What is SSTI?

SSTI (Server-Side Template Injection) is a vulnerability where user input is embedded into a server-side template and then evaluated, so an attacker injects template syntax that the engine executes. Because template engines can reach language objects and functions, SSTI frequently escalates to remote code execution. It is caused by concatenating untrusted input into a template instead of passing it as data, and the fix is to keep user input strictly as rendered data, never template source.

HOW IT WORKS

01 How it works and example

The classic probe is a math expression that only a template engine would evaluate:

- Send `{{7*7}}`. If the response contains 49, the input is being evaluated as a template (the hallmark SSTI test).
- Identify the engine, then escalate. On Jinja2, attackers walk Python objects to reach OS commands, for example via `{{ ''.__class__... }}` gadget chains ending in `os.popen("id").read()`.
- Other engines (Twig, Freemarker) have their own gadgets to reach code execution.

Examples shown for defensive context.

> THE {{7*7}} TEST
>
> *If injecting* `{{7*7}}` *(or* `${7*7}`*) returns* 49*, the application is evaluating your input as a template, a strong SSTI signal that usually leads to code execution.*

HOW TO DEFEND

- Never build templates from user input. Pass user data as context variables to a static template, so it is rendered as data, not code.
- Use logic-less or sandboxed templates where user-supplied templates are unavoidable, and keep the sandbox patched.
- Avoid letting users supply template content at all.
- Validate and contextually encode output.
- Test any feature that renders user-influenced content through a template engine.

SOURCES

[1] OWASP Web Security Testing Guide

[2] MITRE CWE-1336: Improper Neutralization of Special Elements Used in a Template Engine

[3] OWASP Top 10

Test your application for SSTI and 30+ other classes.

securelayer7.net/learn/application-security/server-side-template-injection

Open online