



*{Client Name}*

*{Month} {Year}*

# *{Client Name} Web Application Security Assessment*

## Client Details

Company Name: {Client Name}

Contact Person: {Person Name}

Address: {Address Name}

Email: {Email Address}

Telephone: {Telephone Number}

## Document History

Version	Date	Author	Remark
1.0	{Date}	{Author}	Document Creation

# Table of Contents

<b>1. About Payatu</b>	4
<b>2. Project Details</b>	5
2.1 Executive Summary	5
2.2 Scope and Objective	5
2.3 Project Timeline	5
2.4 Technological Impact Summary	6
2.5 Business Impact Summary	6
2.6 Testing Environment	7
2.7 Vulnerability Chart	8
2.8 Vulnerabilities by Category	9
2.9 Table of Findings	10
2.10 Application Strengths	11
2.11 Application Weaknesses	12
<b>3. Technical Findings</b>	13
3.1 PY-CL-001: Application vulnerable to SSRF through PDF generator	13
3.2 PY-CL-002: Missing Rate Limit Checks at Login Endpoint	17
3.3 PY-CL-003: Ability to delete any user's document with a known ID	19
3.4 PY-CL-004: Ability to delete any user's Note with a known ID	21
3.5 PY-CL-005: Ability to read any user's Note and Document whose ID is known	23
3.6 PY-CL-006: Application vulnerable to XSS via a crafted link in Note	25
3.7 PY-CL-007: Application vulnerable to XSS via Note's title on {Page Name}	27
3.8 PY-CL-008: Application vulnerable to XSS via Annotation's body rendered on {Page Name}	29
3.9 PY-CL-009: Application Vulnerable to XSS via crafted uploaded file's name on {Page Name}	31
3.10 PY-CL-010: Application vulnerable to WebSocket hijacking	33
3.11 PY-CL-011: Security headers missing or set to insecure values	35
3.12 PY-CL-012: GraphQL endpoint returns Error Stack in response	37
<b>4. We Prescribe</b>	39
<b>5. Appendix</b>	41
5.1 References	41
5.2 Observations	41
5.3 Failed Test Cases	42
5.3.1 SQL Injections	42
5.3.2 OAuth/SSO Misconfiguration	44

## 1. About Payatu

Payatu is a research-powered security testing service organization specialized in IoT and embedded products, web, mobile, cloud & infrastructure security assessments, and in-depth technical security training. Our state-of-the-art research, methodologies, and tools ensure the safety of our client's assets.

At Payatu, we believe in following one's passion, and with that thought, we have created a world-class team of researchers and executors who are bending the rules to provide the best security services. We are a passionate bunch of folks working on the latest and leading-edge security technology.

We are proud to be part of a vibrant security community and don't miss any opportunity to give back. Some of the contributions in the following fields reflect our dedication and passion.

- nullcon - nullcon security conference is an annual security event held in Goa, India. After years of effort put in the event, it has become a world-renowned platform to showcase the latest and undisclosed research.
- hardware.io - Hardware security conference is an annual hardware security event held in The Hague, Netherlands. It is being organized to answer emerging threats and attacks on hardware. We aim to make it the largest platform, where hardware security innovation happens.
- Dedicated fuzzing infrastructure - We are proud to be one of the few security research-based companies to own an in-house infrastructure and hardware for distributed fuzzing of software such as browsers, client, and server applications.
- null - It all started with null - The open security community. It's a registered non-profit society and one of the most active security community. null is driven totally by passionate volunteers.
- Open source - Our team regularly authors open source tools to aid in security learning and research.
- Talks and Training: Our team delivers talks/highly technical training in various international security and hacking conference, i.e., DEFCON Las Vegas, BlackHat Las Vegas, HITB Amsterdam, Consecwest Vancouver, nullcon Goa, HackinParis Paris, Brucon Belgium, zer0con Seoul, PoC Seoul to name few.

We are catering to a diverse portfolio of clients across the world, who are leaders in banking, finance, technology, healthcare, manufacturing, media houses, information security, and education, including government agencies. Having various empanelment and accreditations, along with a strong word of mouth, has helped us win new customers. Our thorough professionalism and quality of work have brought repeat business from our existing clients. We thank you for considering our security services and requesting a proposal. We look forward to extending the expertise of our passionate, world-class professionals to achieve your security objectives.

## 2. Project Details



### 2.1 Executive Summary

Security Assessment of {Client Name} web application has been performed, considering below common security issues:

- ✓ If proper access control is implemented across teams.
- ✓ If proper Authorization and Authentication System is implemented.
- ✓ If proper error handling is done to avoid exposing API Keys.
- ✓ If the user input is properly escaped.

Overall security postures of the application are moderate. However, some of the security controls/measures have not been properly thought of/implemented during the design and coding of the application. The exploitation is not very likely due to randomness in identifiers.

The security assessment revealed 0 critical severity security issue, 1 high severity security issue, 11 medium severity issues, 0 low severity issues in this application. The consolidated summary of the assessment has been presented in the Executive Summary section. Additional information is contained within the Detailed Vulnerability Information section of this report.



### 2.2 Scope and Objective

The scope of this assessment was limited to web applications created by {Client Name} located at {URL}



### 2.3 Project Timeline

The security assessment was performed for {Number of Days} days from {Start Date} to {End Date}.



## 2.4 Technological Impact Summary

---

The Payatu security team performs real-time security assessments on the web application. These assessments aim to uncover any security issues in the assessed web application, explain the impact and risks associated with the found issues, and provide guidance in the prioritization and remediation steps.

It was identified that the web application did not implement Access Control features, especially '{Feature Name}' and '{Feature Name}.' Although features are less likely to be exploited due to randomness in the generated document ID. The application also doesn't properly sanitize the user input before rendering on the page, leading to XSS at many endpoints and even SSRF using a combination of features. The web application also lacks rate-limiting at certain endpoints.



## 2.5 Business Impact Summary

---

We identified the following business impacts:

- ▶ An attacker can potentially take over accounts and obtain private information if the attacker conducts a successful XSS attack.
- ▶ An attacker can potentially take over accounts by brute-forcing against a list of known passwords.
- ▶ An attacker can get access to server infrastructure and source code.



## 2.6 Testing Environment

---

To perform web application portal assessment, we have setup Burp Suite proxy tool, Which intercepts the HTTP/HTTPS traffic originated from browser to server.



Detailed instruction to configure Burp Suite proxy tool in the browser:

<https://portswigger.net/support/configuring-your-browser-to-work-with-burp>

For all the mentioned reproduction steps in this report, we assume Burp Suite testing environment setup is done.

Apart from Burp Suite, one more tool, Firefox Multi-Account Container was used to easily shift to different accounts.

The detailed instructions to install and use can be found here.

<https://addons.mozilla.org/en-US/firefox/addon/multi-account-containers/>



## 2.7 Vulnerability Chart

The discovered vulnerabilities table and chart illustrated below provides a snapshot view of the number and severity of issues discovered during this security assessment.

**CRITICAL**

This issue can impact the application severely and should be addressed immediately. Attackers can gain root or superuser access or severely impact system operation.

**HIGH**

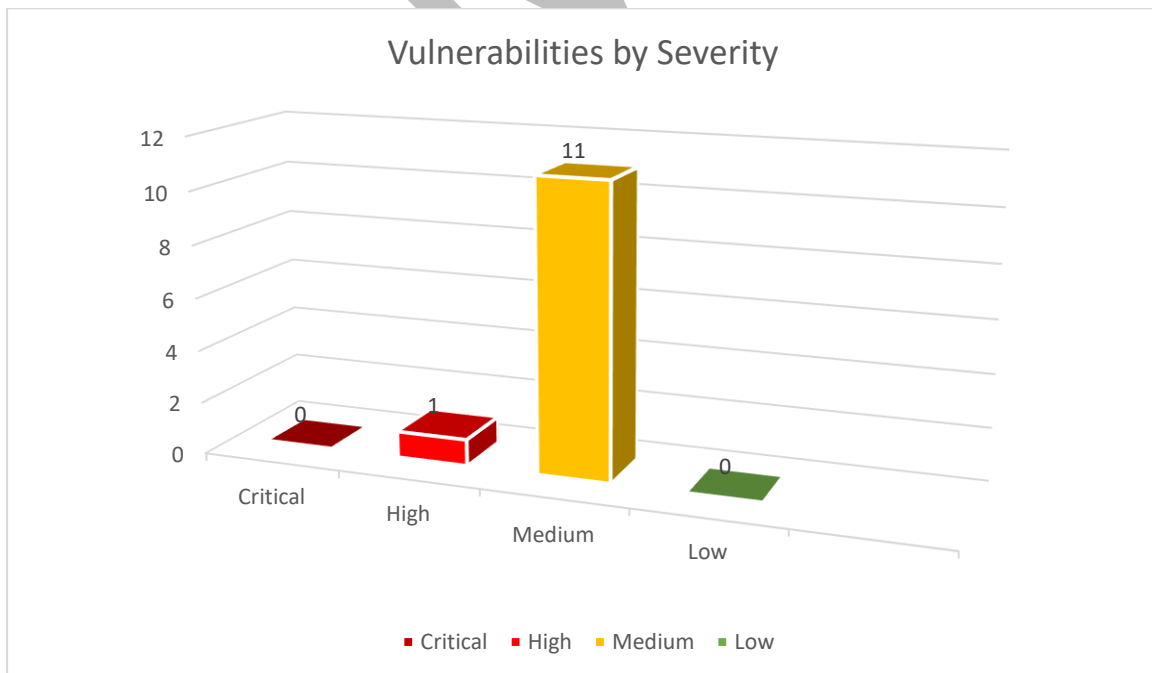
This issue can cause a problem like unprivileged access and should be addressed as soon as possible.

**MEDIUM**

This issue may pose a significant threat over a longer period of time.

**LOW**

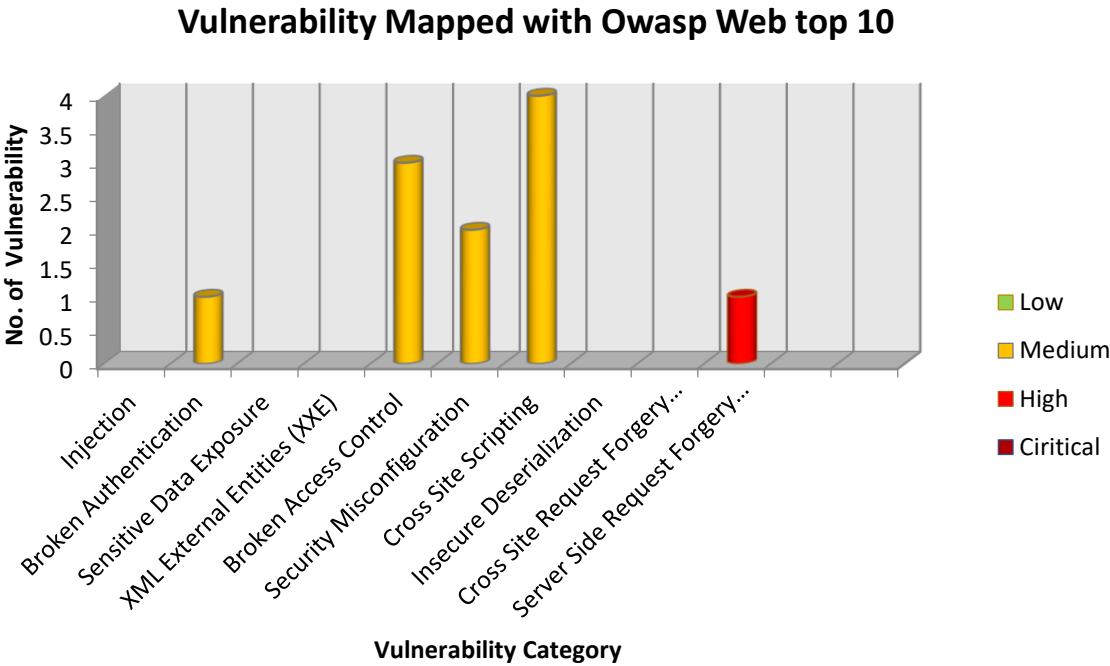
This issue is more likely an information disclosure and may be an acceptable threat.







