CS 542 – Introduction to Software Security
Exercise on Automated Assessment Tools

Binhao Chen (bchen276@wisc.edu), Steven Yang (yang558@wisc.edu)

Due: December 8 at 2:30pm.

1 SpotBugs

1.1 Description of all the security weaknesses identified by the tool that need to be fixed

1.1.1 Security weakness 1: Information Exposure Through An Error Message

The number of security weakness 1: 4

1. Database.java[line 105], 2. TargetServer.java[line 102, 118, 160]

Seriousness and Explanation: High, by printing the stack trace, the sensitive and valuable information such as passwords, internal of the database and the program are exposed to the attacker. Although a SQL injection attack might not initially be successful, an error message could indicate the query's flawed logic, which could potentially include passwords or other sensitive data.

How to fix: Avoid using printStackTrace() method, try to instead print out a more generalized error message to the user and log the sensitive error information into a more secure log server that cannot be reached by unauthorized users.

1.1.2 Security weakness 2: Potential XSS in Servlet

The number of security weakness 2: 18

1. AcountPageServlet.java[line 78, 92, 98, 101, 106, 111, 113], 2. EarnPointsPageServlet.java[line 62, 75, 82], 3. RankListPageServlet.java[line 63, 80, 81, 82], 4. TransferPageServlet.java[line 66, 84, 88, 95]

Seriousness and Explanation: High, an attacker can insert a malicious JavaScript and the user's browser will execute this script as long as this page is loaded. This script could access any cookies, session ids, and sensitive information. This web application is involved with transferring credits, thus it is dangerous to let the attacker access cookies.
How to fix: When call getPatameter, make sure first check if the input parameter has illegal characters that may involve unwanted JavaScript

1.1.3 Security weakness 3: Untrusted servlet parameter

The number of security weakness 3: 6

1. AccountPageServelt.java[line 52], 2. TransferPageServlet.java[line 137, 138],
3.ViewPageServlet.java[line 39, 99 , 100]

Seriousness and Explanation: High, if the return value of getParameter() is not checked, an XSS attack will happen. Thus, the attacker will have access to any cookies and session ids.
How to fix: It is necessary to validate or sanitize those parameters from req.getParameter() before passing them to other servlets (APIs).

1.1.4 Security weakness 4: Trust Boundary Violation

The number of security weakness 4: 1

ViewPageServlet.java[line 99 and line 105]

Seriousness and Explanation: Medium, the data coming from the getParameter() is not checked before it is used to setAttribute(). It becomes easier to mistakenly trust unvalidated data and cause more severe consequences to the system.
How to fix: The untrusted input validation prior to setting a new session attribute is necessary. It is preferable that the data used for a session attribute setting is from a safe/trusted location rather than from user input, which is untrusted.

1.2 Explanation of 7 other weaknesses identified by the tools that are not false positives

(1):Comparison of String objects using == or !=
String objects can be checked for reference equality using the == and!= operators. The same string value may be represented by two different String objects. Instead, we should consider using the equals(Object) method.

(2): Method invokes inefficient new String(String) constructor
The object created by calling the java.lang.String(String) constructor will be functionally identical to the String supplied as an argument, the String(String) constructor wastes memory.

(3): Boxing/unboxing to parse a primitive
To obtain the unboxed primitive value from a String, it is more efficient and safe to simply call the static Integer.parseInt(String) method.

(4): Method invokes inefficient Number constructor;
Integer.valueOf(int) allows caching of values to be done by the compiler, class library, or JVM, which allows the code to run faster than using new Integer(). We should consider using either autoboxing or the valueOf() method when creating instances of Long, Integer, Short, Character, and Byte.

(5): Dead store to local variable
This instruction assigns a value to a local variable "deductPoints", but the variable is not read or used in any subsequent instruction.

(6): method converts an exception into a boolean 'error code' value
This method handles exceptions and returns a boolean indicating whether or not an exception occurred. This nullifies the value of exception handling and allows code to disregard the resulting 'error code' return value. The caller will never know the value of the exception. Therefore, we should simply throw the exception to the caller.

