Scenario tests for user story 2

Group 17

Scenario 1 : Installing the ZAP addon

1- Go to the project root '/Group17'

2- Build the project using: *./gradlew clean build* - A ZAP addon (.zap) is created in '/Group17/zap-addons/addOns/dslpolicyloader/build/zapAddOn/bin/'

3- Start the ZAP proxy program

4- Go to file, load addon and select the .zap addon

5- Verify in the output tab that the addon has been successfully installed.

[6-] The addon can be uninstalled via the addon manager.

Scenario 2: Loading a policy

1 - Consult the existing policy test files under 'other/rule_policy_dsl'. For constructing new policies, please consult the 'Guide for writing rule DSL.pdf' in 'solutions/userstory2'.

2 – In the Zap program, menu bar, go to 'Tools-DSL Policy Loader' and select one or multiple test policies .

3 - A n alert showing the status of the operation will be displayed (should be either a success message or a duplication confirmation if the policy has been loaded.)

Scenario 3: Consulting the currently loaded policies

1- After loading the DSL policies, consulting the currently loaded policies can be done by going to *Tools/ View loaded DSL Policy* - A Java Swing window with currently loaded DSL policies is shown.

2- Select a DSL policy in the list.

3- Click the "Show" button, the selected policy and its rules will be displayed.

Scenario 4: Scanning HTTP traffic scanning and live alerts

After loading the ZAP addon and DSL policies, all visited websites will be checked via imported DSL policies and alerts will be risen whenever rules are violated.

1- Configure your browser to use the ZAP proxy (127.0.0.1:8080) .

2- Visit some websites that violated the laoded rules. In our case, these should raise some :

2.1- Searching “hacker” in www.google.com

2.2- Searching “abc” in www.google.com

2.3- Searching “zerohedge” in www.google.com

2.4- Searching “cern” in www.google.com

2.5- Searching “shouldnotalert” in www.google.com

3 - Check the alert messages in the “Alerts” section..

Expected from 2.1

Alerts: Policy_dsl_example2.Rule_hacker_rule violated

Policy_dsl_example2.Rule_keyword_NOT_rule violated

Policy_dsl_example2.Rule_keyword_OR_rule violated

Policy_dsl_example2.Rule_keyword_list_rule violated

Expected from 2.2

Alerts: Policy_dsl_example2.Rule_keyword_NOT_rule violated

Policy_dsl_example2.Rule_regex_rule violated

Expected from 2.3

Alerts: Policy_dsl_example2.Rule_keyword_NOT_rule violated

Policy_dsl_example2.Rule_keyword_OR_rule violated

Policy_dsl_example2.Rule_keyword_list_rule violated

Policy_dsl_example2.Rule_paranthesis_frenzy_rule violated

Expected from 2.4

Alerts: Policy_dsl_example2.Rule_keyword_AND_frenzy_rule violated

Policy_dsl_example2.Rule_keyword_NOT_rule violated

Expected from 2.5

Alerts: Policy_dsl_example2.Rule_keyword_NOT_rule violated

Scenario 5: Building a report

1- A report with alert's details can be created by going to *Report/ DSL Policy Violations Report .*

2- Choose a destination directory to save the report.

3- Input a name for the report with the extension name of HTML ".html", e.g "report.html"

4- Click save, an alert showing the success of the operation should be displayed.

5- Browse to the saved html report file and open it with a browser.

Example Test Policies

Example Policy test 1

Rule "hacker_rule" "hacker exists in the response body":
response.body.value="hacker";

Rule "zerohedge_rule" "zerohedge exists in the request body":
request.body.value="zerohedge";

Rule "keyword_list_rule" "response body contains at least one of the keywords in the list":
response.body.values=["hacker","zerohedge"];

Rule "keyword_AND_rule" "response body contains both of the keywords":
response.body.value="hacker" and response.body.value="zerohedge";

Rule "keyword_OR_rule" "response body contains at least one of the keywords":
response.body.value="hacker" or response.body.value="zerohedge";

Rule "keyword_NOT_rule" "response body does not contain the keyword":
not (request.body.value="mango");

Rule "regex_rule" "response body matches regex":
response.header.re="abc";

Example Policy test 2

Rule "keyword_AND_paranthesis_rule" "response body contains both of the keywords":
(response.body.value="hacker" and response.body.value="zerohedge") or
(response.body.value="cern");

Rule "paranthesis_frenzy_rule" "checking for numerous parenthesis":
((((((((((response.body.value="zerohedge"))))))))));

Rule "paranthesis_and_frenzy_rule" "checking for numerous parenthesis with and":
response.body.value="zerohedge" and ((((((((((response.body.value="hacker"))))))))));

Rule "not_not_rule" "checking for nested nots":
not (not (response.body.value="cern"));

Example Policy test 3

Rule "hacker_req_header_rule" "hacker exists in the request header":
request.header.re="hacker";

Rule "hacker_resp_header_rule" "hacker exists in the response header":
response.header.value="hacker";

Rule "keyword_NOT_req_header_rule" "request header does not contain the keyword":
not (request.header.re="mango");

Rule "keyword_NOT_resp_header_rule" "response header does not contain the keyword":
not (response.header.value="mango");