AppAttack x Thoth Tech (OnTrack) Project Collaboration

End-Of-Trimester Finding Report

SIT374/764 and SIT378/782

Capstone - Team Project

Deakin University, Australia

Table of Contents

1. Statement of Confidentiality

This document has been meticulously crafted by AppAttack following rigorous testing on **Thoth Tech's OnTrack Project**. AppAttack regards the contents herein as the proprietary and confidential information of the **OnTrack Project** and its capstone leadership team. This information is intended solely for its designated purpose and must not be disseminated to any other vendor, business partner, or contractor without the prior written consent of the **OnTrack Project** or the explicit approval of the capstone unit directors.

Furthermore, no portion of this document may be conveyed, duplicated, copied, or distributed without the prior consent of the parties involved or the capstone directors. It's important to note that the contents of this document are not to be construed as legal advice. While AppAttack provides services that pertain to compliance, litigation, or other legal matters, they should not be interpreted as a substitute for legal counsel and must not be relied upon as such.

The assessment documented herein involves the exploration of vulnerabilities within the **OnTrack Project**, conducted exclusively for training, testing, and security examination purposes. It is essential to emphasize that any vulnerabilities discovered in this process will not impact the security of AppAttack's external or internal infrastructure.

2. Engagement Contacts

AppAttack Contacts		
Primary Contacts	**Title**	**Primary Contact Email**
NIKOLA NICHOLAS KRCEVINAC	Project Co-Lead	s222564328@deakin.edu.au
FILIPE MANUEL FUNINA OLIVEIRA	Project Co-Lead	s222478779@deakin.edu.au
DARRYL JUN OOI	Project Co-Lead	s222216788@deakin.edu.au
SHEHANI NATASHA DE SAYRAH WICKREMASEKERA	Project Co-Lead	s223841302@deakin.edu.au
Secondary Contacts	**Title**	**Secondary Contact Email**
WAHIDULLAH HASHIMI	Jr Co-Lead	s223397352@deakin.edu.au

Director Contact		
Director Name	**Title**	**Director Contact Email**
ANH DINH	Hardhat Acting Director	anh.dinh@deakin.edu.au

3. Executive Summary

AppAttack was tasked to perform a thorough security evaluation of Thoth Tech's OnTrack Project web application, which included both penetration testing and a detailed secure code review. Our evaluation followed industry-standard security practices, and the results were compared against these established benchmarks. The findings and conclusions in this report provide an overview of the application's current security status as of the mid-point in our assessment. The recommendations offered here aim to help address the identified security issues.

The evaluation was conducted using the security guidelines set by the Open Web Application Security Project (OWASP), ensuring a high-quality, best-practice approach. This report summarizes the results of the penetration testing and secure code review carried out by AppAttack between March 24th (Week 4) and May 9th (Week 9) in T1 2025. The tests were performed in a staging environment set up specifically for this purpose, running on a Kali-based VM. During testing, AppAttack used different attack scenarios to assess how well the application could withstand security threats. AppAttack has compiled a list of vulnerabilities discovered during the duration of the engagement, categorized by their severity.

At the midpoint of the engagement (April 7th), AppAttack provided the OnTrack team with the findings of the vulnerabilities discovered to that point (6.1 to 6.11). In response, the OnTrack team remediated two critical vulnerabilities (6.9 and 6.10) - clickjacking and Insecure Direct Object References (IDOR), which were subsequently retested and verified by AppAttack. AppAttack also left comments on the relevant commits and pull requests in OnTrack's GitHub repository to confirm that these vulnerabilities had been successfully addressed and retested.

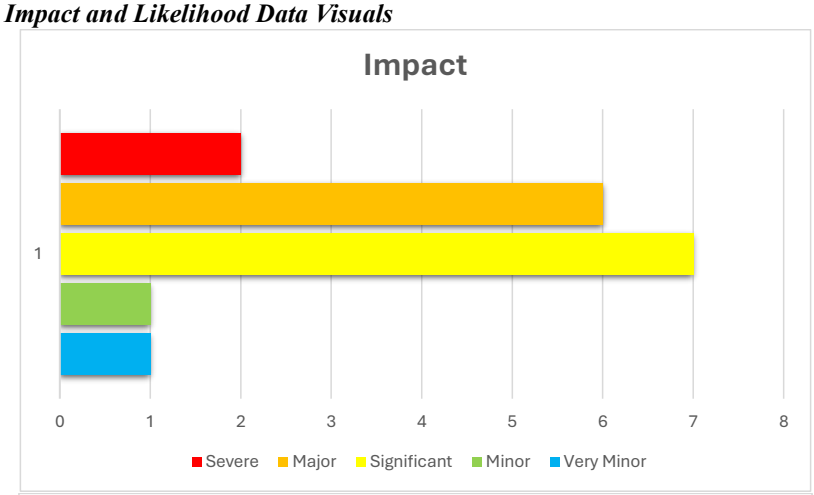
4. Technical Summary

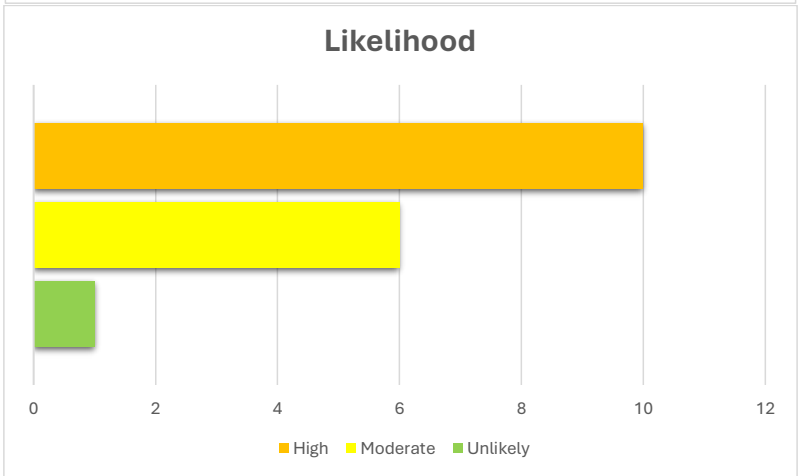
4.1. Risk Rating Summary

No#	Type of Attack	Impact	Likelihood	Risk Rating
1.	Exposed JavaScript Source Map	**Major**	**High**	16
2.	Insufficient Token Entropy	**Very Minor**	**Moderate**	3
3.	Forced Session Fixation	**Significant**	**High**	12
4.	Insufficient Session Binding	**Significant**	**Moderate**	9
5.	Exploitable CORS Configuration	**Significant**	**High**	12
6.	Insecure Token Exposure	**Significant**	**High**	12
7.	Publicly Accessible API Documentation	**Major**	**Moderate**	12
8.	Privilege Escalation via URL	**Minor**	**High**	8
9.	Clickjacking Vulnerability	**Severe**	**High**	20
10.	IDOR Vulnerability	**Major**	**High**	16

11.	Malicious Code Execution	**Major**	**Moderate**	12
12.	Privilege Escalation Chain	**Significant**	**Unlikely**	6
13.	Privilege Escalation via Insecure Cookie Manipulation	**Significant**	**High**	12
14.	Lack of Rate Limiting on Login Endpoint	**Major**	**High**	16
15.	Insecure Transmission of Credentials	**Severe**	**High**	20
16.	Client-Side Template Injection	**Major**	**Moderate**	12
17.	Unauthenticated API Endpoint Disclosure	**Significant**	**Moderate**	9

4.2. Impact and Likelihood Data Visuals





5. Risk Rating Matrix

To provide a clear understanding of the risks posed by these identified vulnerabilities, we have employed a risk matrix. This matrix categorizes vulnerabilities into distinct risk categories based on their potential impact and likelihood of exploitation. The following chart talks about the vulnerabilities that can be classified:

Impact: It relates to an estimation of how impactful an issue is related to how severe it would affect an operation. The impact ranges from very minor to severe. This estimation is made for the current state of an issue and is not to be used as a future predictor.

Impact values				
Very Minor	***Minor***	***Significant***	***Major***	***Severe***
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. It can cause some damage but not enough to impede regular activity.	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally.	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

Likelihood: It relates to the measure of how difficult an issue is to find and exploit. This is not representative of the likelihood of an attacker using a certain exploit in the future but rather how easily it is found.

Likelihood				
Rare	***Unlikely***	***Moderate***	***High***	***Certain***
The event may occur and/or if it did, it occurs in specific circumstances.	The event could occur occasionally and/or could happen (at some point)	The event may occur and/or happen.	The event occurs at times and/or probably happens a lot.	The event is occurring now and/or happens frequently.

Risk rating matrix: The matrix below is used to compare the impact value rating against that of the likelihood rating. Using this process, we can produce an aggregate score in which we believe the overall importance of an issue is and how critical it is to fix. This is an estimated value and will not always be up-to-date and relevant. The ratings given by the matrix are given for vulnerabilities in the present and may not reflect the future.

	Impact				
Likelihood		**Very minor x1**	**Minor x2**	**Significant x3**	**Major x4**	**Severe x5**
	Certain x5	**Low (5)**	**Medium (10)**	**Intermediate (15)**	**High (20)**	**Extreme (25)**

	High x4	**Low (4)**	**Medium (8)**	**Intermediate (12)**	**High (16)**	**High (20)**
	Moderate x3	**Low (3)**	**Medium (6)**	**Medium (9)**	**Intermediate (12)**	**Intermediate (15)**
	Unlikely x2	**Low (2)**	**Low (4)**	**Medium (6)**	**Medium (8)**	**Medium (10)**
	Rare x1	**Low (1)**	**Low (2)**	**Low (3)**	**Low (4)**	**Low (5)**
	No Threat	**Info (0)**	-	-	-	-

Informational	**0**	This level of risk is informational only and does not pose a threat to the organization but is a security best practice.
Low	**0-5**	This level of risk can be accepted if no appropriate strategies can be easily implemented, or it is not cost efficient to do so. This risk should be monitored to ensure that changes are detected and acted upon.
Medium	**6-10**	This level of risk can still be accepted if no appropriate strategies can be easily implemented. Monitoring this risk level should be done more regularly to ensure changes are detected.
Intermediate	**11-15**	This level of risk should not be accepted, and strategies should be aimed at minimizing or getting rid of this risk. This strategy should be developed and introduced within the next 3-6 months.
High	**16-20**	This level of risk should not be accepted. Strategies should be developed and implemented in a hasty manner, usually within 1 month.
Extreme	**21-25**	This level of risk entails a situation where an impact would be so severe that activity would need to cease immediately. Mitigation for the said risk should be taken immediately or plans for strategies should be in place immediately.

This report serves as an invaluable resource to guide the Thoth Tech project "OnTrack". This will help the Thoth Tech team in improving their security posture for their web application. It is our hope that the findings found, and recommendations provided within will enable you to address these vulnerabilities effectively and enhance your project's overall security resilience of the application.

6. OnTrack Vulnerability Findings

6.1: Exposed JavaScript Source Map

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Adam Afzal Awan	AppAttack Pen-Tester	Junior	OnTrack	Filipe Oliveira	No

Was this Finding Successful?
Yes

Finding Description

During a security assessment it has been discovered that the OnTrack application was exposing the JavaScript source map material via **main.js.map** to unauthenticated users. This file was located at **172.18.0.1:4200/main.js.map** and could be accessed directly through the URL or command line tools like wget. The source map contains detailed information about the original source code of the JavaScript used in the application which can pose a serious security risk to the OnTrack Application

Risk Rating

Impact: Major
Likelihood: High

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity.	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally.	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

The exposure of JavaScript source maps poses significant security vulnerabilities such as reverse-engineering of the application. Attackers can gain insight into how the application

was built and operating revealing sensitive structures. Attackers could exploit weaknesses in the application, steal API keys, or inject malicious code.

API keys and tokens are highly sensitive because they can provide unauthorized access to key systems or third-party services. An attacker in possession of exposed API keys can bypass authentication, access sensitive user data and manipulate the application's functionality, leading to devastating consequences such as unauthorized data access, privilege escalation, and further malicious actions.

This exposure could lead the company to experience a compromise to its intellectual property, data theft and data breaches to users, this not only impacts the company's reputation but also severally affects the OnTrack users trust in the application.

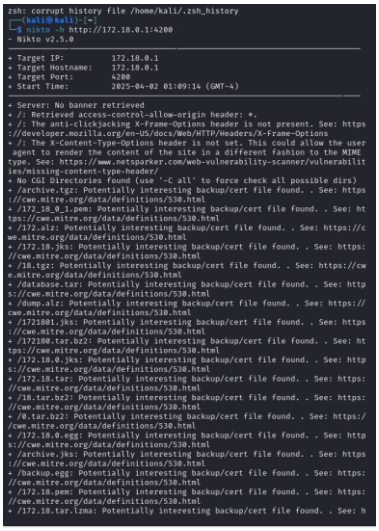
Affected Assets

- **172.18.0.1: 4200/main.js.map**: Exposed JavaScript source map that could reveal the applications source code.
- **API Endpoints**
  - /api/users: Manages user data including personal information, credentials and access rights.
  - /api/service: Handling service configurations.
  - /api/models: Linked to data models that may involve sensitive user or application data.
- **OnTrack Application**: The entire application is vulnerable due to exposed source map and API endpoints
- **User Data**: Sensitive data related to users, could be exposed through the API vulnerabilities
- **Intellectual Property**: Application's code and business logic can be exploited if exposed
- **API Keys**: Sensitive tokens or keys could grant unauthorized access to the system.

Evidence

Step 1: Nikto Scan

First, we start with running a Nikto Scan on the target application to search for vulnerabilities With Nikto -h with the target URL. This brings us a list of interesting backup files.

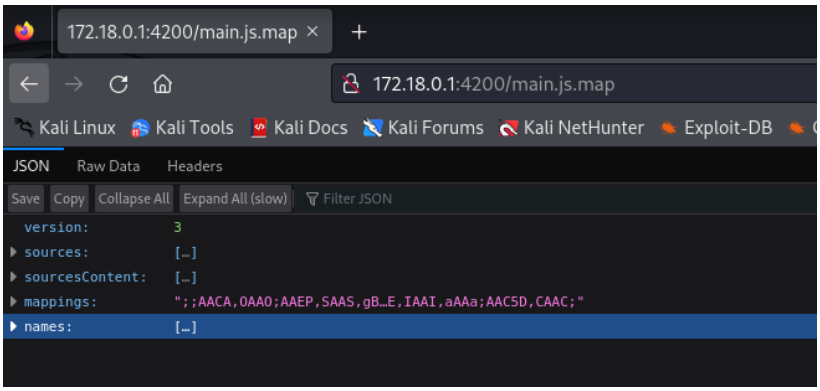


Step 2: Accessing Potentially Exposed File

The file that was first identified from the list of potentially exposed backup files was a .pem file which was shown to have read only permissions. Using the 'head' command we can examine the file's contents. Upon inspection, it revealed HTML structure and JavaScript code. The **@vite/client** line tells us that this is a client-side resource and is a possible vulnerability.

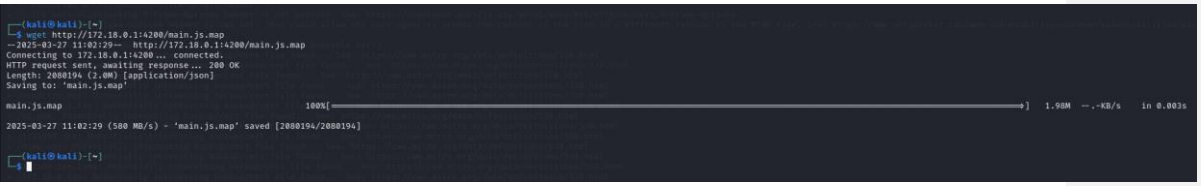
Step 3: Identifying the source map file

As we have determined, the application is using Vite which generates source map files to map JavaScript code back to the original source code. These Source maps are commonly used in development for debugging and tracking down errors. We have now identified the exposed source map and successfully accessed it through URL.



Step 4: Downloading the Source Map

To further analyze the source map, we can use the wget command to download the source map from the web server.



Step 5: Analyzing the Source Map

After downloading the source map, we can use the ‘cat’ command to open and analyze the contents. The source map file contains a mapping from JavaScript code and its original source. This is useful for understanding how the application was structured and its potential security issues.

Step 6: Identifying Sensitive Information

Now that we have access to the source maps content, we can use the ‘grep -i’ command to search for keywords that may expose sensitive information. For example, by using the term ‘api’, we can locate API paths, tokens and keys that may be exposed. These could be critical for securing sensitive information that an attacker could exploit to gain unauthorized access to the applications resources.

Step 7 (Optional): Further Analysis

As we have already identified sensitive information by searching for the keyword ‘api’ we can further analyze by using the keyword ‘token’. This allows us to uncover additional exposed lines related to tokens, which may reveal more critical information that could be exploited by attackers. This provides us with a deeper dive into the source map content, identifying more potential vulnerabilities.

Remediation Advice

- Configuring the server to remove source maps where they are not necessary to reduce the risk of exposed sensitive code this can be done in depending on the framework (Vue, Angular, React) by disabling source maps generation in production.
- Block access to .map files using server rules in Nginx, Apache, Node.js
- Implement access control with role-based access control (RBAC) to ensure only authorized users can view or interact with sensitive files like source maps.
- Minify and obfuscate JavaScript to make it difficult for attackers to read and understand the code.
- Regularly review and clean up source maps to help limit exposure of sensitive code with Nikto and OWASP ZAP.
- Upload source maps privately via Sentry.

References

Nikto Web Scanner
https://cirt.net/Nikto2

Wget Command
https://www.gnu.org/software/wget/manual/

Grep Command
https://www.gnu.org/software/grep/manual/

Vite Documentation
https://vite.dev/

Source Maps
https://developer.chrome.com/blog/sourcemaps

JavaScript Source Maps
https://docs.sentry.io/platforms/javascript/sourcemaps/

Disabling Source Map
https://alan.norbauer.com/articles/disable-source-maps

Contact Details

Adam Afzal Awan
s223169786@deakin.edu.au

Pentest Leader Feedback.

Filipe Oliveira s222478779@deakin.edua.au

- **Please Change Blue text to black in the references and in “was this finding successful?”**
- **Include spacing between steps 4-5**
- **Is this a re-tested Finding? Pleas answer this in the box at the top, as No**
- **Step 1,2 and 3 screenshots need to be retaken as they contain a white filter over the top and are unclear**
- **Affected assets needs to include specifics not just “Ontrack App” (API paths, addresses, etc)**
- **Yes, having the API and tokens are sensitive information, but you must explain why this is an issue and what the attacker could possibly do if they gather this information.**
- **If you can find information on how to fix an issue like the one you found, on the internet you should also provide that in the remediation section**

6.2: Insufficient Token Entropy

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Andrew Duffy	AppAtack	Pen-Tester	OnTrack Web App	Filipe Oliveira	No

Was this Finding Successful?
Yes

Finding Description

An **Insufficient Token Entropy vulnerability** was discovered in the OnTrack web application. Session or Authentication Tokens with insufficient entropy have reduced randomness and present opportunities for threat actors to guess valid tokens through brute force techniques. The Entropy value is the indicator of how resilient a token is to this brute force guessing.

Analysis was conducted using the Burp Suite Sequencer tool, a free tool that is readily available, used for testing token randomness. The Sequencer tool sends thousands of authentication requests to the application and captures the returned authentication token. In this instance 10000 tokens were captured, which represents a significant sample size. The tool will assess the token to determine its actual randomness at a bit level and identify patterns or predictability with the final output or entropy estimate as an indication of its resilience to being brute forced.

As a result, the token randomness was assessed as being **reasonable** with an entropy of approximately **38 bits**. Entropy represents the randomness or unpredictability of a token, and higher entropy means more possible combinations an attacker must guess. The Open Worldwide Application Security Project (OWASP) recommends that session identifiers have at least 64 bits of entropy to prevent brute force attacks. According to OWASP, session IDs with 64 bits of entropy, could take a threat actor approximately 585 years to successfully guess a valid session ID. [1]

The simplistic nature of brute force attacks and ready availability of the tools, tempered with the relative time it would stake to conduct a successful attack, despite teh lower entropy, the likelihood of this vulnerability being exploited is **Moderate**. Additionally, compensating controls such as token invalidation techniques limit the impact of a successfully brute forced token, reducing the impact to **Very Minor**, resulting in an overall risk rating of **Low**.