Below given chart shows the vulnerability matrix based on the category of vulnerabilities.



SA



## 2.9 Table of Findings

Vul ID	Finding	Severity	Status
PY-CL-001	Application vulnerable to SSRF through PDF generator	HIGH	Not Fixed
PY-CL-002	Missing Rate Limit Checks at Log-in Endpoint	MEDIUM	Not Fixed
PY-CL-003	Ability to delete any user's document with a known ID	MEDIUM	Not Fixed
PY-CL-004	Ability to delete any user's Note with a known ID	MEDIUM	Not Fixed
PY-CL-005	Ability to read any user's Note and Document whose ID is known	MEDIUM	Not Fixed
PY-CL-006	Application vulnerable to XSS via a crafted link in Note	MEDIUM	Not Fixed
PY-CL-007	Application vulnerable to XSS via Note's title on {Page Name}	MEDIUM	Not Fixed
PY-CL-008	Application vulnerable to XSS via Annotation's body rendered on {Page Name}	MEDIUM	Not Fixed
PY-CL-009	Application Vulnerable to XSS via crafted uploaded file's name on {Page Name}	MEDIUM	Not Fixed
PY-CL-010	Application vulnerable to WebSocket hijacking	MEDIUM	Not Fixed
PY-CL-011	Security headers missing or set to insecure values.	MEDIUM	Not Fixed
PY-CL-012	GraphQL endpoint returns Error Stack in response	MEDIUM	Not Fixed



## 2.10 WebApplicaition Strengths

During our assessment, we observed the following properties of the application that are well designed and serve towards its strengths:

- ✓ The application implements strong authentication checks at the backend.
- ✓ The application has proper protection for SQL injections vulnerabilities.
- ✓ The application does not leak any sensitive data via logging.
- ✓ During our testing, we did not find any session management issues. The application immediately expires the token after logout.
- ✓ The application does not expose any critical information like the current password through any interface.
- ✓ The application uses HTTPS for all communication.
- ✓ The application's server infrastructure uses a role with the least permissions wherever needed.
- ✓ The application's authentication mechanism provides great protection against authenticated CSRF attacks.



## 2.11 WebApplication Weaknesses

---

- ▶ The application does not properly implement Access Control Mechanism.
- ▶ The application does not properly sanitize user input.
- ▶ The application does not properly implement rate-limiting.
- ▶ The application does not properly configure few header values.

**The vulnerabilities below were identified and verified by Payatu during the process of this web Application Penetration Test for the client. Retesting should be planned to follow the remediation of these vulnerabilities.**

SAMPLE

## 3. Technical Findings

### 3.1 PY-CL-001: Application vulnerable to SSRF through PDF generator

**Potential Impact:** HIGH

**Description:**

During the security assessment of the application, we found that the application doesn't properly sanitize a few input fields, which are leading to XSS. This XSS can be leveraged to get SSRF through PDF generation.

**Affected Resources**

- {Client Name} web Application
  - {URL}

**CVSS Score:**

CVSS Base Score: 7.1  
CVSS Temporal Score: 6.4  
CVSS Environmental Score: 7.7

**CVSS Vector:**

CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:  
H/I:L/A:N/E:P/RL:O/RC:C/CR:H/IR:H/  
AR:L/MAV:N/MAC:L/MPR:L/MUI:N/MS  
:U/MC:H/MI:L/MA:N

**Business Risk:** Loss of intellectual property (source code)  
Loss of customer data (RDS database credentials compromise)

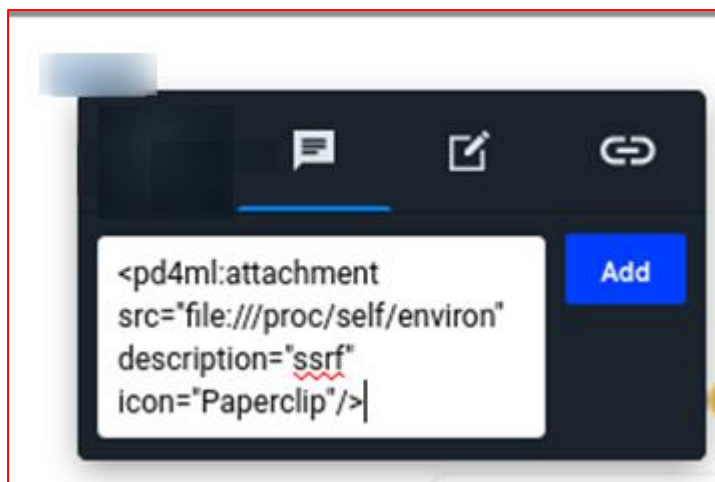
**Technical Risk:** Using this vulnerability, an attacker can make HTTP requests and download server files, including the source code, AWS token, and credentials and environment files.

**Remediation:**

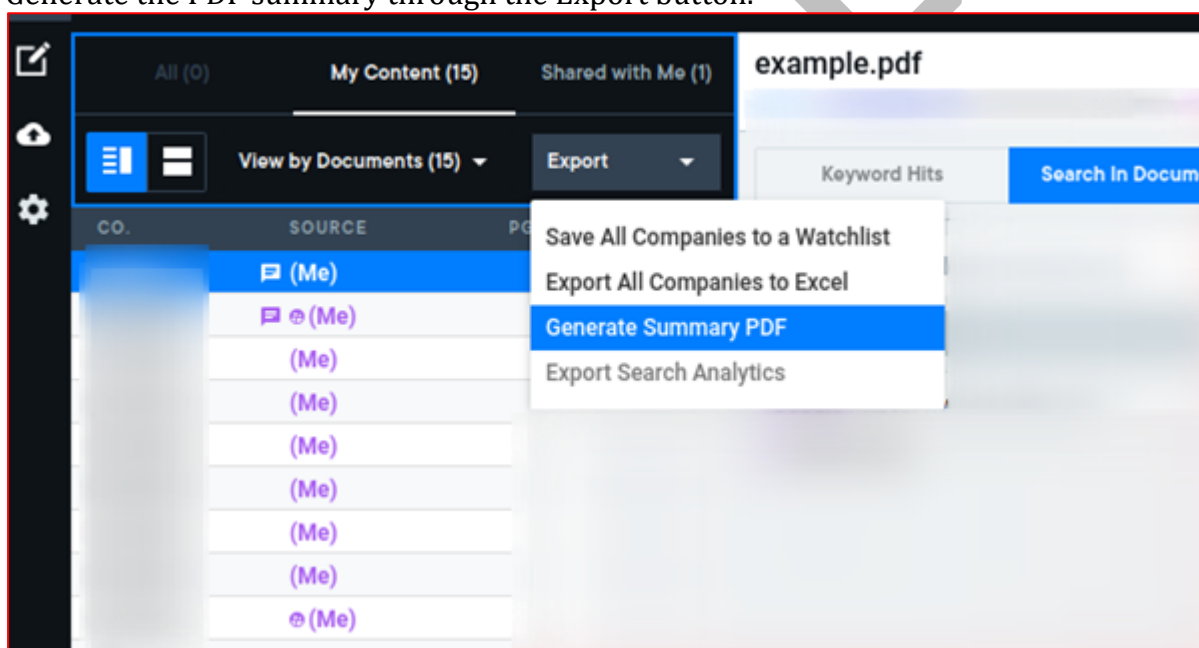
- Escape user input by removing any HTML tags.
- Isolate server-side worker generating PDF from rest of the infrastructure
- Allow only a set of specific harmless HTML tags in the document, which will be converted to PDF.

**Steps to Reproduce:**

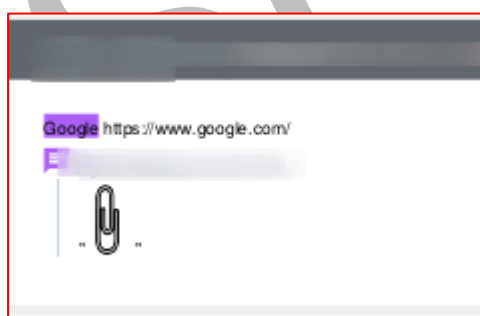
1. Login as any user.
2. Go to any document
3. Create an annotation.
4. Use the following payload in the body of the annotation.  
`<pd4ml:attachment src="file:///proc/self/environ" description="ssrf" icon="Paperclip"/>`



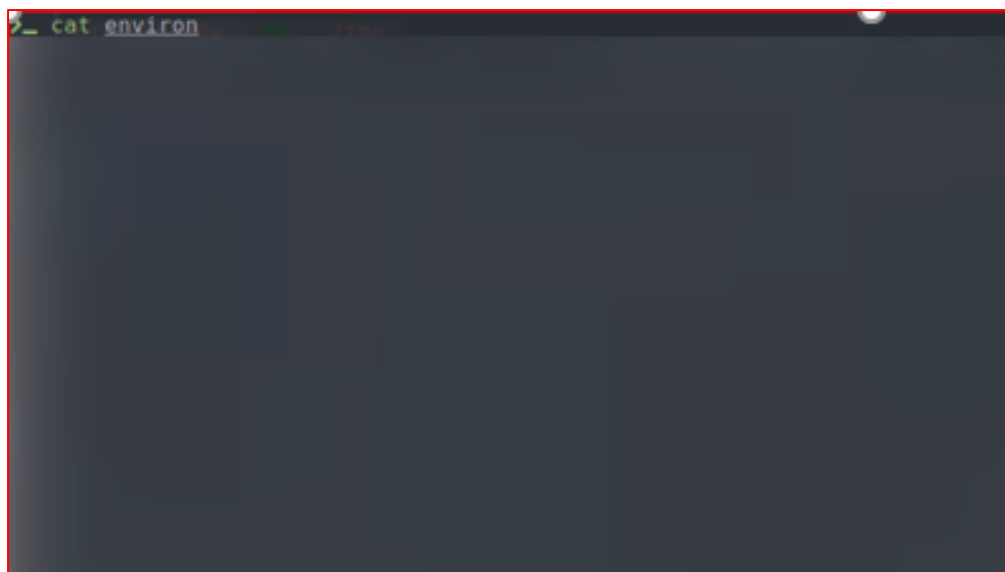
5. Generate the PDF summary through the Export button.



6. Open the downloaded PDF file.
7. Right-click on the paper clip icon, as shown in the image.



8. Click the 'Save Attachment As..' option. Proceed to save the attachment.
9. View the saved file, and you can read the server's process environment file.



### Proof of Concept Exploit

To demonstrate the impact of this vulnerability, we have developed a proof of concept exploit code. The below attached python script can download the source code, and fetch AWS token, which further leads to AWS credentials compromise.

The process of extracting attachment might differ in different tools/OS. You can use the attached python script to extract/view the server file.

### Attached Script (as an object)



### How to use Script?

#### ## Install Dependencies

```
$ pip3 install PyPDF2 requests
```

#### ## Download Server's source code

```
$ python3 extract.py 'file:///proc/self/cwd/webapps/ui.war' --save 'ui.war'
```

#### ## Fetch AWS Access token

```
$ python3 extract.py 'http://169.254.169.254/latest/meta-data/iam/security-credentials/{URL}'
```

Using the above exploit, we were able to retrieve **AWS credentials** from the server's source code. The file is attached for easier access.



Below is the snippet from the same.

```
# Database settings for the Ticker API.
database.
database.
database.
database.
database.

#####
# Quant feed URL
quant.feed.backtesting.service.url=

#####
# Email sending
smtp.
smtp.
smtp.
smtp.
```



## 3.2 PY-CL-002: Missing Rate Limit Checks at Login Endpoint

**Potential Impact:** MEDIUM

**Description:**

It has been observed during the assessment that the application doesn't properly implement rate-limiting on the log-in endpoint. This allows an attacker to brute-force the password for a known email address.

