OWASP ASIDE Project Roadmap

Development Team:

Jing Xie (jxie2@uncc.edu), Bill Chu (billchu@uncc.edu), John Melton (jtmelton@gmail.com)

Goal:

ASIDE is an abbreviation for Assured Software Integrated Development Environment. It is an Eclipse^{TM} Plugin which is a software tool primarily designed to help students write more secure code by detecting and identifying potentially vulnerable code and providing informative fixes during the construction of programs in IDEs. ASIDE may be useful by professional developers as well.

Design Rationales:

1. From Human Computer Interaction standpoint of view, one of the best practices for designing usable tools is to ease users' cognitive burden by recognition instead of recalling. Visual cueing is one such instance. It facilitates subjects' perception of information by providing visualized symbols such as icons, dialogs, markers and etc in the computer world.

2. Some of the vulnerabilities reside in code can be fixed by adding/modifying a few lines of code locally.

3. Program authors have the best understanding of the program. It is easiest to write secure code while their minds are actively focused on program constructing than patching it up later in the development cycle.

Envisioned Benefits:

1. Reinforce secure programming concepts, techniques such as white-list based input validation, output encoding and use of proven library such as OWASP ESAPI. Programmers will be constantly reminded whenever a potential vulnerability is detected, which shapes a mindset that makes security a first class concern throughout the development process.

2. Secure programming knowledge and standards can be shared and enforced to ensure programmers are following best practices.

3. Rules associated with validation code generated by such tools can be written and baked into static analysis tools to help with accuracy of static analysis by reducing false positives.

4. Through facilities such as logs and annotations which can record how security considerations have been addressed, programmers can provide the necessary information needed for security auditing/ assurance/course grading.

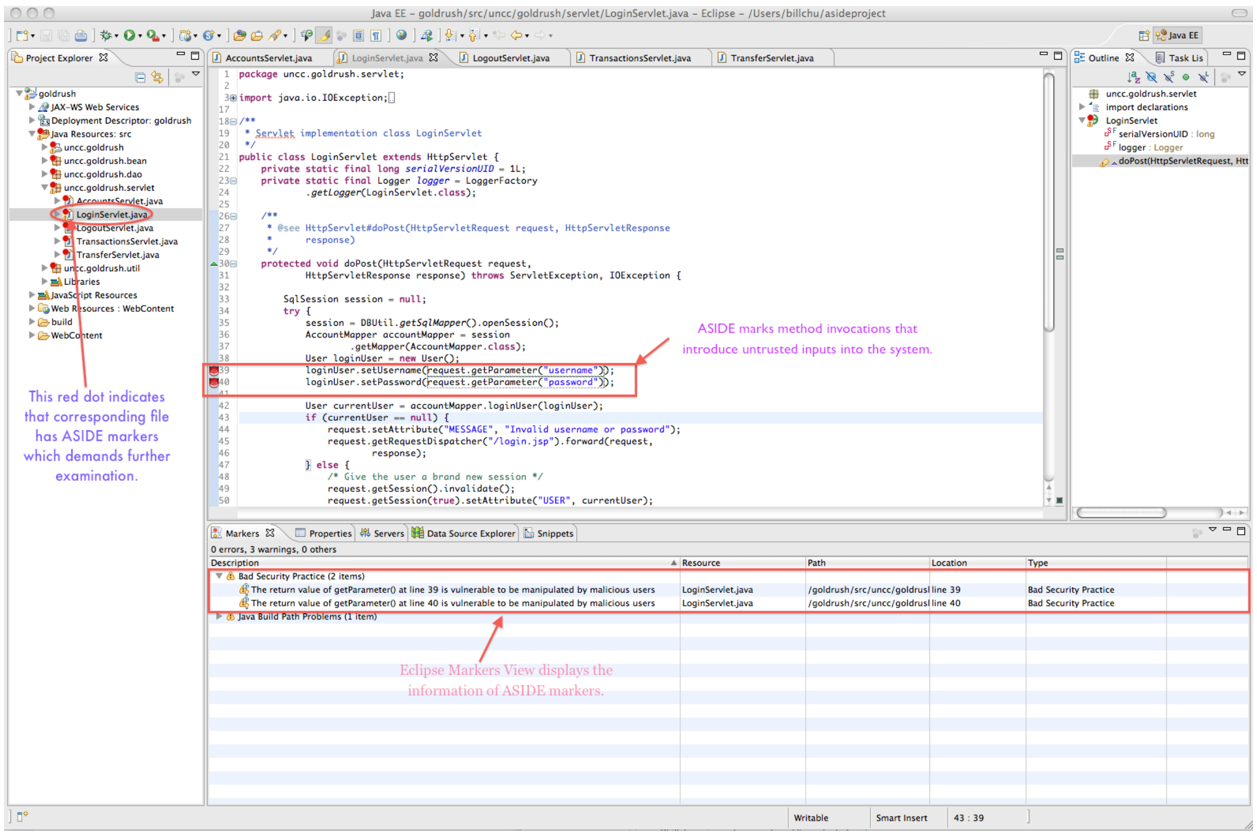
Design Descriptions:

1. Rule based specification of sources of untrusted inputs which we refer as trust boundaries. Rules which we refer to as heuristic rules are defined in XML format currently. The two major types of trust boundaries are: Method invocations (e.g. HttpServletRequest.getParameter(String string)) and Parameter inputs (e.g. the arguments of Java program entrance method main(String[] args)).

2. ASIDE recognizes untrusted inputs, based on specified rules, and prompts developers to select from a library of options for either input validation or output encoding. Such library contains validation rules and encoding mechanisms contributed to and shared by many developers in the same way as an open source project.

3. The validation specifications can include both syntactic as well as semantic validations.

<u>Prototype Demostration:</u>

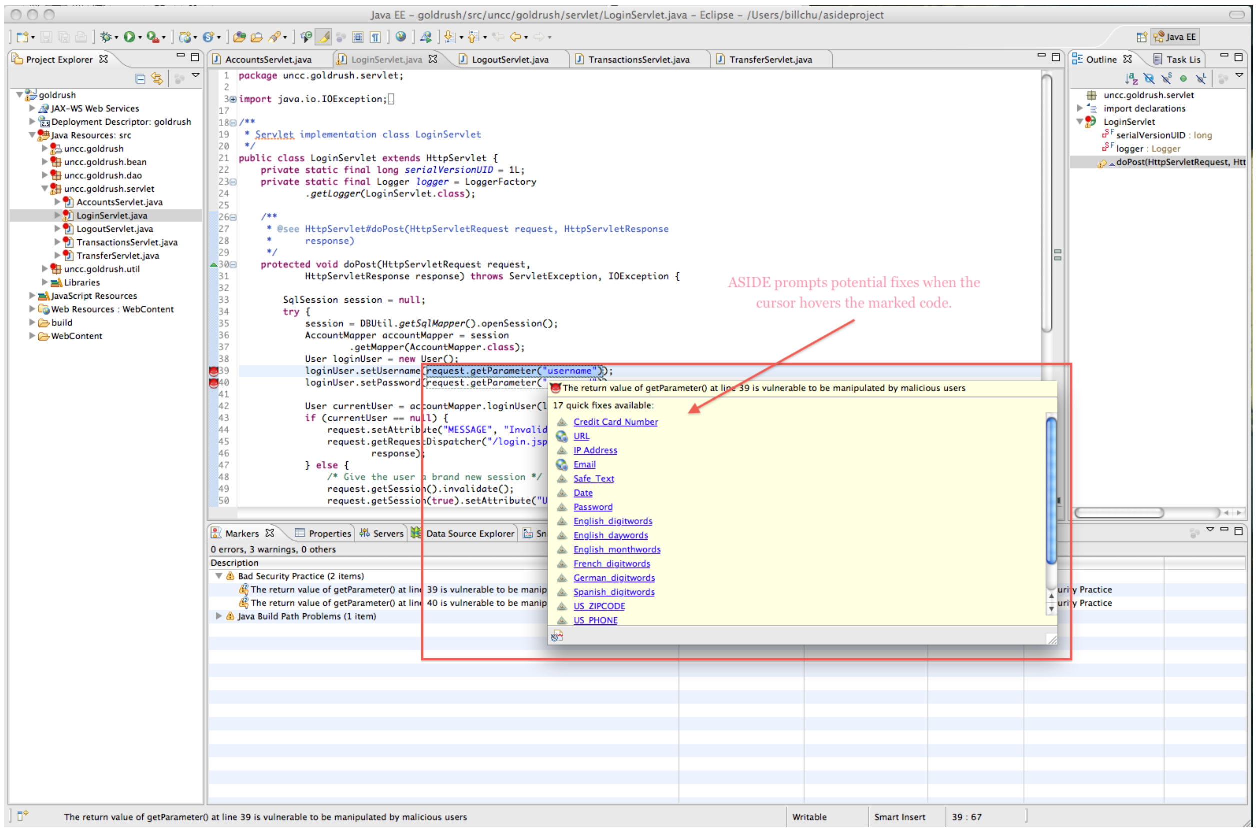


```
 1 package uncc.goldrush.servlet;
 2 
 3 import java.io.IOException;
17 
18 /**
19  * Servlet implementation class LoginServlet
20  */
21 public class LoginServlet extends HttpServlet {
22     private static final long serialVersionUID = 1L;
23     private static final Logger logger = LoggerFactory
24             .getLogger(LoginServlet.class);
25 
26     /**
27      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
28      *      response)
29      */
30     protected void doPost(HttpServletRequest request,
31             HttpServletResponse response) throws ServletException, IOException {
32 
33         SqlSession session = null;
34         try {
35             session = DBUtil.getSqlMapper().openSession();
36             AccountMapper accountMapper = session
37                     .getMapper(AccountMapper.class);
38             User loginUser = new User();
39             loginUser.setUsername(request.getParameter("username"));
40             loginUser.setPassword(request.getParameter("password"));
41 
42             User currentUser = accountMapper.loginUser(loginUser);
43             if (currentUser == null) {
44                 request.setAttribute("MESSAGE", "Invalid username or password");
45                 request.getRequestDispatcher("/login.jsp").forward(request,
46                         response);
47             } else {
48                 /* Give the user a brand new session */
49                 request.getSession().invalidate();
50                 request.getSession(true).setAttribute("USER", currentUser);
```

