Preventing Injection Attacks Report

Ethan Kalika

ITIS 4221/5221

March 14th, 2023

PREVENTING INJECTION ATTACKS REPORT - TUNESTORE

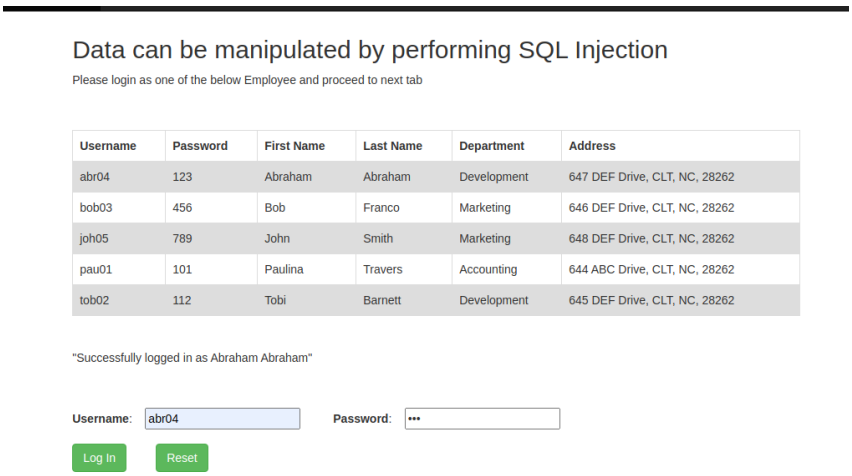
TABLE OF CONTENTS

1.0 General Information

In this project, you are going to fix injection vulnerabilities in the "penetration_test" project distributed in the class VM.

2.0 SQL Injection Mitigations

In the vm under SQL injection attacks page I signed in as Abraham.



Data can be manipulated by performing SQL Injection

Please login as one of the below Employee and proceed to next tab

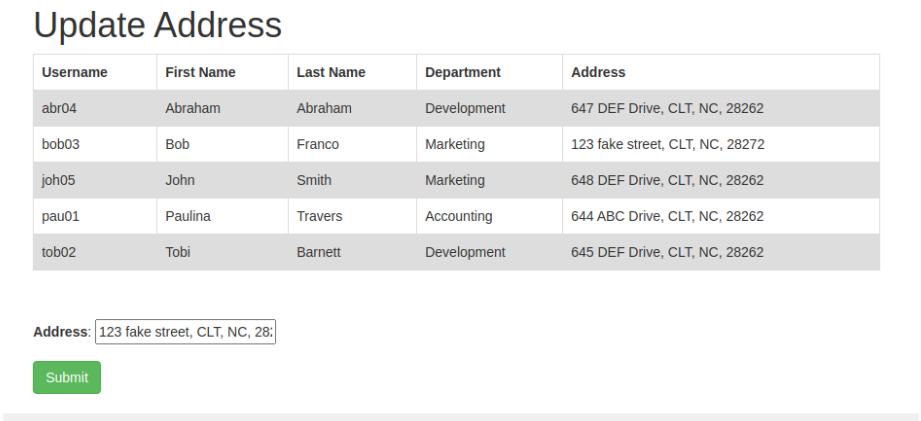
Username	Password	First Name	Last Name	Department	Address
abr04	123	Abraham	Abraham	Development	647 DEF Drive, CLT, NC, 28262
bob03	456	Bob	Franco	Marketing	646 DEF Drive, CLT, NC, 28262
joh05	789	John	Smith	Marketing	648 DEF Drive, CLT, NC, 28262
pau01	101	Paulina	Travers	Accounting	644 ABC Drive, CLT, NC, 28262
tob02	112	Tobi	Barnett	Development	645 DEF Drive, CLT, NC, 28262

"Successfully logged in as Abraham Abraham"

Username: abr04 Password: •••

Log In Reset

You are only supposed to be allowed to modify your address when logged in, however you can also change the address of other users by inserting the code "where username = 'bob03'#" to the address field.



