Module 11 - Server Side Attacks

▼ Server side infrastructure

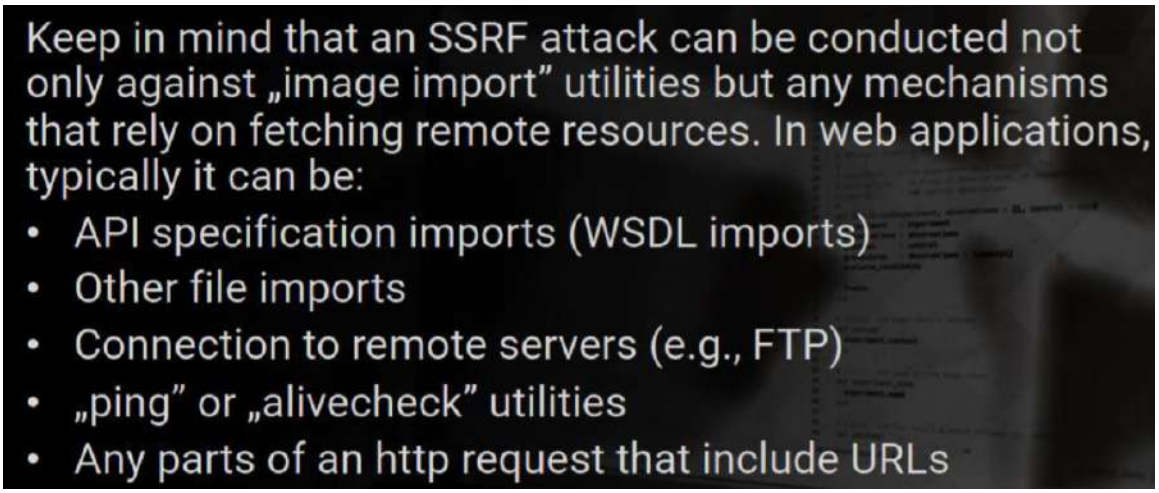
Understanding Modern Infrastructure

- Content Delivery Network (CDN)
- WAF / IDS
- Load Balancer

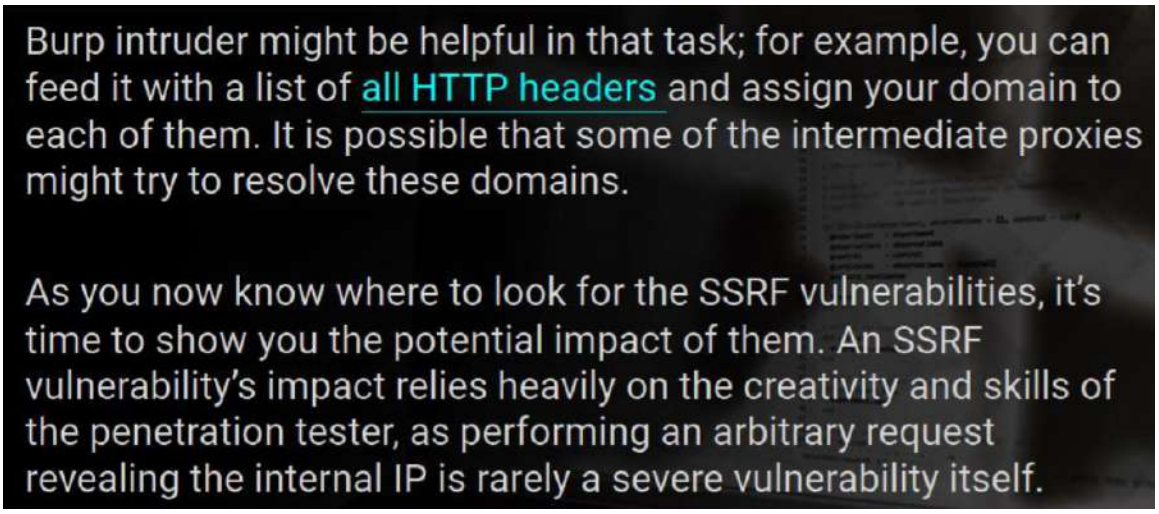
Abusing Intermediate Devices

▼ Server Side Request Forgery (ssrf)