---

**Affected Resource**

- {Client Name} Web Application
  - {URL}

---

**CVSS Score:**

CVSS Base Score: 5.3  
CVSS Temporal Score: 4.8  
CVSS Environmental Score: 6.8

**CVSS Vector:**

CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:  
L/I:N/A:N/E:P/RL:T/RC:C/CR:H/IR:H/  
AR:L/MAV:N/MAC:L/MPR:N/MUI:N/MS  
:U/MC:L/MI:L/MA:N

---

**Business Risk:** Successful exploitation of this vulnerability allows an attacker to successfully take over a user account.

---

**Technical Risk:** An attacker can brute force the user's credentials.

---

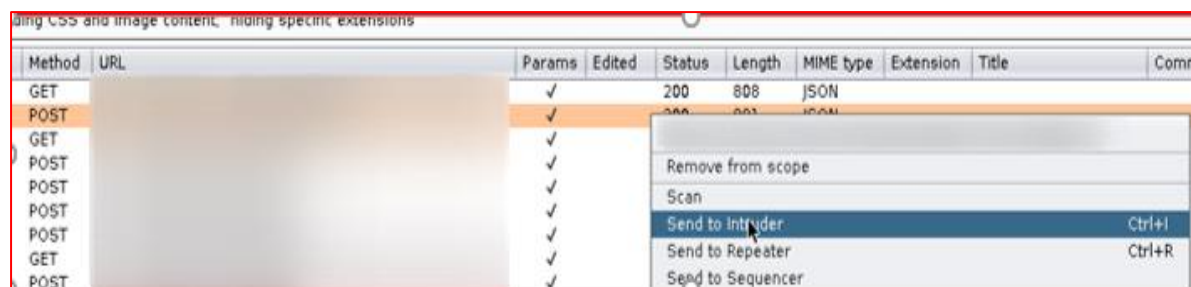
**Remediation:**

- Add limit on the number of times an unsuccessful attempt can be made for an account within a specific time.
- Add a captcha to prevent against automated brute force attack. Use industry-standard captcha like ReCaptcha.

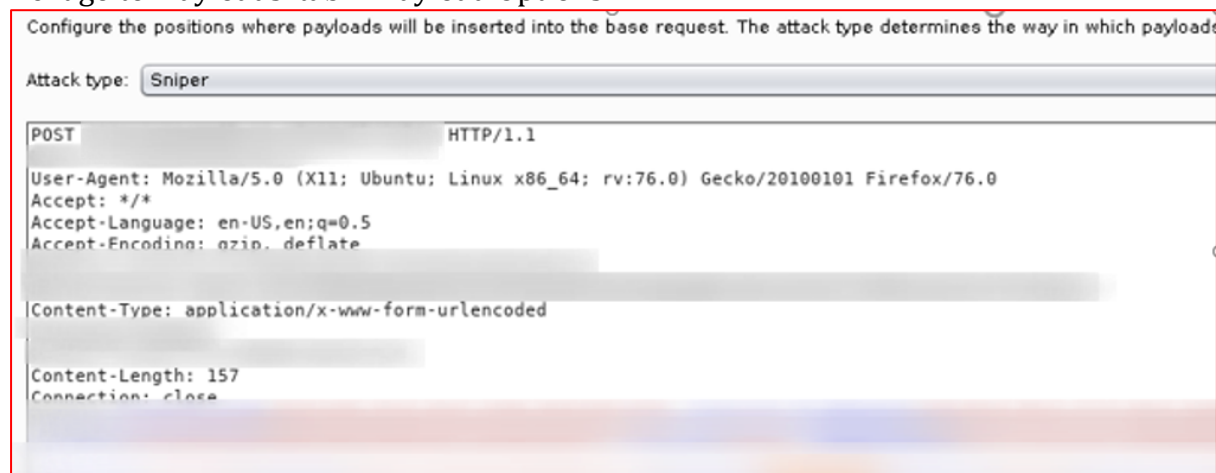
---

**Steps to Reproduce:**

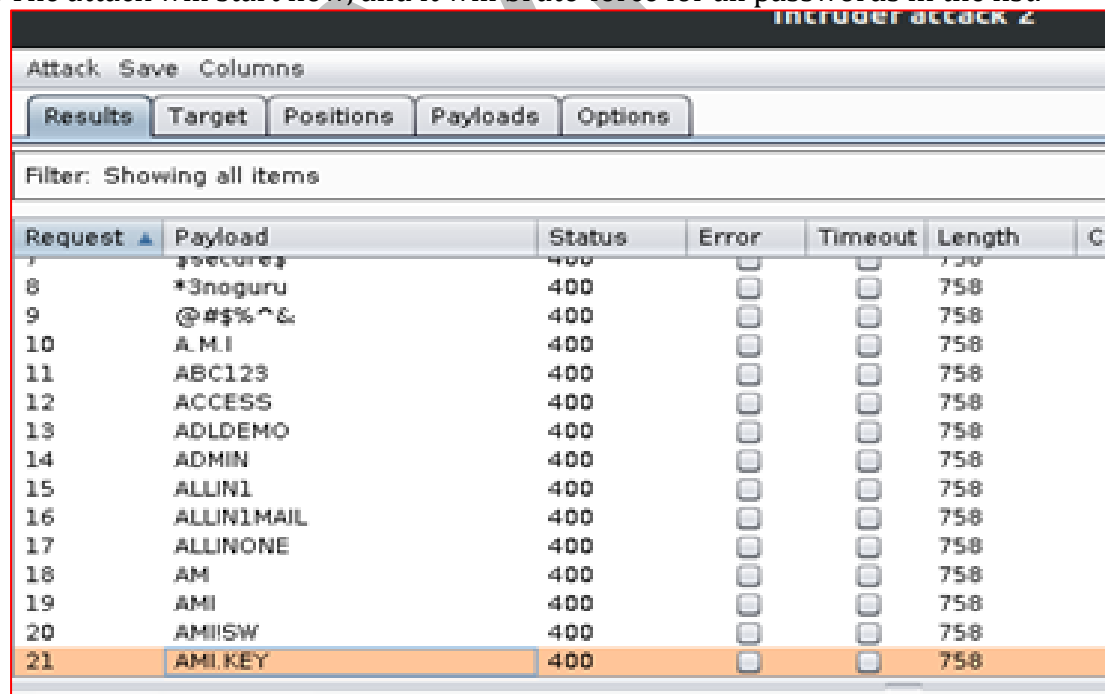
1. Go to {URL}
2. Enter the username and password.
3. Now go to Burp Suite > Proxy > HTTP History.
4. Select the request with endpoint '{URL}'.
5. Send the request to Intruder. Check the image for reference.
6. Go to the Intruder > Position tab. Refer to the image.



7. Click on 'Clear \$', then select the password as highlighted and click 'Add \$'.
8. Next go to 'Payloads' tab > 'Payload Options'.



9. Expand the 'Add from list...' and select 'passwords'.
10. Now click the 'Start Attack' on top-right corner.
11. The attack will start now, and it will brute-force for all passwords in the list.



12. It can be seen that there is not rate-limiting by analyzing the server response. The server checks for the validity of the password and returns the 400 or 200 status accordingly.

### 3.3 PY-CL-003: Ability to delete any user's document with a known ID

**Potential Impact:** MEDIUM

#### Description:

During the security assessment of the application, we found that the application allows a user to delete any document if the ID of that document is known.

#### Affected Resource

- {Client Name} Web Application
  - {URL}

#### CVSS Score:

CVSS Base Score: 4.3  
 CVSS Temporal Score: 3.9  
 CVSS Environmental Score: 3.5

#### CVSS Vector:

CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:  
 L/A:N/E:P/RL:T/RC:C/CR:H/IR:H/AR:M/M  
 AV:N/MAC:H/MPR:L/MUI:N/MS:U/MC:N/  
 MI:L/MA:N

**Business Risk:** This can lead to loss of sensitive and important documents.

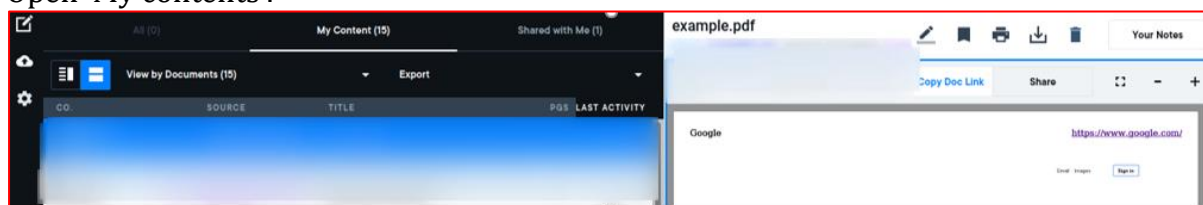
**Technical Risk:** This allows an attacker to delete any document whose ID is known to the attacker. The victim will lose access to the document.

#### Remediation:

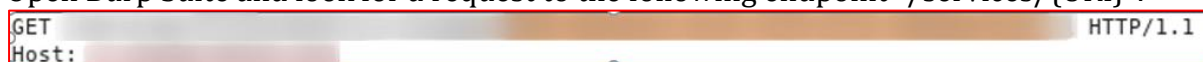
- Implement proper authorization checks. Ensure that the user owns the specific document before performing delete action.

#### Steps to Reproduce:

1. First, we need to get the document ID, which we want to delete.
2. Open 'My contents'.



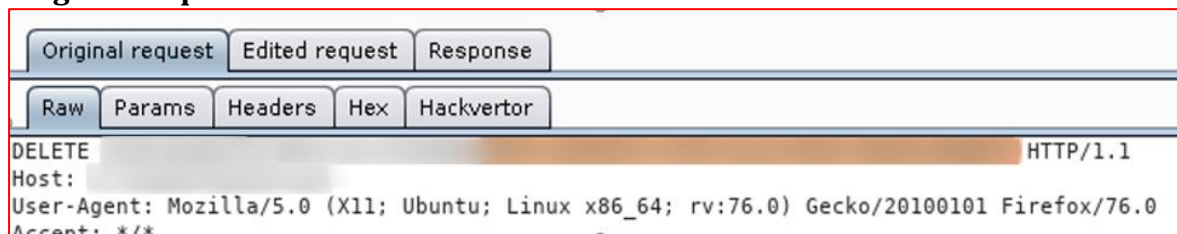
3. Click on the document which needs to be deleted.
4. Open Burp Suite and look for a request to the following endpoint “/services/{URI}”.



5. Here the document ID is '{Document-ID}'.

6. Now, we start the attack—log in as any different user.
7. Turn on intercept in Burp Suite.
8. Click the delete icon of any document (other than Note).
9. Intercept the request to '{URL}'
10. Modify the request params by replacing previously noted ID. Refer to the image.