Update Address

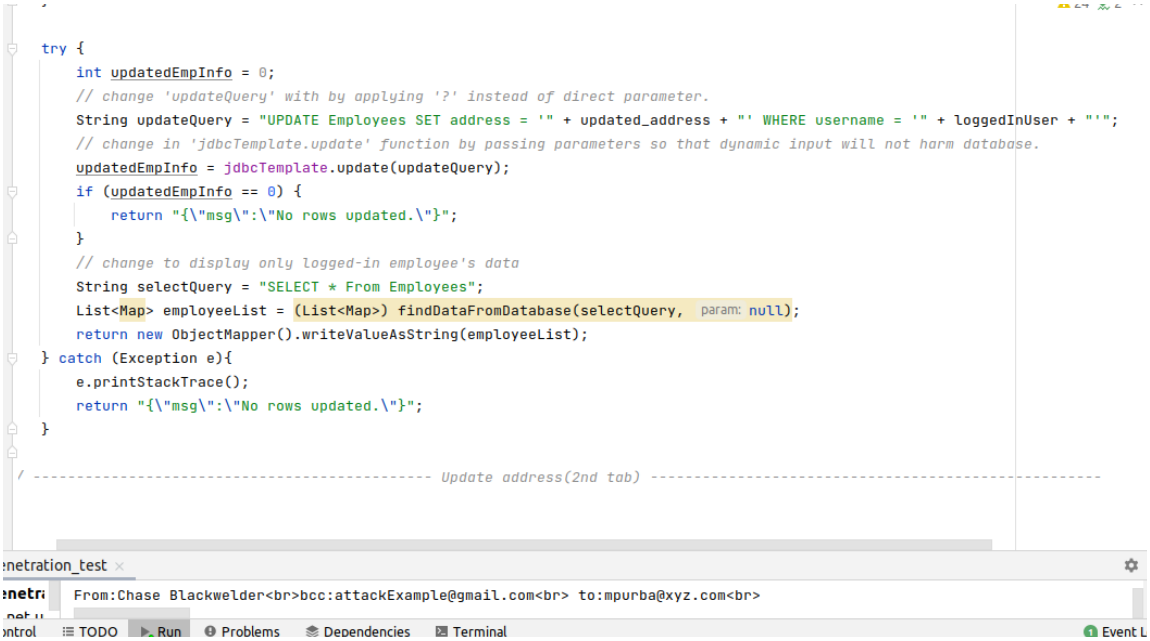
Username	First Name	Last Name	Department	Address
abr04	Abraham	Abraham	Development	647 DEF Drive, CLT, NC, 28262
bob03	Bob	Franco	Marketing	123 fake street, CLT, NC, 28272
joh05	John	Smith	Marketing	648 DEF Drive, CLT, NC, 28262
pau01	Paulina	Travers	Accounting	644 ABC Drive, CLT, NC, 28262
tob02	Tobi	Barnett	Development	645 DEF Drive, CLT, NC, 28262

Address: 123 fake street, CLT, NC, 28;

Submit

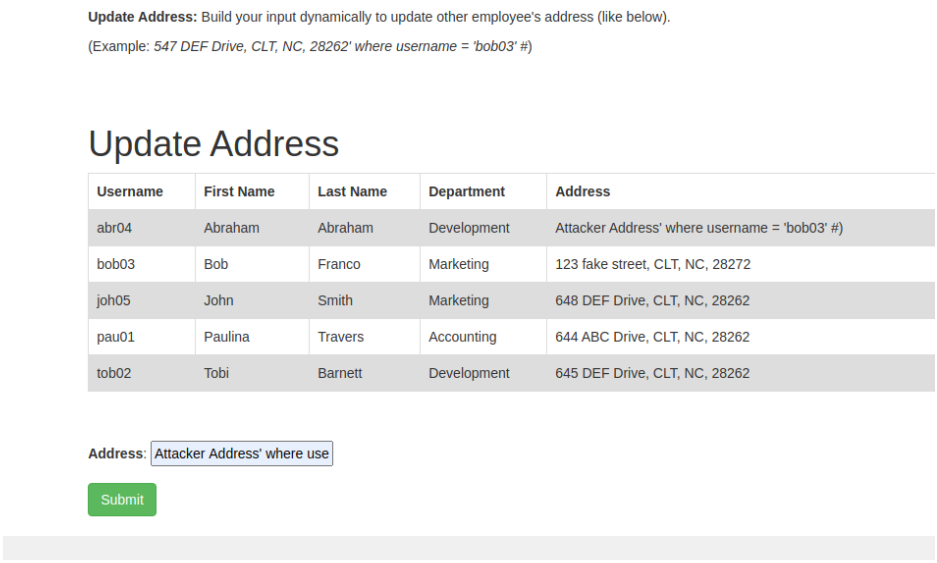
I adjusted Bob's address while still logged in as Abraham. This code allowed me to accomplish this.

The updateQuery string in this code simply adds to the user's input, which means that if the user adds code like we did, it will be executed.



I modified the code to look like this.

