Testing the OWASP Java HTML Sanitizer

Steven Aldinger, Michael Stenhouse, Marta Pancaldi, and Seth Stoudenmier

Computer Science Dept., College of Charleston

Introduction

OWASP Java HTML Sanitizer was originally chosen for its multitude of documentation and our group's understanding of Java. This particular open source project aims to prevent any malicious HTML code from being injected into a web application, as well as organize poorly written code. The source repository came with an extensive test suite that we used to help guide our test plan. To start we took tests from the OWASP Java HTML Sanitizer to make sure that the source code would compile. Look down to Figure 1 and 2 to see an example of the tests run from the OWASP HTML Sanitizer.

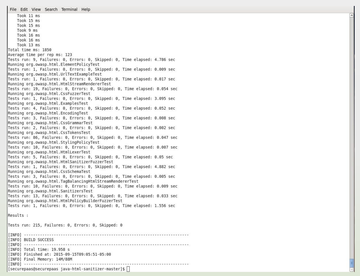

Figure 1 (Shows output of a successful "mvn clean install" to test and build the HTML Sanitizer.)

Figure 2 (Shows some example code from one of the tests. It is verifying that the htmlDecoder is working properly.)

Test Framework

The three methods tested within OWASP Java HTML Sanitizer were decodeHtml, cssContent, and sanitize. The organization for our test framework is demonstrated in Figure 3.

```
/TestAutomation
    /project
        /src
        /bin
        / ...
    /scripts
        runAllTests (some scripting extension)
        ... other helper scripts
    /testCases
        testCase1.txt
        testCase2.txt
        ...
    /testCasesExecutables
        testCase1(may be folder or file)
        ...
    /temp (for output from running tests ... be sure to clean at start of runAllTests)
        testCase1results (might be folder or file)
        ...
    /oracles
        testCase1Oracle (might be folder or file)
        ...
    /docs
        README.txt
    /reports
        testReport.(txt or html)
        ...
```

Figure 3 (Structure for test framework.)

The requirements for each method tested are as follows:

- decodeHtml handles HTML entities to produce a string containing only valid unicode scalar values
- cssContent handles escape sequences and strips any quotes from the input
- sanitize handles removal of elements from HTML strings

Each test case is contained in a separate .txt file. Each file follows a simple template:

1. ## test number or ID
2. ## requirement being tested
3. ## component being tested
4. ## method being tested
5. ## command-line arguments
6. ## expected outcomes

To run these test cases a driver is used for each method. A single script is invoked to access the folder of drivers and run all of our test cases. An example driver for the decodeHtml method is seen in Figure 4.

```
public static void main(String[] args){
        try{
                String theOracle = args[1];
                String theTest = args[0];

                String result = org.owasp.html.Encoding.decodeHtml(theTest);
                if(theOracle.contains("\\n")){
                        //System.out.println("Oracle contains: \\n");
                        theOracle = theOracle.replace("\\n","\n");
                }


                if((result).equals(theOracle)){
                        try{
                                System.out.println("org.owasp.html.Encoding");
                                System.out.println("decodeHtml(String)");
                                System.out.println(args[0]);
                                System.out.println(args[1]);
                                System.out.println("passed\n");

                        }catch(Exception e){
                                e.printStackTrace();
                        }

                }
                else{
                        try{
                                System.out.println("org.owasp.html.Encoding");
                                System.out.println("decodeHtml(String)");
                                System.out.println(args[0]);
                                System.out.println(args[1]);
                                System.out.println("failed\n");

                        }catch(Exception e){
                                e.printStackTrace();
                        }
                }
        }catch(Exception e){
                System.out.println("Exception: Possibly incorrect number of arguments.");
                e.printStackTrace();
        }
}
```

Figure 4 (Driver for the tests cases for the decodeHtml method.)

Results

After running all tests a HTML file is outputted containing a chart for each test case and other information regarding each. This HTML table can be seen in Figure 5.