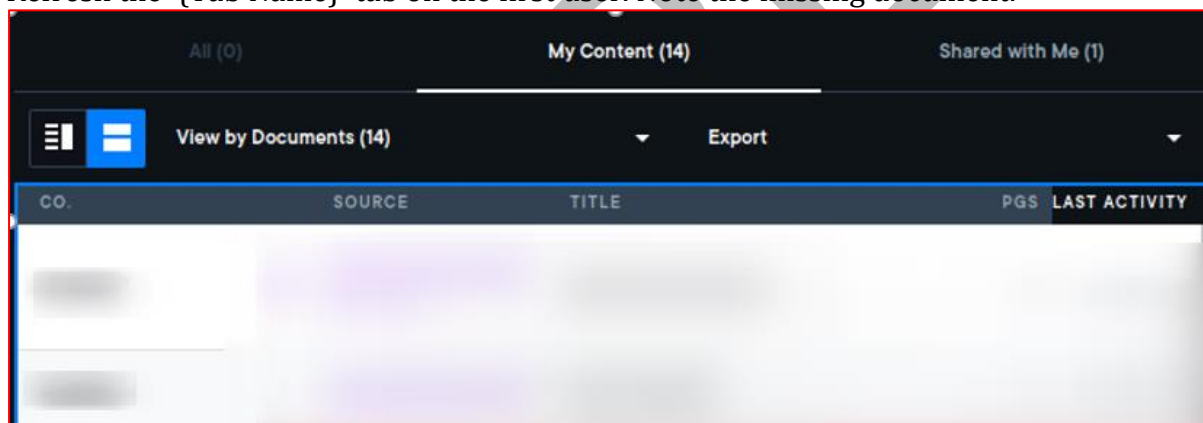
#### Original Request



#### Edited Request



11. Refresh the '{Tab Name}' tab on the first user. Note the missing document.



### 3.4 PY-CL-004: Ability to delete any user's Note with a known ID

**Potential Impact:** MEDIUM

**Description:**

During the security assessment of the application, we found that the application allows a user to delete any note if the ID of that Note is known.

**Affected Resources**

- {Client Name} web Application
  - {URL}

**CVSS Score:**

CVSS Base Score: 4.3  
CVSS Temporal Score: 3.9  
CVSS Environmental Score: 3.5

**CVSS Vector:**

CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:  
N/I:L/A:N/E:P/RL:T/RC:C/CR:H/IR:H/  
AR:M/MAV:N/MAC:H/MPR:L/MUI:N/M  
S:U/MC:N/MI:L/MA:N

**Business Risk:** This can lead to users losing sensitive and important notes.

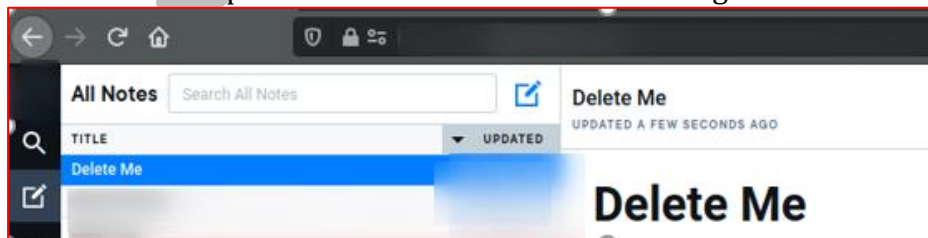
**Technical Risk:** This allows an attacker to delete any note whose ID is known to the attacker. The victim will lose access to the Note. The Note will just keep loading on the browser.

**Remediation:**

- Implement proper authorization checks. Ensure that the user owns the specific Note before performing delete action.

**Steps to Reproduce:**

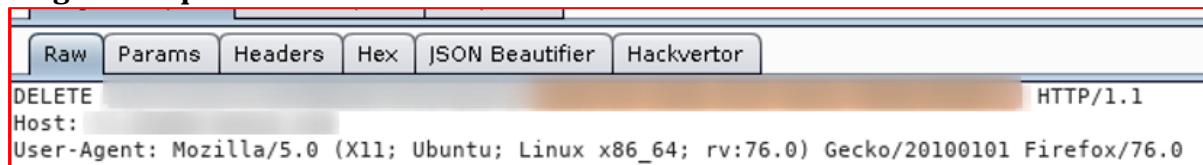
1. First, we need to get the note ID, which we want to delete.
2. Open the notes tab.
3. Click on the Note, which needs to be deleted.
4. The Note's ID is present in the URL. Refer to the image.



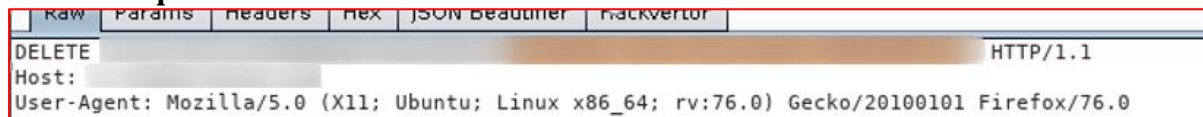
5. Here the note ID is '{Note ID}'.
6. Now, we start the attack—log in as any different user.
7. Turn on intercept in Burp.
8. Click the delete icon of any note.

9. Intercept the request to '{URL}'.
10. Modify the request params by replacing previously noted ID in URL and Body. Refer to the image.

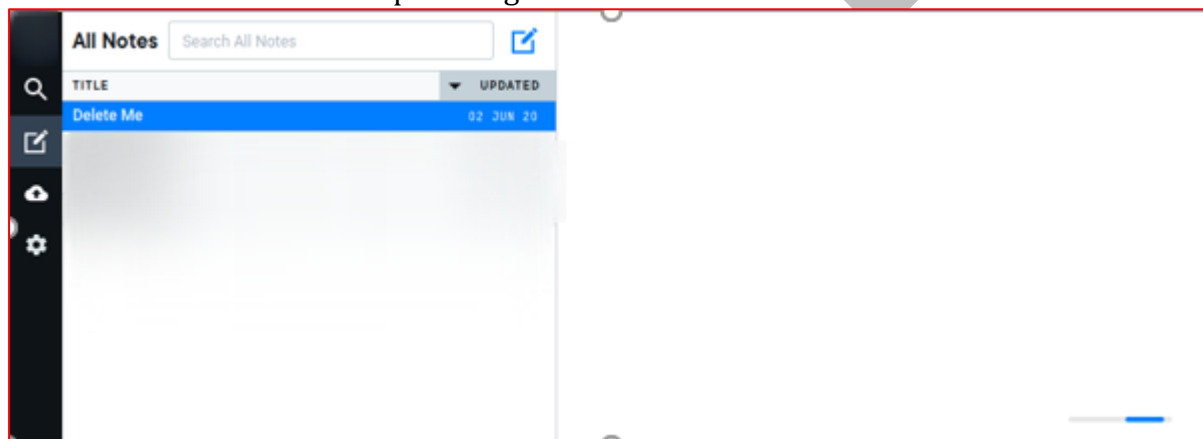
### Original Request



### Edited Request

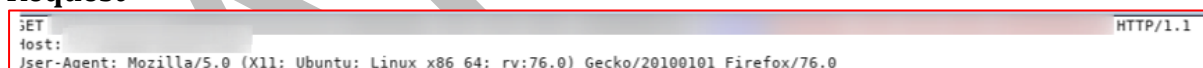


11. Refresh the '{Tab Name}' tab on the first user.
12. Click on the Note. It will keep loading the contents.

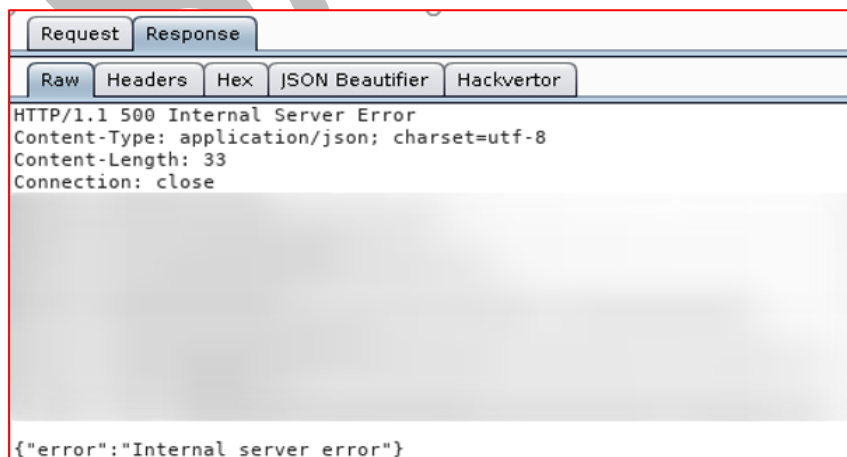


13. On checking the Burp Suite. It can be noted that the server returns 500.

### Request



### Response



### 3.5 PY-CL-005: Ability to read any user's Note and Document whose ID is known

**Potential Impact:** MEDIUM

**Description:**

While performing pen testing, it was observed that all uploaded documents are saved using the same API call, which accepts temporarily generated document ID. This call then saves the provided documents to the user's content. Additionally, it was observed that there were no proper checks in which IDs can be provided through the '{Function Name}' call.

**Affected Resource**

- {Client Name} web Application
  - {URL}

**CVSS Score:**

CVSS Base Score: 5.4  
CVSS Temporal Score: 3.9  
CVSS Environmental Score: 3.5

**CVSS Vector:**

CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:  
L/I:L/A:N/E:P/RL:T/RC:C/CR:H/IR:H/A  
R:M/MAV:N/MAC:H/MPR:L/MUI:N/MS:  
U/MC:N/MI:L/MA:N

**Business Risk:** An attacker can leverage this attack to get access to any user's documents or notes.

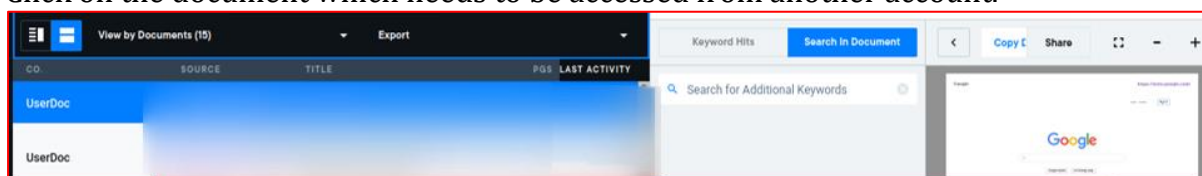
**Technical Risk:** An attacker can use this vulnerability to get read access (also, delete access as discussed in PY-CL-003) to any document whose ID is known to the attacker.

**Remediation:**

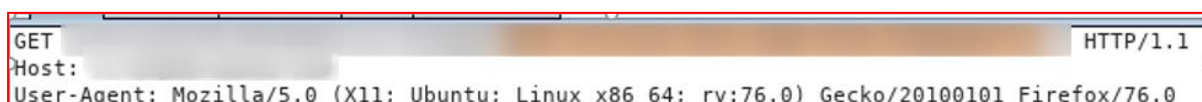
Employ proper authorization checks. Ensure that the document ID used is not claimed by any user, or the ID belongs to a temporary user.

**Steps to Reproduce:**

1. First, we need to get the document ID, which we want to delete.
2. Open '{Tab Name}'.
3. Click on the document which needs to be accessed from another account.



4. Open Burp Suite and look for a request to the following endpoint "{URL}".



5. Here the document ID is '{Document ID}'.
6. Now, start the attack. Log in as any other user.
7. Go to upload document options.
8. Upload a document. Don't click the 'Save' button yet.
9. Turn on Intercept in Burp Suite.
10. Click the 'Save' button.
11. Intercept the request to '{URL}'.
12. Replace the document ID with previously noted ID in the URL and body.

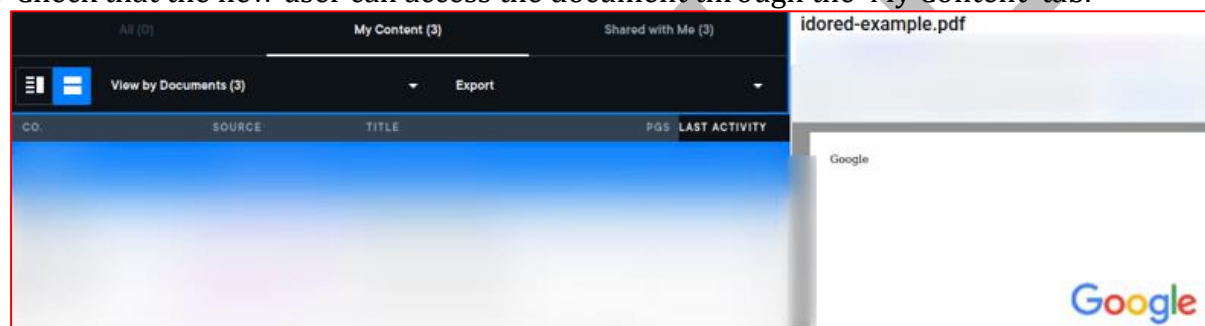
#### Original Request

```
s":[{"id": "example.pdf"}, {"originalFileName": "example.pdf"}], "cust
```

#### Edited Request

```
s":[{"id": "idored-example.pdf"}, {"originalFileName": "idored-example.pdf"}], "c
```

13. Forward the request.
14. Check that the new user can access the document through the 'My Content' tab.



#### Note:

- In case of attacker and victim are in the same team, then the document disappears from the victim's list.
- In case the attacker and victim are from different teams, then they both can access the document.