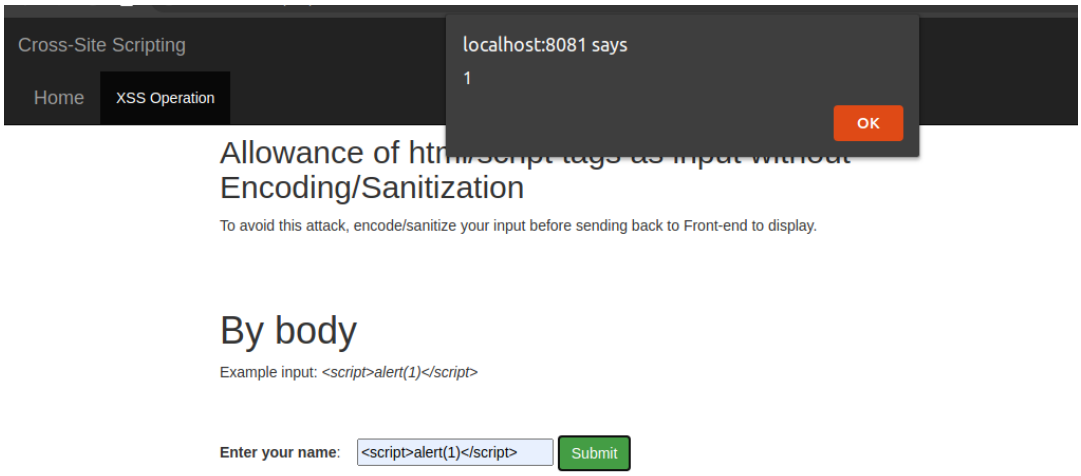
This is what occurs when a sql injection is attempted now.



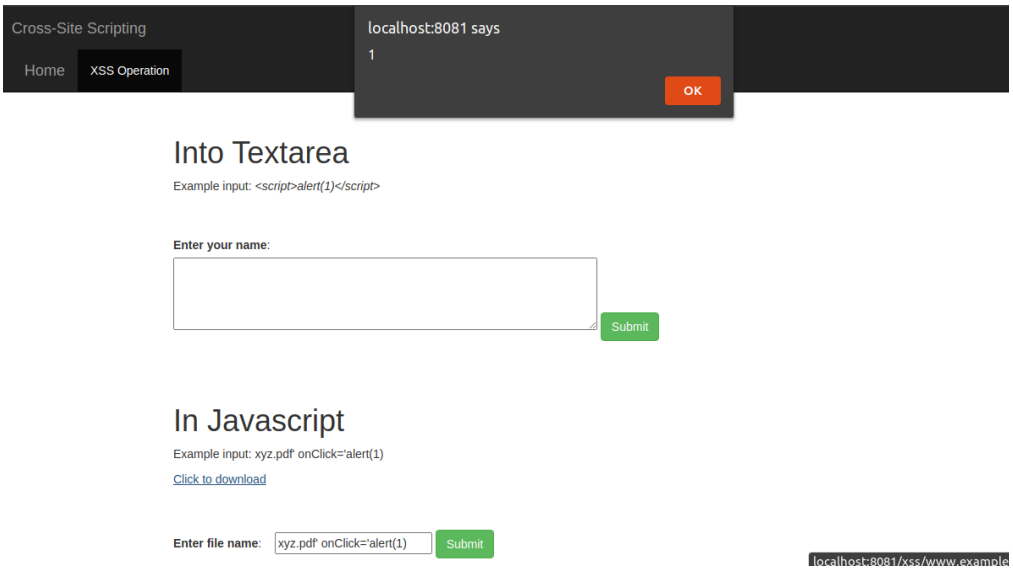
My address has been altered with the sql injection code rather than altering Bob's.

3.0 XSS Mitigations

Here are two instances of cross-site scripting in action: one is javascript and the other is html, and both utilize the alert(1) sample code that is shown in the image.



^ HTML



^JavaScript

This is conceivable because the functions were not encoded, allowing them to be executed if they were coded. The wrong code and correct code are shown below.