Further analysis with a larger sample size (approximately 20,000 tokens) will likely result in an even lower assessed entropy and randomness score.

Risk Rating - Low

Impact: **Very Minor**
Likelihood: **Moderate**

[1] OWASP. Session Management. https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity.	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally.	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

While the test conducted doesn't directly exploit a token vulnerability, the results are indicative of an underlying vulnerability. The presence of session tokens with low entropy provides an opportunity for threat actors to exploit and opportunity to obtain a valid session token through brute forcing techniques. A successful compromise could expose user accounts or sensitive data through authenticated requests. Although compensating session management controls are in place to limit the exposure to this vulnerability, weaknesses in token generation below the industry recommendations may undermine user trust in the applications security.

Affected Assets

OnTrack Web Applicaiton Authentication/Session Token implementation. Any API, endpoint of functionality that relies on a valid authentication token is potentially vulnerable.

Evidence

Step 1: Configure Burp Suite and Firefox

1. Configure Firefox to route all traffic through the Burp Suite proxy, ensuring that requests made to the application are processed through Burp and can be reviewed in the HTTP history.

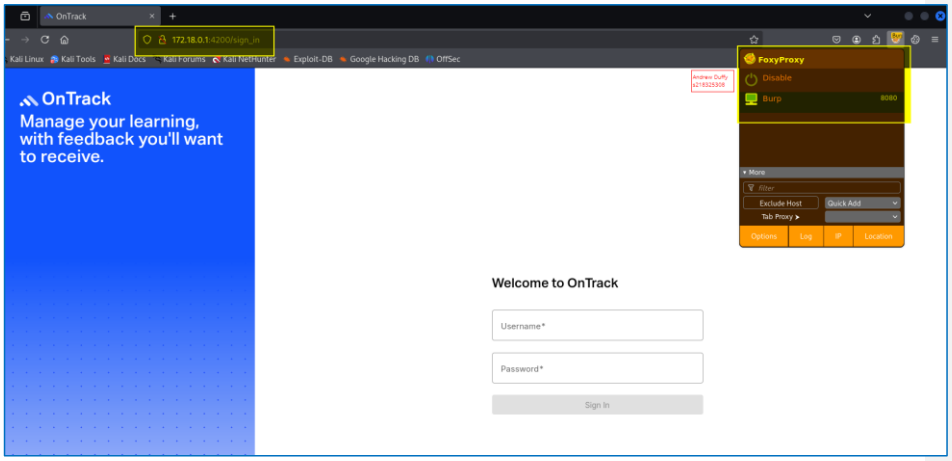


Figure – Firefox configured to route traffic through Burp Suite

Step 2: Login to the application review HTTP history in Burp Suite

1. Login to the application as student_1.

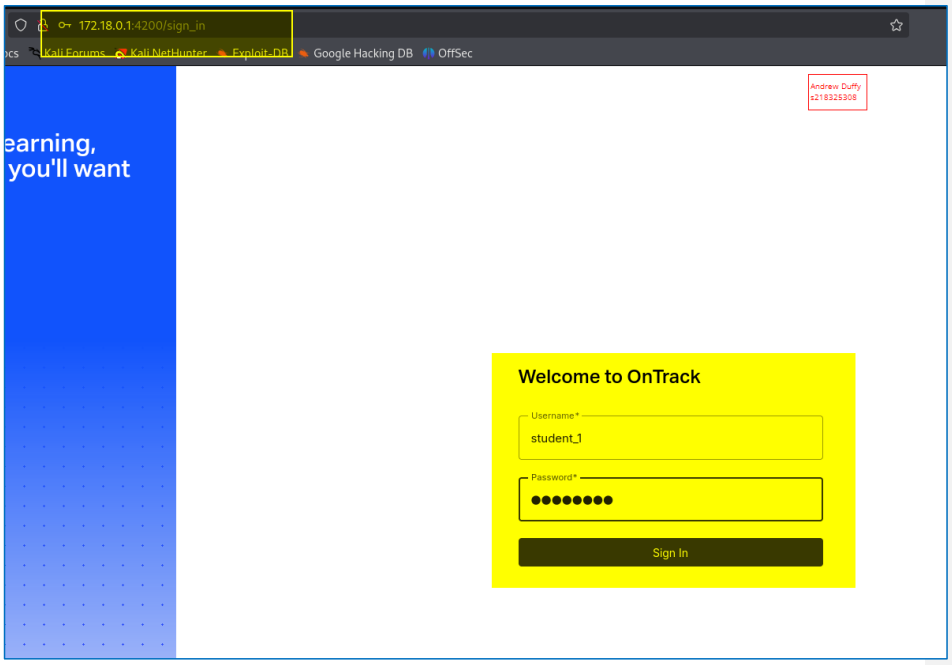


Figure – Login to application

2. Go to Burp Suite and navigate to Proxy, HTTP history and identify the HTTP request that returns an authorisation token.

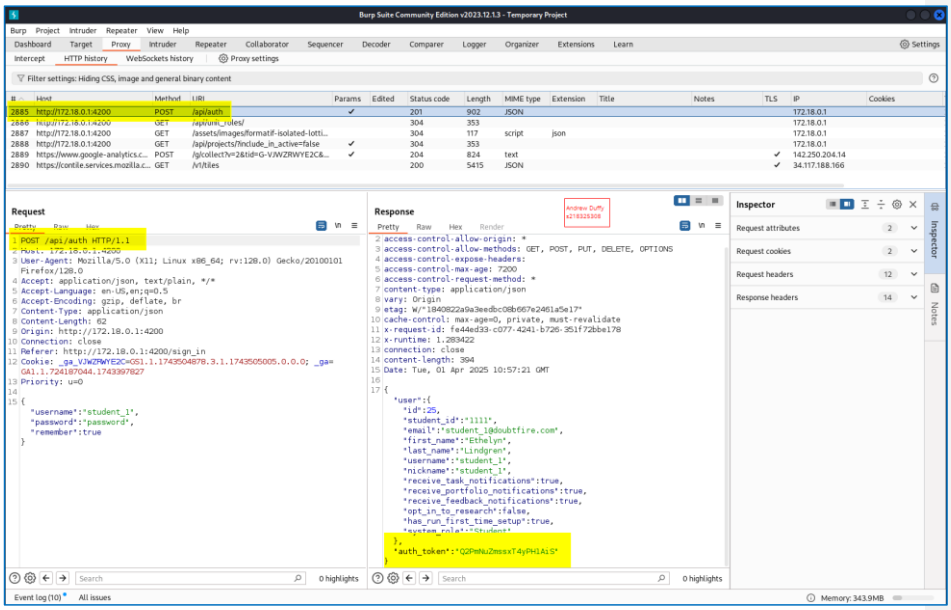


Figure – Identifying the authorisation token request / response

Step 3: Send authorisation request to Burp Suit Sequencer tool

1. Right click within the 'Request' box and select 'Send to Sequencer' in the pop up menu. Then navigate to the Sequencer tab and click on Configure to set the token location for the tool.

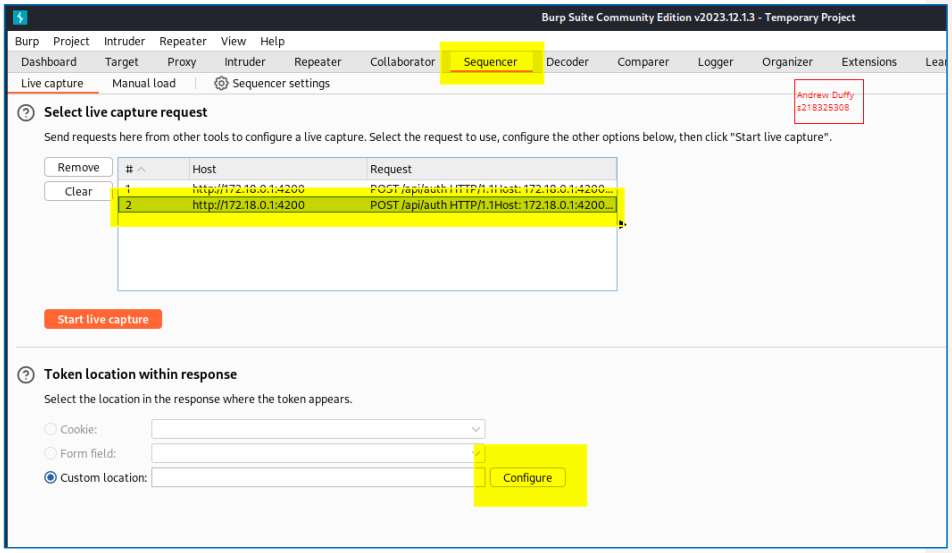


Figure – Navigate to Sequencer, select configure

2. In the new window pops up to configure the token location, scroll down to the token and highlight just the characters between the quotations then click ok. This will set the token for the tool and will populate the ‘Custom location’ field.

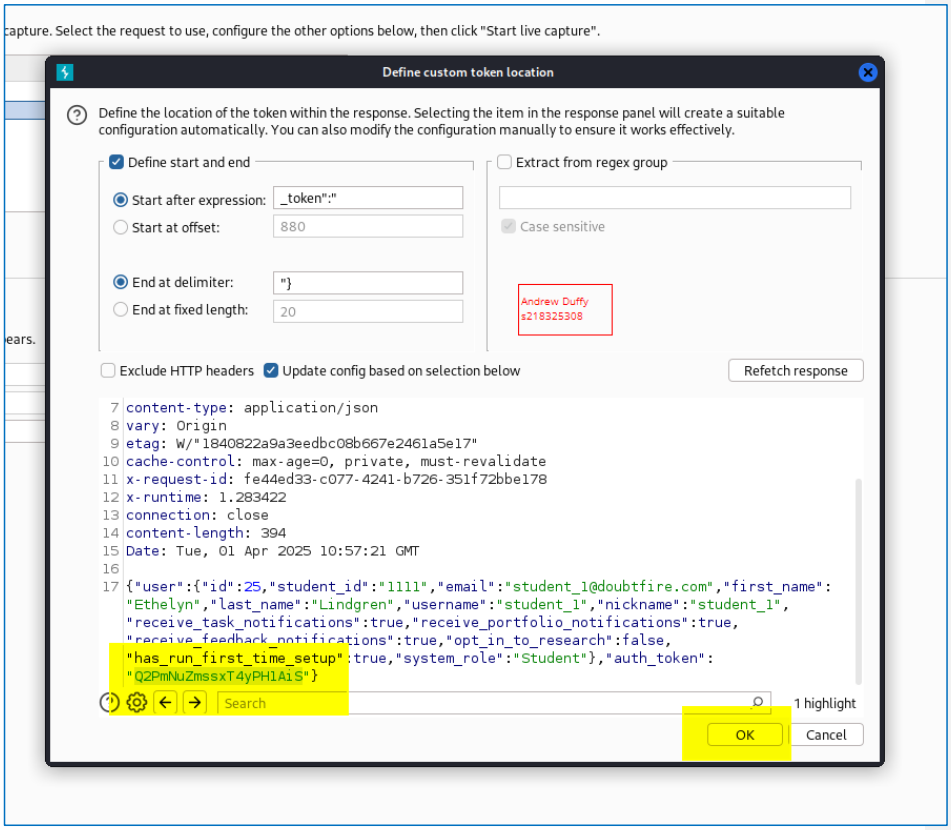


Figure – Setting token location

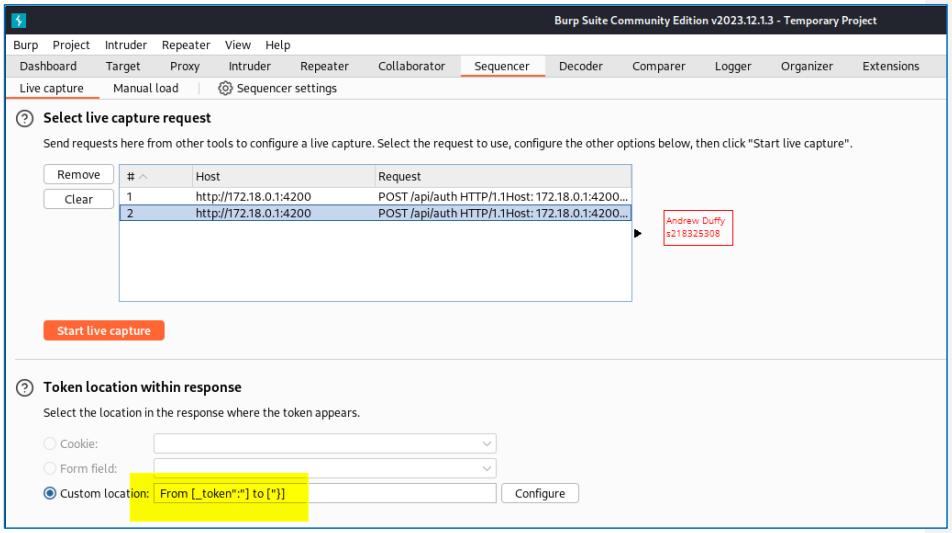


Figure – Custom token location populated within the tool

Step 3: Start live capture

1. Click ‘Start live capture’ this will open another window where the tool will send authorisation requests to the application and capturing the authorisation token. Let the tool run for approximately 10000 requests, this is significant enough to get an accurate indication of the token's strength. Once 10000 requests have been captured click ‘Stop’ then ‘Analyze now’. This will present a summary of the token's overall randomness and entropy.

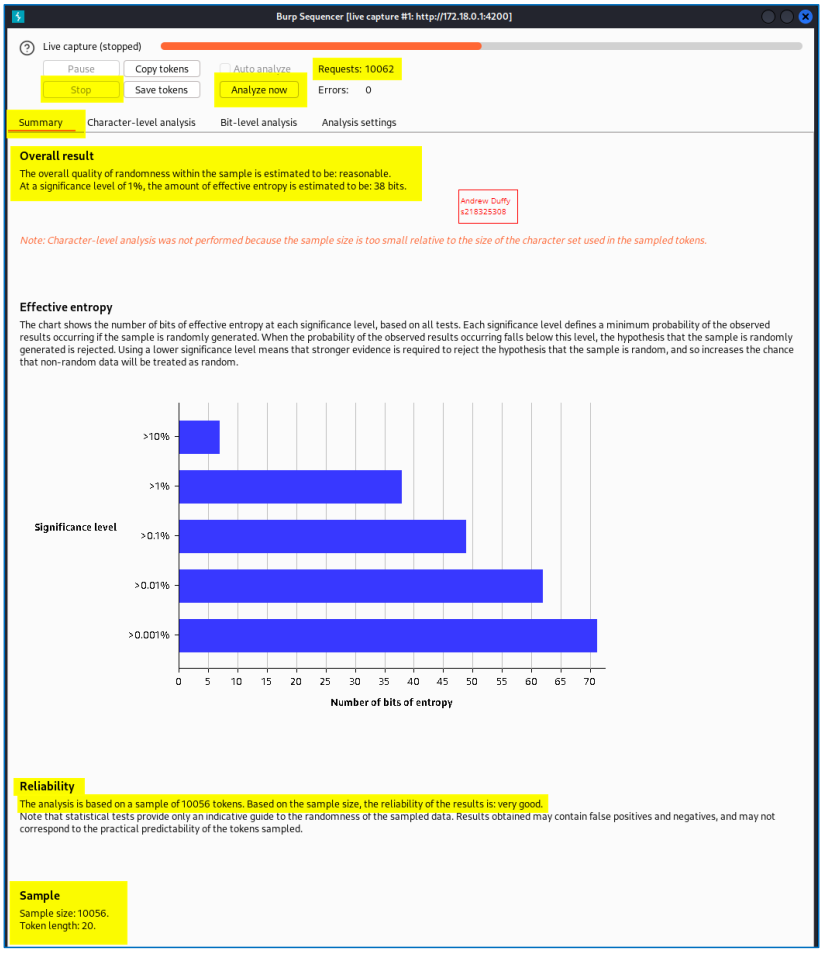


Figure – Summary of token analysis

2. Further analysis results are available to better understand bit predictability.

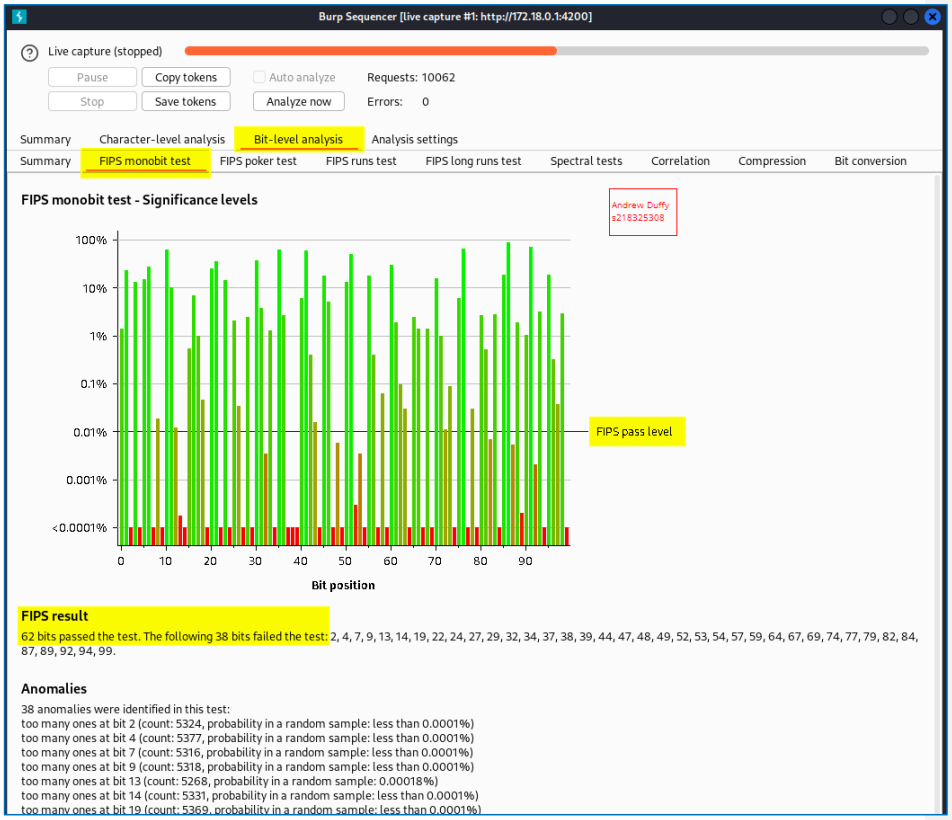


Figure – Further analysis is available under each specific test

Remediation Advice

The application currently uses a session token generator with lower than recommended entropy, making tokens more predictable and vulnerable to brute-force attacks.

Mitigations:

- Review current session identifier implementation and uplift in line with OWASP recommendations, uplift to ensure at least 64 bits of entropy.
- Use a strong CSPRNG (Cryptographically Secure Pseudorandom Number Generator) must be used to generate session IDs.
- Retest session token using same method (e.g. Burp Suite Sequencer) to confirm improvements have been implemented effectively.

References

- OWASP - Session Management Cheat Sheet
- OWASP - Insufficient Session ID Length
- CWE - CWE-331: Insufficient Entropy

Contact Details

Name/Teams: Andrew Duffy
Email: s218325308@deakin.edu.au

Pentest Leader Feedback.

Filipe Oliveira s222478779@deaki.edu.au

- Make sure blue text is changed to black - Done
- Some bits of your text are not size 12 times new roman – Fixed, I note the sub heading ar size 13. And the table at top is 10.
- Expand on affected assets e.g any API endpoints or pages protected by session tokens – The affected item is the token, or the token generation implementation. Then by extention the whole application. -- Updated.
- You should also add more references e.g Burp, owasp auth cheat sheet - Added
- Unsure if at the bottom of step 2 there is a missing image or just accidental repeat of (Figure – Student level user accessing admin level endpoint) - Fixed. Removed.

well done!

6.3: Session Hijacking – Forced Session Fixation

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Andrew Duffy	AppAtack	Pen-Tester	OnTrack Web App	Nicholas Krcevinac	No

Was this Finding Successful?
Yes

Finding Description

A Session Hijacking vulnerability was discovered in the OnTrack web application due to forced session persistence through manipulation of the logout process. In this instance, a malicious actor can maintain access to a valid session token beyond logout by intercepting and 'dropping' the logout request (DELETE /api/auth). This prevents the application from properly invalidating the session when the user logs out, allowing the threat actor to continue issuing authenticated requests with the stolen token.