### 3.6 PY-CL-006: Application vulnerable to XSS via a crafted link in Note

**Potential Impact:** MEDIUM

**Description:**

During the security assessment of the application, we found that the application allows a user to add links in the notes. This link can point to JavaScript code, which will be executed when a user clicks on the link.

**Affected Resources**

- {Client Name} web Application
  - {URL}

**CVSS Score:**

CVSS Base Score: 5.4  
CVSS Temporal Score: 4.9  
CVSS Environmental Score: 8.2

**CVSS Vector:**

CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:  
L/I:L/A:N/E:P/RL:O/RC:C/CR:H/IR:H/A  
R:L/MAV:N/MAC:L/MPR:L/MUI:R/MS:C  
/MC:H/MI:L/MA:N

**Business Risk:** An attacker can take over a victim's account using this vulnerability.

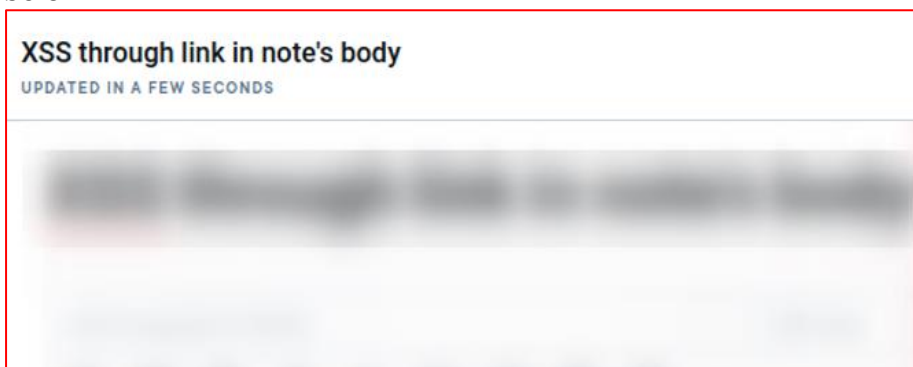
**Technical Risk:** Using this vulnerability, an attacker can get access to the victim's cookies, which contains the OAuth tokens. This can give the attacker full access to the victim's account.

**Remediation:**

- Ensure that the link has a valid schema, like http:// or https://

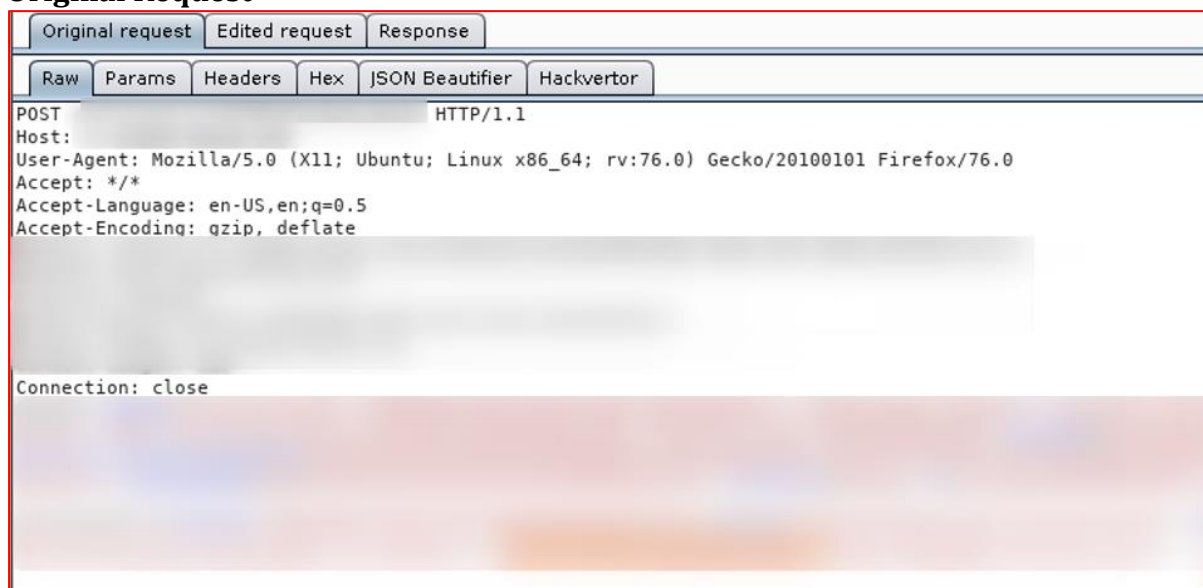
**Steps to Reproduce:**

1. Log in as any user
2. Create a new note. Give a random title and body content.
3. Turn on 'Intercept' in Burp Suite.
4. Now paste some well-formatted link in the body of the Note. Refer to the image below.

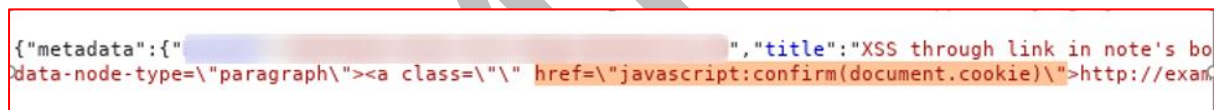


- Intercept the request to '{URL}' endpoint.
- Change the link in the request's body. Refer to the image below.

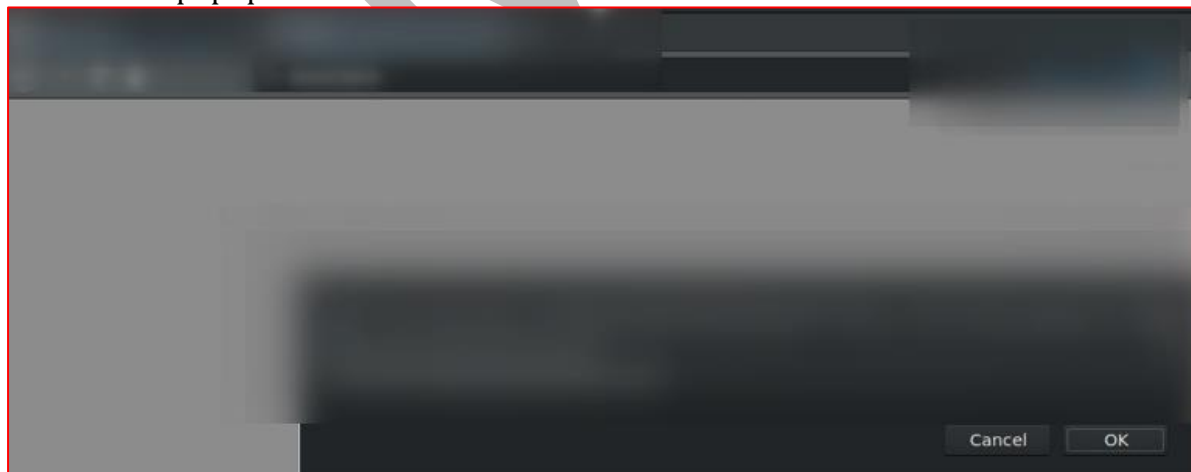
### Original Request



### Edited Request



- Now refresh the page on the browser. Click the link.
- Observe the popup in the new tab.



### 3.7 PY-CL-007: Application vulnerable to XSS via Note's title on {Page Name}

**Potential Impact:** MEDIUM

**Description:**

During the security assessment of the application, we found that the application doesn't properly sanitize annotation's title when displaying in '{Tab Name}' tab, leading to XSS. The application renders user's input (note's title) via 'dangerouslySetInnerHTML' property causing XSS.

**Affected Resources**

- {Client Name} web Application
  - {URL}

**CVSS Score:**

CVSS Base Score: 5.4  
CVSS Temporal Score: 4.9  
CVSS Environmental Score: 8.2

**CVSS Vector:**

CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:  
L/I:L/A:N/E:P/RL:O/RC:C/CR:H/IR:H/A  
R:L/MAV:N/MAC:L/MPR:L/MUI:R/MS:C  
/MC:H/MI:L/MA:N

**Business Risk:** An attacker can take over a victim's account using this vulnerability.

**Technical Risk:** Using this vulnerability, an attacker can get access to the victim's cookies, which contains the OAuth tokens. This can give the attacker full access to the victim's account.

**Remediation:**

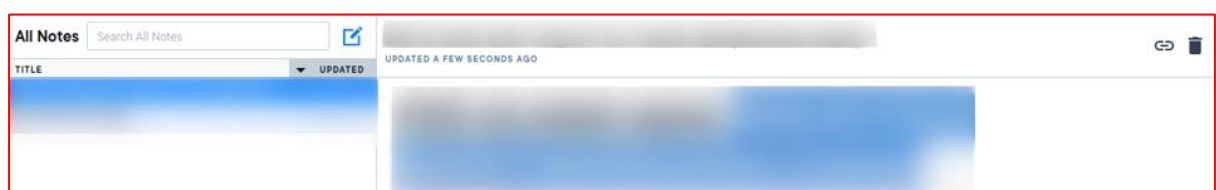
Avoid using 'dangerouslySetInnerHTML' property in React.JS

Ref: <https://reactjs.org/docs/dom-elements.html#dangerouslysetinnerhtml>

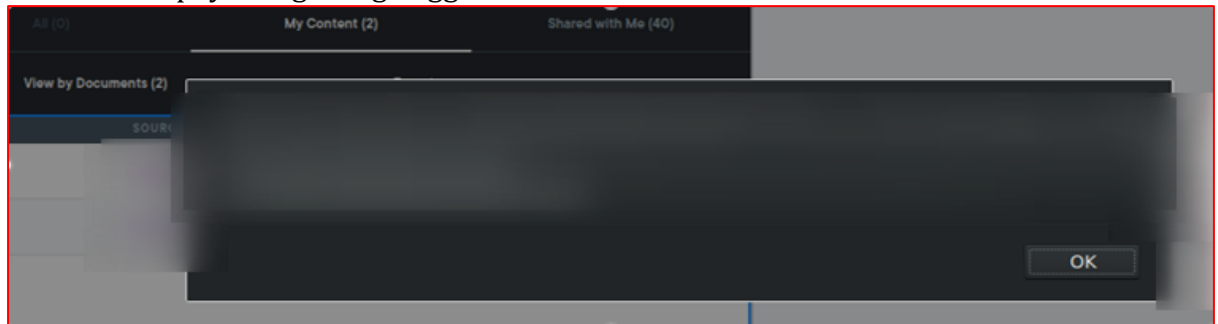
**Steps to Reproduce:**

1. Log in as any user.
2. Create a new note.
3. Add XSS payload in Note's title. Refer to the image below.

**Payload:** <img src='//s.s' onerror=alert(document.cookie)>



4. Wait until the Note is saved.
5. Open the '{Tab Name}' tab.
6. Observe the payload getting triggered.



### 3.8 PY-CL-008: Application vulnerable to XSS via Annotation's body rendered on {Page Name}

**Potential Impact:** MEDIUM

**Description:**

During the security assessment of the application, we found that the application doesn't properly sanitize annotation's body, while rendering on the side pane, leading to XSS.