Application.java | index.html | XssController.java

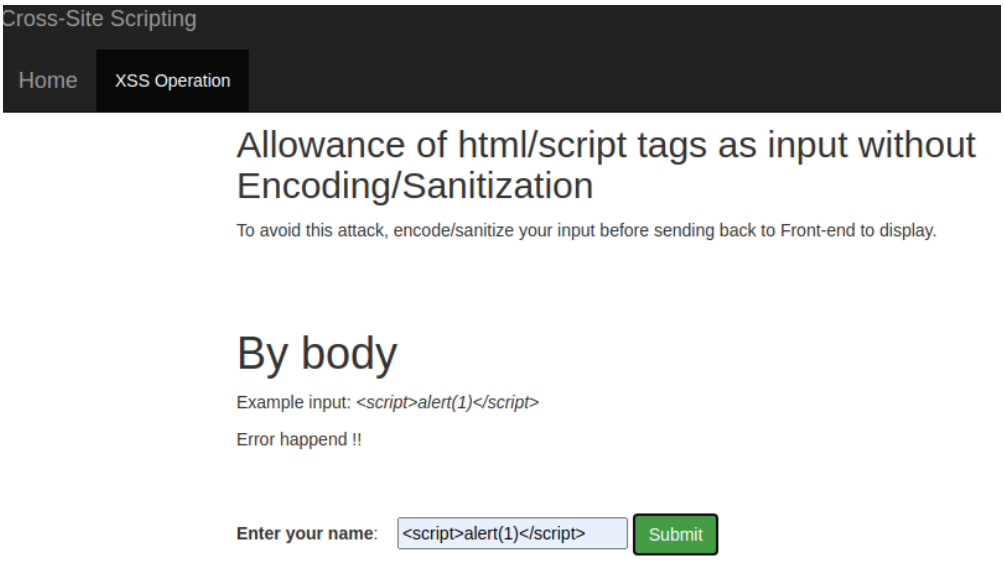
```
13      public String xss_index() { return "xss/index"; }
16
17      @GetMapping("/body_xss")
18      @ResponseBody
19      public String body_xss(@RequestParam String body_tagVal) throws Exception {
20          return body_tagVal;
21      }
22
23      @GetMapping("/textarea_xss")
24      @ResponseBody
25      public Object textarea_xss(@RequestParam String textarea_tagVal) throws Exception {
26          return textarea_tagVal;
27      }
28
29      @GetMapping("/js_xss")
30      @ResponseBody
31      public Object js_xss(@RequestParam String js_tagVal) throws Exception {
32          return js_tagVal;
33      }
34
35  }
```

^ broken code

```java
public class XssController {

    @GetMapping("/")
    public String xss_index() { return "xss/index"; }

    @GetMapping("/body_xss")
    @ResponseBody
    public String body_xss(@RequestParam String body_tagVal) throws Exception {
        return escapeHtml(body_tagVal);
    }

    @GetMapping("/textarea_xss")
    @ResponseBody
    public Object textarea_xss(@RequestParam String textarea_tagVal) throws Exception {
        return escapeHtml(textarea_tagVal);
    }

    @GetMapping("/js_xss")
    @ResponseBody
    public Object js_xss(@RequestParam String js_tagVal) throws Exception {
        return escapeJavaScript(js_tagVal);
    }
```

^ fixed code

This will resolve the XSS issue, and here is my effort to do XSS on the corrected code.



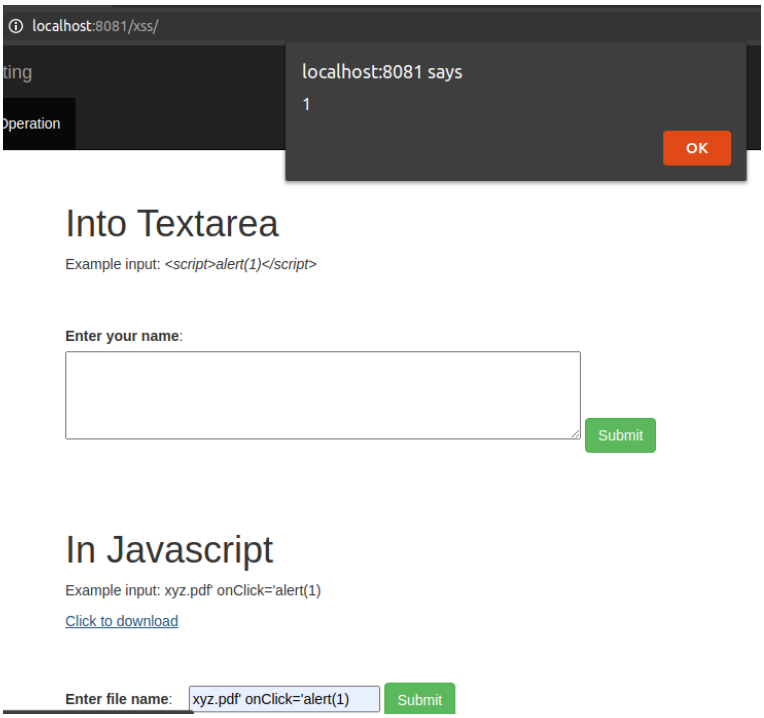
^ HTML encoding

^ java encoding

It's crucial to understand that using the incorrect encoding function won't solve the issue.
This is an illustration of what happens if you do html encoding on javascript.

```
    @GetMapping("/js_xss")
    @ResponseBody
    public Object js_xss(@RequestParam String js_tagVal) throws Exception {
        return escapeHtml(js_tagVal);
    }

}
```

^ wrong encode



^ still broken

Choosing the right encoding function is crucial.

