Instructions to replicate the Empirical Results for RQ2

These instructions refer to the replicability package for the paper

P. X. Mai, F. Pastore, A. Goknil and L. Briand, "Metamorphic Security Testing for Web Systems," 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), 2020, pp. 186-197, doi: 10.1109/ICST46399.2020.00028.

The proposed approach was applied to discover vulnerabilities in two case studies: an industrial Web system (IWS) and Jenkins, an open source system. We tested the two systems against the metamorphic relations (MRs) that target the vulnerabilities affecting them (4 for IWS and 8 for Jenkins).

Our replicability package does not include IWS data because of confidentiality restrictions. The following instructions describe how to replicate the empirical evaluation results for the Jenkins case study.

1. Configure the Jenkins server

A prepared Jenkins server is available from the file "latestJenkins_2_121_1.ova". This is a virtual machine which runs on VirtualBox. This Jenkins server virtual machine should be downloaded, then be imported to VirtualBox (e.g., as “UbuntuJenkins 2.121.1” virtual machine).

To ensure the consistency between the test input data and Jenkins server, this virtual machine should be configured as follows:

- Add a new adapter “vboxnet0” to VirtualBox (VirtualBox Manager/ Tools/Network/Create), then configure the new adapter at Tab Properties/vboxnet0:
    - Adapter: Enable Server with the address 192.168.56.1/24
    - Enable DHCP server with address 192.168.56.100/24 and the lower address bound 192.168.56.102.
- Enable Network Adapter 2 of the virtual machine as a Host-only Adapter (Select the “UbuntuJenkins 2.121.1” virtual machine in the list of virtual machines in VirtualBox, then choose Setting/Network/Adapter 2/Enable Network Adapter/Attached to “Host-only adapter” with Name “vboxnet0”).
- Start the Jenkins server virtual machine, then check its IP address. If it does not have the IP address 192.168.56.102/24, manually change the IP address to 192.168.56.102/24.

An additional VM, to be used to configure a slave for Jenkins is provided. Its name is slave.ova.

* Jenkins 2.1.2.1

  - Ubuntu account: jenkinsuser/123

  - Jenkins accounts:

    + admin/adminPass

    + user1/user1Pass

    + user2/user2Pass

* Slave machine

  - Ubuntu account: slave/slaveubuntu

- User for jenkins (/home/slaveJenkins)

    + Username for jenkins: slaveJenkins

    + Pass: slavePass

- Keypass for SSH: slavePass

- Username for jenkins: mai

- Pass: maiMaster

2. Prepare the Test Environment

We provide an Eclipse project including a catalog of 22 MRs and the Java code generated from these MRs. The project is located at “EclipseProjectSMRL.zip".

Users can edit the provided MRs using the SMRL editor. To ensure anonymity we provide an Eclipse distribution including our plugin at " eclipseSMRL_Linux_x64.tar.gz".

Our framework relies on Selenium HQ to automate interaction with Web systems. To use Selenium for this purpose, you need to download and install "chromedriver", a tool that enables the use of the Chrome browser through Selenium HQ. Please download the chromedriver from the following URL https://chromedriver.storage.googleapis.com/index.html?path=2.44/

The chromedriver should be copied in "/usr/local/bin/". We tested it using MacOS.

2.1. Editing the provided MRs.

Please download, decompress, and run the provided Eclipse distribution in an empty eclipse workspace.

To edit the provided MRs please decompress the file "EclipseProjectSMRL.zip". This will create a folder named "ICST2020".

After decompressing the file, please import the project ICST2020 in the Eclipse workspace. At this point you can open an available MR (e.g., OTG_AUTHZ_002.smrl) by double clicking on it. The relation should be automatically opened in the SMRL editor. The syntax of the MR should be properly highlighted, if this does not happen open the MR after selecting the editor manually. If it does not work, you can still replicate the paper results because we provide the generated Java classes (see Section 3 below).

After editing and saving each .smrl file, the SMRL editor plugin automatically generates a corresponding java class in the folder "src-gen/srml/mr/owasp" (e.g., "src-gen/srml/mr/owasp /OTG_AUTHZ_002.java").

3. Run the Test Cases

We provide a JUnit test class including twelve test cases, each verifying Jenkins against one MR targeting the Jenkins vulnerabilities. The junit class is named "TestJenkins" and is contained in the package "icst2020.tests".

The source inputs and the configuration file for our MR framework are available in the folder "testData/Jenkins". This folder contains the following files:

- jenkinsSysConfig.json it contains the system under test’s configuration information, for example:
    - inputFile, which identifies the input file containing the source inputs.

    - userParameter and passwordParameter, which indicate the name of the HTML form parameters used to provide username and password. E.g., “j_username” and “j_password” in case of Jenkins.
    - loginURL and logoutURL, which identify the URLs to log into or log out from the system.
- Input_FULL.json, which contains the source inputs derived from crawled data and manual test cases.
- Input_crawljax.json, which contains the source inputs automatically crawled by Crawljax.
- filePaths.txt, which contains the file paths returned by the SMRL function "RandomFilePaths". We populated it with all the files in the Jenkins folder in the virtual machine (i.e., /var/lib/jenkins).
- adminFilePaths.txt, which contains the file paths returned by the SMRL function "AdminFilePaths". It contains paths belonging to folders where files should not be written/created by normal users (i.e., not admin) during execution. For Jenkins, these are the folders belonging to the config tree.

To replicate our results:

- Start the Jenkins system (start the Jenkins virtual machine).
- Execute a test case belonging the TestJenkins class.
- Test execution is logged in the Eclipse console. When a test case discovers a vulnerability, failure information is printed (see "FAILURE" keyword) and the test case is indicated as failing by JUnit. We suggest to redirect console output to file.

We provide the log files generated during our empirical evaluation in the zip file “RQ2-executionLogs.zip" a description of the content of the logs is provided in the document named "RQ2-ExecutionLogs.pdf".