0 errors, 3 warnings, 0 others

Description	Resource	Path	Location	Type
Bad Security Practice (2 items)				
The return value of getParameter() at line 39 is vulnerable to be manipulated by malicious users	LoginServlet.java	/goldrush/src/uncc/goldrusl	line 39	Bad Security Practice
The return value of getParameter() at line 40 is vulnerable to be manipulated by malicious users	LoginServlet.java	/goldrush/src/uncc/goldrusl	line 40	Bad Security Practice
Java Build Path Problems (1 item)				

Figure 1. ASIDE detects and identifies code that imports untrusted inputs through HttpServletRequest.getParameter(String param) method.



Java EE – goldrush/src/uncc/goldrush/servlet/LoginServlet.java – Eclipse – /Users/billchu/asideproject

```
1  package uncc.goldrush.servlet;
2
3  import java.io.IOException;
17
18 /**
19  * Servlet implementation class LoginServlet
20  */
21 public class LoginServlet extends HttpServlet {
22     private static final long serialVersionUID = 1L;
23     private static final Logger logger = LoggerFactory
24             .getLogger(LoginServlet.class);
25
26     /**
27      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
28      *      response)
29      */
30     protected void doPost(HttpServletRequest request,
31             HttpServletResponse response) throws ServletException, IOException {
32
33         SqlSession session = null;
34         try {
35             session = DBUtil.getSqlMapper().openSession();
36             AccountMapper accountMapper = session
37                     .getMapper(AccountMapper.class);
38             User loginUser = new User();
39             loginUser.setUsername(request.getParameter("username"));
40             loginUser.setPassword(request.getParameter("
41
42             User currentUser = accountMapper.loginUser(l
43             if (currentUser == null) {
44                 request.setAttribute("MESSAGE", "Invalid
45                 request.getRequestDispatcher("/login.jsp
46                         response);
47             } else {
48                 /* Give the user a brand new session */
49                 request.getSession().invalidate();
50                 request.getSession(true).setAttribute("U
```

The return value of getParameter() at line 39 is vulnerable to be manipulated by malicious users

17 quick fixes available:

- Credit Card Number
- URL
- IP Address
- Email
- Safe_Text
- Date
- Password
- English_digitwords
- English_daywords
- English_monthwords
- French_digitwords
- German_digitwords
- Spanish_digitwords
- US_ZIPCODE
- US_PHONE

