MSC11-J. Do not let session information leak within a servlet

Java servlets often must store information associated with each client that connects to them. Using member fields in the `javax.servlet.http.HttpServlet` to store information specific to individual clients is a common, simple practice. However, doing so is a mistake for the following reasons:

- In any Java servlet container, such as Apache Tomcat, `HttpServlet` is a singleton class (see MSC07-J. Prevent multiple instantiations of singleton objects for information related to singleton classes). Therefore, there can be only one instance of member variables, even if they are not declared static.
- A servlet container is permitted to invoke the servlet from multiple threads. Consequently, accessing fields in the servlet can lead to data races.
- If two clients initiate sessions with the servlet, the servlet can leak information from one client to the other client.

Java servlets provide an `HttpSession` class for storing session-specific data, which is encoded in each web request. Use of this class prevents both cross-session information leakage and data races.

Noncompliant Code Example

This noncompliant code example creates a servlet that prompts the user for an email address, then repeats the address back to the user. The previous address is stored in the `lastAddr` variable, which is an instance field.

```
public class SampleServlet extends HttpServlet {

  private String lastAddr = "nobody@nowhere.com";

  public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html>");

    String emailAddr = request.getParameter("emailAddr");

    if (emailAddr != null) {
      out.println("Email Address:");
      out.println(sanitize(emailAddr));
      out.println("<br>Previous Address:");
      out.println(sanitize(lastAddr));
    };

    out.println("<p>");
    out.print("<form action=\"");
    out.print("SampleServlet\" ");
    out.println("method=POST>");
    out.println("Parameter:");
    out.println("<input type=text size=20 name=emailAddr>");
    out.println("<br>");
    out.println("<input type=submit>");
    out.println("</form>");

    lastAddr = emailAddr;
  }

  public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {
    doGet(request, response);
  }

  // Filter the specified message string for characters
  // that are sensitive in HTML.
  public static String sanitize(String message) {
    // ...
  }
}
```

Because the `HttpServlet` class is a singleton, there is only one `lastAddr` field shared by every client who accesses the servlet. Consequently, the contents of the `lastAddr` field can be the previous setting of the field by a different client. Also, because this code example lacks thread-safety, it is possible for the `lastAddr` field to take on a stale value should two clients request the parameter simultaneously, which violates VNA01-J. Ensure visibility of shared references to immutable objects.

Noncompliant Code Example

In this noncompliant code example, the `lastAddr` field is static. It more accurately reflects the fact that there is never more than a single instance of the field. However, this code has the same behavior as the previous noncompliant code example and also violates VNA01-J. Ensure visibility of shared references to immutable objects.

```
public class SampleServlet extends HttpServlet {

  private static String lastAddr = "nobody@nowhere.com";

  // ... Other methods unchanged
}
```

Noncompliant Code Example

In this noncompliant code example, the `lastAddr` field is static and is protected from concurrent access by a separate lock object, as is recommended by LCK00-J. Use private final lock objects to synchronize classes that may interact with untrusted code. This approach guarantees thread-safety in the servlet. However, the servlet can still return the email address provided by a different session.

```
public class SampleServlet extends HttpServlet {

  private static String lastAddr = "nobody@nowhere.com";
  private static final Object lastAddrLock = new Object();

  public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html>");

    String emailAddr = request.getParameter("emailAddr");

    if (emailAddr != null) {
      out.println("Email Address::");
      out.println(sanitize(emailAddr));
      synchronized (lock) {
        out.println("<br>Previous Email Address::");
        out.println(sanitize(lastAddr));
      }
    };

    out.println("<p>");
    out.print("<form action=\"");
    out.print("SampleServlet\" ");
    out.println("method=POST>");
    out.println("Parameter:");
    out.println("<input type=text size=20 name=emailAddr>");
    out.println("<br>");
    out.println("<input type=submit>");
    out.println("</form>");

    synchronized (lock) {
      lastAddr = emailAddr;
    }
  }

  public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {
    doGet(request, response);
  }

  // Filter the specified message string for characters
  // that are sensitive in HTML.
  public static String sanitize(String message) {
    // ...
  }
}
```

Compliant Solution

This compliant solution stores the `lastAddr` parameter in the `HttpSession` object, which is provided as part of the `HttpServletRequest`. The servlet mechanism keeps track of the session, providing the client with the session's ID, which is stored as a cookie by the client's browser. The other information in the session, including the `lastAddr` attribute, is stored by the server. Consequently, the servlet provides the last email address that was presented to the servlet in the same session (avoiding race conditions with requests from other sessions). The local variables, which temporarily hold data in this example, are not vulnerable to race conditions.

```
public class SampleServlet extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html>");

    String emailAddr = request.getParameter("emailAddr");
    HttpSession session = request.getSession();
    Object attr = session.getAttribute("lastAddr");
    String lastAddr = (attr == null) ? "null" : attr.toString();

    if (emailAddr != null) {
      out.println("Email Address::");
      out.println(sanitize(emailAddr));
      out.println("<br>Previous Email Address::");
      out.println(sanitize(lastAddr));
    };

    out.println("<p>");
    out.print("<form action=\"");
    out.print("SampleServlet\" ");
    out.println("method=POST>");
    out.println("Parameter:");
    out.println("<input type=text size=20 name=emailAddr>");
    out.println("<br>");
    out.println("<input type=submit>");
    out.println("</form>");

    session.setAttribute("lastAddr", emailAddr);
  }

  public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {
    doGet(request, response);
  }

  // Filter the specified message string for characters
  // that are sensitive in HTML.
  public static String sanitize(String message) {
    // ...
  }
}
```

Risk Assessment

Use of nonstatic member fields in a servlet can result in information leakage.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
MSC11-J	Medium	Likely	High	P6	L2

Automated Detection

Tool	Version	Checker	Description
Findbugs	2.0.3	**MSF_MUTABLE_SERVLET_FIELD** **MTIA_SUSPECT_STRUTS_INSTANCE_FIELD** **MTIA_SUSPECT_SERVLET_INSTANCE_FIELD**	Implemented
Fortify	6.10.0120	**Singleton_Member_Field**	Implemented
SonarQube	6.7	**S2226**	

Related Guidelines

MITRE CWE	CWE-543, Use of Singleton Pattern Without Synchronization in a Multithreaded Context

Bibliography

[Apache 2015]	Apache Tomcat
[Fortify 2014]	Fortify Diagnostic
[J2EE API 2013]	HttpServletRequest