Remote Code Execution from SSTI in the Sandbox: Automatically Detecting and Exploiting Template Escape Bugs

Yudi Zhao, Yuan Zhang, and Min Yang, *Fudan University*

https://www.usenix.org/conference/usenixsecurity23/presentation/zhao-yudi

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.

USENIX'23 Artifact Appendix: <Remote Code Execution from SSTI in the Sandbox: Automatically Detecting and Exploiting Template Escape Bugs>

Yudi Zhao[1,¶], Yuan Zhang[1,¶], and Min Yang[1]

[1]*School of Computer Science, Fudan University, China*
[¶]*co-first authors*

A Artifact Appendix

A.1 Abstract

This artifact is a tool named TEFuzz for paper <Remote Code Execution from SSTI in the Sandbox: Automatically Detecting and Exploiting Template Escape Bugs>. TEfuzz can detect the template escape bugs in the template engine and generate exploit synthesis.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The artifact don't have any risk for evaluators while executing your artifact to their machines security, data privacy or others ethical concerns. Evaluators can run this artifact with confidence.

A.2.2 How to access

Evaluators can access the artifact and source code by github repository (https://github.com/seclab-fudan/TEFuzz).

A.2.3 Hardware dependencies

None

A.2.4 Software dependencies

The TEFuzz part of the artifact run under ubuntu 18.04 and require support for Python3.8 and related libraries, a list of which is included in the *requirements.txt* file in the github repository. The TE driver part of the artifact needs to run in Apache2 and PHP7.2.34 environment. We provide a complete docker image on github repository, and evaluators can directly pull the docker image without manual configuration.

A.2.5 Benchmarks

None

A.3 Set-up

A.3.1 Installation

First clone the source code from the Github repository, set up Python3.8 environment.

```
git clone https://github.com/seclab-fudan/TEFuzz
sudo apt-get update
sudo apt-get install python3.8 python3-pip
python3.8 -m pip install -r requirements.txt
cd $YOUR_TEFUZZ_PATH/CodeWrapper
composer install
sed -i 's/protected $attributes;/public
    $attributes;/g' vendor/nikic/php-parser/lib/
    PhpParser/NodeAbstract.php
```

Listing 1: Bash command

Then pull the docker image , evaluators need to mount docker's '/var/www/html/tefuzz' directory to the host so that tools can read the information.

```
docker pull altm4nz/tefuzz:1.0
docker run -itd -p 80:80 -v /var/www/html/tefuzz:/
    var/www/html/tefuzz --name tefuzz altm4nz/
    tefuzz:1.0
docker cp tefuzz:/tmp/tefuzz/ /var/www/html/
docker cp tefuzz:/tmp/seed/ $YOUR_TEFUZZ_PATH/
    result/
docker exec -it tefuzz /bin/bash -c 'service
    apache2 start'
```

Listing 2: Docker command

A.3.2 Basic Test

After installing the artifact, evaluators can perform a basic test using the *'python3 check.py '* command to determine whether the artifact was successfully installed. If *success* is displayed, the evaluators can continue the follow test evaluation.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): TEFuzz can detect template escape bugs in the template engine of the data set and generate exploit synthesis.

A.4.2 Experiments

(E1): [Vulnerability detection] [30 human-minutes + 50 compute-hour + 5GB disk]
Preparation: After the artifact have been successfully installed, set *TARGET_IP* as your docker ip, set *TE_NAME* in *'config.py'* to the target template engine name, such as 'smarty'.
Execution: Run the artifact using *'python3 main.py'*. Detecting a different template engine requires re-running the artifact after replacing the *TE_NAME* parameter.
Results: After completion of the run, the TEFuzz will output the number of generated exploit synthesises, this result corresponding to the third column of Table 2 in the paper. For example, "generate 3 EXP" means 3 exploit synthesises were generated. Meanwhile, TEFuzz will output intermediate data in the process of vulnerability detection, including the number of seeds, the number of Testcases generated, the number of interesting Testcases , etc., which correspond to Table 6 in the paper.

A.5 Notes on Reusability

If you want to evaluate more template engines by using my artifact, first you need to collect the seed data for that template engine, store it in the *result/seed* directory, and configure the appropriate adaption rules (optional). The most important thing is that you need to set up a TE driver environment for the new template engine, The TE driver environment can refer to the environment in the docker image.

If you want to test new seeds in an existing template engine, it's easier to just update your seeds into the *result/seed* directory.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2023/.