**Affected Resources**

- {Client Name} web Application
  - {URL}

**CVSS Score:**

CVSS Base Score: 5.4  
CVSS Temporal Score: 4.9  
CVSS Environmental Score: 8.2

**CVSS Vector:**

CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:  
L/I:L/A:N/E:P/RL:O/RC:C/CR:H/IR:H/A  
R:L/MAV:N/MAC:L/MPR:L/MUI:R/MS:C  
/MC:H/MI:L/MA:N

**Business Risk:** An attacker can take over a victim's account using this vulnerability.

**Technical Risk:** Using this vulnerability, an attacker can get access to the victim's cookies, which contains the OAuth tokens. This can give the attacker full access to the victim's account.

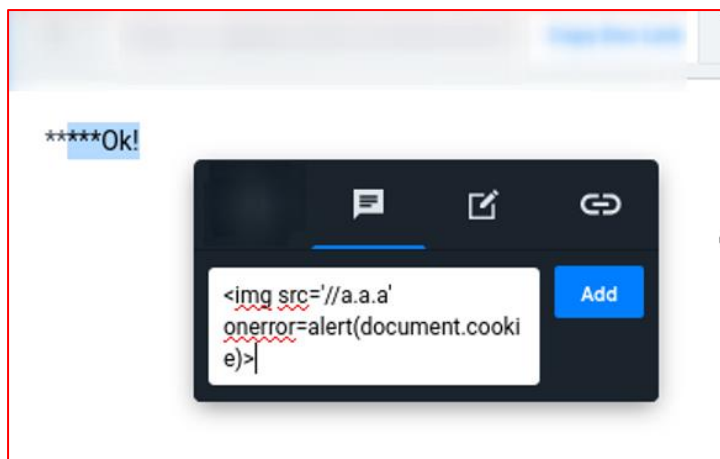
**Remediation:**

- Escape HTML tags from user input.
- Avoid using 'dangerouslySetInnerHTML' property in React.JS  
Ref: <https://reactjs.org/docs/dom-elements.html#dangerouslysetinnerhtml>

**Steps to Reproduce:**

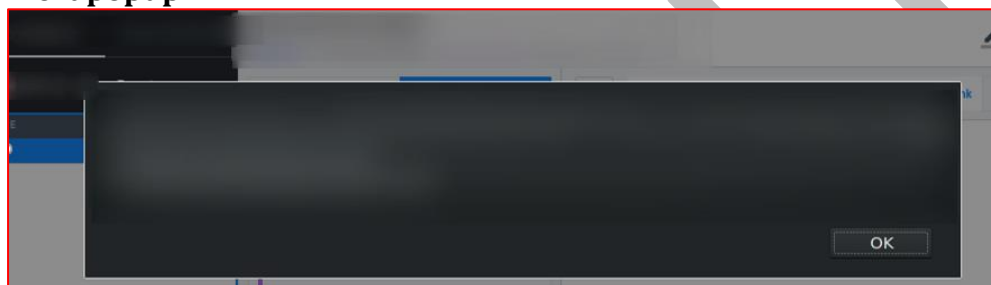
1. Log in as any user.
2. Open any document.
3. Create an annotation.
4. Use XSS Payload in the annotation's body. Refer to the image below.

**Payload:** <img src='//a.a.a/' onerror=alert(document.cookie)>'

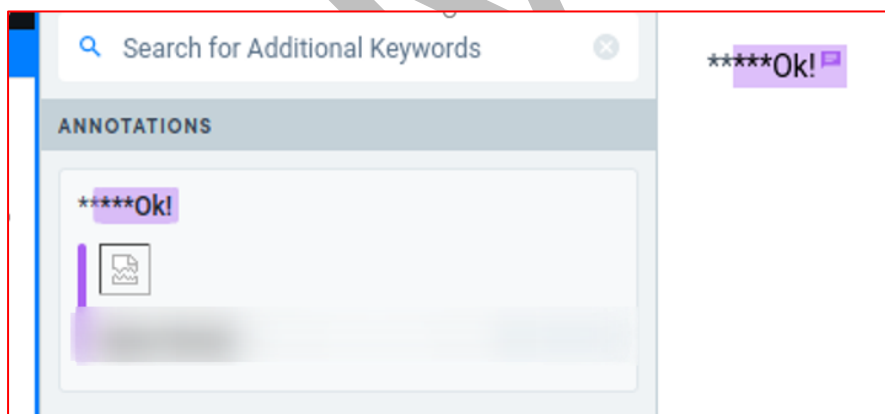


5. Wait for some time and observe the pop-up. Refer to the image below.

### Alert popup



### Broken Image:



### 3.9 PY-CL-009: Application Vulnerable to XSS via crafted uploaded file's name on {Page Name}

**Potential Impact:** MEDIUM

**Description:**

During the security assessment of the application, we found that the application doesn't properly sanitize file names. The application renders the file name using React.JS's 'dangerouslySetInnerHTML' property, leading to XSS.

**Affected Resources**

- {Client Name} web Application
  - {URL-1}
  - {URL-2}

**CVSS Score:**

CVSS Base Score: 5.4  
CVSS Temporal Score: 4.9  
CVSS Environmental Score: 8.2

**CVSS Vector:**

CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:  
L/I:L/A:N/E:P/RL:O/RC:C/CR:H/IR:H/A  
R:L/MAV:N/MAC:L/MPR:L/MUI:R/MS:C  
/MC:H/MI:L/MA:N

**Business Risk:** An attacker can take over a victim's account using this vulnerability.

**Technical Risk:** Using this vulnerability, an attacker can get access to the victim's cookies, which contains the OAuth tokens. This can give the attacker full access to the victim's account.

**Remediation:**

Avoid using 'dangerouslySetInnerHTML' property in React.JS

Ref: <https://reactjs.org/docs/dom-elements.html#dangerouslysetinnerhtml>

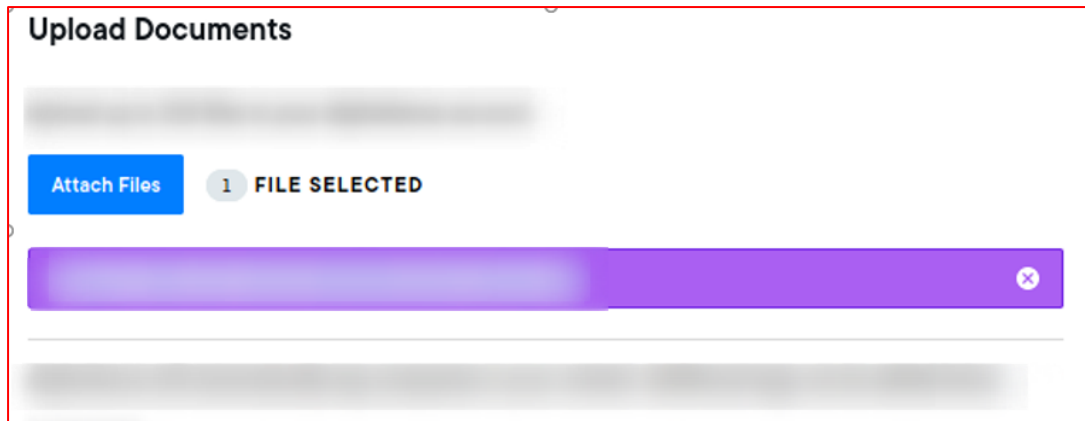
**Steps to Reproduce:**

1. Login as any user.
2. Create pdf on your machine with the following name (any name will do which has XSS payload)

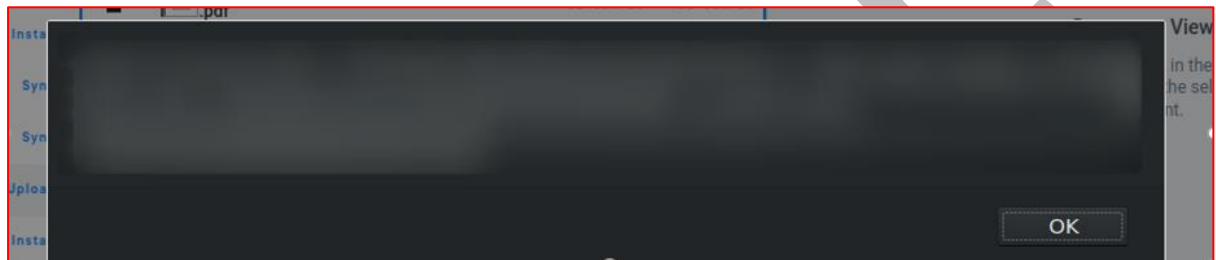
**XSS Payload in the file name:**

I<img src='abc.com' onerror=alert(document.cookie)>.pdf

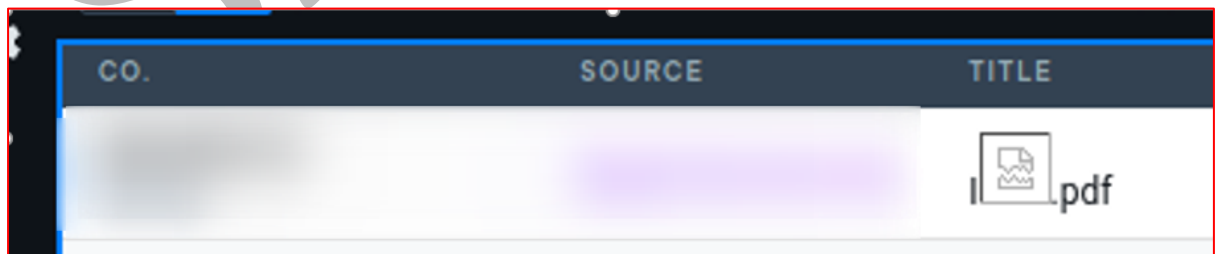
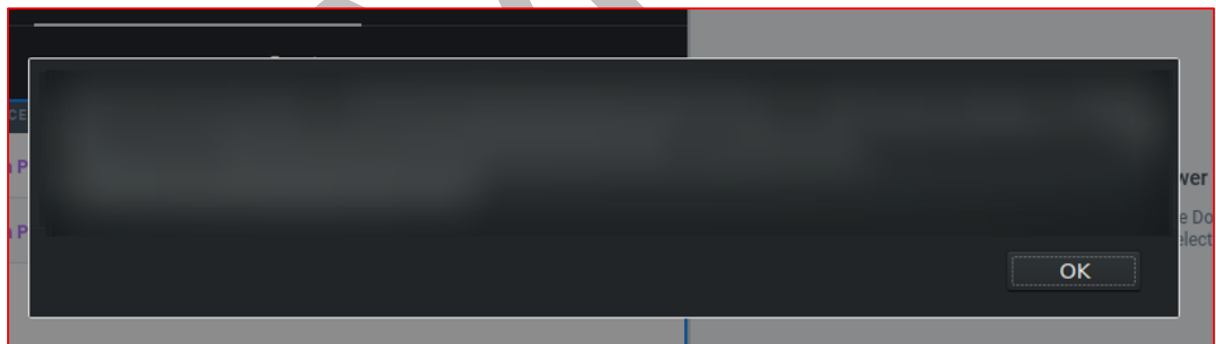
3. Go to 'Upload Document'.
4. Upload the previously renamed document and click 'Save'.



- Now, go to {Tab Name} > {Tab Name} tab. Observe the popup. Refer to the image below.



- Now, go to '{Tab Name}'. Observe the popup. Refer to the images below.





## 3.10 PY-CL-010: Application vulnerable to WebSocket hijacking

**Potential Impact:** MEDIUM

### Description:

During the security assessment of the application, we found that the application uses WebSocket to send changes made in a note as updates. The WebSocket integration here doesn't validate the 'Origin' header.

