Server-Side Template Injection

Jakub Czyszczonik

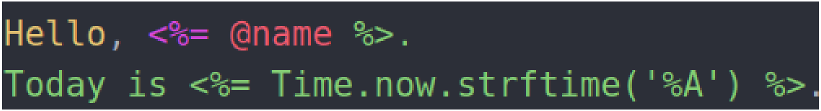
Template Engines

Why we use template engines:

- Generate static HTML page,
- Less (AJAX) calls for content,
- Simple and fast for development,
- Reusable.

Example Code

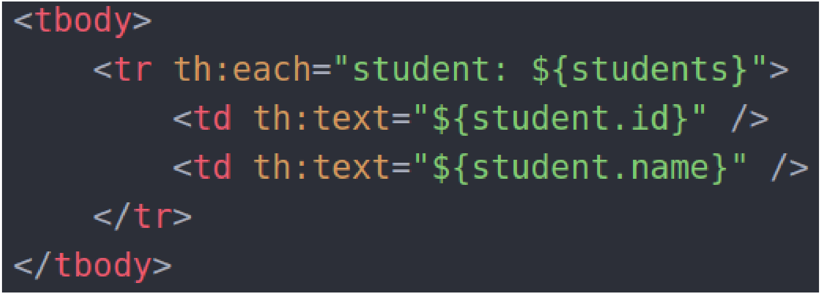
Ruby ERB:



```
Hello, <%= @name %>.
Today is <%= Time.now.strftime('%A') %>.
```

Example Code

Java Spring Thymeleaf:



```
<tbody>
    <tr th:each="student: ${students}">
        <td th:text="${student.id}" />
        <td th:text="${student.name}" />
    </tr>
</tbody>
```

Attack Stages

1. Detect

Attack Stages

1. Detect
2. Identify

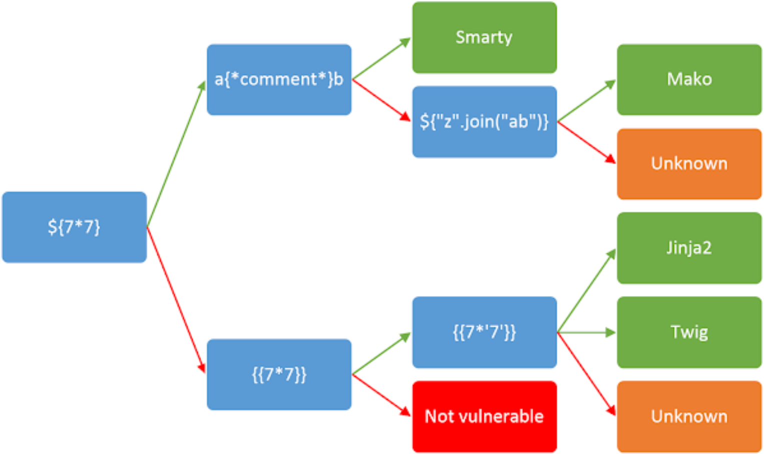
Attack Stages

1. Detect
2. Identify
3. Exploit

Identify

Engine	Injection code
jinja2	{{ %s }}
Mako	${ %s }
ERB	<%= %s %>
Dust	{# %s } or { %s } or {@ %s }
Velocity	#set($x=1+1)${x}

Identify Graph



https://portswigger.net/research/server-side-template-injection

Tplmap

Features:

- Detection and Identification vulnerability
- Run and execute shell command on the target
- Upload and Download files
- Can be used as a Burp Suite extension

Use cases

1. Remote Code Execution

Use cases

1. Remote Code Execution
2. Add, change, remove data or code in server

Use cases

1. Remote Code Execution
2. Add, change, remove data or code in server
3. Steal sensitive data

Use cases

1. Remote Code Execution
2. Add, change, remove data or code in server
3. Steal sensitive data
4. Shout down servers

Prevention

1. Resignation from templates

Prevention

1. Resignation from templates
2. Using Save Engines (Logic-less templates, save mode)

Prevention

1. Resignation from templates
2. Using Save Engines (Logic-less templates, save mode)
3. Sand-boxing (black / white list, processing special characters)

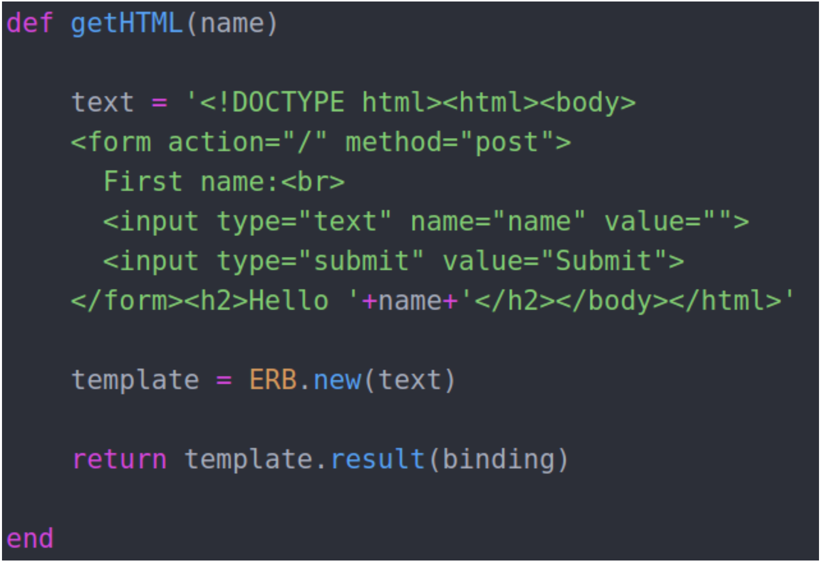
Prevention

1. Resignation from templates
2. Using Save Engines (Logic-less templates, save mode)
3. Sand-boxing (black / white list, processing special characters)
4. Hardening (Virtual environment, file access rights, well configured SELinux or grsecurity policies)

Good Practices

Defence in depth

Example Attack



```ruby
def getHTML(name)

    text = '<!DOCTYPE html><html><body>
    <form action="/" method="post">
      First name:<br>
      <input type="text" name="name" value="">
      <input type="submit" value="Submit">
    </form><h2>Hello '+name+'</h2></body></html>'

    template = ERB.new(text)

    return template.result(binding)

end
```

Bibliography

- Book: Bezpieczeństwo aplikacji webowych, Securitum 2019, INSB: 978-83-954853-0-5
- https://portswigger.net/research/server-side-template-injection
- https://ajinabraham.com/blog/server-side-template-injection-in-tornado
- https://github.com/epinna/tplmap
- https://github.com/DiogoMRSilva/websitesVulnerableToSSTI