- Most places to look for them:
  - load profile pictures from URL
  - or similar functionalities

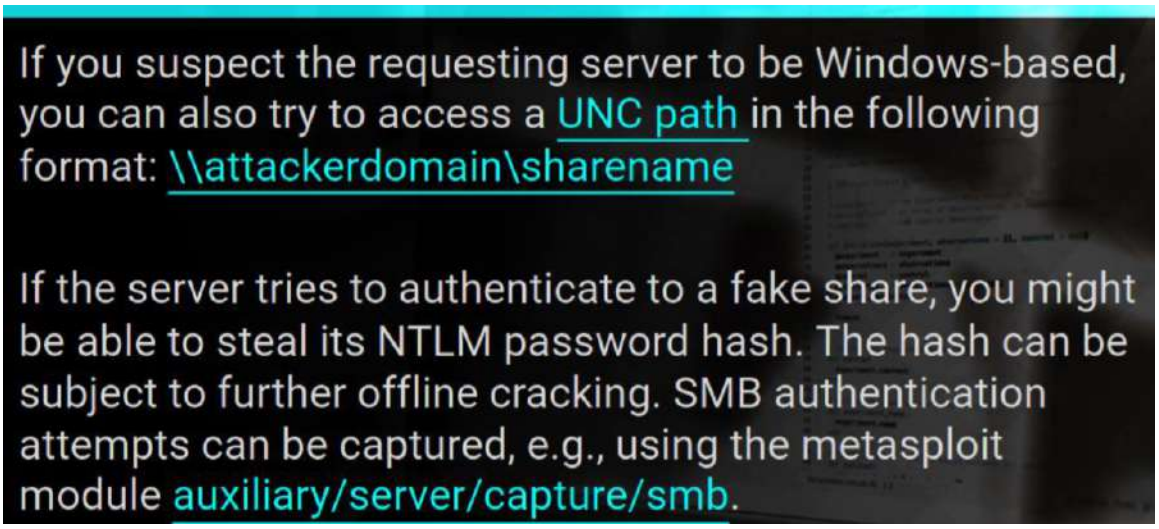


- I can use burp intruder to issue a blind ssrf request to my server using http headers



