





```
Project: Alpha - (file_path)/XSSRequestWrapper.java

import java.util.regex.Pattern;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletRequestWrapper;

public class XSSRequestWrapper extends HttpServletRequestWrapper {

    public XSSRequestWrapper(HttpServletRequest servletRequest) {
        super(servletRequest);
    }

    @Override
    public String[] getParameterValues(String parameter) {
        String[] values = super.getParameterValues(parameter);

        if (values == null) {
            return null;
        }

        int count = values.length;
        String[] encodedValues = new String[count];
        for (int i = 0; i < count; i++) {
            encodedValues[i] = stripXSS(values[i]);
        }

        return encodedValues;
    }

    @Override
    public String getParameter(String parameter) {
        String value = super.getParameter(parameter);

        return stripXSS(value);
    }

    @Override
    public String getHeader(String name) {
        String value = super.getHeader(name);
        return stripXSS(value);
    }

    private String stripXSS(String value) {
```

```
        return encodedValues;
    }

    @Override
    public String getParameter(String parameter) {
        String value = super.getParameter(parameter);

        return stripXSS(value);
    }

    @Override
    public String getHeader(String name) {
        String value = super.getHeader(name);
        return stripXSS(value);
    }

    private String stripXSS(String value) {
        if (value == null) {
            // NOTE: It's highly recommended to use the ESAPI library and uncomment the following line to
            // avoid encoded attacks
            // value = ESAPI.encoder().canonicalize(value);

            // Avoid null characters
            value = value.replaceAll("", "");

            // Avoid anything between script tags
            Pattern scriptPattern = Pattern.compile("<script>.*</script>", Pattern.CASE_INSENSITIVE);
            value = scriptPattern.matcher(value).replaceAll("");

            // Avoid anything in a src="" type of expression
            scriptPattern = Pattern.compile("src=[\"'].*[\"']\\\"", Pattern.CASE_INSENSITIVE | Pattern.MULTILINE | Pattern.DOTALL);
            value = scriptPattern.matcher(value).replaceAll("");

            scriptPattern = Pattern.compile("src=[\"'].*[\"'].*[\"']\\\"", Pattern.CASE_INSENSITIVE | Pattern.MULTILINE | Pattern.DOTALL);
            value = scriptPattern.matcher(value).replaceAll("");

            // Remove any lonesome </script> tag
            scriptPattern = Pattern.compile("</script>", Pattern.CASE_INSENSITIVE);
            value = scriptPattern.matcher(value).replaceAll("");
```

```
        // Avoid expression(...) expressions
        scriptPattern = Pattern.compile("expression\\((.*)\\)", Pattern.CASE_INSENSITIVE | Pattern.MULTILINE | Pattern.DOTALL);
        value = scriptPattern.matcher(value).replaceAll("");

        // Avoid javascript: expressions
        scriptPattern = Pattern.compile("javascript:", Pattern.CASE_INSENSITIVE);
        value = scriptPattern.matcher(value).replaceAll("");

        // Avoid vbscript: expressions
        scriptPattern = Pattern.compile("vbscript:", Pattern.CASE_INSENSITIVE);
        value = scriptPattern.matcher(value).replaceAll("");

        // Avoid onload= expressions
        scriptPattern = Pattern.compile("onload(.*?)=", Pattern.CASE_INSENSITIVE | Pattern.MULTILINE | Pattern.DOTALL);
        value = scriptPattern.matcher(value).replaceAll("");
    }

    return value;
}
```