During testing, the logout action was triggered from a browser session with a privileged user. However, by intercepting the DELETE /api/auth request in Burp Suite and 'dropping' (deleting or not sending) that request, the session token remained valid and usable for a period of time. The captured privileged user's token was then able to continue to be used for issuing requests, which included privileged actions, even after the user had logged out.

This vulnerability demonstrates that session termination is not enforced on the server-side when a user logs out, and that token invalidation relies solely on the logout request reaching the server. This allows a threat actor to force a session to persist beyond its intended lifecycle, maintaining access and increasing the window of opportunity for exploit.

Given the simplicity of the attack; the chance of intercepting logout requests in compromised or untrusted environments, and the simplicity in identifying the Delete request, the likelihood is that this vulnerability will be targets is High. However, compensating controls like time-based token expiry that appeared to be in place, reduce the impact to Significant. This results in an overall Risk Rating of Intermediate.

Risk Rating - Intermediate

Impact: **Significant**
Likelihood: **High**

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular	Risk that holds minor form of impact, but not significant enough to be of threat. Can	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can	Risk that holds major impact to be of threat. Will cause damage that will impede regular	Risk that holds severe impact and is a threat. Will cause critical damage that can

activity can continue.	cause some damage but not enough to impede regular activity.	impede regular activity but will be able to run normally.	activity and will not be able to run normally.	cease activity to be run.

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

This vulnerability demonstrates a weakness in the application's logout process that allows an attacker to maintain access to a valid session token even after the legitimate user has logged out. By intercepting and dropping the logout request, an attacker can prevent the session from being properly invalidated on the server side. This forced session persistence creates an opportunity for unauthorised use of a stolen token beyond the expected session lifespan. Allowing a threat actor to conduct actions and access potentially sensitive data that was restricted to the privileges to the stolen token potentially compromising confidentiality and integrity. Compensating controls such as time-based token expiry may reduce risk by limiting the exposure window available to the threat actor.

Affected Assets

- OnTrack Web Applicaiton Authentication/Session Token implementation. Any API, endpoint of functionality that relies on a valid authentication token is potentially vulnerable.
- DELETE /api/auth

Evidence

The attack chain below simulates a threat actor intercepting an active user's session traffic. Stealing the session token and identifying and intercepting when the user attempts to log out of the application. The threat actor can then intercept and drop the DELETE request, which prevents the back end from invalidating the session token, thus keeping it alive until secondary controls, such as time-based de-authentication occur. Although this example is done with provided testing accounts, and the set up for traffic interception is simplified due to a testing environment, a motivated threat actor only needs to overcome the hurdle of identifying and intercepting requests made by the user to be able to then exploit this vulnerability. As seen below, the threat actor will have permissions available to the token that is compromised.

Step 1: Set up Burp Suite and login

1. Configure Firefox to route traffic through the Burp Suite proxy. Then Login as a privileged user (aadmin).

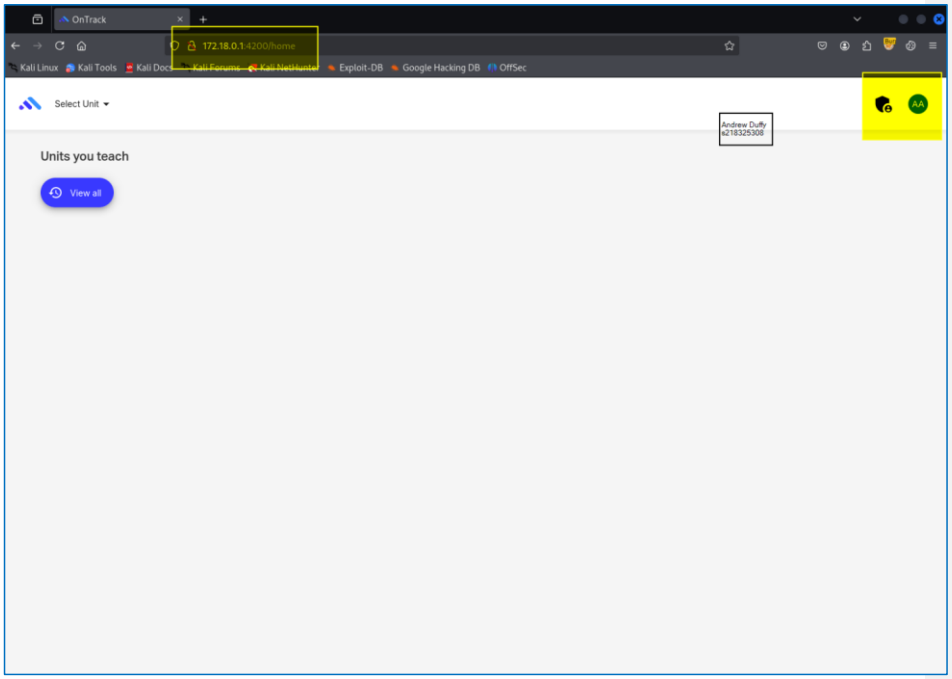


Figure – Logged in as aadmin in browser

1. Configure Firefox to route traffic through the Burp Suite proxy. Then Login as a privileged user (aadmin).

Step 2: Simulate intercepting a logout request

1. Open Burp Suite and set to intercept traffic. Then log out of the privileged user and intercept the DELETE request. Save the auth-token details.

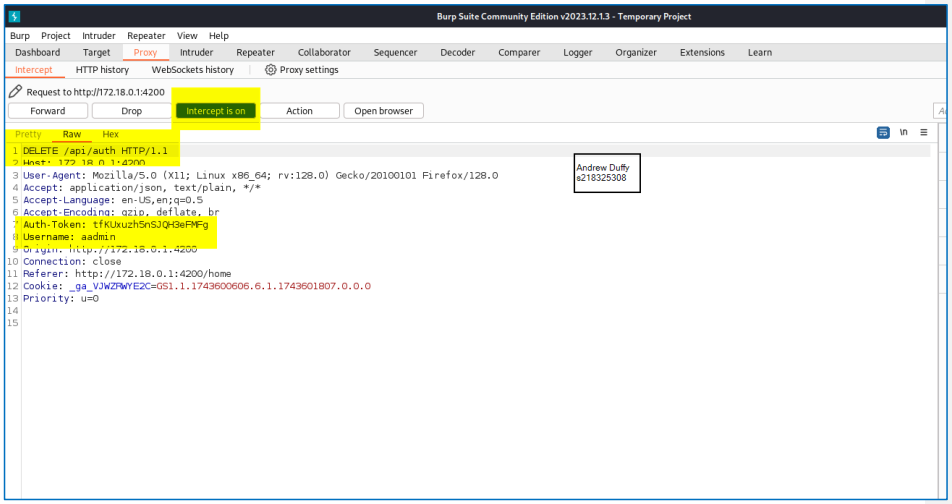


Figure – Intercepted DELETE request during logout.

2. Drop that request using Burp Suite. This will prevent it being transmitted and actioned. IE. The request to delete the authentication token will never be completed.

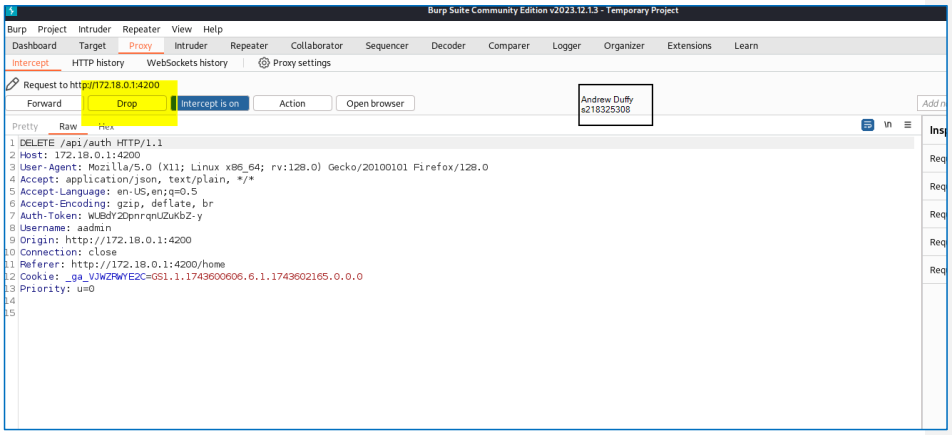


Figure – Drop the request

Step 3: Exploit the token

1. *The following is one example of how this token can be used, to demonstrate the criticality of the vulnerability should it be fully exploited.* The token can now be exploited by a low-level user for a limited time – this is the same exploitation as seen in the 'Insufficient Session Binding' vulnerability. The malicious user can now make privileged requests and replace their token with the stolen token.

2. For Demonstration - On the unprivileged account, using the Broken Access Control vulnerability, to navigate to the **/admin/units** endpoint. This is a privileged endpoint accessible due to a broken access control vulnerability.

3. Normally the low-level user cannot execute the privileged function of creating a new unit.

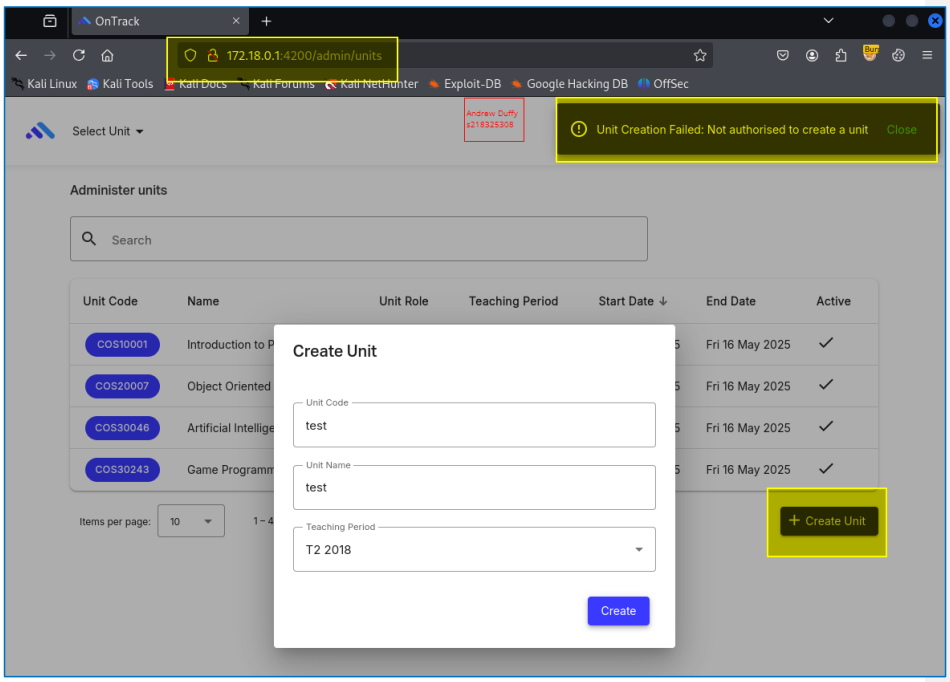


Figure – Student level user attempting to execute admin level function

3. Reattempt to create a new unit and intercept the request in Burp Suite. Complete the required details and turn on the proxy intercept. Intercept the request to execute creating the unit. Note the next image is to evidence the same token was used for the next steps.

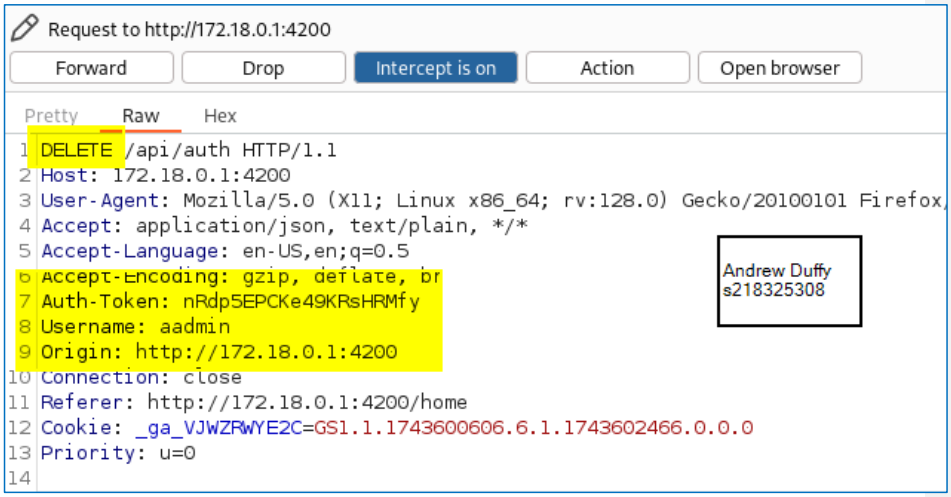


Figure – New Token Grabbed for exploitation

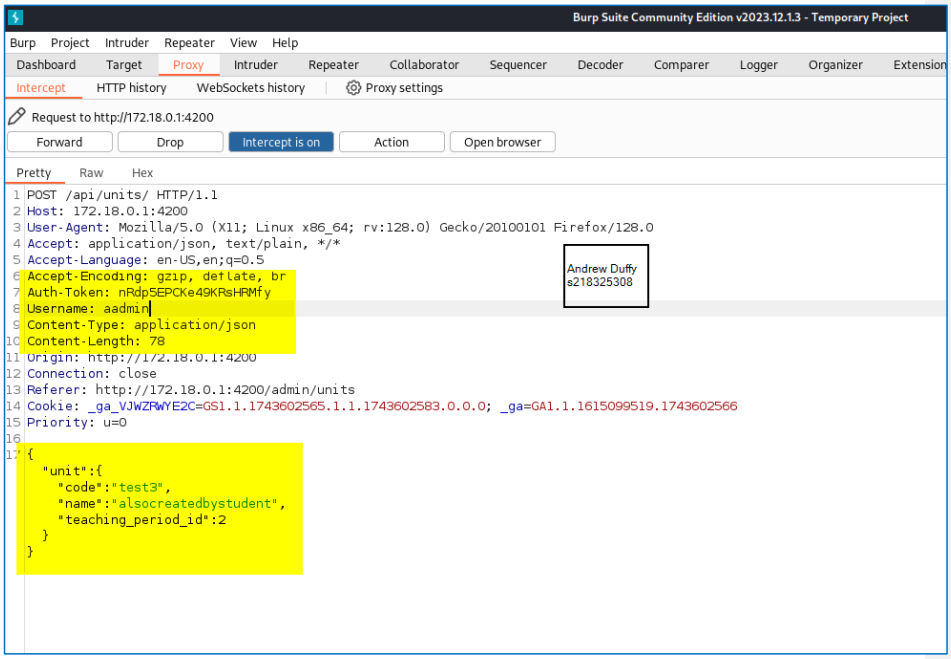


Figure – Creating a new unit, replacing authentication details

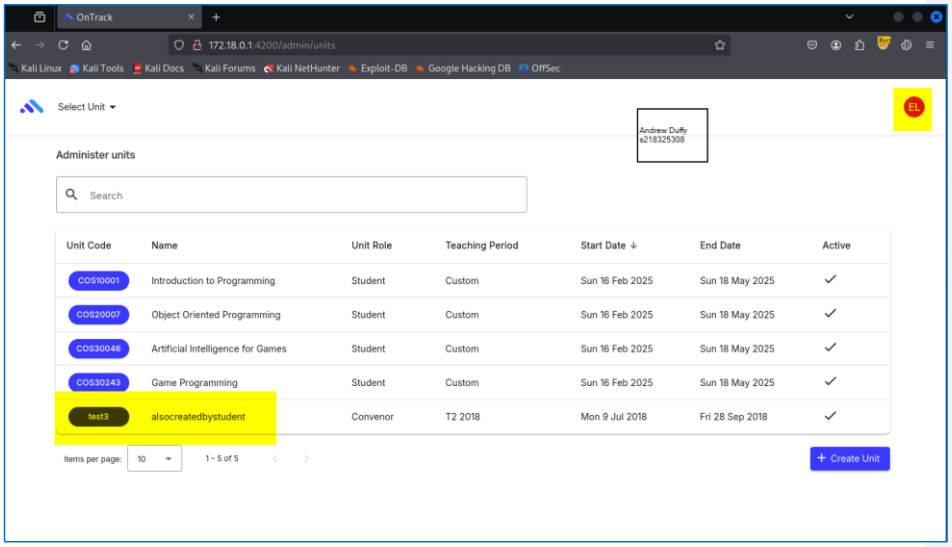


Figure – New unit successfully created by low level user

Remediation Advice

The application does not enforce strong session binding between the client and session token, which allows for session token reuse from unauthorised environments.

Mitigations:

- Implement session binding mechanisms by associating session tokens with client-specific attributes (e.g. IP address, User-Agent, device fingerprint) and validating them on each request.
- Disable Web Browser Cross-Tab Sessions – once a user has logged in and a session has been established a user must re-authenticate if a new web browser tab or window is opened against the same web application.

References

- OWASP Top 10 – Broken Access Control
- OWASP Session fixation
- OWASP Session Managment Cheat Sheet

Contact Details

Name/Teams: Andrew Duffy
Email: s218325308@deakin.edu.au

Pentest Leader Feedback.

Nicholas Krcevianc

- The document is perfect and valid

- The only thing that I would add before the first step in evidence it a general description of what the attack entails, and how this is an internal attack using credentials given to us. So the Ontrack team understand the type of attack this is. - Quick breakdown added before step 1.
- Perfect Done

6.4: Session Hijacking - Insufficient Session Binding

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Andrew Duffy	AppAtack	Pen-Tester	OnTrack Web App	Filipe Oliveira	No

Was this Finding Successful?
Yes

Finding Description

A Session Hijacking vulnerability was discovered in the OnTrack web application as a result of insufficient session binding. Session binding is the process of associating a session token with specific client attributes such as IP address, user's browser, or a device fingerprint. Without strong session binding, a valid session token can be reused across different devices or locations, which increases the risk of session hijacking if a token is intercepted or stolen.

During testing, a valid authentication token and username captured using Burp Suite, from an active privileged user session (convenor) were inserted into an intercepted request made by a lower-privileged user (student). The request was to carry out privileged user actions and is blocked by default for the low-level user. However, replacing the low-level token with the privileged token and forwarding the request, it was accepted and processed successfully by the application.

This vulnerability demonstrated that no additional validation was in place to bind the session token to the original user's environment. This vulnerability can be exploited by an attacker who successfully gains access to an active session token (e.g. via Cross Site Scripting (XSS), social engineering, or sniffing). Insufficient session binding provides an opportunity for threat actors to reuse the stolen token from another browser or device and impersonate the original user. During testing the stolen token enabled the low-level user to conduct privileged actions but was unable to take over the privileged session at this time.

Due to the common nature of this vulnerability, it should be expected that threat actors will attempt to search for and exploit it with a Moderate likelihood of occurring. Exploitation allowed a threat actor to perform some administrative actions but was not able to fully comprise the privileged session. It was also noted that compensating controls appeared to be in place, which included a time-based session token rotation which greatly limits the window of opportunity for exploitation and reducing the impact to Significant, giving an overall Risk of Medium.

Risk Rating - Medium

Impact: Significant
Likelihood: Moderate

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity.	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally.	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

While the test did not demonstrate full session takeover, the results indicate a significant weakness in the application's session handling. The lack of session binding allows a valid session token to be reused across different browsers or devices simultaneously. If a session token is successfully intercepted or obtained by a threat actor, it can then be reused to impersonate the original user and perform any privileged actions available to the original user. Depending on the victim's token, this vulnerability presents a risk of unauthorised access to sensitive data or privileged functionality. Although time-based session expiry mechanisms may reduce the window of opportunity, the absence of session binding increases the risk of token reuse and may compromise the confidentiality and integrity of the data within the application.

Affected Assets

OnTrack Web Applicaiton Authentication/Session Token implementation. Any API, endpoint of functionality that relies on a valid authentication token is potentially vulnerable.