4.0 Command Injection Mitigations

You may cause a computer to execute more instructions by adding more commands to the area that requests an IP address.

Inject Command

PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=55 time=9.42 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=55 time=9.35 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=55 time=9.41 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 9.353/9.393/9.418/0.028 ms
logs
Penetration_test.iml
pom.xml
README.md
src
target
itis42215221

IP Address: 8.8.8.8 && ls && whoami

Submit

^ The ping command is combined with the ls and whoami commands in the code.

```
onseBody
c Object command_injected(@RequestParam String ip_address) {
ap<String, String> response_data = new HashMap<~>();
ry {
    String output = "";
    String[] command = {"/bin/bash", "-c", "ping -c 3 " + ip_address};
    Process proc = Runtime.getRuntime().exec(command);
    proc.waitFor();

    String line = "";
    BufferedReader inputStream = new BufferedReader(new InputStreamReader(proc.getInputS
    BufferedReader errorStream = new BufferedReader(new InputStreamReader(proc.getErrorS
    while ((line = inputStream.readLine()) != null) {
        output += line + "<br/>";
    }
    inputStream.close();
    while ((line = errorStream.readLine()) != null) {
        output += line + "<br/>";
    }
    errorStream.close();
    proc.waitFor();

    response_data.put("status", "success");
```

^The broken code is shown above.

This is incorrect since the command just converts the input into an array string, which causes the commands ls and whoami to be appended to the string and executed.

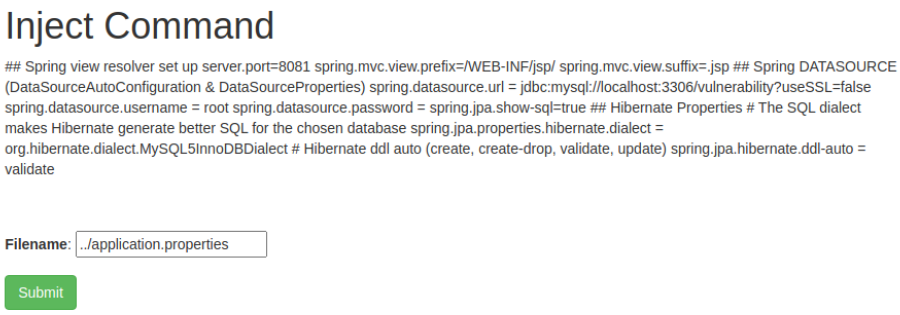
```java
public Object command_injected(@RequestParam String ip_address) {
    Map<String, String> response_data = new HashMap<~>();
    try {
        String output = "";
        String[] command = {"/bin/bash", "-c", "ping -c 3 " + ip_address};
        ProcessBuilder processBuilder = new ProcessBuilder();
        processBuilder.command("ping","-c","3",ip_address);
        Process p = processBuilder.start();
        p.waitFor();

        String line = "";
        BufferedReader inputStream = new BufferedReader(new InputStreamReader(p.getInputStream()));
        BufferedReader errorStream = new BufferedReader(new InputStreamReader(p.getErrorStream()));
        while ((line = inputStream.readLine()) != null) {
            output += line + "<br/>";
        }
        inputStream.close();
        while ((line = errorStream.readLine()) != null) {
            output += line + "<br/>";
        }
        errorStream.close();
        p.waitFor();
```

If further information is added when using the process builder, the computer will no longer recognize the IP address.

Does not currently recognize the address.

5.0 Path Manipulation Mitigations

You can shift the path to a location you weren't supposed to reach depending on the page by adding some creative code. There are several ways to accomplish this. This is an illustration of route alterations.



^opening the application's settings page

This is feasible because there is no data sanitation, which means that you must create a separate program to prevent the faulty code,../, from functioning.

```
29
30    etMapping("/viewFile")
31    esponseBody
32    blic Map<String, String> view_file(@RequestParam String file_name) throws Exception {
33      Map<String, String> response_data = new HashMap<~>();
34
35      try {
36          Resource resource = resourceLoader.getResource( s: "classpath:files/" + file_name
37          File file = resource.getFile();
38          String text = new String(Files.readAllBytes(file.toPath()));
39
40          response_data.put("status", "success");
41          response_data.put("msg", text);
42          return response_data;
43      } catch (IOException e) {
44          e.printStackTrace();
45          response_data.put("status", "error");
46          response_data.put("msg", "No output found");
47          return response_data;
48      }
```