### Affected Resources

- {Client Name} web Application
  - {URL-1}
  - {URL-2}

### CVSS Score:

CVSS Base Score: 4.3  
 CVSS Temporal Score: 4.1  
 CVSS Environmental Score: 3.7

### CVSS Vector:

CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:N/E:H/RL:O/RC:C/CR:H/IR:H/AR:M/MAV:N/MAC:H/MPR:N/MUI:R/M S:U/MC:N/MI:L/MA:N

**Business Risk:** N/A

**Technical Risk:** Upon successful exploitation of this vulnerability, an attacker can send updated over WebSocket from any domain.

### Remediation:

- Check 'Origin' header to be from a trusted/allowed domain.

### Steps to Reproduce:

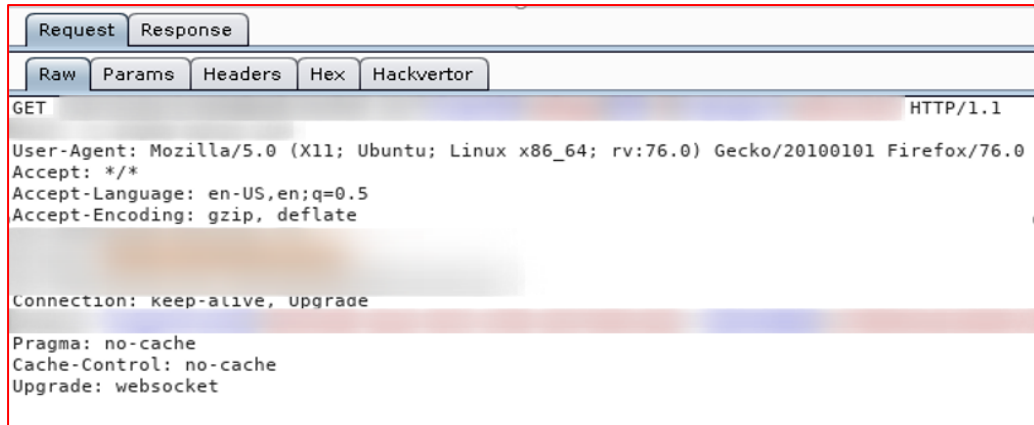
1. Go to any domain other than '{Domain Name}'.
2. Open a WebSocket to '{URL}' using the console.

JavaScript Code:

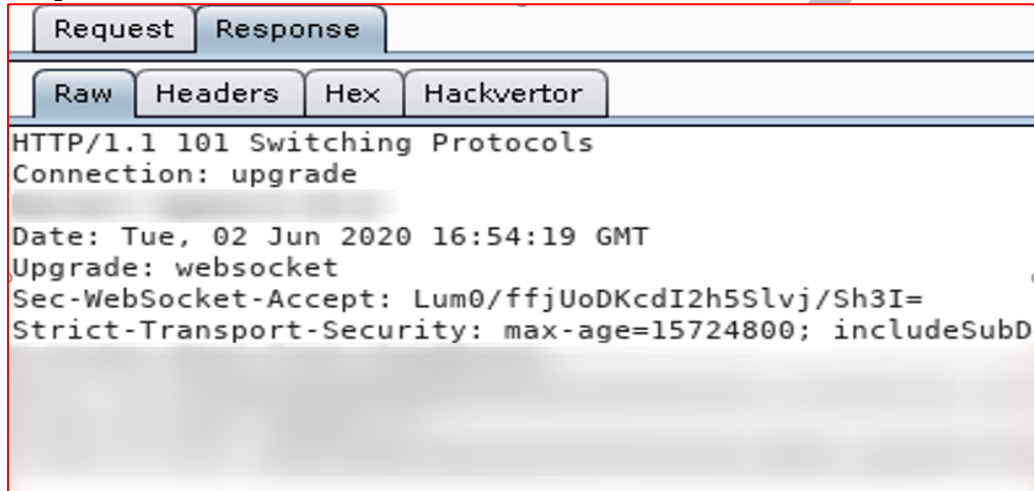
```
new WebSocket(<url>);
```

3. Observe that the connection is established successfully, and messages are being transferred.

### Request:



### Response:



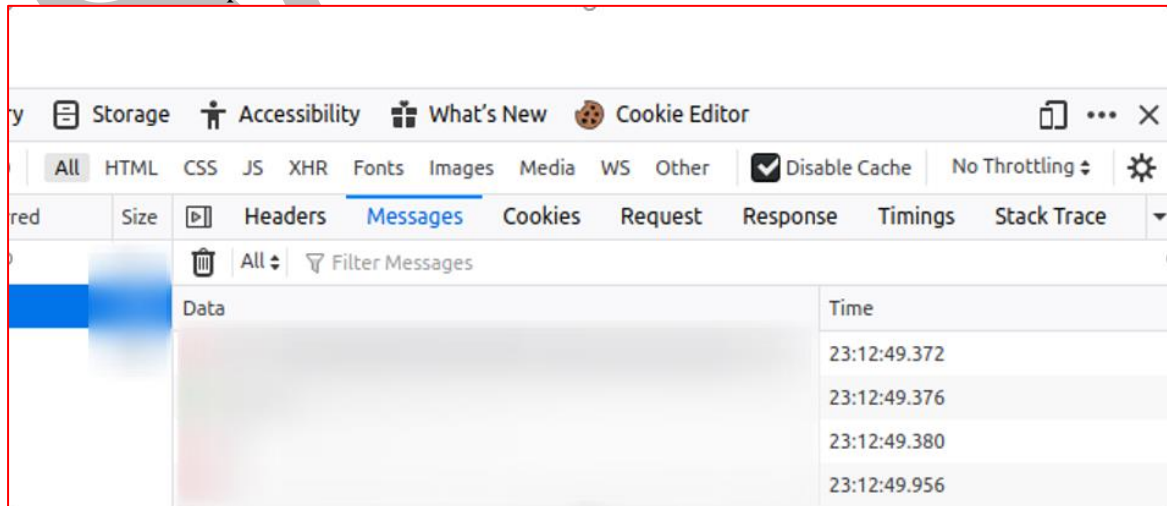
### Proof of Concept Exploit

- Upload the following file at any location from where it can be served.



websocket.html

- Open the HTML file in the browser.
- Observe the script has established communication with WebSocket.



## 3.11 PY-CL-011: Security headers missing or set to insecure values

**Potential Impact:** MEDIUM

### Description:

During the security assessment of the application, we found that the application is missing standard headers, or if present, their values are set to insecure values.

Misconfigured Header:

- Cookie
- Access-Control-Allow-Origin

Missing Headers:

- X-Frame-Options

### Affected Resources

- {Client Name} web Application
  - {URL-1}
  - {URL-2}

### CVSS Score:

CVSS Base Score: 4.3

CVSS Temporal Score: 4.1

CVSS Environmental Score: 3.7

### CVSS Vector:

CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:  
L/I:N/A:N/E:H/RL:O/RC:C/CR:H/IR:H/  
AR:M/MAV:N/MAC:H/MPR:N/MUI:R/M  
S:U/MC:L/MI:N/MA:N

**Business Risk:** Successful exploitation of this vulnerability can lead to the attackers getting access to sensitive user data.

**Technical Risk:** An attacker can use this vulnerability for a variety of attacks like clickjacking, or can exploit misconfigured CORS to read sensitive data of the user.

### Remediation:

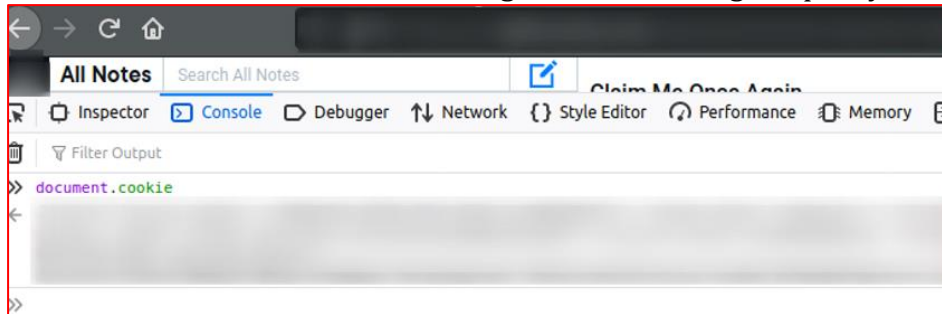
- Set 'X-Frame-Options' on REST API endpoints and content which should not be frameable
- Set cookies with 'HttpOnly' and 'SameSite' option.
- Set 'Access-Control-Allow-Origin' only where necessary and to specific domain values rather than a wildcard.

### Steps to Reproduce:

#### 1. Cookie

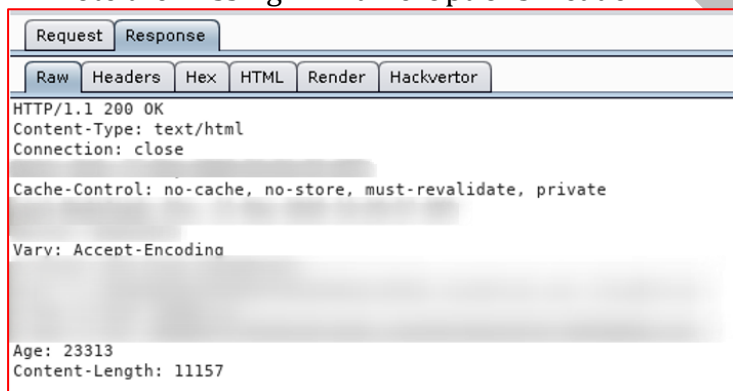
- Go to {URL}
- Open developer console
- Enter 'document.cookie'

- Observe that all the cookies are being shown. Meaning 'HttpOnly' is not set.



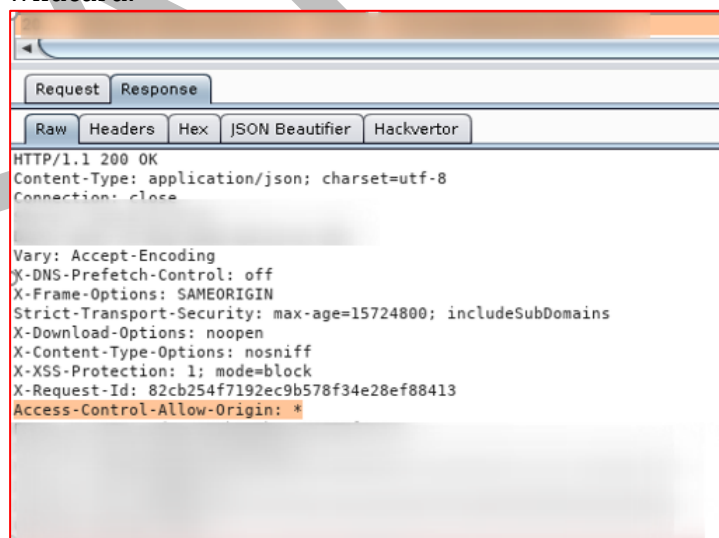
## 2. X-Frame-Options

- Request to {URL}
- Observer the headers in Burp Suite.
- Note the missing 'X-Frame-Options' header.



## 3. Access-Allow-Origin

- Open the Burp Suite history tab.
- Request to endpoint '{URL}'
- Observe the headers. Note the 'Access-Control-Allow-Origin' set to wildcard.



## 3.12 PY-CL-012: GraphQL endpoint returns Error Stack in response

**Potential Impact:** MEDIUM

### Description:

During the security assessment of the application, we found that the application is returning error stack for any error on GraphQL endpoint. This might lead to server configuration, and the file path being leaked. Such an error stack should be disabled in production.