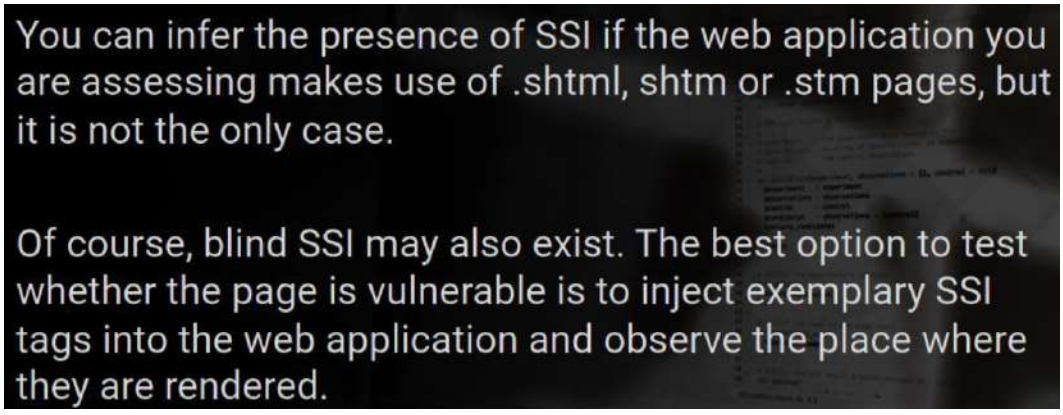
Abusing URL Structure

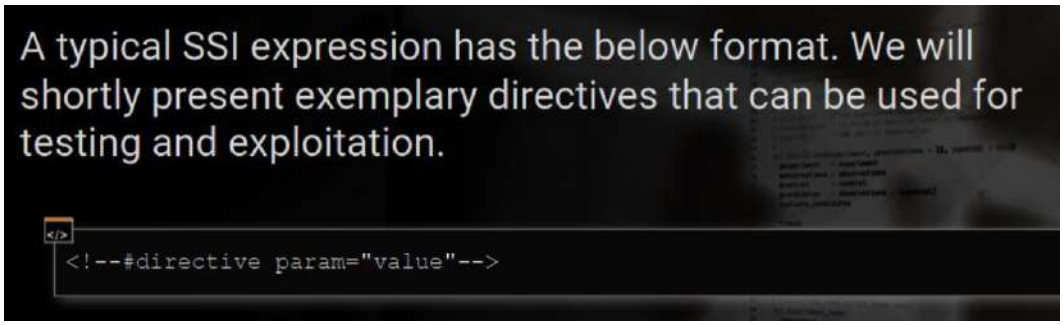
- Search URL schemas for PHP language



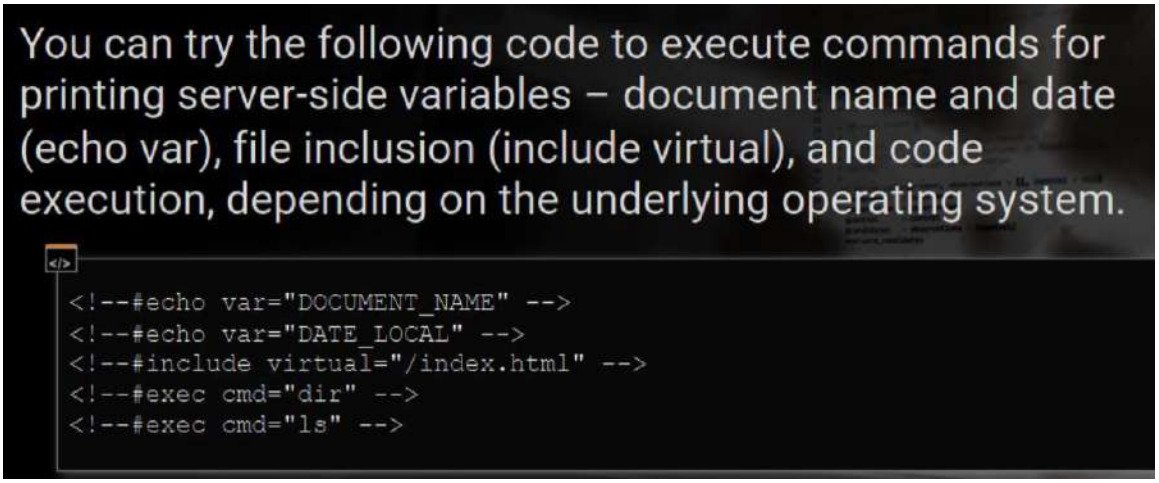
▼ Server Side Include

Can practice SSI in bWAPP (ubuntu)



