### 7.2D: Session management and CSRF attacks

I implemented security solutions to address two critical vulnerabilities in a web
application: some session management issues and possible CSCF attack threats. That is
why the features I have introduced to the current Spring-based online apps are custom
exception, cross-site the program I am working on now is protected against scripting
attacks.

#### 1) CSRF Attack Prevention

- ✓ Cross Site Request Forgery aka Cross-Site Reference Forgery is a web security flaw where an attacker can perform some activities on behalf of any authorized user without that user ever being knowledgeable about it. I implemented a security check into the code of the Controller class for the Coach to decrease the possibility of CSRF attacks. This check is part of Hitachi's HTTP inspection procedure and involves a check on the Referer portions of Web requests and the elimination of unauthorized requests from admittance to the Web.
- ✓ Referrer Header Comparison: Thus, by using the Referer header of the first request made, I performed the changeAge operation. Next, I extracted the Referer from the second request as Referer2, selected Referers 1 two Referers. I further thought that the request could be a kind of CSRF or attack since the session got cancelled if the data above did not equal something.

```
ClientSAD dum - mew ClientSADImpl();
dum.updateClient(client);
 * In a Of-coachwellage involve-toding-course encountries
   · Ed / 02-coechwelsepp-apring | votario-custing-course-soc
 • $7 > 02-vol-coachwebapp (secure coding course source
 + 0s 09-coachwettapp-spring
 + Cy D4-coathwelsop-spring
  + 8) > 05-coachwebopp-spring (music coding exerce-occ
  * 29 06-coachwetopp spring-hiberrate
 * Ed - 07-coachwelsop-spring-hibernale (secure coding
 + 24 + 08-session-management liccure emfro-course su
         • 29 Jana Bessacross
         . as Referenced Utvaries
                                                                                                                                                                                                                                            (lientAb) das = new (lientAbSimpl(1)

Rechittpfession sussion = (MockEttpSession) repeat.getSession();

String uname = (String) tession_epintribute('sper');

(lient_cluer o'dos_retrieveClentIname);

(lient_setApploap);

dece_updateClentIc(lient);

medi_abdStribute'=mession', "how map has been updated");

retrier_outdect.
        + in-build
 • Ed • 10-authornation (virus) coultry-course-assures in
 + 3g > 11-authorisation (secure-coding-course-sources incl.

    By - Theurbristies protected on the during many in the second man
```

- ✓ This can be mostly prevented by matching the Referer field of a request with the first request that came through the initial page request. This ensure that the request originate from the same website safely and will also prevent the other unauthorized activities that are instigated fake websites.
- When the mitigating strategy will not work: This security measure, however, will not be effective if the user's browser or the middle proxies modify or strip off the Referer header for personal reasons. When used in isolation this is inadequate because the attacker is capable of spoofing the Referer header and bypass this check. However, the two that are most effective and should be used together include the CSRF tokens and the referrer validation.