Evidence

Step 1: Set up separate browser instances

1. This requires two isolated browser contexts. one for unprivileged user (student_1), one for privileged user (aadmin). Start Burp Suite. Open Firefox and set traffic to be forwarded through the Burp proxy. This is one of the isolated browsers

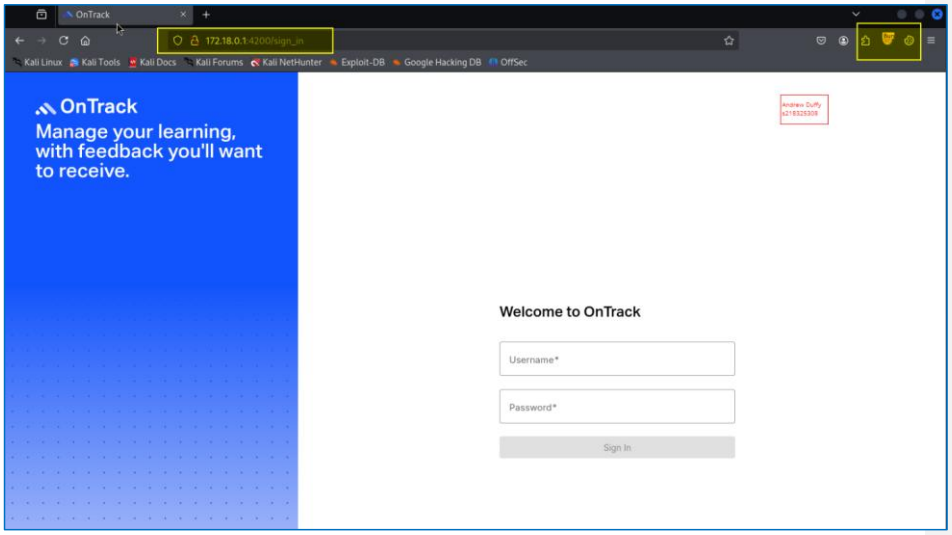


Figure – Browser 1, Firefox sending traffic through Burp Suite

2. Navigate to the proxy tab in Burp Suite and select open browser. This will open a chromium-based browser tied to Burp Suite. This will act as the second isolated browser instance.

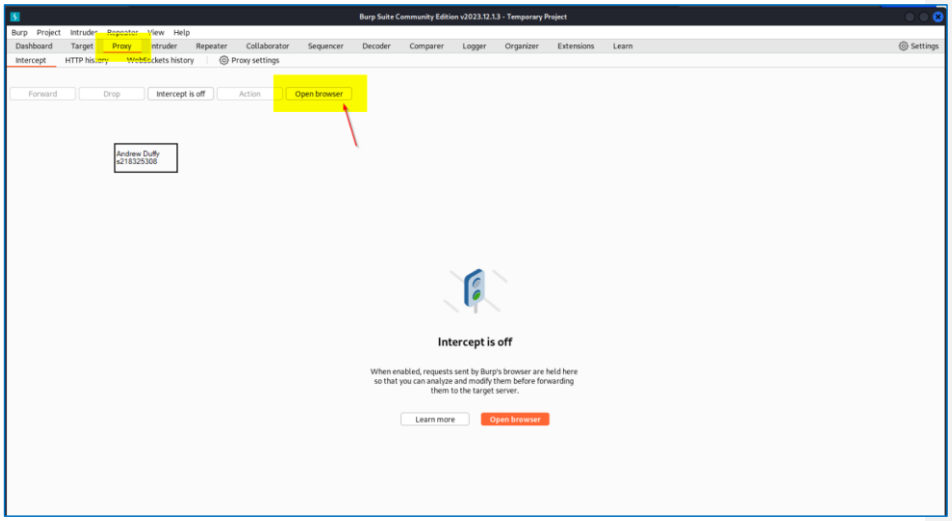


Figure – Open Burp inbuilt browser

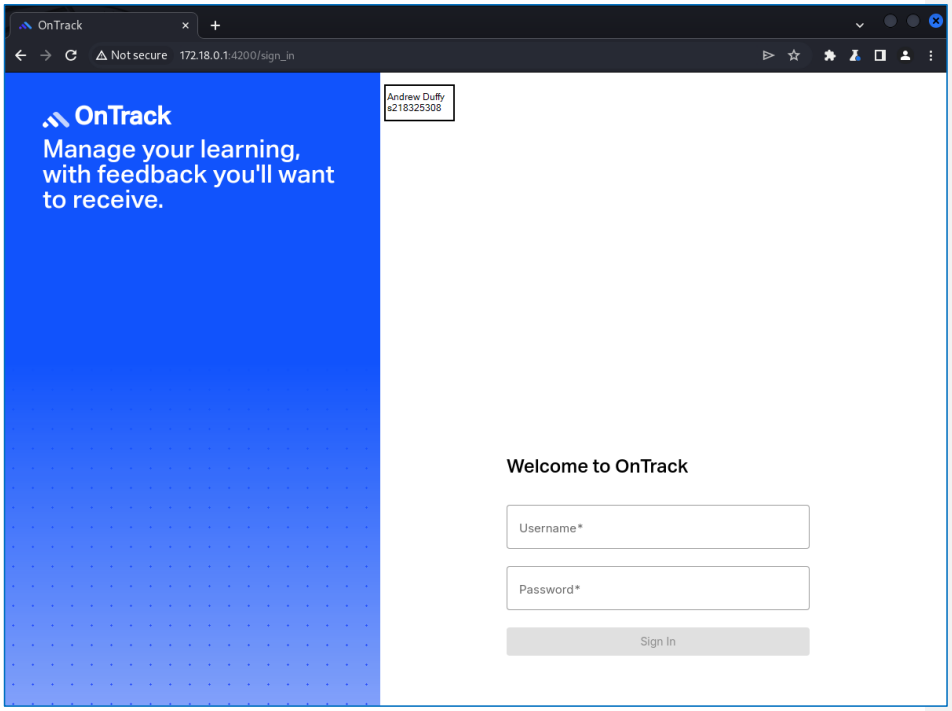


Figure – Burp Suite inbuilt browser

Step 2: Login to each account

1. Login to two separate accounts using the individual browsers. In the Firefox browser as student_1. This will navigate you to /home once authenticated.

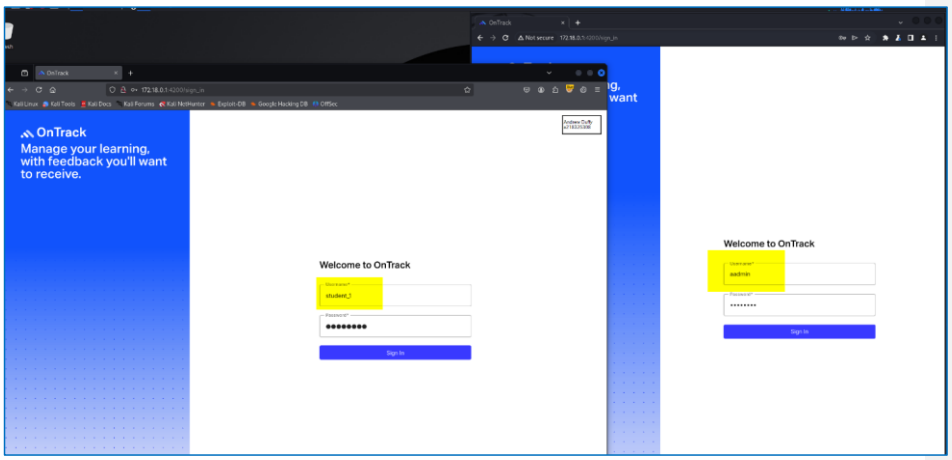


Figure – Set up both users

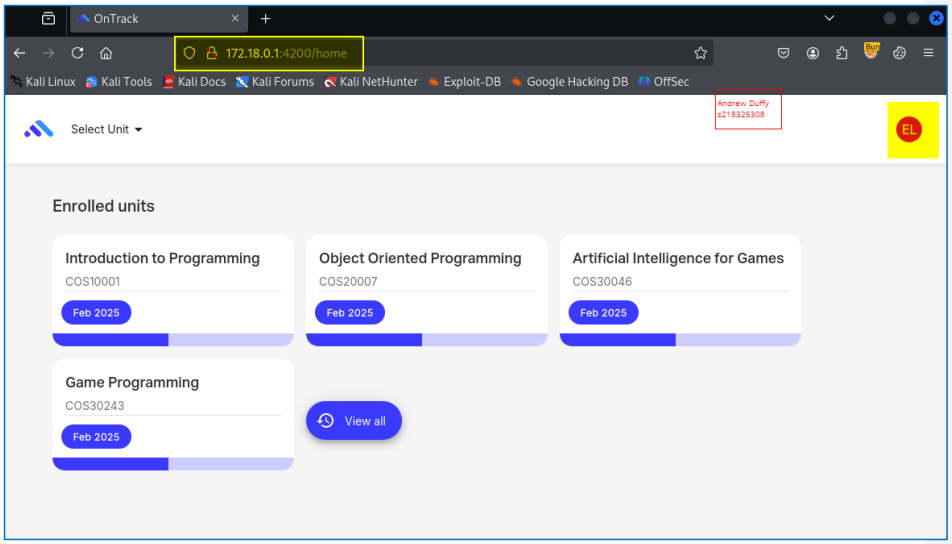


Figure – Logged in a student_1 a student level user.

2. Login to aadmin using the Burp browser. This will present the admin home page.

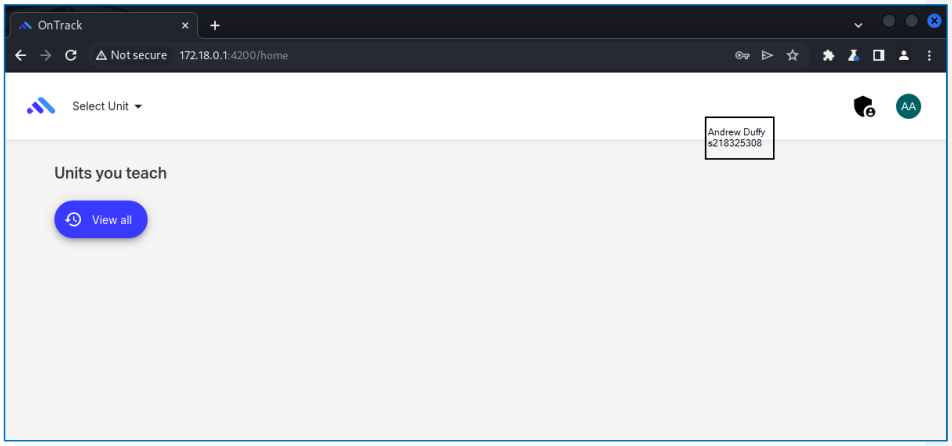


Figure – Logged in as aadmin in browser 2

Step 3: Emulate capturing privileged token

1. Open Burp Suite and review the HTTP history. Find the authorisation request and save the token that is provided in the response.

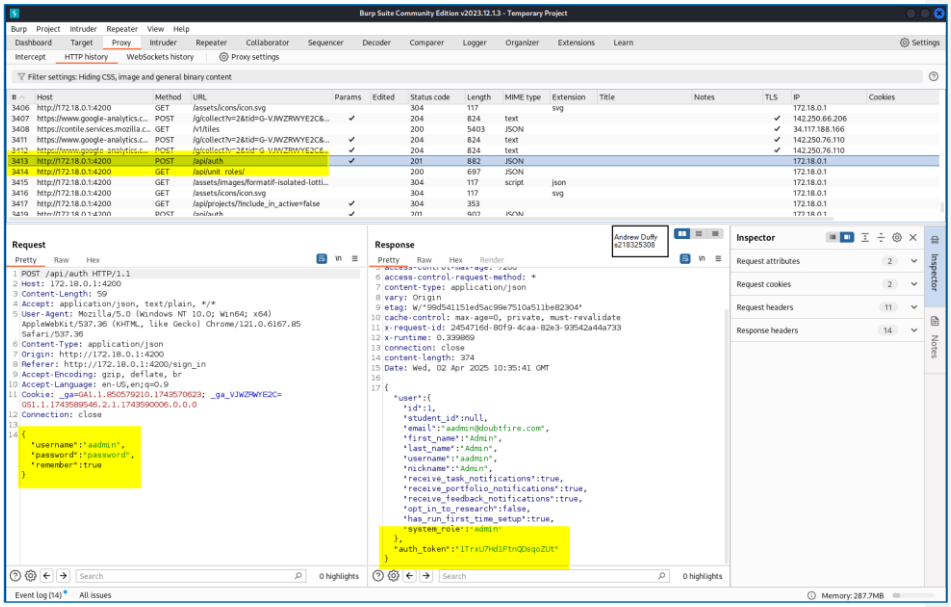


Figure – 'Obtaining' victim token

Step 4: Execute attack

1. For Demonstration - On the unprivileged account, using the Broken Access Control vulnerability, to navigate to the **/admin/units** endpoint. This is a privileged endpoint accessible due to a broken access control vulnerability.

2. Normally the low-level user cannot execute the privileged function of creating a new unit.

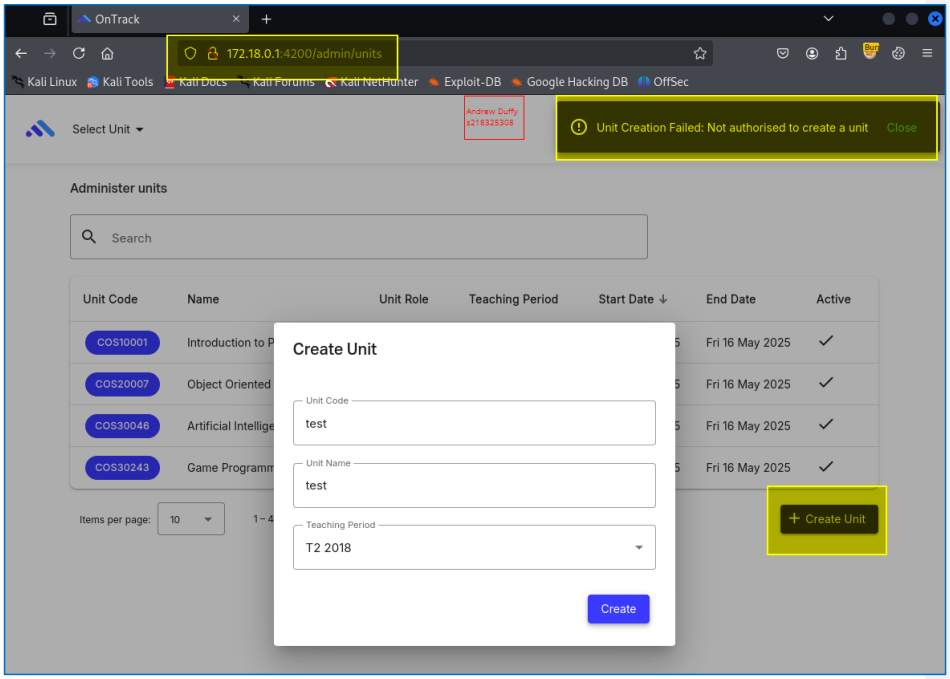


Figure – Student level user attempting to execute admin level function

3. Reattempt to create a new unit and intercept the request in Burp Suite. Complete the required details and turn on the proxy intercept.

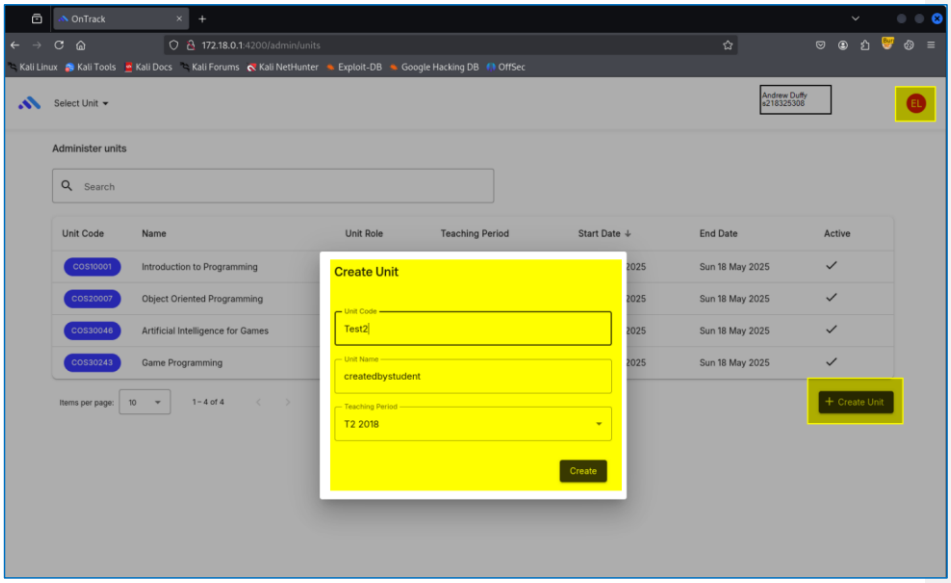


Figure – Fill out required details

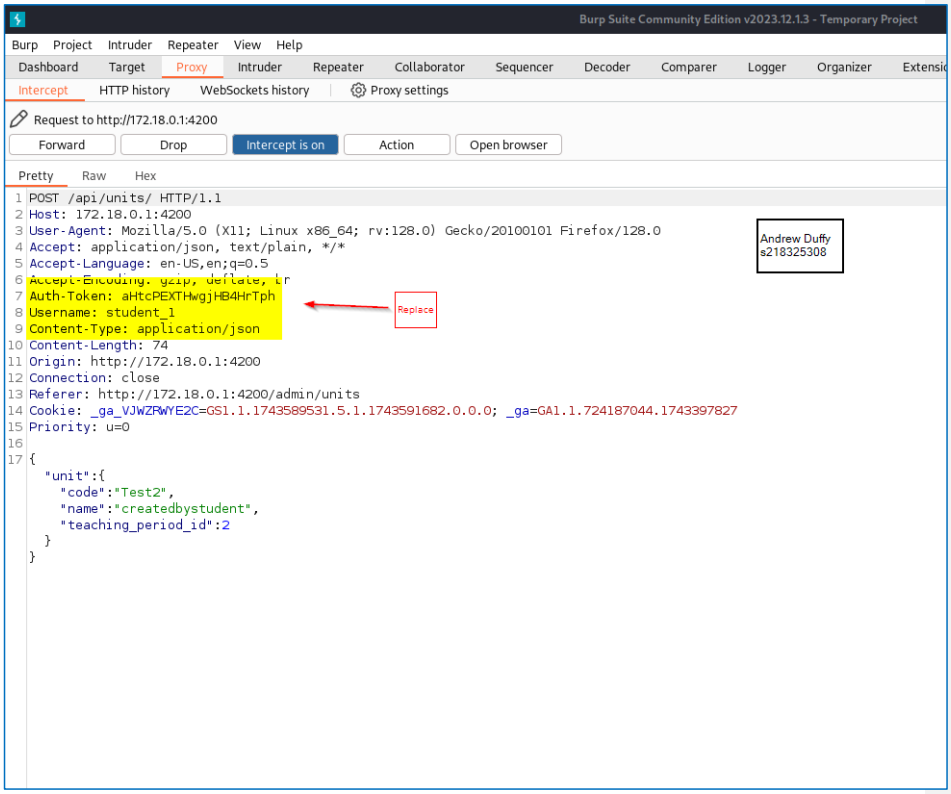


Figure – Review the intercepted request. Identify Authentication token and username.

4. Replace authentication token and username with ‘stolen’ token and username.

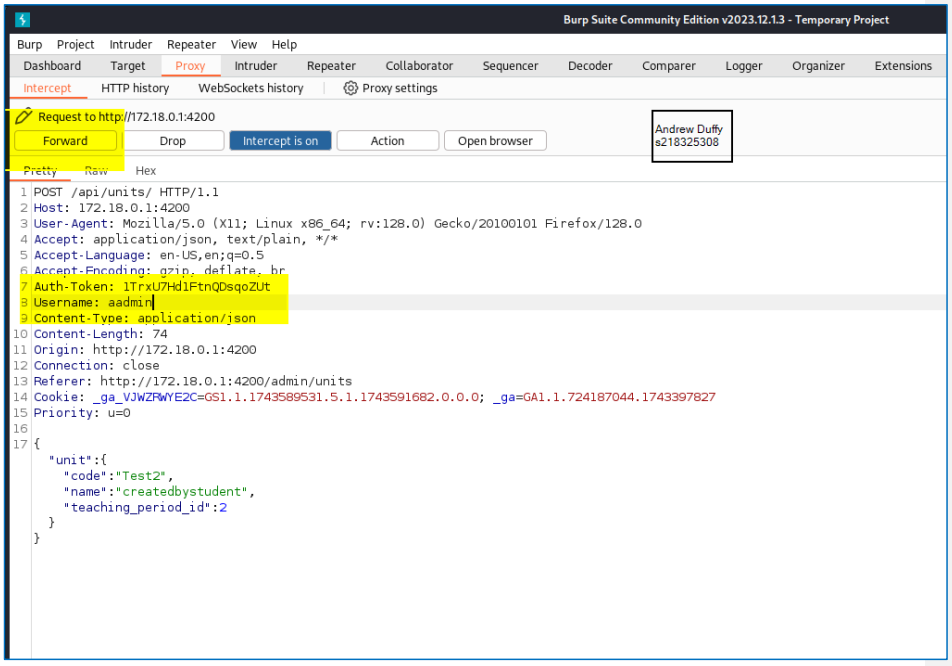


Figure – Fill out required details

5. Forward the request – this will successfully add the newly created unit, validating the session token can be used outside the original context.

