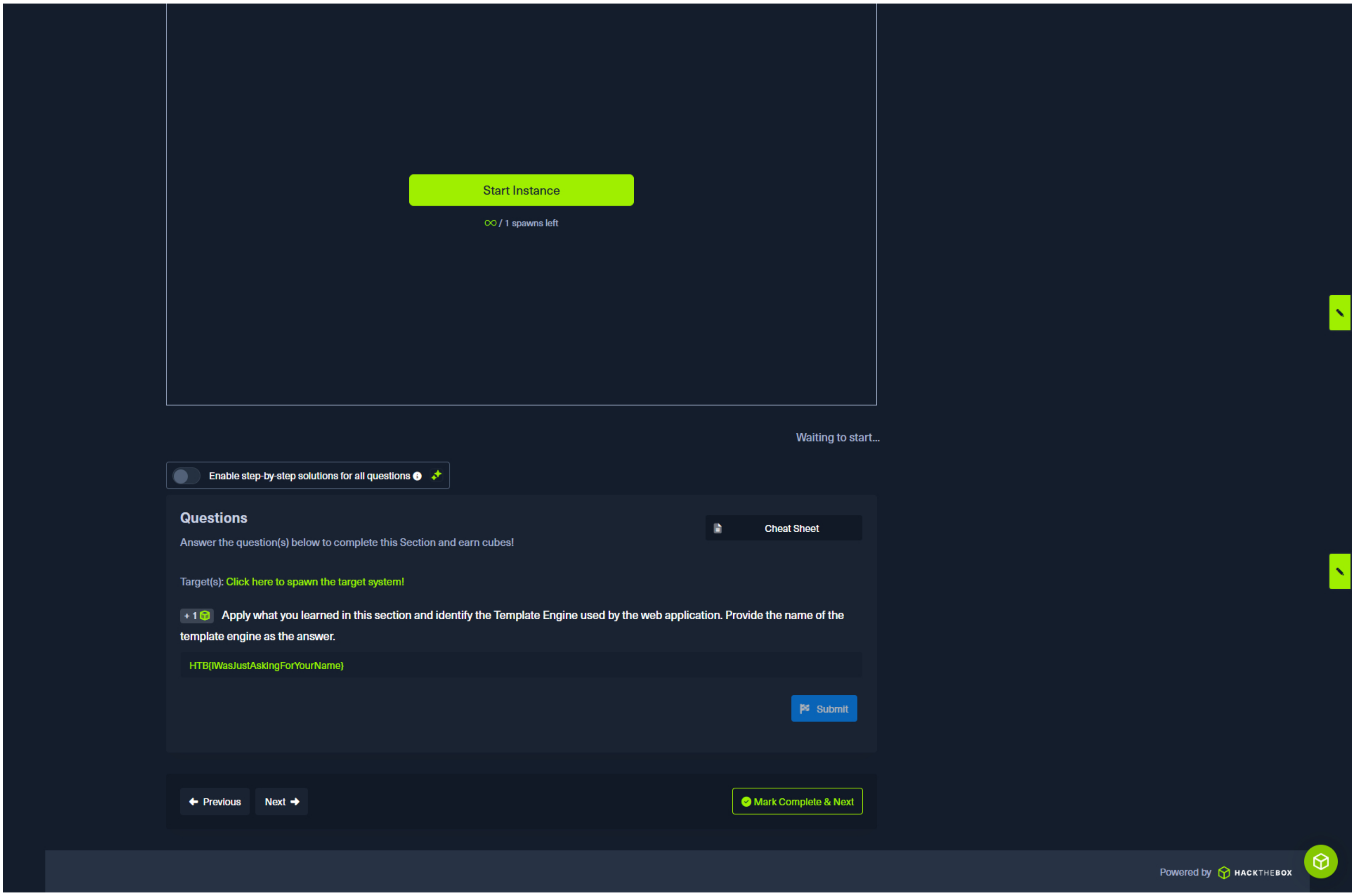
This time, the payload was executed by the template engine. Therefore, we follow the green arrow and inject the payload `{{7*'7'}}`. The result will enable us to deduce the template engine used by the web application. In Jinja, the result will be `7777777`, while in Twig, the result will be `49`.

Connect to Pwnbox
Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK 150ms

ⓘ Terminate Pwnbox to switch location



Questions

Answer the question(s) below to complete this Section and earn cubes!

Target(s): Click here to spawn the target system!

+ 1 Apply what you learned in this section and identify the Template Engine used by the web application. Provide the name of the template engine as the answer.

HTB{IWasJustAskingForYourName}