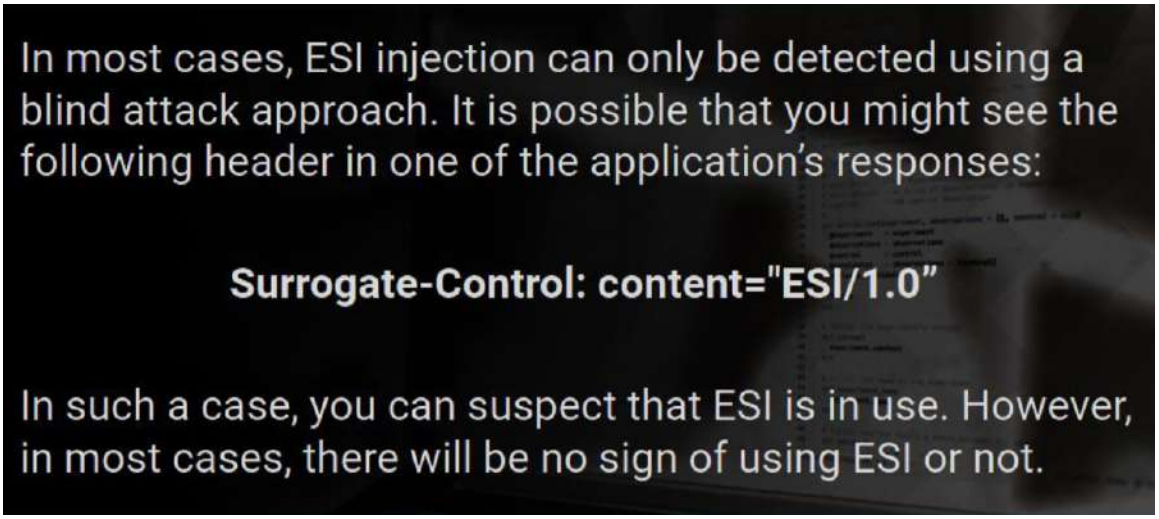


SSI Expressions

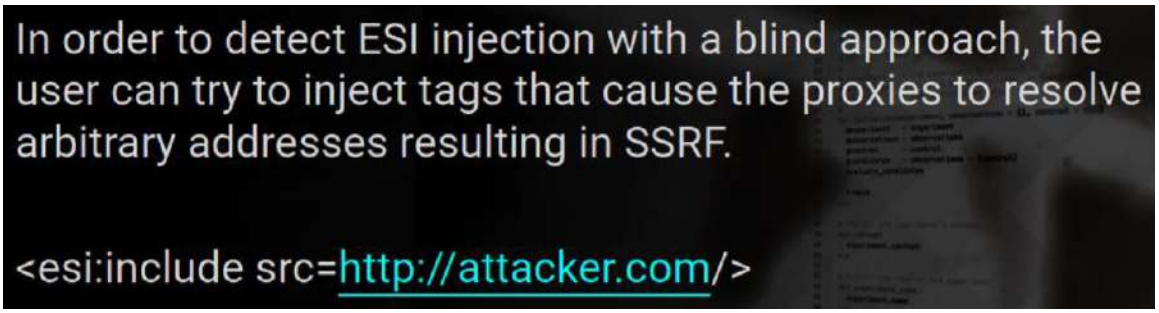


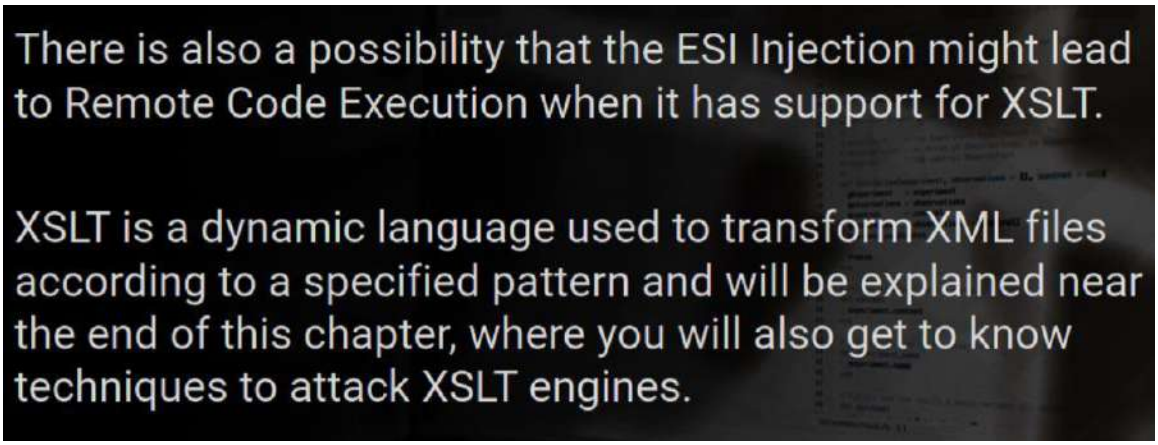
ESI Expressions

ESI Detection



ESI Exploitation



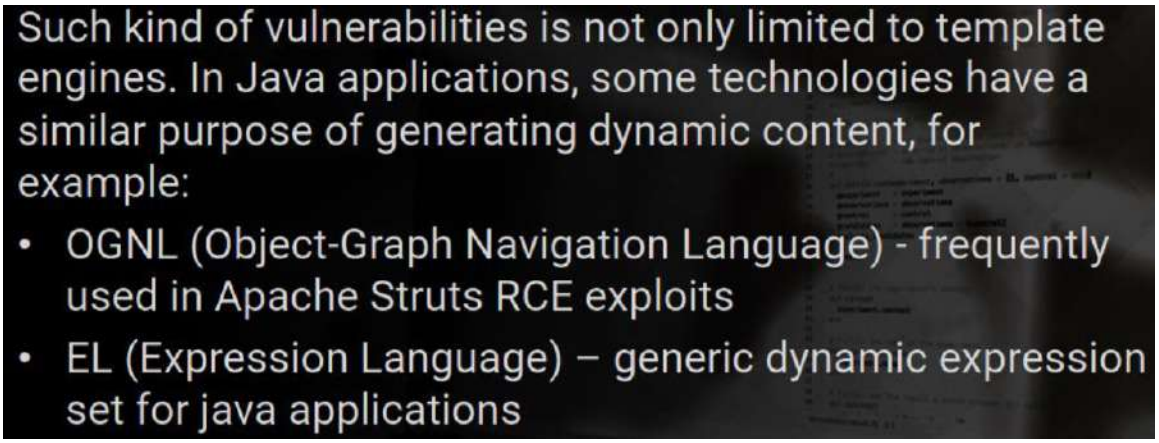


▼ language evaluation

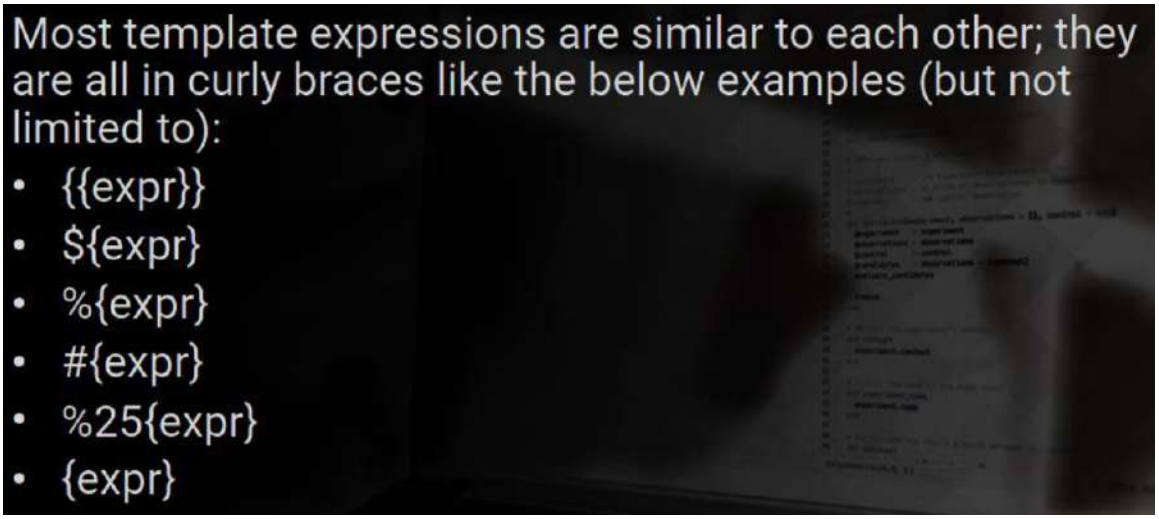
- Language evaluation includes vulnerabilities such as:
  - double evaluation
  - server-side template injections
  - expression language injections

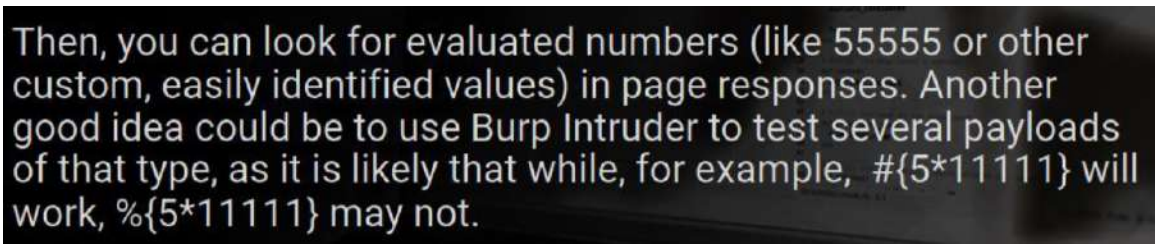
Template Engines

- PHP (Twig, Smarty)
- Python (Flask, Jinja)
- Java (Freemarker)



