

SSTI

BY UNCLE RAT



Agenda

- ▶ What is SSTI?
- ▶ Why SSTI?
- ▶ Different SSTI Contexts
- ▶ Attack strategy
 - ▶ Detection
 - ▶ Identification of the templating engine used
 - ▶ Exploiting SSTI



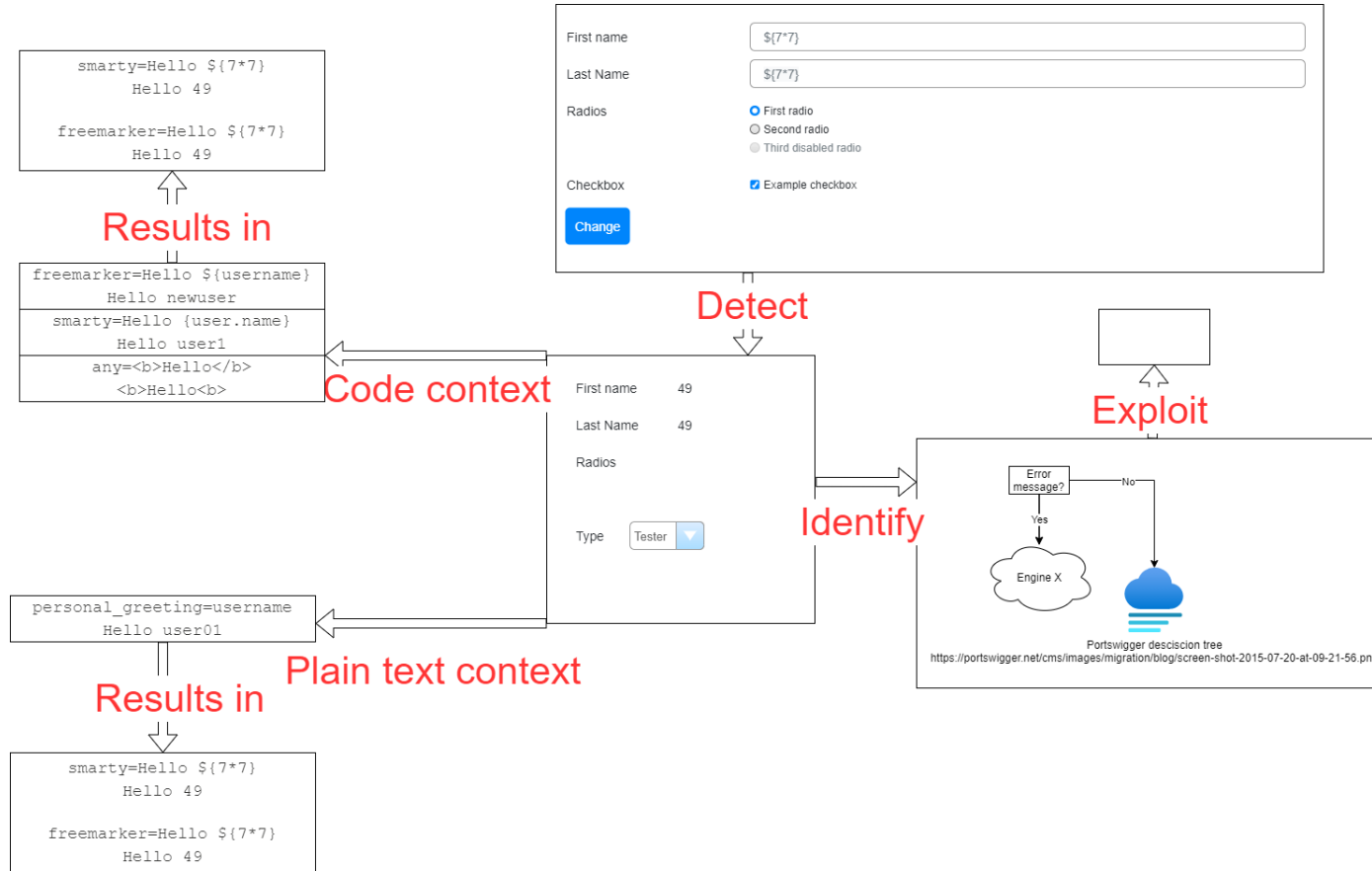
What is SSTI?



What is SSTI?

- ▶ Sometimes templating engines are being used
 - ▶ Twig - <https://twig.symfony.com/>
 - ▶ Tornado - <https://ajinabraham.com/blog/server-side-template-injection-in-tornado>
- ▶ Developers can sometimes feed unsanitised data to engines
- ▶ This allows for arbitrary code execution





Why SSTI?



Why SSTI?

- ▶ SSTI is very hard to detect
 - ▶ Often mistaken for XSS
 - ▶ Often missed fully
- ▶ Can easily lead to Remote Code Execution



Identify the
templating
engine



Identify the templating engine

- ▶ Sometimes templating engines give an error when you fuzz them
- ▶ If they don't follow the flowchart laid out by albinowax over at
 - ▶ <https://portswigger.net/research/server-side-template-injection>



Attack strategy - Detection



Attack strategy - Detection

- ▶ XSS similarities
- ▶ Insert attack vector into every field possible
 - ▶ $\{\{7*7\}$
- ▶ If it resolves we might have an SSTI or CSTI
 - ▶ Now we identify the templating engine



Exploiting SSTI



Exploiting SSTI

- ▶ READ THE MANUAL
- ▶ Key areas of interest are:
 - ▶ 'For Template Authors' sections covering basic syntax.
 - ▶ 'Security Considerations' - chances are whoever developed the app you're testing didn't read this, and it may contain some useful hints.
 - ▶ Lists of builtin methods, functions, filters, and variables.
 - ▶ Lists of extensions/plugins - some may be enabled by default.



Exploiting SSTI

- ▶ Smarty: `{php}echo `id`;{/php}`
 - ▶ Very simple executing `id` command on target
- ▶ Mako:

```
<%  
import os  
x=os.popen('id').read()  
%>  
${x}
```
- ▶ Do you research into other templating engines and what kind of PoC you need for those