## 2) Improved Error Handling

- I made the web application more robust when I developed a global error handler for it by using the @ExceptionHandler annotation on the surface class of the project known as CoachController.java. This one simply takes it and reports it at the highest level of severity until an exception is caught which in the end is caught.
- ✓ In order to log mistakes at a very high level, I used java.util.Logger. This warrants that critical concerns are recorded as they are later deliberated subsequently within the premises.

```
GhequestMapping(*/change*)
public String changeAge(@MequestPeran(*age*) int age, Model model, MttpServletRequest request) {
    // Logic to prevent ESRF attacks
    this.referer2 = request.getHeader("referer*);
    if (!this.referer1.equals(this.referer2)) {
if is secure-coding-course-sources (secure-coding-course
★ 34 > 01-coachwebapp | secure-coding-course-sources mi
+ sil > 02-coachwebapp-spring (secure-coding-course-soc
* ## > 02-vull-coachwebapp (secure-coding-course-source)
                                                                                             HttpSession ression = request.getSession(false);
if (session != null) (
> 03-coachwebapp-spring
                                                                                                   session.invalidate(); // Invalidate session if (SRF is detected
+ @ 04-coachwebapp-spring
► 2 05-coachwebapp-spring [secure-coding-course-sec
                                                                                             model.addAttribute('message', 'CSRF attack detected. Session invalidated.');
return 'error'; // Redirect to custom error page

    → 06-coachwebapp-spring-hibernate

* $4 > 07-coachwebapp-spring-hibernate [secure-coding-
                                                                                        ClientDAD dao - new ClientDAOImpl():

▼ $4 > 08-session-management [secure-coding-course-sc
                                                                                        MockHttpSession session = (MockHttpSession) request.getSession();
String uname = (String) session.getAttribute("user");
Client client = dao.retrieveClient(uname);

★ JAXE-WS Web Services

   * Tava Resources
   . m. Referenced Libraries
                                                                                        doo.updateClient(client);
                                                                                        model.addAttribute("message", "Your age has been updated");
return "workout";
  A De build

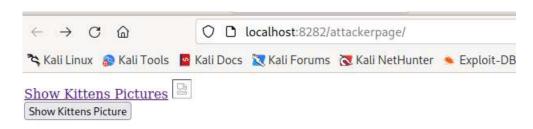
    ■ > 10-authorisation [secure-coding-course-sources mu

                                                                                         estimulandar(Exception:class)
• Bit > 11-authorisation (secure-coding-course-sources ma
                                                                                  public String handleException(Exception ex, Model model) (
    java.util logging.Logger.gottogger(CosthController.class.gutName()) logginvel:5EVENE, mukl, en);
    model.addAttribute("errorMessage", "A server error occurred. Please try again."))

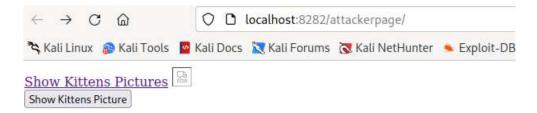
    ► 12-authorisation (secure-coding-course-sources mo

★ Bit > 13-authorisation | secure-coding-course-sources ma
• Ed > 14-authorisation (secure-coding-course-sources ma
* Did > 15-authorisation (secure-coding-course-sources ma | 86 )
```

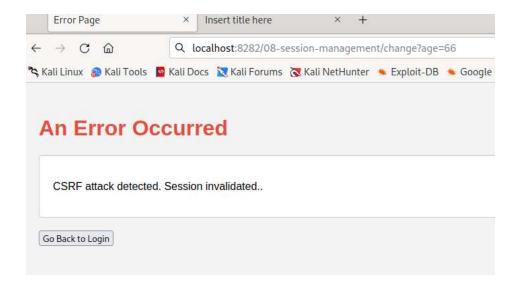
✓ What I only provided was the regular personalized error message that prevents the visitor from seeing the new/common 500 page and safeguard against any processed data if the visitor was logged in.

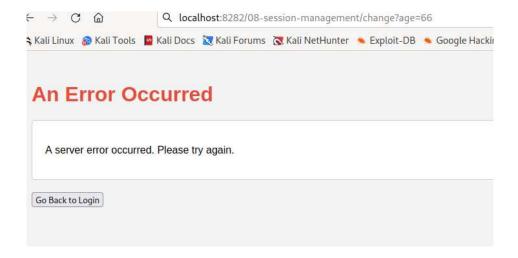






✓ A generic error page with the specific error message overlay shows instead of the internal server error 500 when the show kitten picture button is clicked.





# Prevention logic

- ✓ Global exception handling system should encompass all the uncaught mistakes to provide visibility and lead to an occurrence of an exception in the SEVERE level while presenting an understandable message during mistaken operations. In this case mistakes that can lead to the revelation of how the program operates are suppressed and from leakage.
- When the mitigating strategy will not work: But if the logging is not controlled, or its security is breached, then things with this system can go horribly wrong. Standard logs indicate the occurrence of exception information, which is violation of security, however, good log file encryption and, above all, good encryption and control of access can reduce this risk.