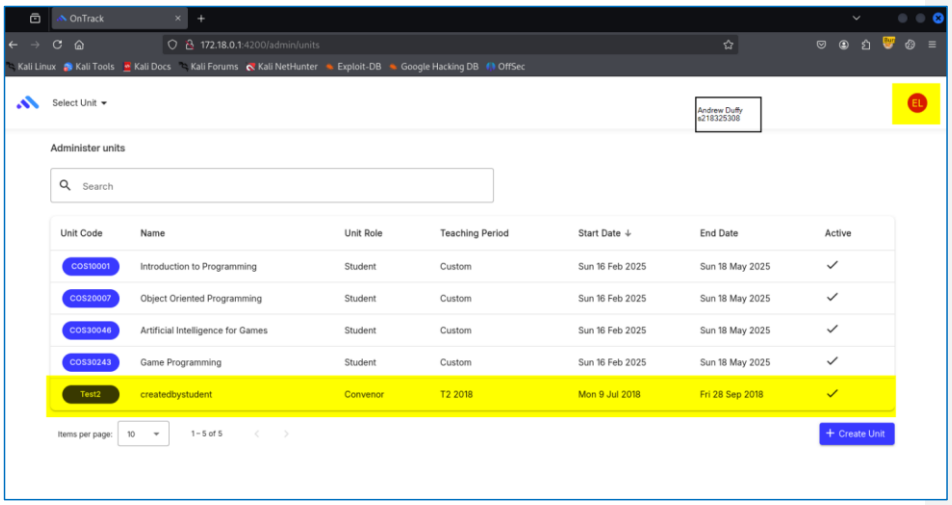


Figure – New unit added by low level user

Remediation Advice

The application does not enforce strong session binding between the client and session token, which allows for session token reuse from unauthorised environments.

Mitigations:

- Implement session binding mechanisms by associating session tokens with client-specific attributes (e.g. IP address, User-Agent, device fingerprint) and validating them on each request.
- Disable Web Browser Cross-Tab Sessions – once a user has logged in and a session has been established a user must re-authenticate if a new web browser tab or window is opened against the same web application.

References

- OWASP Top 10 – Broken Access Control
- OWASP Session Managment Cheat Sheet

Contact Details

Name/Teams: Andrew Duffy
Email: s218325308@deakin.edu.au

Pentest Leader Feedback.

Great find! Excellent work.
Filipe Oliveira S222478779@deakin.edu.au

6.5: Exploitable Cross-Origin Resource Sharing (CORS) Configuration

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Fahad Alshamri	Ontrack	Pentester	AppAttack	**Nicholas Krcevinac**	NO

Was this Finding Successful?
Yes

Finding Description

The OnTrack web application located at http://172.18.0.1:4200 is configured to respond with Access-Control-Allow-Origin: *, which allows cross-origin requests from any domain. This insecure CORS policy can be exploited by a remote attacker to retrieve API responses using JavaScript from a malicious page hosted on another domain.

Risk Rating

Impact: Significant
Likelihood: High

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity.	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally.	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

An attacker could host a malicious site that silently performs unauthorized API requests to the OnTrack server on behalf of a logged-in user or retrieve sensitive API responses if endpoints are exposed. This could result in data leaks, unauthorized access, or session manipulation if tokens are mismanaged.

Affected Assets

- OnTrack Web Frontend (http://172.18.0.1:4200)
- OnTrack API endpoints (/api/users)

Evidence

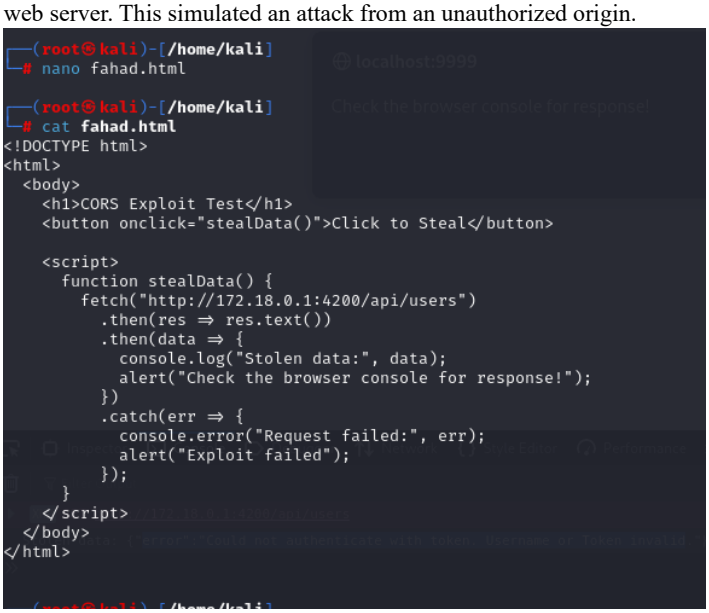
CORS Proof of Concept (POC)

Step 1: Confirm CORS Policy

To determine whether CORS was improperly configured, a simple `curl` request was issued to the main application URL using the `-I` flag to fetch headers. The response included the header `Access-Control-Allow-Origin: *`, which indicated the server was allowing requests from any domain without restrictions.

Step 2: Create Malicious HTML Page

A malicious HTML file (`fahad.html`) was created locally. The file contained a script using the JavaScript Fetch API to send a GET request to the endpoint `/api/users` of the OnTrack web server. This simulated an attack from an unauthorized origin.



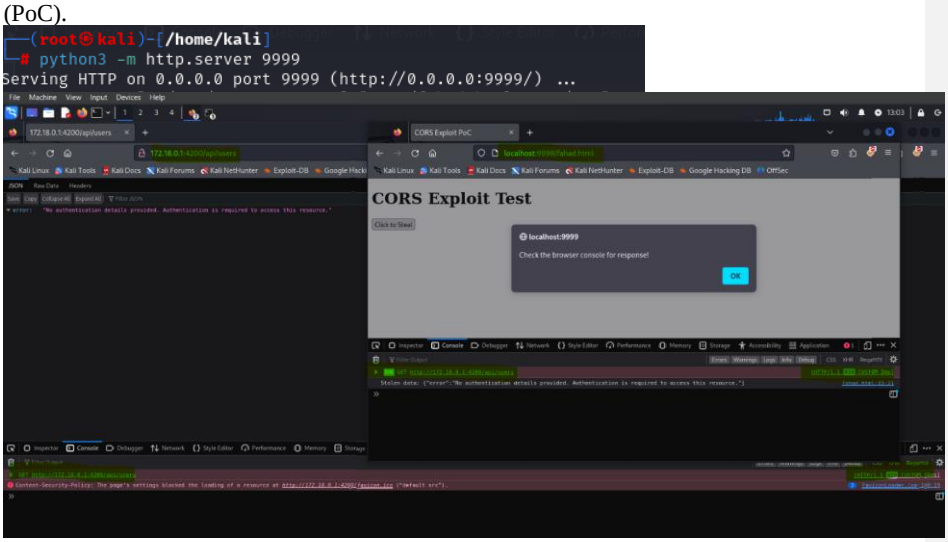
```
┌──(root㉿kali)-[/home/kali]
└─# nano fahad.html

┌──(root㉿kali)-[/home/kali]
└─# cat fahad.html
<!DOCTYPE html>
<html>
  <body>
    <h1>CORS Exploit Test</h1>
    <button onclick="stealData()">Click to Steal</button>

    <script>
      function stealData() {
        fetch("http://172.18.0.1:4200/api/users")
          .then(res => res.text())
          .then(data => {
            console.log("Stolen data:", data);
            alert("Check the browser console for response!");
          })
          .catch(err => {
            console.error("Request failed:", err);
            alert("Exploit failed");
          });
      }
    </script>
  </body>
</html>
```

Step 3: Host and Load the File & Trigger the Exploit

The fahad.html file was accessed from a different origin (localhost:9999), simulating a malicious domain. When the button was clicked, the browser issued a request to http://172.18.0.1:4200/api/users. The OnTrack server responded with a 419 error and JSON response indicating “No authentication details provided,” confirming that the server processed the cross-origin request. While no sensitive data was retrieved, the fact that a response was returned confirms the vulnerability. This is presented as a Proof of Concept (PoC).



Explanation:

This demonstration confirms a Cross-Origin Resource Sharing (CORS) misconfiguration in the OnTrack web application. The crafted fahad.html page, served from http://localhost:9999, successfully initiated a cross-origin GET request to http://172.18.0.1:4200/api/users. Although the server responded to the request, the response returned an authentication error due to the absence of a valid token. While no sensitive data was exfiltrated, the fact that the server accepted and responded to a cross-origin request from an untrusted origin indicates a CORS misconfiguration. This proves the vulnerability exists and should be labeled as a Proof of Concept (PoC).

In a real-world scenario, if an attacker can obtain or guess a valid session token (e.g., through XSS, phishing, or a malicious extension), they could abuse the misconfigured CORS policy to access sensitive API endpoints from a malicious domain — effectively bypassing the same-origin policy.

Remediation Advice

To mitigate this vulnerability, the server's Cross-Origin Resource Sharing (CORS) policy must be restricted. Specifically, the Access-Control-Allow-Origin header should not be set to * in production environments. Instead, it should explicitly list trusted domains, such as https://yourdomain.com, which are permitted to interact with the API. Additionally, if

credentials (like cookies or tokens) are used in requests, Access-Control-Allow-Credentials should be set to true, and Access-Control-Allow-Origin must not be a wildcard. Implementing proper CORS rules ensures that only authorized frontends can interact with backend services, preventing malicious third-party origins from making unauthorized API calls.

References

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
https://cwe.mitre.org/data/definitions/942.html
https://blog.securelayer7.net/owasp-top-10-security-misconfiguration-5-cors-vulnerability-patch/

Contact Details

Name: Fahad Alshamri.
Email: falshamri@deakin.edu.au

Pentest Leader Feedback.

Nicholas Krcevinac:

- **Inconsistent Address Reference**: There is a contradiction between the documentation and the visual evidence. In your write-up, you state that the malicious HTML file sends requests to http://172.18.0.1:4200, yet in the screenshot provided, the exploit is executed from http://localhost:9999/fahad.html. This mismatch can confuse the OnTrack Team. Please update the screenshot or the written steps to match your actual testing environment and ensure consistency across your evidence and description.
- **Text Color Formatting**: Avoid using blue text for headers or body text in formal documentation. It affects readability and professionalism. Stick to default formatting or use black text for a cleaner, consistent, and professional appearance. I would recommend to look in the "4.Ready for Final Report" for what we are looking for in documentation and follow the AppAttack Findings Checklist.docx.

  Also, if you do this is future reports there is a high chance that we will fail you straight away.

- **Proof of Concept Clarity**: While your script executes correctly and reaches the intended endpoint, the response returned an **authentication error**. This means sensitive data was not exfiltrated, but the server did respond — indicating a **CORS misconfiguration**. Given this, it would be more accurate to label your finding as **proof of concept (PoC)**. Please make this clear in the description to avoid overrepresenting the exploit's impact and in your evidence.
- **Additional Suggestions**:
  - You could enhance your finding by checking if any endpoints do return sensitive data when a valid token is present.
  - Include a clearer conclusion section or recommendation summary for better readability.

6.6: Insecure Token Exposure via Client-Side Storage and HTTP Headers

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Filipe Oliveira	OnTrack	Pen-Tester	AppAttack	Darryl Ooi	No

Was this Finding Successful?
Yes

Finding Description

Upon successful login to the OnTrack platform, the application issues a persistent authentication token that is sent in every subsequent API request via HTTP headers. This token provides full user session access without needing credentials again. The application is served over unencrypted HTTP meaning this token is transmitted in plaintext and can be intercepted using tools such as Wireshark on the same network.

Risk Rating

Impact: Significant
Likelihood: High

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity.	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally.	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

If an attacker gains access to a user's authentication token, they can fully impersonate the victim without needing a password. This could lead to exposure of academic records, assignment submissions and may allow unauthorised access to admin modifications. If this occurred on a live, production grade system over an open or shared network, it could result in

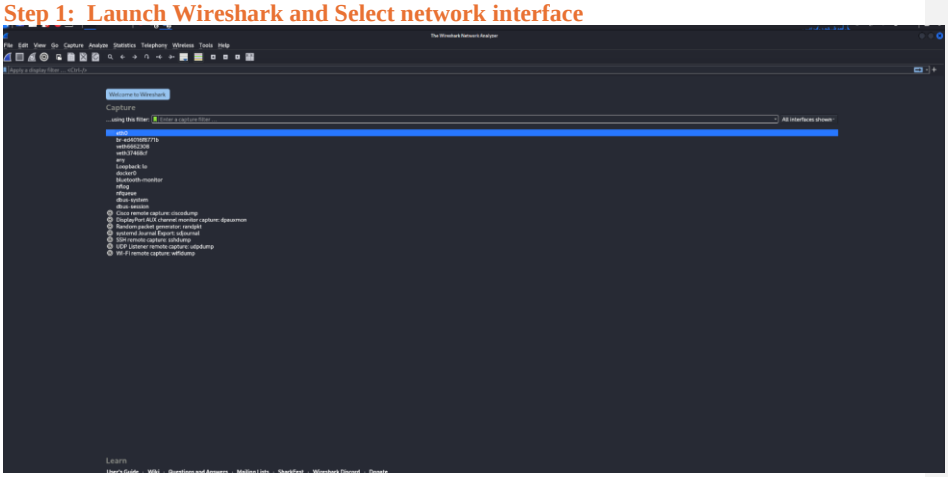
unauthorised access to student information, data privacy violations and erosion of trust in the system's security.

Affected Assets

- Ontrack web application
- API endpoints using Authentication tokens
- Student session tokens.

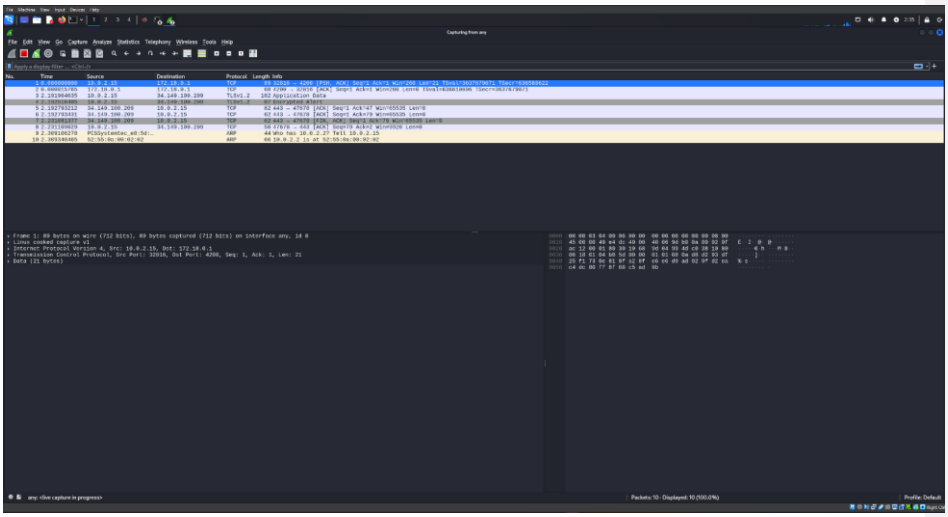
Evidence

Step 1: Launch Wireshark and Select network interface



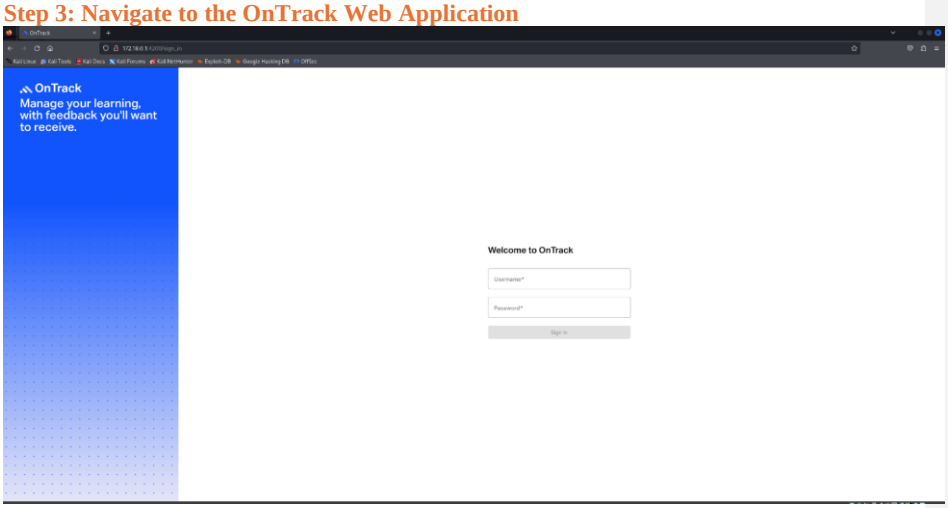
Open Wireshark and select the interface labeled" any" to monitor all the network traffic.

Step 2: begin capturing packets

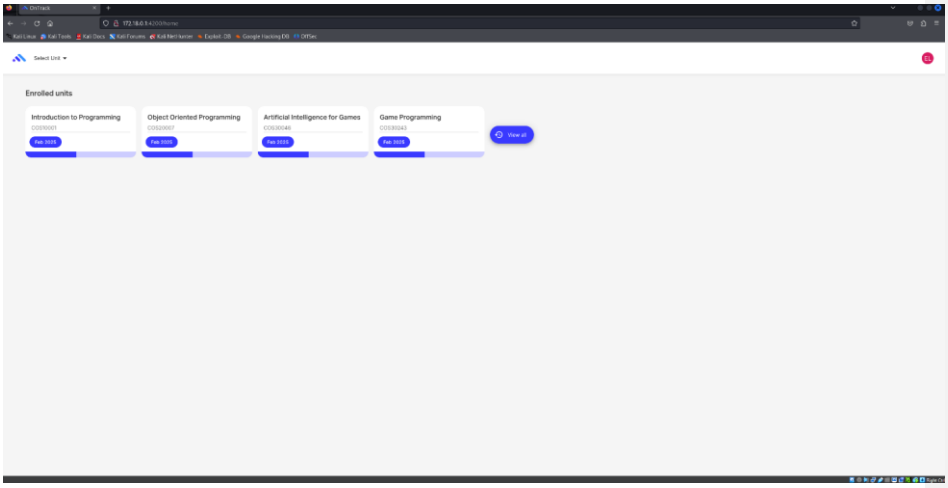


Click the blue fin under the top left File menu to begin live packet capture.

