# jails - python

RCE: `{{ self.__init__.__globals__.__builtins__.__import__('os').popen('cat flag.txt','r').read()}}`
Link where I stumbled upon the self command: https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/18-Testing_for_Server-side_Template_Injection

Link where I stumbled upon the final RCE: https://podalirius.net/en/articles/python-vulnerabilities-code-execution-in-jinja-templates/#the-templatereference-object

1. At first I thought I had to pass in arguments inside the address `127.0.0.1:5000/?filter={{'abc'}}` which the server did not complain but I wasn't getting any visible results whatsoever.
2. I started looking into the app.py file and saw that you're using the `render_template_string` command, but I didn't end up finding anything to exploit with that command.
3. I also saw that the template variable ends at the end with `% name` , so I thought that you're using modulo to encrypt something in a cipher and I started searching about creating ciphers using the modulo operator in python but that did not lead to anything.
4. I started to think that if there is a form, the exploit would be injected through this form, so I started to search injections that used forms in jinja2.
5. I started looking into SSTI exploits in jinja2 when I stumbled upon the link that I referenced above, that's when I found out about the self parameter. I tested it and voila, it returned a TemplateReference None string in the "For debugging use only" section inside the inspect element.
6. Then I searched about the TemplateReference and how can I expand it and finally stumbled across the final link which I referenced above, that led me down using the system command, using ls at first, to list the contents of the directory, then seeing a file called flag.txt and opening it with cat.