### 3) Session Management Fix

- ✓ This is based on firsthand experience: this person found a blatant session management vulnerability in the current program; even if, one of the browser windows was closed, an authenticated session can still remain alive in other browsers something that it should not. This is a clear security threat as it will facilitate, for instance, a user logging out of one browser session but remaining logged in to another.
- ✓ To ensure the session of the user is cleared properly after logout, I added a logout form on workout.jsp and defined /logout in LoginController.java.

### Form for Logout in Workout.jsp

```
* Bh > main
                                                  5∈ <head>
                                                  6 <meta charset="ISO-8859-1">
  + Rb > Java
                                                  7 <title>SIT218 Secure Coding - First Spring-based dynamic web app</title>
    + B→ edu
                                                  8 </head>
      ▼ 🗁 > deakin
                                                 10 <hl>${nessage}</hl>
         + @ > sit218

▼ Strip > coachwebapp

                                                 129 <form action="${pageContext.request.contextPath}/workout" method="GET">
                                                        Another workout advice?
              * P> controller
                                                         <input type="submit" value="Yes" />
                 A > CoachController.java
                                                 15 </forms

→ HttpServletRequest.java

                                                 17= <form action="change" method="GET">
                  A HttpSession.java
                                                 18
                                                        Change age to?
                                                        <input type="number" name="age"/>
                                                 19
                  A > LoginController.java
                                                         <input type="submit"/>
              → D > dao
                  A > ClientDAO.java
                                                 23= #form action="$(pageContext.request.contextPath)/logout" method="POST">
                 IR > ClientDAO(mpL)ava
                                                       Logout2 cinput type="submittovalue=17es":/>
              + m > entity
                                               26. </body>
      hibernate.cfg.xml
                                                 27 «/html>
  ▼ 200 > webapp
```

#### The logout function within LoginController.java

```
■ ≥ 08-session-management (secure-coding-course-sour)

                                                                          foo.setHttpOnly(true);
                                                                         response.addCookie(foo);
  🕨 🔩 Deployment Descriptor: 08-session-management
                                                        35
36
37
38
39
48
  . ... JAX-WS Web Services
                                                                         HttpSession session = request.getSession();
                                                                          session.setAttribute("user", client.getUsername());
 ▶ 🎘 Java Resources
  » m Referenced Libraries
                                                                          return "redirect:/workout";
  * Es build
                                                                             el.addAttribute("errorMessage", "Mrong combination of password and username");
                                                        41
42
43
44
45
46
47
48
49
50
51
 * DE > SEC
                                                                         return "plain-login";
    - Par main
      ₩ [ava
        → 😝 > edu
                                                                     uestHacaing(*/loggot*)
                                                                public String logout(MttpServletRequest request) (
           + pr > deakin
                                                                     HttpSession session - request; getSession(false);
              + (a) > sit218
                                                                     if (session i= mull) (
                                                                         presignationalidate(): // Invalidate session on los

▼ Sp > coachwebapp

                  - D controller
                                                                     return fredirect /showlogints // Redirect to tools were
                       A > CoachController.java
```

# Prevention Logic

- ✓ The session management ensures when a specific user logs out, then the session is destroyed in all the browsers in use. To protect the data, the system fully logs out the user stripping them of all access privileges to the protected resources by removing all of their sessions from the LoginController
- When the mitigating strategy will not work: This tactic however can prove to be quite dangerous if the sessions are handled in the wrong manner; instance; the sessions may be client side cached or the attackers change session tokens in a commonly referred to as session fixation assault. To bolster up this defense, measures like; the use of secure cookies to reduce the risks of session hijacking, while the use of ID regeneration after each login should be established.