Step 3: Navigate to the OnTrack Web Application

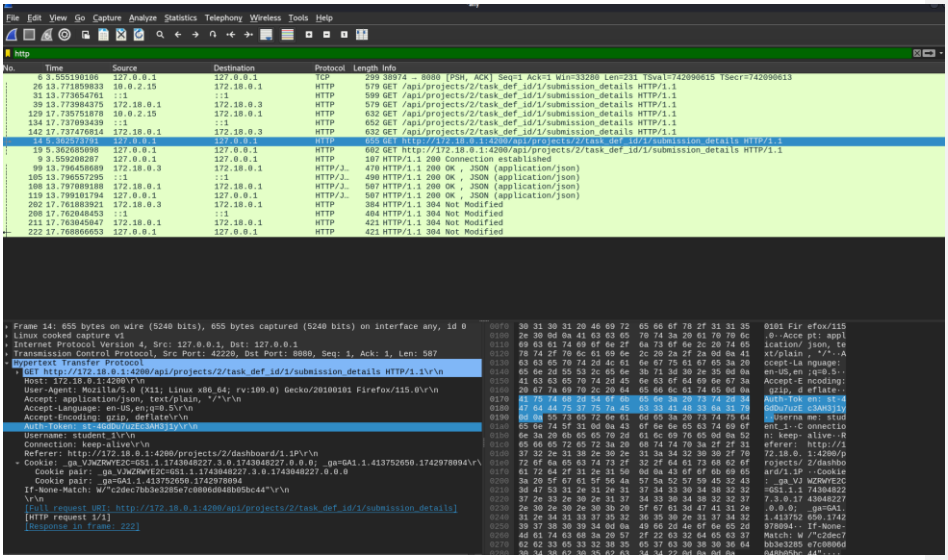


Step 4: Login with student credentials



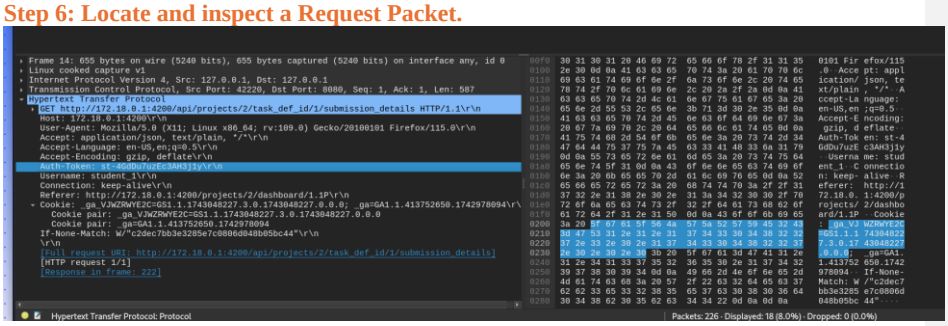
Login with student_1 and password you can also do anything on ontrack, like clicking on a subject.

Step 5: Apply HTTP display filter



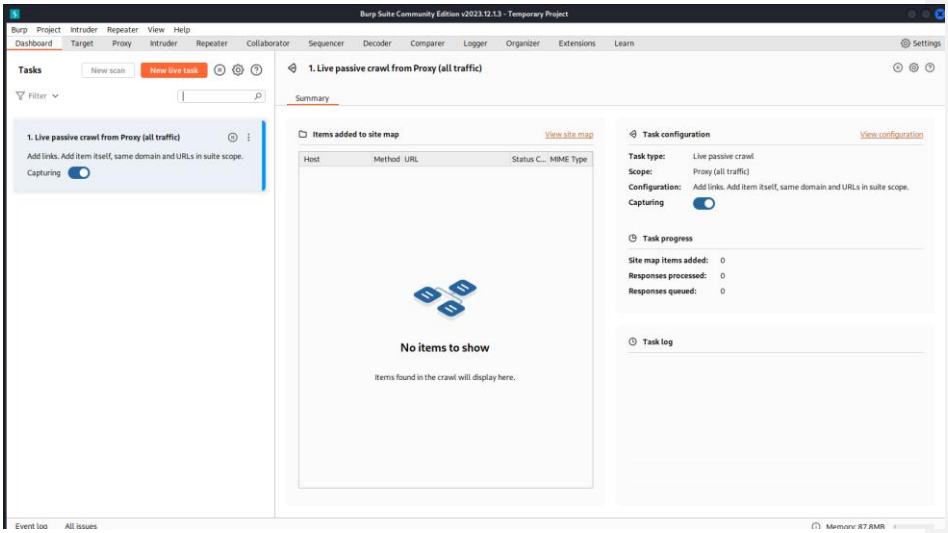
Type http in the filter bar at the top

Step 6: Locate and inspect a Request Packet.

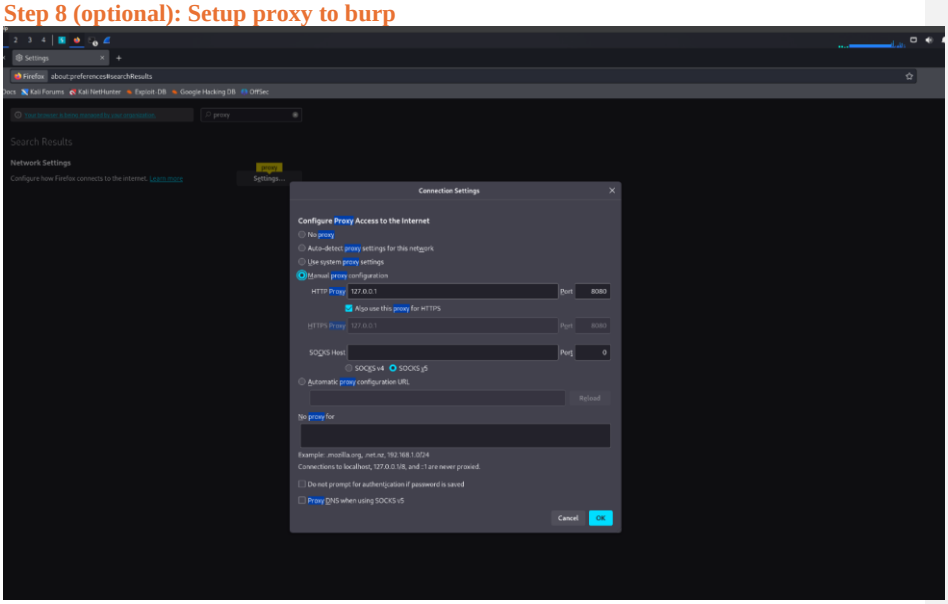


Click on any packet and find the Hypertext protocol subheading and click expand, if it is a packet that has been sent after your login it will contain cookie session data and the authentication token for the login Aswell as the username

Step 7(optional): Setup burp suite

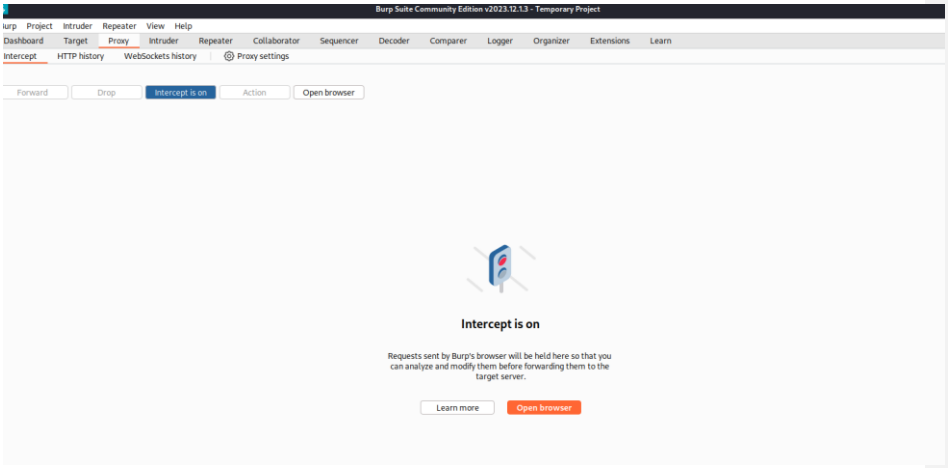


Step 8 (optional): Setup proxy to burp



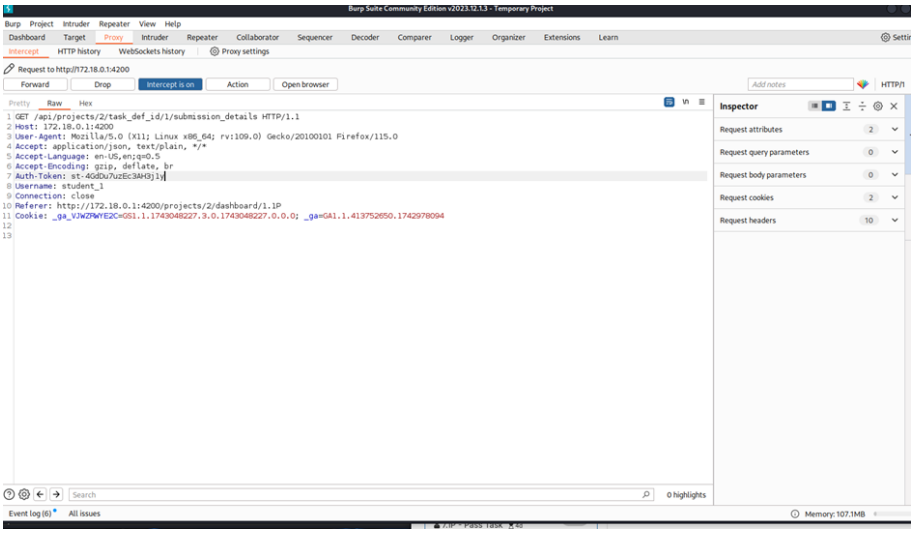
Go to settings in your browser and turn the proxy and input the proxy ip and port

Step 9(optional): Locate and inspect a Request Packet.



Turn on intercept and go ontrack.

Then once you load into OnTrack there should be information that allows each section of ontrack to load before you then “forward” it to the server. As you can see below once i logged in i can see the packets being sent to the server and me, with all the information with auth token and cookies.



Remediation Advice

- Enforce HTTPS (TLS) to encrypt all network communication and prevent sniffing.
- Avoid sending long lived tokens in headers over unsecured connections.

- Move to only HTTPonly secure cookies to store tokens where java script cannot access them.
- Implement token expiration and refresh mechanisms.

References

OWASP Session Management Cheat Sheet
https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

Wireshark Official Website (Download & Docs) https://www.wireshark.org/

Burp Suite Repeater Documentation

https://portswigger.net/burp/documentation/desktop/tools/repeater

Okta Developer Blog – Why You Should Always Use HTTPS

https://developer.okta.com/blog/2019/08/22/why-you-should-always-use-https

OWASP Cheat Sheet Series Main Page (Optional for extra references)
https://cheatsheetseries.owasp.org/

Contact Details

Filipe Oliveira s222478779@deakin.edu.au

Pentest Leader Feedback.

Good work!

6.7: Publicly Accessible API Documentation via Swagger

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Filipe Oliveira	OnTrack	Pen-Tester	AppAttack	Wahidullah Hashimi	No

Was this Finding Successful?
Yes

Finding Description

The swagger-generated Api documentation is publicly accessible via the unauthenticated endpoint: http://172.18.0.1:4200/api/swagger_doc.json this exposes the entire backend Api structure, including sensitive internal routes like PUT /api/users/{id}and GET /api/users
By utilizing this documentation, an attacker can discover hidden API routes not exposed through the front end of the website. Although the app prevents unauthorized role change and other protected actions, the ability to submit those requests without proper feedback shows weakaccess controls and poor feedback handling.

Risk Rating

Impact: Major
Likelihood: Moderate

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity.	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally.	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

If an attacker gains access to a user's authentication token, they can fully impersonate the victim without needing a password. This could lead to exposure of academic records, assignment submissions and may allow unauthorised access to admin modifications. If this occurred on a live, production grade system over an open or shared network, it could result in

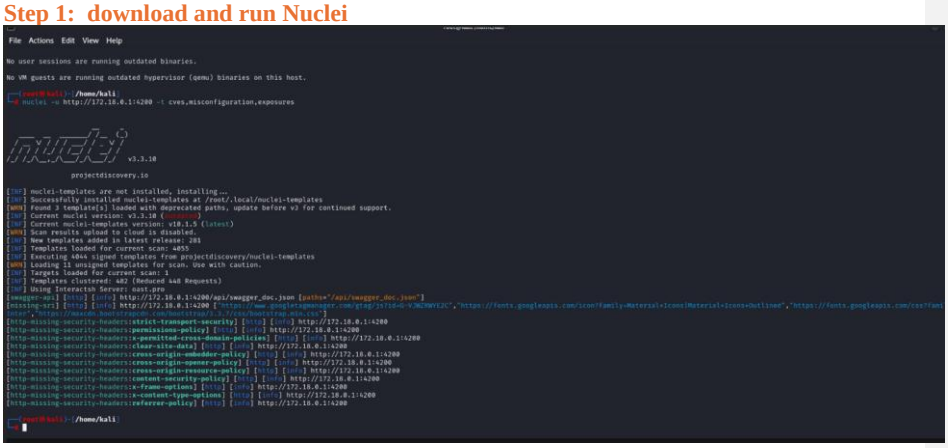
unauthorised access to student information, data privacy violations and erosion of trust in the system's security.

Affected Assets

- GET /api/swagger_doc.json
- All endpoints defined in swagger

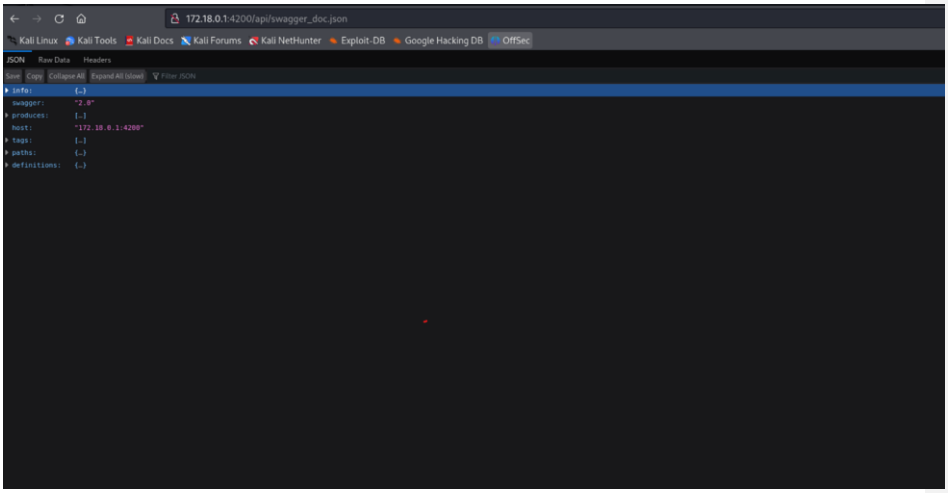
Evidence

Step 1: download and run Nuclei

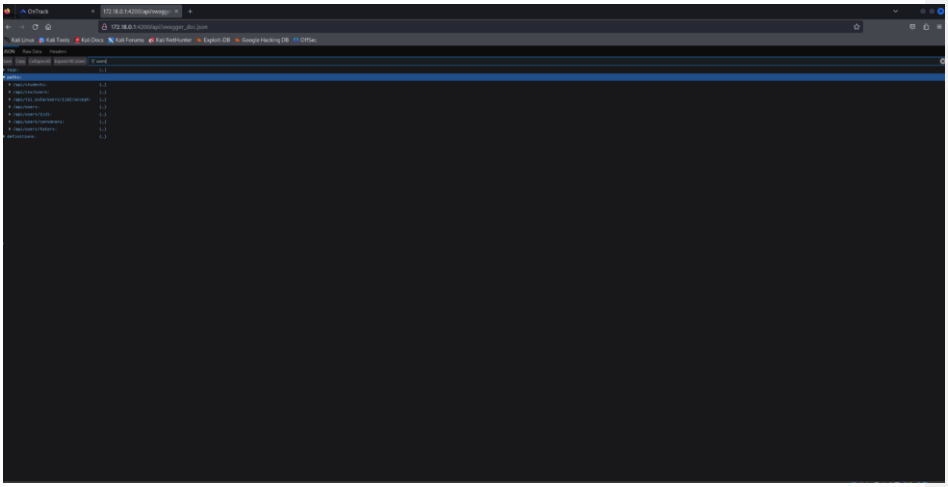


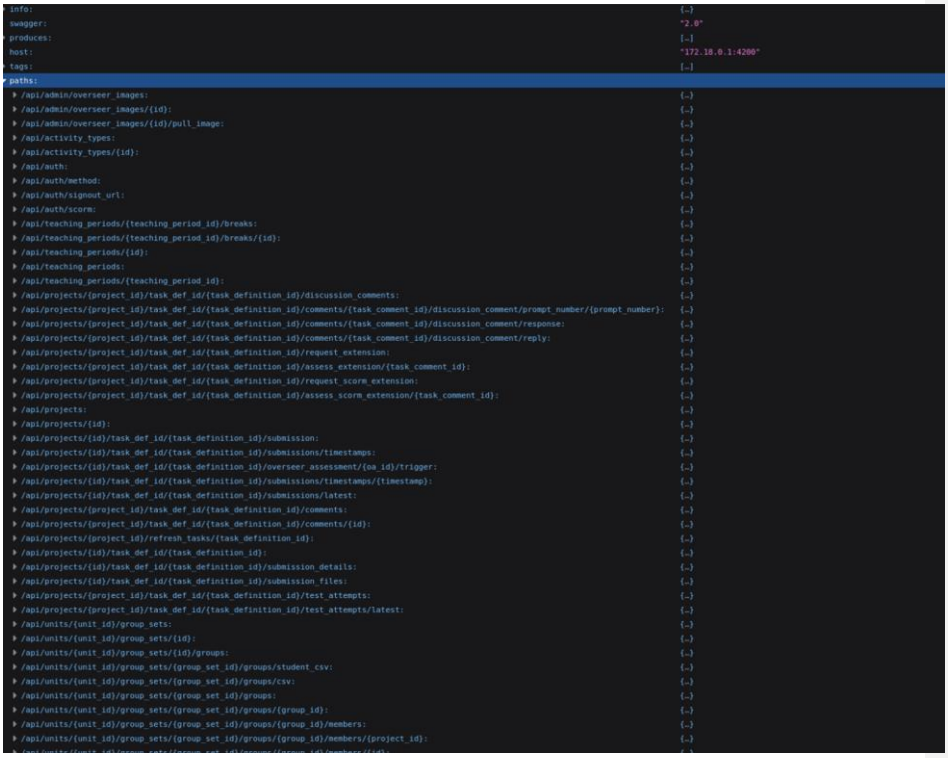
In this above screenshot we can see one of the vulnerabilities is a public API along with its URL to access it

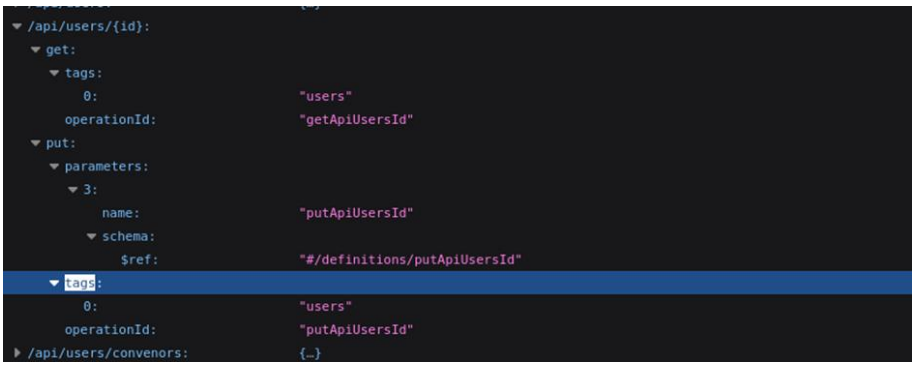
Step 2: Copy and paste the URL in browser



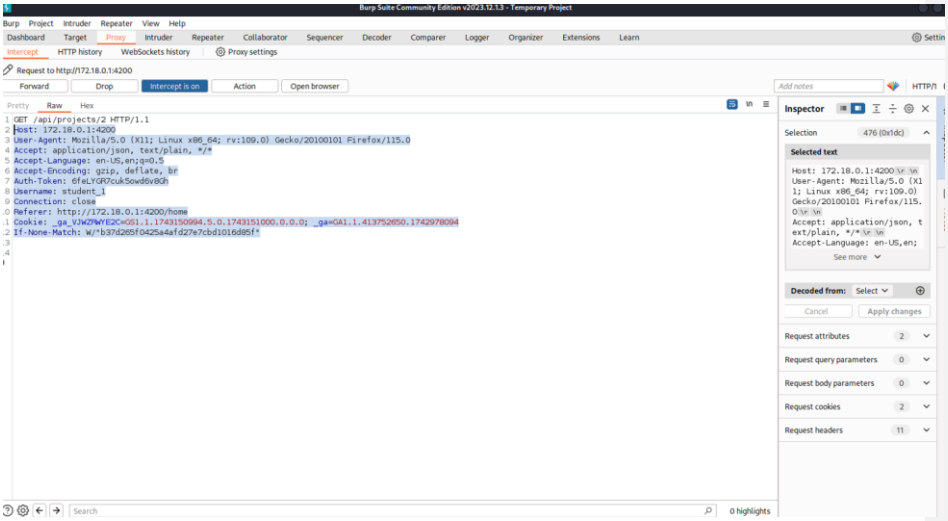
we now have access to the API and can go through it to reveal end points like /api/users





