1. Reflected XSS
    - https://github.com/securitydatspring2019/xssDemoServer
    - Eksempel på phishing mail:
      Dear user

      We have implemented a cool new feature on our super-searcher. Please try it, using the following link, but you might want to read the section below "Safety on the Web" first:

      http://www.dat-security.dk/xss/SearchDemo

      Safety on the web: Is this link safe?
      If you read this before you clicked the link you are “wise” :-)

      You should never click untrusted links sent via mails. If you want to be safe, open your browser and navigate to the usual address used to access our site, and login as usual.

      You can then verify that the link above (using the first part of the address) is the same as the one you normally use. If it is, it’s safe to come back and click the link.

      We hope you will enjoy our new fast search feature

      Hacker Inc.
2. Stored XSS
    - https://github.com/Lars-m/xssServer
3. DOM XSS
    -

- Ved brug af user validation på input forms og validation for det returneret input. Dette vil gøre det umuligt for en hacker at opsnappe ting som document.cookie.
- Brug sanitizer for at undgå f.eks. <script> tag som user input.
- Brug encoder for at f.eks. få HTML kodet input kodet om til rå tekst.
- Brug escape formatering i user inputs f.eks. “/” til at escape “<” i et <script> tag.

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    String plainUserData = request.getParameter("input");
    String encoded = "<b>Change me</b> to show the original string, HTML-Encoded";
    String simpleTags = "<u>Change me</u> to allow simple tags (bold and italic), but no other tags";
    String noTags = "<b>Change me</b> to allow NO tags at all";
    PolicyFactory policy = new HtmlPolicyBuilder()
            .allowElements("b", "i")
            .toFactory();
    PolicyFactory policy2 = new HtmlPolicyBuilder()
            .toFactory();
    String safeHTML2 = policy2.sanitize(noTags);
    String safeHTML = policy.sanitize(simpleTags);
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet Sanitizer</title>");
        out.println("</head>");
        out.println("<body style=\"font-family:sans-serif\">");
        out.println("<h2>Encoder + Sanitizer Demo</h2>");
        out.println("<p>Use your browsers 'inspect' button to se the actual content of the string below</p>");
        out.print(plainUserData);
        out.println("<br/><br/>");
        out.println(Encode.forHtml(encoded));
        out.println("<br/><br/>");
        out.println(safeHTML);
        out.println("<br/><br/>");
        out.println(safeHTML2);
        out.print("<div style=\"margin-top:25px;\"><a href=\"" + request.getContextPath() + "/index.html\">Home</a></div>");
        out.println("</body>");
        out.println("</html>");
    }
}
```