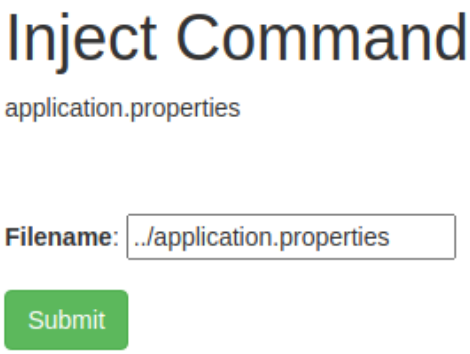
^merely creates a string from the input

```java
@GetMapping("/")
public String path_manipulation_index() { return "path_manipulation/index"; }

@GetMapping("/viewFile")
@ResponseBody
public Map<String, String> view_file(@RequestParam String file_name) throws Exception {
    Map<String, String> response_data = new HashMap<~>();

    try {
        Resource resource = resourceLoader.getResource( s: "classpath:files/" + file_name);
        File file = resource.getFile();
        SecurityEnhancedAPI secure = new SecurityEnhancedAPI();
        String text = secure.getFileName(file.getName());

        response_data.put("status", "success");
        response_data.put("msg", text);
        return response_data;
    } catch (IOException e) {
        e.printStackTrace();
        response_data.put("status", "error");
        response_data.put("msg", "No output found");
        return response_data;
t roles0_.user_id as user_id1_2_0_, roles0_.role_id as role_id2_2_0_, role1_.id as id1_0_1_, role1_.
```

^ code was repaired, but it uses the security-enhanced API, which I'll demonstrate below.

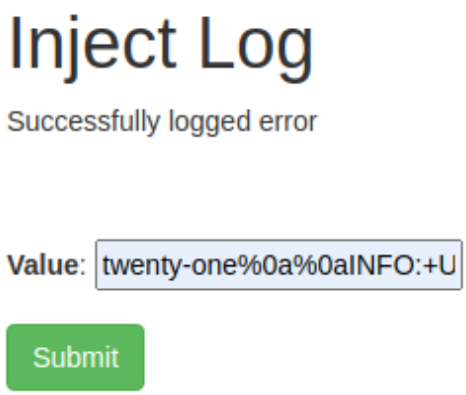
```java

public class SecurityEnhancedAPI {
    public String getFileName(String temp) throws FileNotFoundException{
        String regex = "^[\\w,\\s-]+\\.[A-Za-z]{3,10}$";
        Pattern pattern = Pattern.compile(regex);
        Matcher match = pattern.matcher(temp);


        if(match.find()){
            return temp;
        }else{
            throw new FileNotFoundException("Wrong fileName");
        }
    }
```

I'm attempting to manipulate the route in the fixed code using this function, which is invoked in the code.

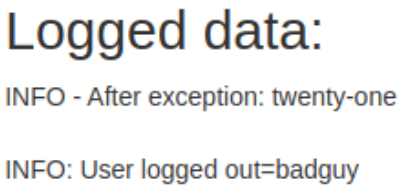


6.0 Log Forging Mitigations

You may get the logged base to run code and create bogus events by placing it in a field that is logged.



^ logged code



^ that code made it say that the user badguy has logged out.

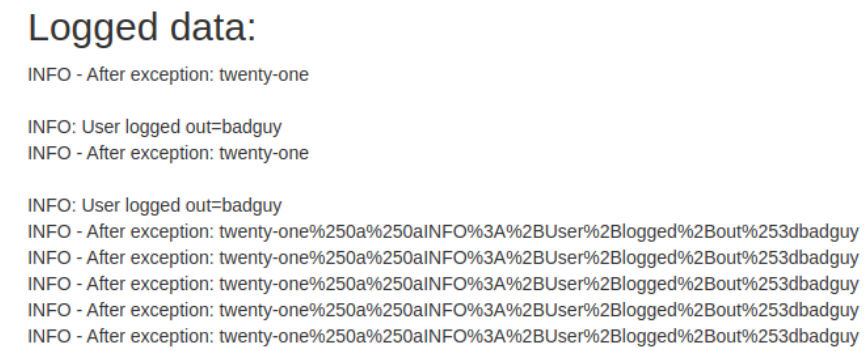
broken code down below

```
nseBody                                                                    10
: Object log_injected(@RequestParam String log_value) {
ap<String, String> response_data = new HashMap<~>();
gger logger = LogManager.getLogger(Log_injectionController.class);
y {
   SimpleLayout layout = new SimpleLayout();
   FileAppender appender = new FileAppender(layout, filename: "./logs/Custom_log_file.log"
   logger.removeAllAppenders();
   logger.addAppender(appender);
   logger.setLevel(Level.DEBUG);
   logger.setAdditivity(true);

   log_value = java.net.URLDecoder.decode(log_value, StandardCharsets.UTF_8.name());
   Integer parsed_log_value = Integer.parseInt(log_value);
   logger.info("Value to log: " + parsed_log_value);

   response_data.put("status", "success");
   response_data.put("msg", "Successfully logged without error");
   return response_data;
catch (Exception e) {
   logger.info("After exception: " + log_value);
   response_data.put("status", "error");
   response_data.put("msg", "Successfully logged error");
   return response_data;
```