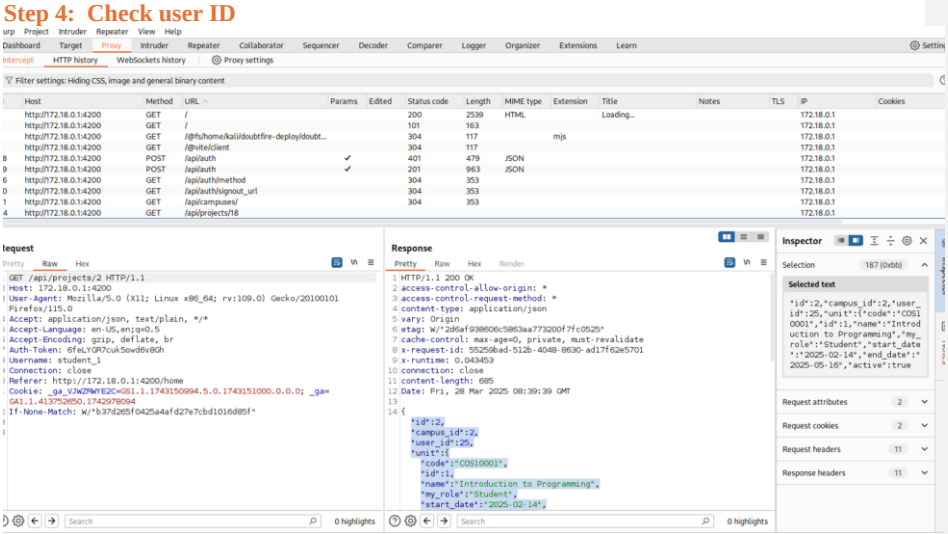


Step 3: Open and setup burp suite



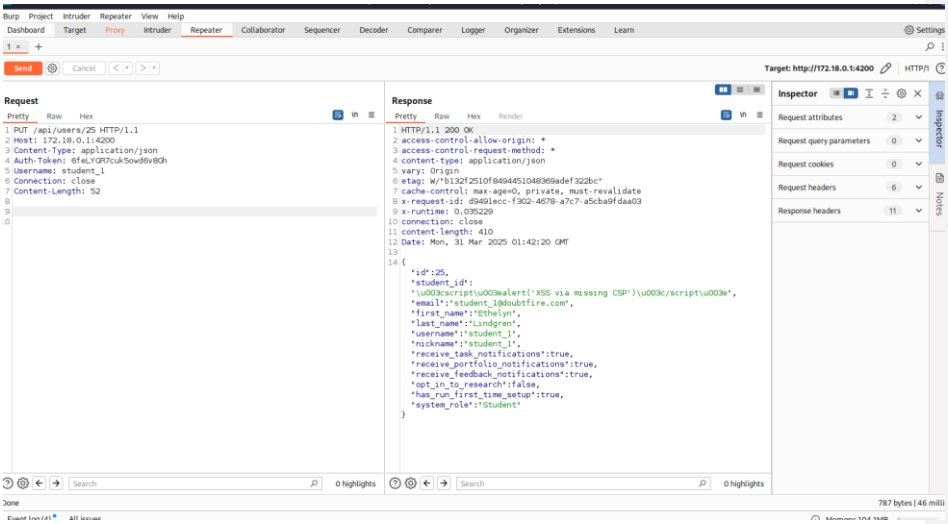
So we login as a student as normal and keep sending the requests through while still intercepting the packets. We take note of our authentication token and cookies.

Step 4: Check user ID

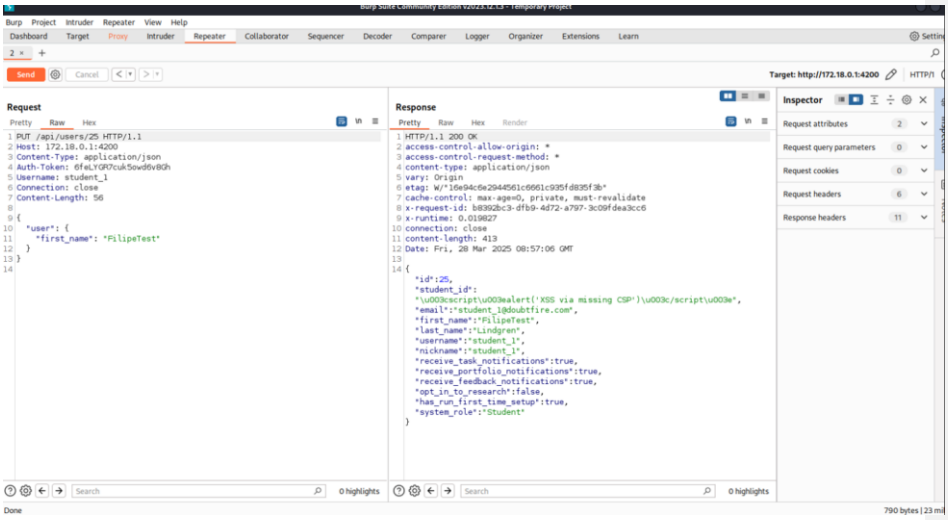


Next go to http history and sift through packets that have /api . This shows me all the fields associated with my user and can correctly note that my user is 25.

Step 5: Go to the repeater tab



As seen above this now shows the information of student one as a whole. Because we know how the backend API is setup, we can push requests and changes to this information.



On the left is the packet that I tested and pushed onto the system, on the right is the confirmation of the server information. (I tried to do system role too, however that seemed to be properly authenticated before making changes, so I could not change that.)

Remediation Advice

- Restrict access to API documentation endpoints like swagger_doc.json

- Require authentication and authorisation before exposing backend routes and parameters
- Implement strict validation on all sensitive user attributes(role_id)
- Ensure proper response code (403 Forbidden) for unauthorised updates.

References

OWASP Session Management Cheat Sheet
https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

Wireshark Official Website (Download & Docs) https://www.wireshark.org/

Burp Suite Repeater Documentation

https://portswigger.net/burp/documentation/desktop/tools/repeater

Okta Developer Blog – Why You Should Always Use HTTPS

https://developer.okta.com/blog/2019/08/22/why-you-should-always-use-https

OWASP Cheat Sheet Series Main Page (Optional for extra references)
https://cheatsheetseries.owasp.org/

Contact Details

Filipe Oliveira s222478779@deakin.edu.au

Pentest Leader Feedback.

Overall, great work! The finding is well-documented with clear steps to follow. However, please ensure consistency in font size and style, as there were some inconsistencies in your report. I have corrected them for you. Additionally, there were a few minor grammatical issues, such as the use of a lowercase “i” in the middle of a sentence and some missing commas, but these have been corrected for you as well.

6.8: Privilege Escalation, tutor accessing admin sites

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Jackson Anton Bouwman	AppAttack	Pen-Tester	OnTrack Web App	Darryl Ooi	No

Was this Finding Successful?
Yes

Finding Description

Issue: The admin institution settings pages are accessible to unauthorized users. This is a result of inadequate access control mechanism for sensitive URLs.

How: From the home page as a tutor, manually entering the path in the browsers URL and gained access. Through this, authentication can be bypassed granting access to administrator pages.

Risk Rating

Impact: Minor
Likelihood: High

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity.	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally.	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

Administrative actions were unsuccessful not allowing access to any sensitive data. The Broken Access Control allowing lower-level users to access administrative endpoints presents both an operational and representative risk. If an attacker can chain this vulnerability with others or if further authentication bypasses are discovered, the attacker would then be able to access and modify sensitive data. This data includes that of user information which if accessed will have a detrimental effect on user trust in the application and Deakin, with the risk of Personally Identifiable Information being exposed which would result in likely legal implications.

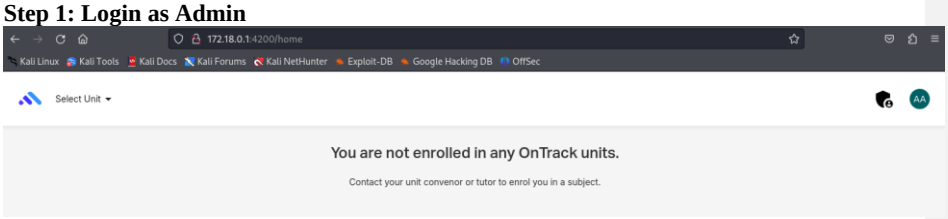
Affected Assets

OnTrack Web API:

- Endpoint: http://172.18.0.1:4200/admin/units
- Endpoint: http://172.18.0.1:4200/admin/institution-settings
- Endpoint: http://172.18.0.1:4200/admin/users
- Role: Tutor

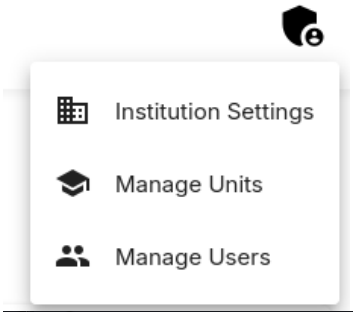
Evidence

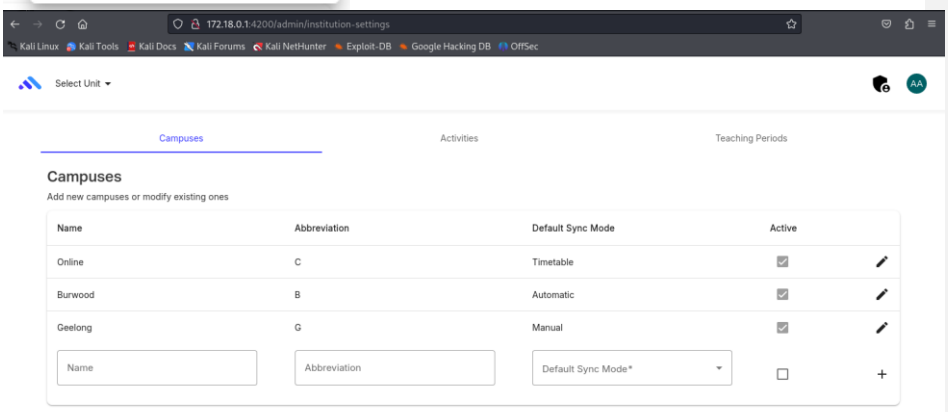
Step 1: Login as Admin



Enter the login credentials for the admin being aadmin and password, then you will be redirected to the home page.

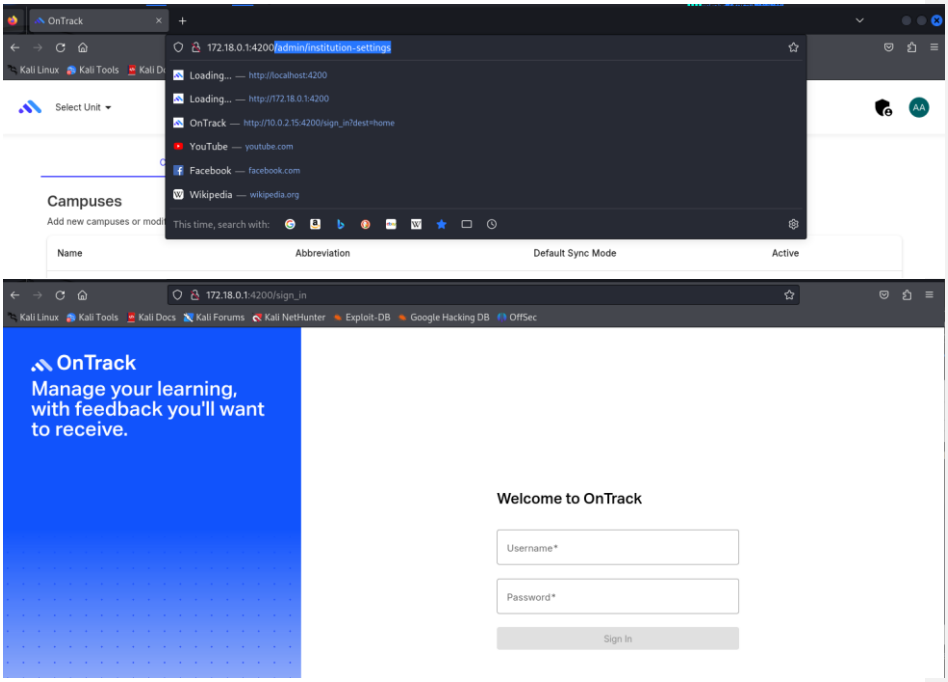
Step 2: Navigate to the sensitive pages





Click the 1st icon on the top right and click on institution settings, then the same for manage units and manage users

Step 3: Copy the paths and log out



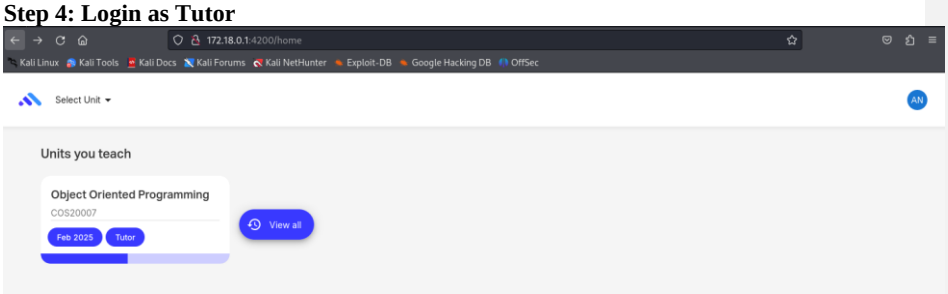
Keep the window open and open a new ontrack page in another tab then log out. You should have the following links.

http://172.18.0.1:4200/admin/units

http://172.18.0.1:4200/admin/institution-settings

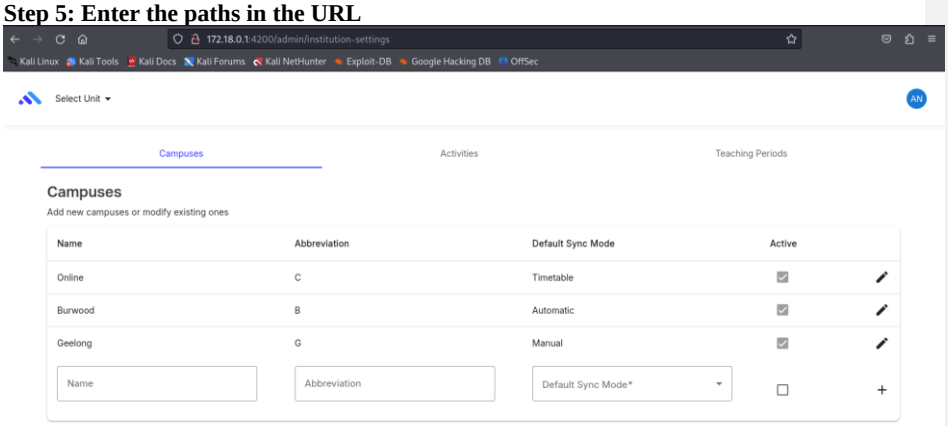
http://172.18.0.1:4200/admin/users

Step 4: Login as Tutor



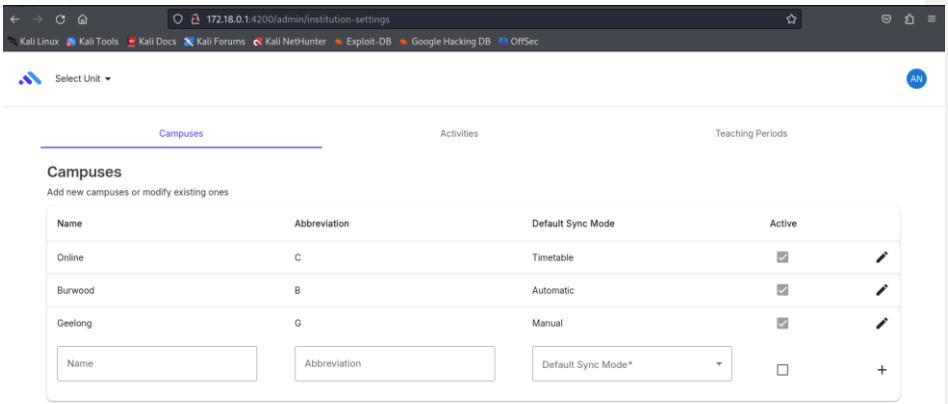
Enter the login credentials for the tutor being atutor and password, then you will be redirected to the home page.

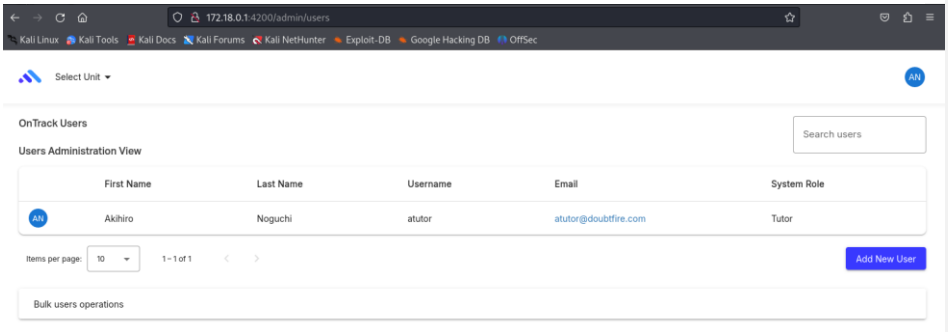
Step 5: Enter the paths in the URL

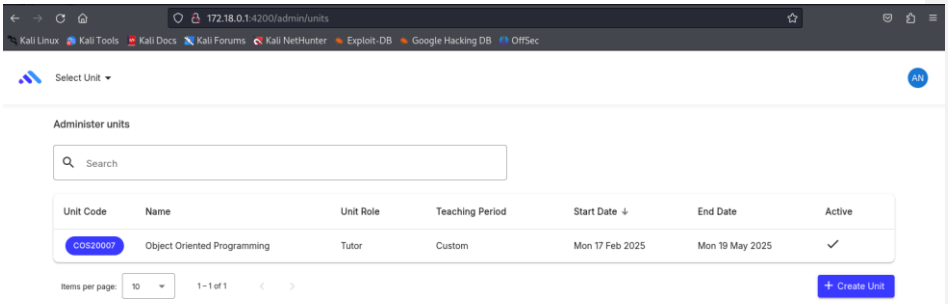


Enter the path to the admin path in the URL being /admin/institution-settings etc then be redirected

Step 6: Be redirected to the admin pages







Although there isn't any sensitive data and no modifications can be applied to the page, the tutor account has elevated privileges enabling the access to the admin pages.

Remediation Advice

The vulnerability arises due to inadequate access control measures, in this case a tutor was able to manually access the admin institution settings page via the URL.

To avoid this specifically, input validation must be implemented on URL paths, verifying user roles against the URL request. Another action that can be taken is the implementation of multi-factor authentication for administrative users accessing privileged pages. Using HTTP 403 Forbidden for unauthorised attempts to access sensitive URLs with minimal information displayed in these messages is another action that can be taken to keep sensitive information secure.

References

No tools required:
Login atutor , password
Path to enter admin/institution-settings

Contact Details

Jackson Anton Bouwman
Student, Deakin university
S222238893@deakin.edu.au

Pentest Leader Feedback.

- Please rename the document to follow the naming conventions outlined in the findings workflow document.
- Please change all the blue text to black.
- Role and project were incorrect, which I have fixed for you.
- Provide more information and details in your finding description. Also mention the specific URL/endpoints.
- In the risk rating section, impact and likelihood should just be single words as shown below:

  Risk Rating
  Impact: Major
  Likelihood: High

- In the evidence section, demonstrate how you were able to find the /admin/institution-settings endpoint to be used for privileged access.
- Please demonstrate successful modification of the Campuses, Activities, and Teaching Periods which you claimed possible in the evidence description. I was unable to do this in my testing. Thus, if it is not possible to modify these, please adjust all sections of your report to reflect this and reassess your impact risk rating.
- Don’t use localhost address use one of the other two external network addresses and using localhost provides higher access. Use the 10.0.01 or the other

6.9: Clickjacking Vulnerability (Nessus Scan and VM Execution)

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Nikola Nicholas Krcevinac	OnTrack	Pen-Tester	AppAttack	Darryl Ooi	No

Was this Finding Successful?
Yes

Finding Description

The OnTrack web application was found to be vulnerable to Clickjacking. This type of attack happens when the attacker loads the legitimate application within an <iframe> and tricks the user into clicking elements they didn't intend to interact with, like a login button or form submission.