(7): Method makes literal string comparisons passing the literal as an argument
The code is written in the form of str.equals("quit"), which will trigger a null pointer exception if the str is null. Calling equals() or compareTo() on the string literal and passing the variable as an argument, then the null exception could never happen. We should always write String str = ... "someOtherString".equals(str); or "someOtherString".compareTo(str);

2 OWASP Dependency Check

2.1 Description of 7 of the problems associated with the vulnerable dependencies found by the tool.

(1): CVE-2017-7656

In Eclipse Jetty, versions 9.2.x and older, 9.3.x (all configurations), and 9.4.x (non-default configuration with RFC2616 compliance enabled), HTTP/0.9 is handled poorly. An HTTP/1 style request line (i.e. method space URI space version) that declares a version of HTTP/0.9 was accepted and treated as a 0.9 request. If deployed behind an intermediary that also accepted and passed through the 0.9 version (but did not act on it), then the response sent could be interpreted by the intermediary as HTTP/1 headers. This could be used to poison the cache if the server allowed the origin client to generate arbitrary content in the response.

(2): CVE-2022-31160
jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of jQuery. Versions prior to 1.13.2 are potentially vulnerable to cross-site scripting. Initializing a checkboxradio widget on an input enclosed within a label makes that parent label contents considered as the input label. Calling '.checkboxradio("refresh")' on such a widget and the initial HTML contained encoded HTML entities will make them erroneously get decoded. This can lead to potentially executing JavaScript code. The bug has been patched in jQuery UI 1.13.2. To remediate the issue, someone who can change the initial HTML can wrap all the non-input contents of the 'label' in a 'span'. CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

(3): CVE-2019-10247
In Eclipse Jetty version 7.x, 8.x, 9.2.27 and older, 9.3.26 and older, and 9.4.16 and older, the server running on any OS and Jetty version combination will reveal the configured fully qualified directory base resource location on the output of the 404 error for not finding a Context that matches the requested path. The default server behavior on jetty-distribution and jetty-home will include at the end of the Handler tree a DefaultHandler, which is responsible for reporting this 404 error, it presents the various configured contexts as HTML for users to click through to. This produced HTML includes output that contains the configured fully qualified directory base resource location for each context. CWE-200 Information Exposure

(4): CVE-2020-11023
In jQuery versions greater than or equal to 1.0.3 and before 3.5.0, passing HTML containing <option> elements from untrusted sources - even after sanitizing it - to one of jQuery's DOM manipulation methods (i.e. .html(), .append(), and others) may execute untrusted code. This problem is patched in jQuery 3.5.0.

(5): CVE-2021-34428
For Eclipse Jetty versions $\leq$ 9.4.40, $\leq$ 10.0.2, $\leq$ 11.0.2, if an exception is thrown from the SessionListenersessionDestroyed() method, then the session ID is not invalidated in the session ID manager. On deployments with clustered sessions and multiple contexts this can result in a session not being invalidated. This can result in an application used on a shared computer being left logged in.

(6): CVE-2020-27216
In Eclipse Jetty versions 1.0 thru 9.4.32.v20200930, 10.0.0.alpha1 thru 10.0.0.beta2, and 11.0.0.alpha1 thru 11.0.0.beta2O, on Unix like systems, the system's temporary directory is shared between all users on that system. A collocated user can observe the process of creating a temporary sub directory in the shared temporary directory and race to complete the creation of the temporary subdirectory. If the attacker wins the race then they will have read and write permission to the subdirectory used to unpack web applications, including their WEB-INF/lib jar files and JSP files. If any code is ever executed out of this temporary directory, this can lead to a local privilege escalation vulnerability.

(7): CVE-2020-27223
In Eclipse Jetty 9.4.6.v20170531 to 9.4.36.v20210114 (inclusive), 10.0.0, and 11.0.0 when Jetty handles a request containing multiple Accept headers with a large number of "quality" (i.e. q) parameters, the server may enter a denial of service (DoS) state due to high CPU usage processing those quality values, resulting in minutes of CPU time exhausted processing those quality values. CWE-400 Uncontrolled Resource Consumption ('Resource Exhaustion')