^The log may be executed as code as it is not being encoded.

```
34          Logger logger = LogManager.getLogger(Log_injectionController.class);          10
35          try {
36              SimpleLayout layout = new SimpleLayout();
37              FileAppender appender = new FileAppender(layout, filename: "./logs/Custom_log_file.
38              logger.removeAllAppenders();
39              logger.addAppender(appender);
40              logger.setLevel(Level.DEBUG);
41              logger.setAdditivity(true);
42
43              log_value = java.net.URLEncoder.encode(log_value, StandardCharsets.UTF_8.name());
44              Integer parsed_log_value = Integer.parseInt(log_value);
45              logger.info("Value to log: " + parsed_log_value);
46
47              response_data.put("status", "success");
48              response_data.put("msg", "Successfully logged without error");
49              return response_data;
50          } catch (Exception e) {
51              logger.info("After exception: " + log_value);
52              response_data.put("status", "error");
53              response_data.put("msg", "Successfully logged error");
54              return response_data;
55          }
```

^ The log will stop running what is logged by switching the decodes to encode.



^Instead of executing and impacting the logs, the entire line of code is now recorded.

7.0 XPath Injection Mitigations

When you provide an accepted email, the xml request returns the id but with the code 'or 1='1>, which changes how the request behaves. You can get the code and all the ids here.

^Instead of only the mpurba ID, all user IDs are contained in the [].

```
esponseBody
blic Object xpath_injected(@RequestParam String email_address) {
    Map<String, String> response_data = new HashMap<~>();
    try {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        factory.setNamespaceAware(true);
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document doc = builder.parse( uri: "src/main/resources/files/customer.xml");

        XPathFactory xpathFactory = XPathFactory.newInstance();
        XPath xpath = xpathFactory.newXPath();

        List<String> id_list = new ArrayList<>();
        XPathExpression expression = xpath.compile( expression: "/customers/customer[email = '" + email_address + "']/
        NodeList nodes = (NodeList) expression.evaluate(doc, XPathConstants.NODESET);
        for (int i = 0; i < nodes.getLength(); i++)
            id_list.add(nodes.item(i).getNodeValue());

        response_data.put("status", "success");
        response_data.put("msg", Arrays.toString(id_list.toArray()));
        return response_data;
    } catch (Exception e) {
        e.printStackTrace();
```

The id doesn't have data sanitization because it uses the user's recently obtained email address, which makes this hack possible.

```
50      try {
51          DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
52          factory.setNamespaceAware(true);
53          DocumentBuilder builder = factory.newDocumentBuilder();
54          Document doc = builder.parse( uri: "src/main/resources/files/customer.xml");
55
56          XPathFactory xpathFactory = XPathFactory.newInstance();
57          XPath xpath = xpathFactory.newXPath();
58
59          List<String> id_list = new ArrayList<>();
60
61          SimpleVariableResolver resolver = new SimpleVariableResolver();
62          resolver.addVariable(new QName( namespaceURI: null, localPart: "email_val"),email_address);
63          xpath.setXPathVariableResolver(resolver);
64          XPathExpression expression = xpath.compile( expression: "/customers/customer[email =$email_val]/id/text()");
65
66          NodeList nodes = (NodeList) expression.evaluate(doc, XPathConstants.NODESET);
67          for (int i = 0; i < nodes.getLength(); i++)
68              id_list.add(nodes.item(i).getNodeValue());
69
70          response_data.put("status", "success");
71          response_data.put("msg", Arrays.toString(id_list.toArray()));
72          return response_data;
```

By including the simple variable resolver Apart from an email address, it won't let more information to be attributed. The code for the simple variable resolver is shown.

```java
package net.uncc.app.xpath_injection;

import javax.xml.namespace.QName;
import javax.xml.xpath.XPathVariableResolver;
import java.util.HashMap;
import java.util.Map;

public class SimpleVariableResolver implements XPathVariableResolver {

    private final Map<QName, Object> vars = new HashMap<QName,Object>();

    public void addVariable(QName name, Object value){ vars.put(name,value);}

    public Object resolveVariable(QName variableName){ return vars.get(variableName);}
}
```

Trying the identical code from previously on the corrected Xpath results in the following.