Test Number	Class	Method	Requirement	Input	Output	Oracle	Result
1	org.owasp.html.Encoding	decodeHtml(String)	decodeHtml handles HTML entities to produce a string containing only valid unicode scalar values
\n	\n	passed
2	org.owasp.html.Encoding	decodeHtml(String)	decodeHtml handles HTML entities to produce a string containing only valid unicode scalar values
\n	\n	passed
3	org.owasp.html.Encoding	decodeHtml(String)	decodeHtml handles HTML entities to produce a string containing only valid unicode scalar values	&#x02ziggy	&#x02ziggy	&#x02ziggy	passed
4	org.owasp.html.Encoding	decodeHtml(String)	decodeHtml handles HTML entities to produce a string containing only valid unicode scalar values	&#0a;	\n	\n	passed
5	org.owasp.html.Encoding	decodeHtml(String)	decodeHtml handles HTML entities to produce a string containing only valid unicode scalar values
\n	\n	passed
6	org.owasp.html.CssGrammar	cssContent(String)	cssContent handles escape sequences and strips any quotes from the input	'foo'	foo	foo	failed
7	org.owasp.html.CssGrammar	cssContent(String)	cssContent handles escape sequences and strips any quotes from the input	\61zimuth	azimuth	azimuth	passed
8	org.owasp.html.CssGrammar	cssContent(String)	cssContent handles escape sequences and strips any quotes from the input	\61ser	taser	taser	passed
9	org.owasp.html.CssGrammar	cssContent(String)	cssContent handles escape sequences and strips any quotes from the input	foo	foo	foo	passed
10	org.owasp.html.CssGrammar	cssContent(String)	cssContent handles escape sequences and strips any quotes from the input	\'foo\'	\'foo\'	foo	failed
11	org.owasp.html.CssGrammar	cssContent(String)	cssContent handles escape sequences and strips any quotes from the input	'	'	'	passed
12	org.owasp.html.CssGrammar	cssContent(String)	cssContent handles escape sequences and strips any quotes from the input	\'	\'	\'	passed
13	org.owasp.html.CssGrammar	cssContent(String)	cssContent handles escape sequences and strips any quotes from the input	\'\2\2\2\2\'	\'\'\'\'	\'\'	failed
14	org.owasp.html.CssGrammar	cssContent(String)	cssContent handles escape sequences and strips any quotes from the input	\2\2\2	\'\'	\'\'	passed
15	org.owasp.html.CssGrammar	cssContent(String)	cssContent handles escape sequences and strips any quotes from the input	\'	\'	\	failed
16	org.owasp.html.CssGrammar	cssContent(String)	cssContent handles escape sequences and strips any quotes from the input	\x'	\x'	\n	failed
17	org.owasp.html.Sanitizers	FORMATTING.sanitize(String)	sanitize handles removal of elements from HTML strings	Hello,World!\n	Hello,World!\n	Hello,World!\n	passed
18	org.owasp.html.Sanitizers	FORMATTING.sanitize(String)	sanitize handles removal of elements from HTML strings	>\n	>\n	>\n	passed
19	org.owasp.html.Sanitizers	FORMATTING.sanitize(String)	sanitize handles removal of elements from HTML strings	Hello,<boniclick=alert(1337)>World!	Hello,World!	Hello,World!	passed
20	org.owasp.html.Sanitizers	BLOCKS.sanitize(String)	sanitize handles removal of elements from HTML strings	Hello,World!	Hello,World!	Hello,World!	passed
21	org.owasp.html.Sanitizers	BLOCKS.sanitize(String)	sanitize handles removal of elements from HTML strings	Hello,World!\n	Hello,World!	Hello,World!	passed
22	org.owasp.html.Sanitizers	BLOCKS.sanitize(String)	sanitize handles removal of elements from HTML strings	Hello,World!\n	Hello,World!	Hello,World!	passed
23	org.owasp.html.Sanitizers	BLOCKS.sanitize(String)	sanitize handles removal of elements from HTML strings	Hello,World!	Hello,World!	Hello,World!	passed
24	org.owasp.html.Sanitizers	BLOCKS.sanitize(String)	sanitize handles removal of elements from HTML strings				passed
25	org.owasp.html.Sanitizers	BLOCKS.sanitize(String)	sanitize handles removal of elements from HTML strings	Hello	Hello	Hello	passed

Figure 5 (HTML table to display our 25 test cases and all of the information regarding them.)

Figure 6 contain the outputted results of each test case prior to the completion of the HTML table. They follow the template mentioned early about test cases.

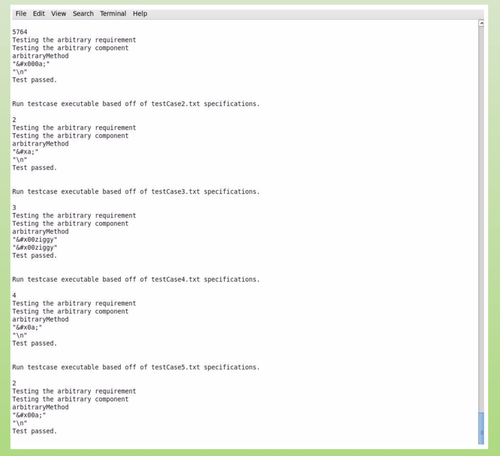

Figure 6 (Series of test cases shown as the tests are performed. After they are all performed the HTML table mentioned in Figure 5 is created.)

Faults

After performing our tests we injected some faults into the original code. One example of a fault can be seen in Figure 7. Notice the comment next to the second conditional which details what was changed.

```
public static String cssContent(String token) {
  int n = token.length();
  int pos = 0;
  StringBuilder sb = null;
  if (n >= 2) {
    char ch0 = token.charAt(0);
    if (ch0 == '"' && ch0 == '\'') { // FAULT: Changed an || to an &&
    //if (ch0 == '"' || ch0 == '\'') {
      if (ch0 == token.charAt(n - 1)) {
        pos = 1;
        --n;
        sb = new StringBuilder(n);
      }
    }
  }
```

Figure 7 (Shows an example of one of the faults that were injected.)

Our fault broke a few of our test cases, but did not break every single one of them. This adds another level of tests to our particular methods.

Bibliography

1. **"OWASP Java HTML Sanitizer Project." OWASP. Web. 29 Nov. 2015. <https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project>.**

CONTACT

Steven Aldinger:
aldingerst@g.cofc.edu

Michael Stenhouse:
stenhousems@g.cofc.edu

Marta Pancaldi:
pancaldim@g.cofc.edu

Seth Stoudenmier:
stoudenmiersh@g.cofc.edu

QR Code