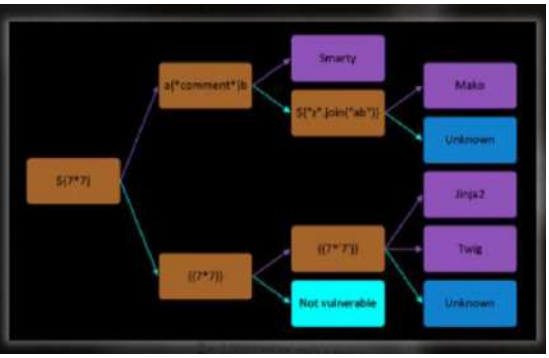
Detecting Template Injection





Confirming Template Injection

You can also use the following diagram to help you with profiling this type of vulnerability, whether it is a template or expression language injection.



To better identify the technology, you can first:

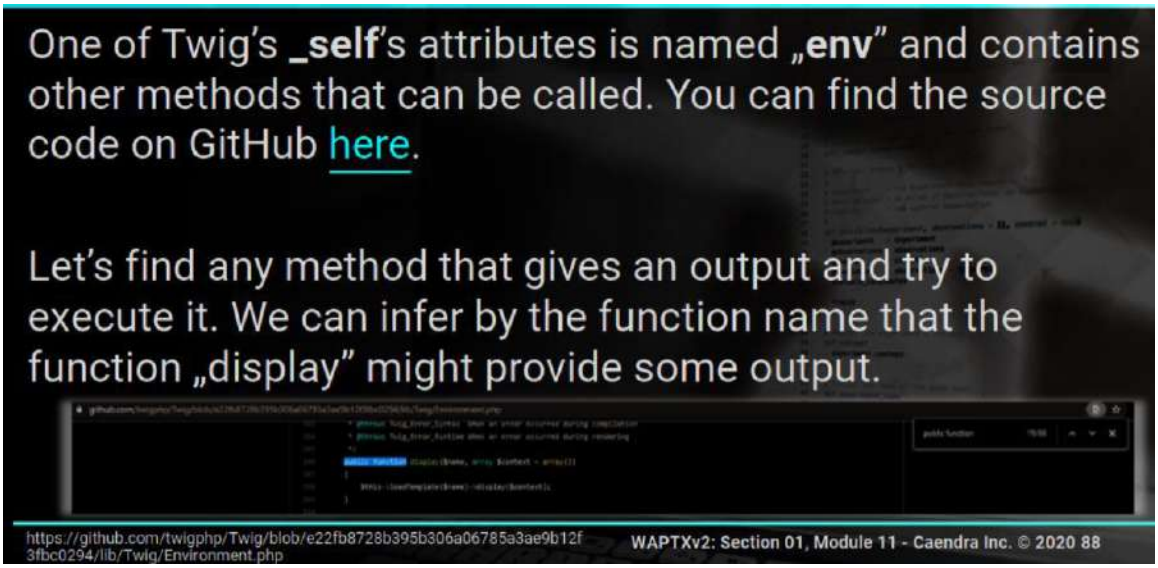
- Observe which is the generic technology of the application. If it is java (e.g., you see it uses .jsp extensions), then you can suspect it is an expression language / OGNL.
- Use the diagram from slide 79 as it contains popular behavior of template engines when handling expressions.
- Try to inject unclosed curly braces (be careful as there is a chance you might permanently disable the attacked webpage); this might provoke verbose error disclosing the underlying technology.
- Observe other verbose errors for technology names.

Exploiting Template Injection

For example, if you are dealing with the PHP template engine called Smarty, the RCE payload can be as simple as the one-liner below:

{php}echo `id`;{/php}

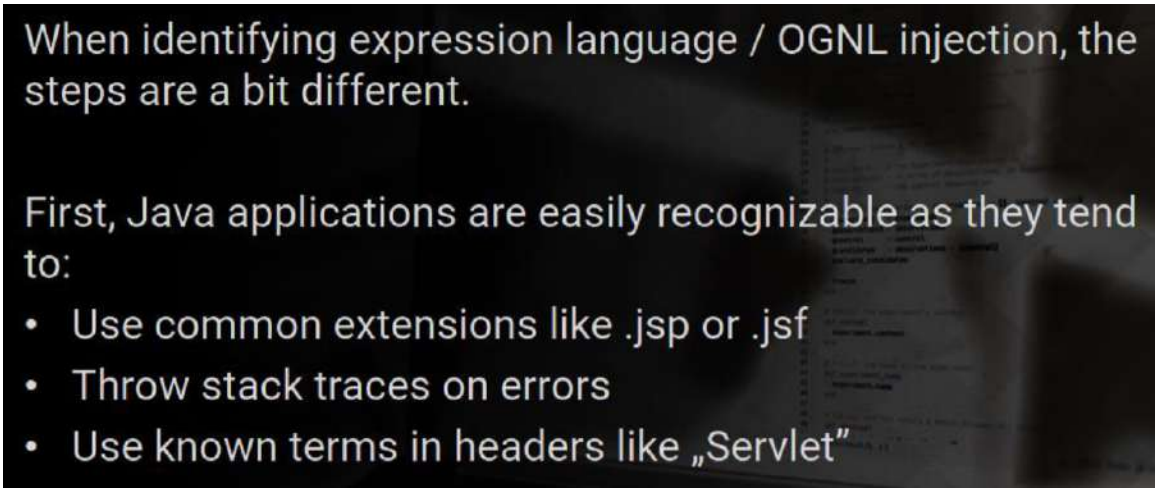

The Python engine Mako is also very straightforward, as we see below:
<%
import os
x=os.popen('id').read()
%>
${x}

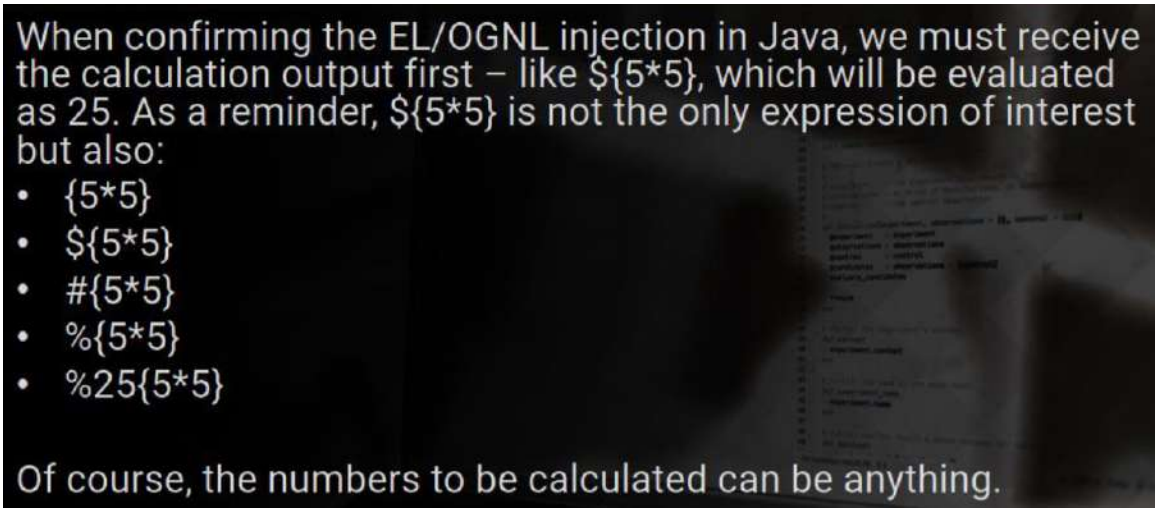

One of Twig's **_self**'s attributes is named „**env**" and contains other methods that can be called. You can find the source code on GitHub here.
Let's find any method that gives an output and try to execute it. We can infer by the function name that the function „display" might provide some output.
https://github.com/twigphp/Twig/blob/e22fb8728b395b306a06785a3ae9b12f3fbc0294/lib/Twig/Environment.php
WAPTXv2: Section 01, Module 11 - Caendra Inc. © 2020 88

##SSTI to XSS##


Apart from the aforementioned template injection vulnerability (of unknown impact) you can also observe an XSS vulnerability if you type in, for example:
{{<svg/onload=confirm(1)>}}
192.168.139.200 says
1
OK
Cancel

Expression Language





Attacking XSLT engines

- XSLT engine