The input just won't be read by it.

8.0 SMTP Header Injection Mitigations

You may change the headers that indicate where to send emails by adding code to the email's name field. The code inserted in this instance was nbcc:attackExample@gmail.com.

From:Chase Blackwelder
bcc:attackExample@gmail.com
to:mpurba@xyz.com
Message:you have been hacked

First Name: Chase Blackwelder\nbcc:atta

Email: mpurba@xyz.com

Comment: you have been hacked

Submit

^The bcc field wasn't present before; it's now there.

Here is the broken code

```
19    public String smtp_header_index() { return "smtp_injection/index"; }
22
23    @GetMapping("/form")
24    @ResponseBody
25    public String smtp_header_submit(@RequestParam String customer_firstName,@RequestPa
26
27        String name = customer_firstName;
28        String email = customer_email;
29        String comment = customer_comments;
30        String to = "root@localhost";
31        String subject = "My Subject";
32
33        String headers = "From:" + name + "\\n" + " to:" + email + "\\n";
34        String[] split = headers.split( regex: "\\\\n");
35        String y="";
36        for (int i = 0; i < split.length; i++) {
37            y += split[i];
38            y += "<br>";
39        }
40        System.out.println(y);
41        return y + " Message:" + comment;
42    }
43
```

The additional code is performed because, as previously, there is no data sanitization.

Here is the fixed code

SecurityEnhancedAPI.java | SmtpController.java

```
29          SecurityEnhancedAPI secure = new SecurityEnhancedAPI();
30
31          String name = null;
32          try {
33              name = secure.getSafeString(customer_firstName);
34          } catch (FileNotFoundException e) {
35              e.printStackTrace();
36          }
37          String email = null;
38          try {
39              email = secure.getEmail(customer_email);
40          } catch (FileNotFoundException e) {
41              e.printStackTrace();
42          }
43          String comment = customer_comments;
44          String to = "root@localhost";
45          String subject = "My Subject";
46
47          String headers = "From:" + name + "\\n" + " to:" + email + "\\n";
48          String[] split = headers.split( regex: "\\\\n");
49          String y="";
50          for (int i = 0; i < split.length; i++) {
51              y += split[i];
```

^ Using the security-enhanced API, it verifies that the email and name are valid strings for usage.

Here is the API code

getSafeString

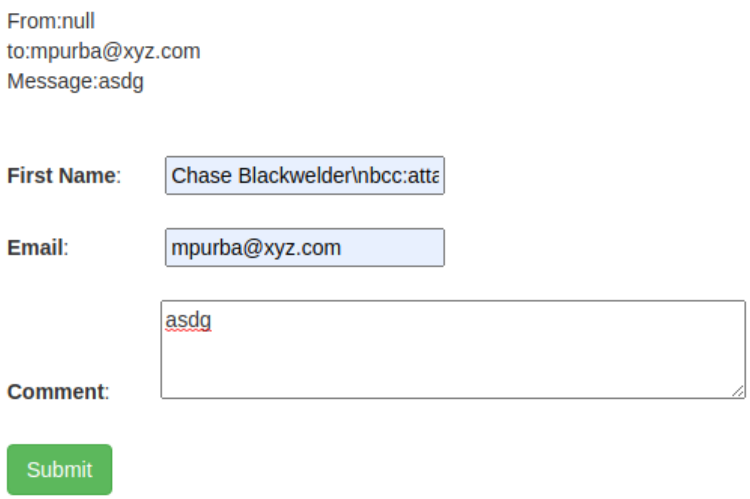
SecurityEnhancedAPI.java | SmtpController.java

```
19      }
20
21      public String getSafeString(String temp) throws FileNotFoundException{
22          String regex = "^[.\\p{Alnum}\\p{Space}]{0,1024}$";
23          Pattern pattern = Pattern.compile(regex);
24          Matcher match = pattern.matcher(temp);
25
26
27          if(match.find()){
28              return temp;
29
30          }else{
31              throw new FileNotFoundException("WrongSafeString");
32          }
33      }
34
```

getEmail

```
public String getEmail(String temp)throws FileNotFoundException{
    String regex = "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}";
    Pattern pattern = Pattern.compile(regex);
    Matcher match = pattern.matcher(temp);

    if(match.find()){
        return temp;
    }else{
        throw new FileNotFoundException("WrongEmail");
    }
```

Here is the attack on the fixed code



The bcc label was not added to the smtp header by the same assault, as can be seen.