### Affected Resources

- {Client Name} web Application
  - {URL}

### CVSS Score:

CVSS Base Score: 4.3  
CVSS Temporal Score: 4.1  
CVSS Environmental Score: 3.7

### CVSS Vector:

CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:  
L/I:N/A:N/E:H/RL:O/RC:C/CR:H/IR:H/  
AR:M/MAV:N/MAC:H/MPR:N/MUI:R/M  
S:U/MC:L/MI:N/MA:N

**Business Risk:** An attacker might be able to leak Intellectual Property.

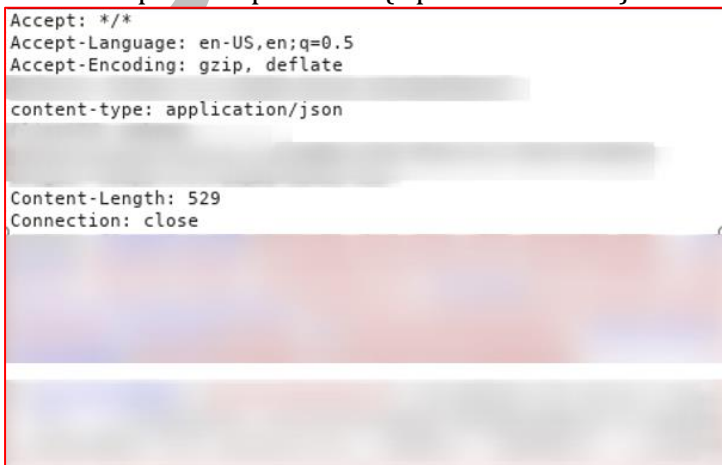
**Technical Risk:** Attacker might leak to server configuration and file path.

### Remediation:

- Disable error stack.
- Use custom messages for errors.


### Steps to Reproduce:

1. Visit {URL}. Let page load
2. Open Burp Suite and find the request to GraphQL endpoint.
3. Look for specific operation '{Operation Name}'.



```
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
content-type: application/json
Content-Length: 529
Connection: close
```

4. Send the request to the 'Repeater'.
5. Edit the request to add '(id: -1)' as shown image below.



```
s": {},  
"query
```

6. Observe the error stack in response.



```
{  
  "errors": [  
    {  
      "message": "Access to user: -1 is not allowed",  
      "path": [  
        "user"  
      ],  
      "extensions": {  
        "code": "FORBIDDEN",  
        "serviceName": "core".  
      },  
      "variables": {},  
      "exception": {  
        "stacktrace": [  
          "ForbiddenError: Access to user: -1 is not allowed",  
        ]  
      }  
    }  
  ]  
}
```

## 4. We Prescribe

The below table provides an at-a-glance view of the remediation solution and the best practices that should be taken into consideration to improve the security posture of the scoped-in application:

Vuln. ID	Recommendation and Best Practices
<b>PY-CL-001</b>	<ul style="list-style-type: none"> <li>• Use standard libraries for generating PDFs.</li> <li>• Whitelist selected HTML tags</li> </ul>
<b>PY-CL-002</b>	<ul style="list-style-type: none"> <li>• Restrict the number of attempts from one IP</li> <li>• Restrict the number of failed attempts for a single user</li> <li>• User industry-standard Captchas.</li> </ul>
<b>PY-CL-003</b>	<ul style="list-style-type: none"> <li>• Ensure the ownership of the document before performing an action</li> <li>• Implement a checklist to ensure proper attention is given to 'Authorization' at every level.</li> </ul>
<b>PY-CL-004</b>	<ul style="list-style-type: none"> <li>• Ensure the ownership of the document before performing an action</li> <li>• Implement a checklist to ensure proper attention is given to 'Authorization' at every level.</li> </ul>
<b>PY-CL-005</b>	<ul style="list-style-type: none"> <li>• Limit the use of uploaded documents to limited functions.</li> <li>• Ensure that an uploaded document is destroyed after the request is processed or the document is saved for future use.</li> <li>• Check the type of document that can be accepted.</li> </ul>
<b>PY-CL-006</b>	<ul style="list-style-type: none"> <li>• Limit the schema of the URL that can be used.</li> </ul>
<b>PY-CL-007</b>	<ul style="list-style-type: none"> <li>• Strip HTML tags from user input.</li> <li>• Ensure that the developers follow the library/tool's recommendations.</li> </ul>
<b>PY-CL-008</b>	<ul style="list-style-type: none"> <li>• Strip HTML tags from user input.</li> <li>• Ensure that the developers follow the library/tool's recommendations.</li> </ul>
<b>PY-CL-009</b>	<ul style="list-style-type: none"> <li>• Strip HTML tags from user input.</li> <li>• Ensure that the developers follow the library/tool's recommendations.</li> </ul>
<b>PY-CL-010</b>	<ul style="list-style-type: none"> <li>• 'Origin' header should be validated to be from the allowed site.</li> </ul>
<b>PY-CL-011</b>	<ul style="list-style-type: none"> <li>• Developers should be made aware of such standard security practices to help avoid misconfiguration.</li> </ul>
<b>PY-CL-012</b>	<ul style="list-style-type: none"> <li>• The Custom error should be displayed instead of the full Error Stack.</li> </ul>

**Looking at the types and severity of vulnerabilities found, we would recommend:**

- The client should properly escape user input.
- Also, we would recommend that the server should perform proper Authorization checks.

SAMPLE



## 5. Appendix



### 5.1 References

In this section, we have given additional data regarding the vulnerabilities that we have found during the engagement to provide better clarity and more insight into the bugs

1. **Server-Side Request Forgery (SSRF):** In a Server-Side Request Forgery (SSRF) attack, the attacker can abuse functionality on the server to read or update internal resources. The attacker can supply or modify a URL which the code running on the server will read or submit data to, and by carefully selecting the URLs, the attacker may be able to read server configuration such as AWS metadata, connect to internal services like HTTP enabled databases or perform post requests towards internal services which are not intended to be exposed.

Ref: [https://owasp.org/www-community/attacks/Server\\_Side\\_Request\\_Forgery](https://owasp.org/www-community/attacks/Server_Side_Request_Forgery)

2. **Cross-Site Scripting (XSS):** Cross-Site Scripting attacks are a type of injection in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end-user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page.

Ref: <https://owasp.org/www-community/attacks/xss/>

### 5.2 Observations

In the observation section, we document any specific issue that we have observed, which, while does not directly lead to any vulnerability, maybe something that needs to be looked at from the perspective of a functional bug or configuration issue.

**We haven't observed anything unusual in this project from functionality or configuration perspective.**



In the failed test case section, we document some of the application strengths that we have observed during our testing. This serves as a view of the best practices that are being followed by the client.

### 5.3.1 SQL Injections

#### Description

We tested the application for SQL injection, but all the endpoints seemed to have proper protection against SQL injection attacks. Additionally, GraphQL had introspection disabled..

#### Testing Process

While using the application, use SQL injection payload at various endpoints. For example, during editing notes.

#### Request:

```
{ "metadata": { "noteId": "[REDACTED]", "title": "SQLi Test", "accessRights": "EVERYONE" }, "body": "<p data-node-type='paragraph'>Body '\</p>" data-bbox="118 492 872 520"/>
```

**Response:** 200 observed with no errors in the body.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 234
Connection: close
Server: [REDACTED]
Date: Fri, 05 Jun 2020 17:01:26 GMT
Access-Control-Allow-Origin: *
Vary: Accept-Encoding
Strict-Transport-Security: max-age=15724800; includeSubDomains
X-Cache: Miss from cloudfront
```

Also, by manually editing GraphQL queries. Refer to the images below.

## Original Request

```
query {
```

## Edited Request

```
query {
```

## Edited Request's Response

```
{
  "errors": [
    {
      "message": "Access to user: 3133 is not allowed",
      "path": [
        "user"
      ]
    }
  ]
}
```

```
{
  "errors": [
    {
      "message": "Access to user: 3133 is not allowed",
      "path": [
        "user"
      ]
    }
  ],
  "extensions": {
    "code": "INTERNAL_SERVER_ERROR"
  }
}
```

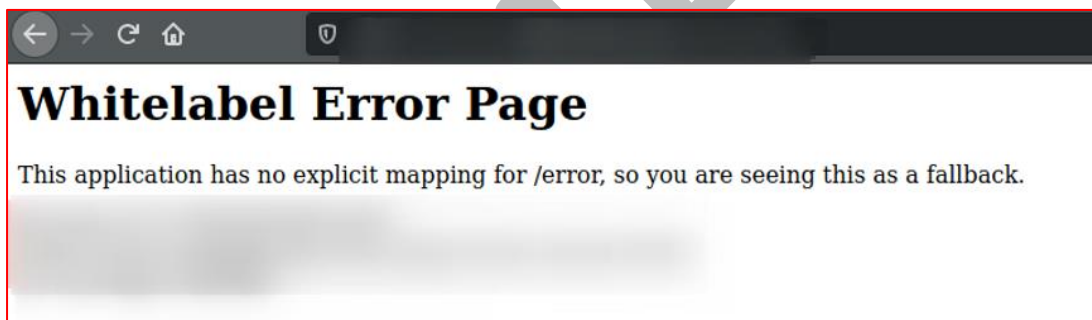
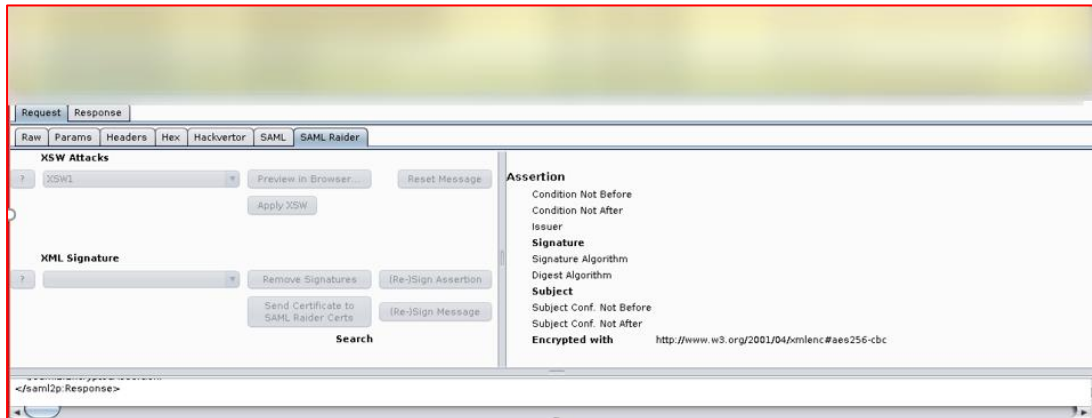
## 5.3.2 OAuth/SSO Misconfiguration

### Description

The application seemed to handle SSO and OAuth exception and respond properly.

### Testing Process

Tested SSO configuration by Burp. The burp configuration test against multiple test cases like manipulating signature, removing it, wrapping it, etc. For all those modifications, we received an error from the server.



The server threw a proper error when trying to manipulate parameters in the OAuth token generation process.

## Original Request

```
GET / HTTP/1.1
Host: 
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:77.0) Gecko/20100101
Firefox/77.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: 
ClientId: 
Authorization: 
Connection: close
```

## Corresponding Response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Connection: close
Server: 
Date: 
Vary: Accept-Encoding
Cache-Control: no-cache, no-store, max-age=0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=15724800; includeSubDomains

Content-Length: 176
```

## Highlighted part to be removed

```
GET / HTTP/1.1
Host: 
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:77.0) Gecko/20100101
Firefox/77.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://rc.alpha-sense.com/dashboard
ClientId: 
Authorization: bearer 
Connection: close
```

## Response to edited request

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
Connection: close
Server: [REDACTED]
Date: [REDACTED]
Cache-Control: no-cache, no-store, max-age=0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
Cache-Control: no-store
Pragma: no-cache
WWW-Authenticate: Bearer realm="oauth", error="unauthorized",
error_description="Full authentication is required to access this resource"
Strict-Transport-Security: max-age=15724800; includeSubDomains

[REDACTED]

{"error": "unauthorized", "error_description": "Full authentication is required to access this resource"}
```

The server always threw 'Unauthorized error' whenever Authorization token was missing.