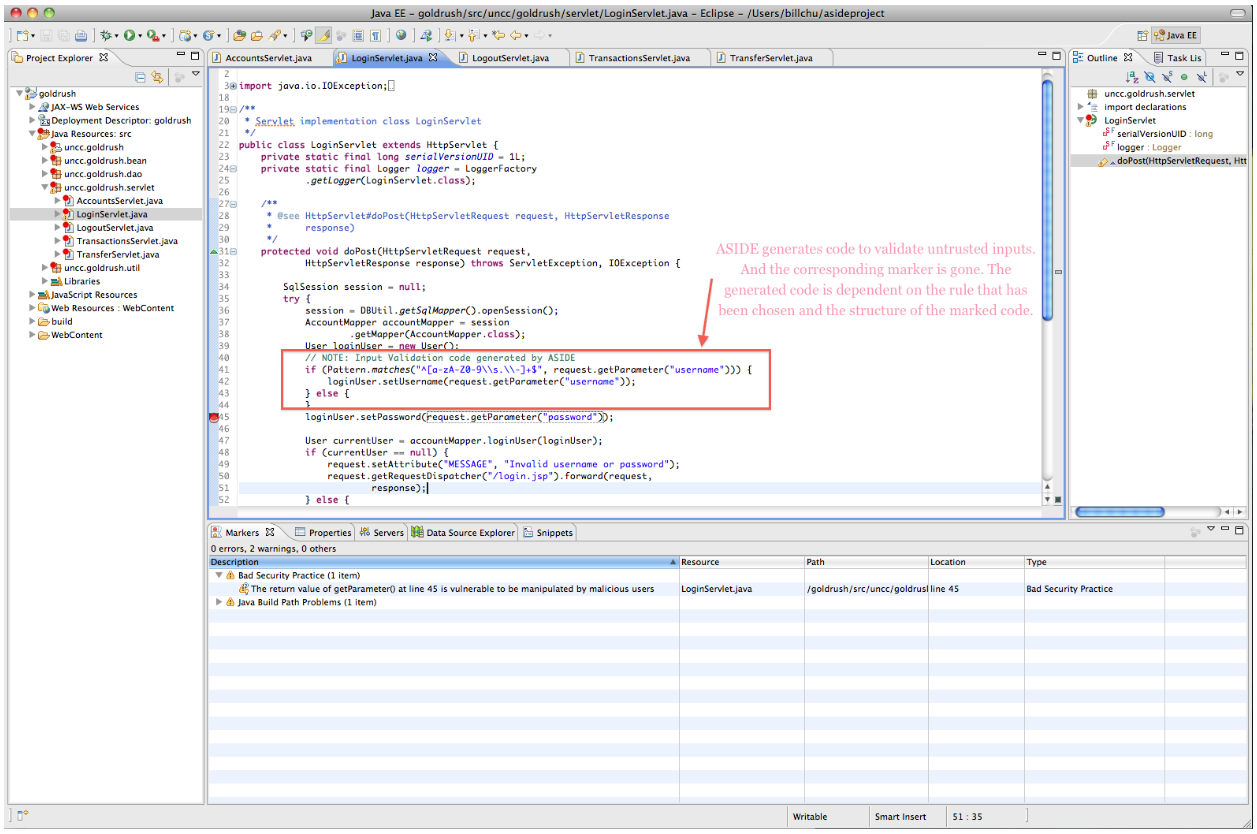
0 errors, 3 warnings, 0 others

Description
- Bad Security Practice (2 items)
  - The return value of getParameter() at line 39 is vulnerable to be manip
  - The return value of getParameter() at line 40 is vulnerable to be manip
- Java Build Path Problems (1 item)

urity Practice
urity Practice

The return value of getParameter() at line 39 is vulnerable to be manipulated by malicious users

Figure 2. ASIDE prompts potential fixes when the cursor hovers on the marked code.



```
 2
 3 import java.io.IOException;
18
19 /**
20  * Servlet implementation class LoginServlet
21  */
22 public class LoginServlet extends HttpServlet {
23     private static final long serialVersionUID = 1L;
24     private static final Logger logger = LoggerFactory
25             .getLogger(LoginServlet.class);
26
27     /**
28      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
29      *      response)
30      */
31     protected void doPost(HttpServletRequest request,
32             HttpServletResponse response) throws ServletException, IOException {
33
34         SqlSession session = null;
35         try {
36             session = DBUtil.getSqlMapper().openSession();
37             AccountMapper accountMapper = session
38                     .getMapper(AccountMapper.class);
39             User loginUser = new User();
40             // NOTE: Input Validation code generated by ASIDE
41             if (Pattern.matches("^[a-zA-Z0-9\\s.\\-]+$", request.getParameter("username"))) {
42                 loginUser.setUsername(request.getParameter("username"));
43             } else {
44             }
45             loginUser.setPassword(request.getParameter("password"));
46
47             User currentUser = accountMapper.loginUser(loginUser);
48             if (currentUser == null) {
49                 request.setAttribute("MESSAGE", "Invalid username or password");
50                 request.getRequestDispatcher("/login.jsp").forward(request,
51                         response);
52             } else {
```

ASIDE generates code to validate untrusted inputs. And the corresponding marker is gone. The generated code is dependent on the rule that has been chosen and the structure of the marked code.

0 errors, 2 warnings, 0 others

Description	Resource	Path	Location	Type
Bad Security Practice (1 item)				
The return value of getParameter() at line 45 is vulnerable to be manipulated by malicious users	LoginServlet.java	/goldrush/src/uncc/goldrusl	line 45	Bad Security Practice
Java Build Path Problems (1 item)				

Figure 3. ASIDE applies selected fix via code refactoring.

Advanced Features:

Design of advanced features to address more subtle vulnerabilities such as broken access control and CSRF are still under development.

Milestones:

We will strive for releasing our Alpha version in mid 2011.