The root cause of this issue is the absence of key security headers in the HTTP response, mainly the **X-Frame-Options** and **Content-Security-Policy (frame-ancestors)** headers which exploit the lack of these HTTP headers. This allows the OnTrack site to be embedded in a malicious site as demonstrated in the evidence section further down. Also, in evidence you will see the Nessus scan screenshots of the findings.

This vulnerability was confirmed using a simple crafted HTML page to frame the OnTrack website. When opened in a browser, it rendered the full interface behind a transparent layer and placed a click button on top, proving the success of the attack.

Risk Rating

Impact: **Severe**
Likelihood: **High**

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity.	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally.	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

This vulnerability poses a high risk to the user trust and application integrity for the university for teacher and students using OnTrack. An attacker could socially engineer techniques to make the user visit a malicious site and provide legitimate OnTrack application embedded in a hidden iframe. This can lead to:

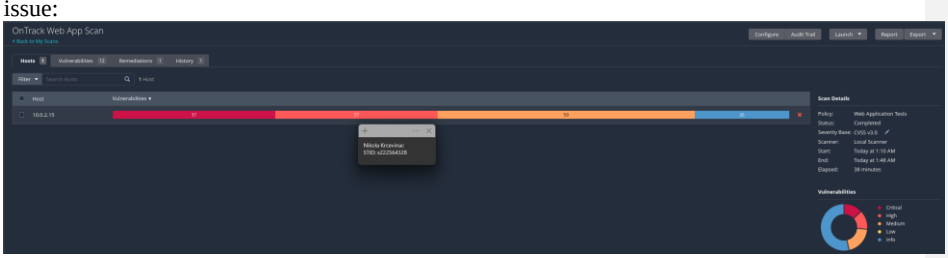
- Unauthorized account access
- Accidental permission changes
- Credential theft
- Reduced trust in the application's security

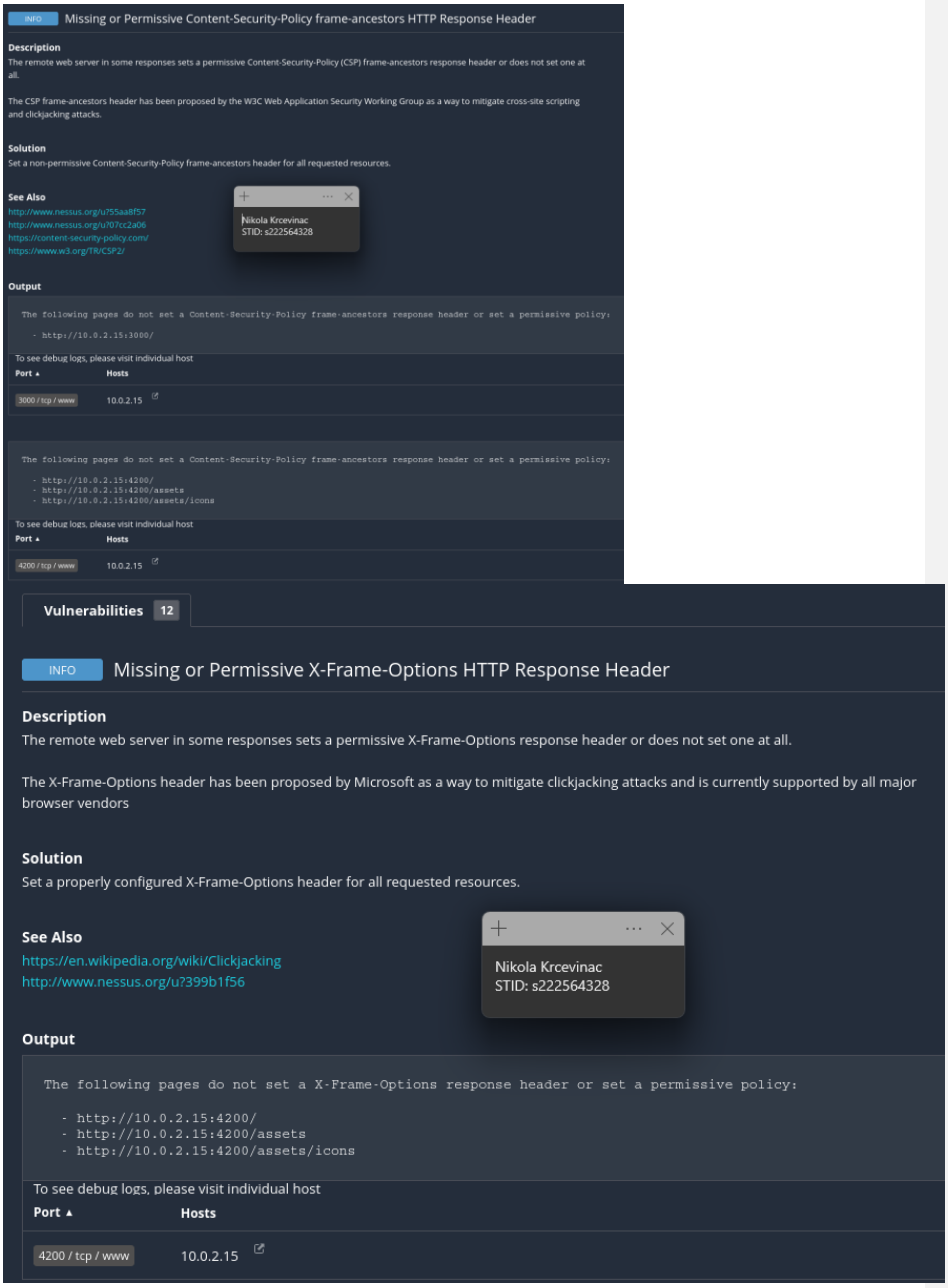
Affected Assets

- http://10.0.2.15:3000/ (Doubtfire API)
- http://10.0.2.15:4200/ (OnTrack Frontend)
- http://10.0.2.15:4200/assets
- http://10.0.2.15:4200/assets/icons

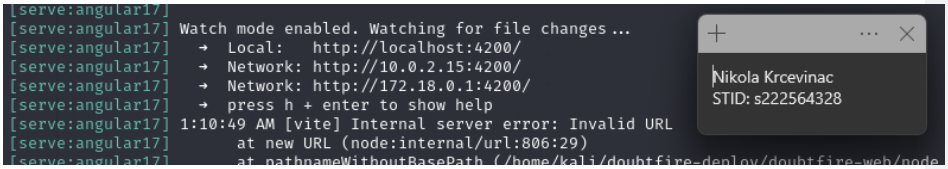
Evidence

For your evidence, I have provided below three images showing my Nessus scan with the hosts, port output, plugins and their description explaining what attacks can happen for this issue:





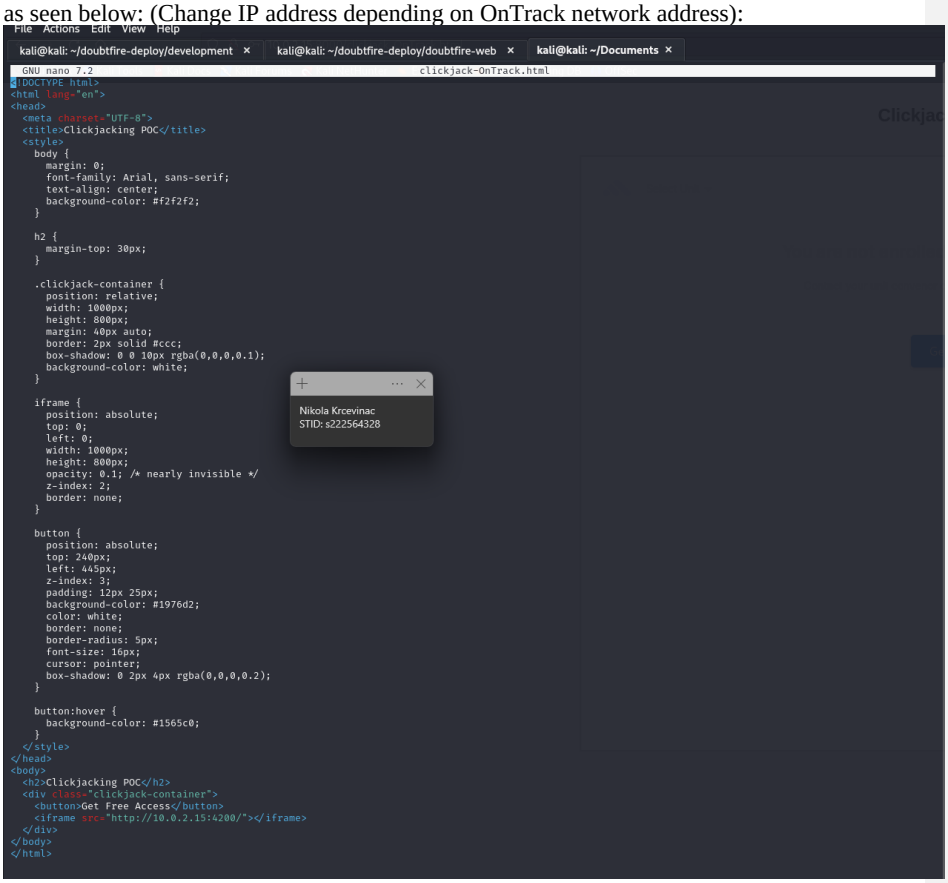
This below is the network address assigned to the OnTrack website on my Kali VM machine as reference:



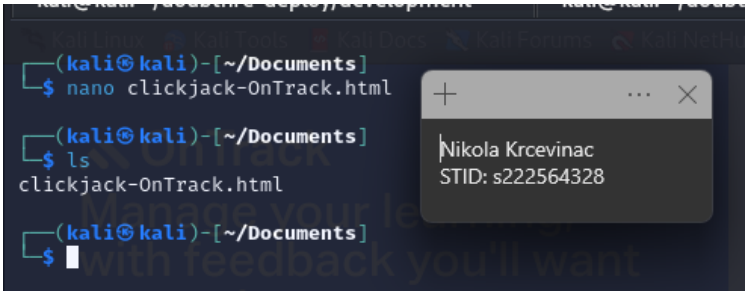
Clickjacking Proof of Concept (POC)

Step 1: Create a Malicious HTMP File

1. First, I use “nano clickjack-OnTrack.html” to create a malicious HTML file for this attack as seen below: (Change IP address depending on OnTrack network address):



The image below shows the file that has been created and the folder it is in:



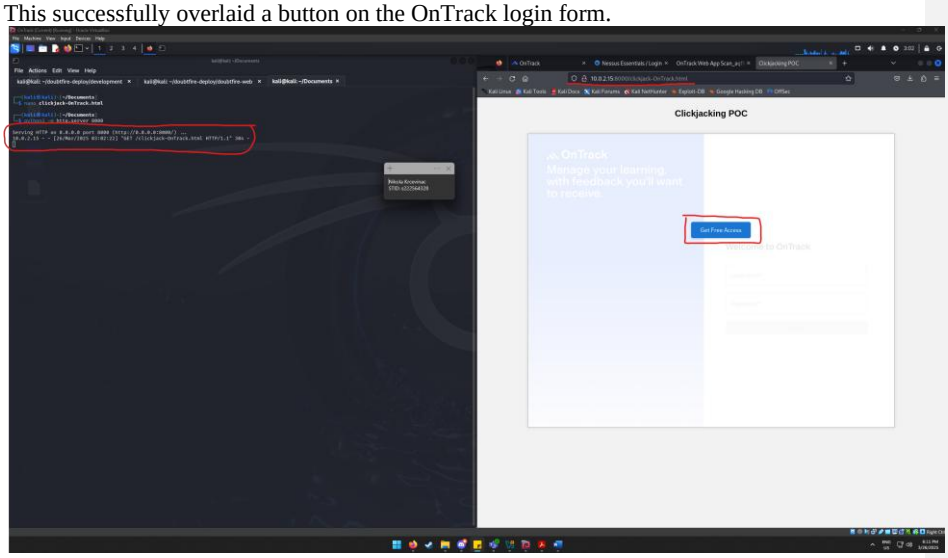
Step 2: Serve the HTML File

1. I will now use “python3 -m http.server 8000’ to run my html:

2. Now that I am running my server I can now visit the OnTrack site at “http://10.0.2.15:8000/clickjack.html” which I will continue in step 3.

Step 3: Verifiction of the Vulnerability

Below I accessed the malicious page via browser at “http://10.0.2.15:8000/clickjack.html”. This successfully overlaid a button on the OnTrack login form.



Explanation:

This demonstration illustrates a **Clickjacking vulnerability** affecting the OnTrack web application. While the "Click Me" button in this proof-of-concept (PoC) does not currently trigger any real action, its purpose is to show that a malicious actor could **embed the OnTrack site into an invisible or transparent iframe** and overlay deceptive content (like fake buttons) on top of it.

If an attacker aligned their fake button over a **real and functional UI element** (e.g., "Sign In", "Confirm", "Allow access", or even "Delete Account"), the user could be tricked into clicking that **underlying real button** while believing they are interacting with the attacker's page.
In a real-world scenario, this could be used to:

- Hijack user sessions
- Trigger actions without the user's consent
- Abuse user privileges if the session is authenticated
- Execute unintended or dangerous operations in the context of the user

Therefore, although this demo is surface level, it **proves that the application is vulnerable** to clickjacking and that **additional security controls are required to prevent this type of attack** from being exploited by a real adversary.

Remediation Advice

To mitigate this vulnerability, the OnTrack development team should:

1. **Set the X-Frame-Options header** to DENY or SAMEORIGIN to prevent the application from being embedded in external frames.
   X-Frame-Options: DENY
2. **Configure a Content Security Policy** with the frame-ancestors directive:
   Content-Security-Policy: frame-ancestors 'none';
3. Regularly scan for missing HTTP headers and implement CSP best practices.

References

- Internal Nessus Report
- Clickjacking OWASP
- X-Frame-Options and Content Security Policy

Contact Details

Name/Teams: Nicholas Krcevinac
Email: s222564328@deakin.edu.au

Pentest Leader Feedback.

Great job Nicholas!

6.10: Insecure Direct Object Reference (IDOR) – Unauthorized Access to Staff Information

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Nicholas Krcevinac	AppAtack	Pen-Tester	OnTrack Web App	Darryl Ooi	No

Was this Finding Successful?
Yes

Finding Description

An **IDOR vulnerability** was discovered in the OnTrack web application, which allows a student-level authenticated user to access sensitive staff information by manipulating API request parameters. The endpoint /api/units/{unit_id} returns full details of the academic staff members associated with any unit, including their:

- Full names
- Email addresses
- Roles (Convenor, Tutor)
- Usernames and nicknames

This data is returned without any access control or role-based filtering, violating the principle of least privilege.

Risk Rating

Impact: **Major**
Likelihood: **High**

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity.	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally.	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

The exposure of personally identifiable information (PII) like staff email addresses and usernames increases the risk of targeted phishing, identity theft, and social engineering attacks. It undermines user privacy, breaks compliance with data protection laws (e.g., GDPR), and could erode user trust in the platform, especially from academic staff.

Affected Assets

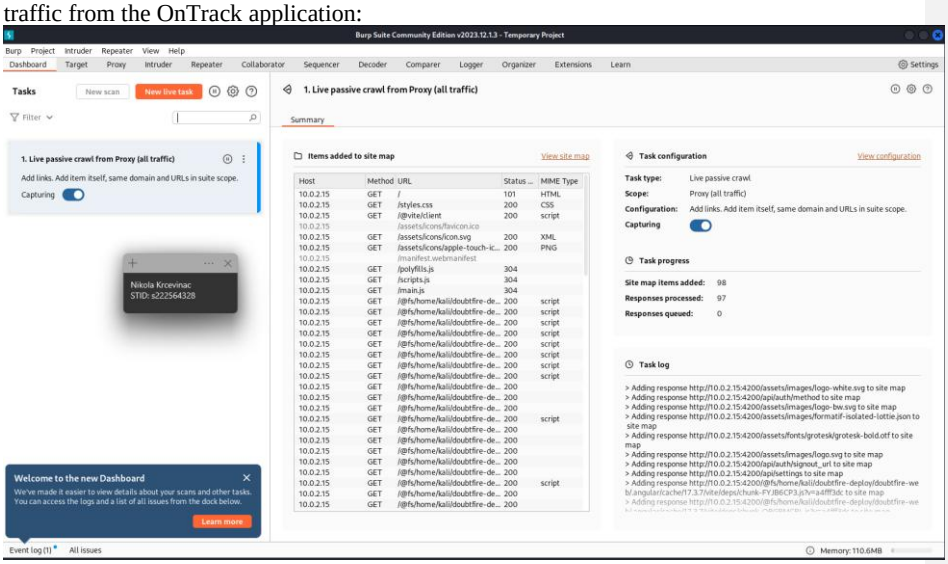
OnTrack Web API:

- Endpoint: GET /api/units/{unit_id}
- Role: Affects *all users* with access to the platform, especially student users gaining unintended access.

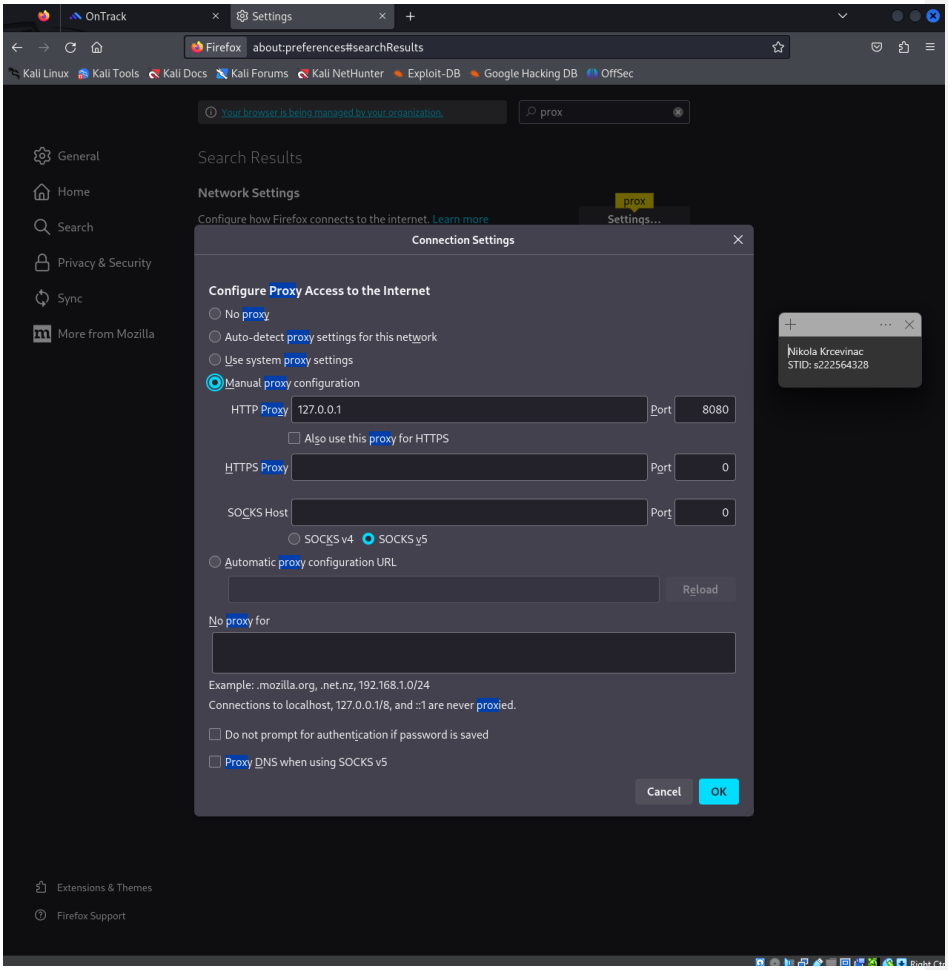
Evidence

Step 1: Configure Burp Suite

1. Make sure to install and have burp suite running as Burp Suite set to intercept and map live traffic from the OnTrack application:

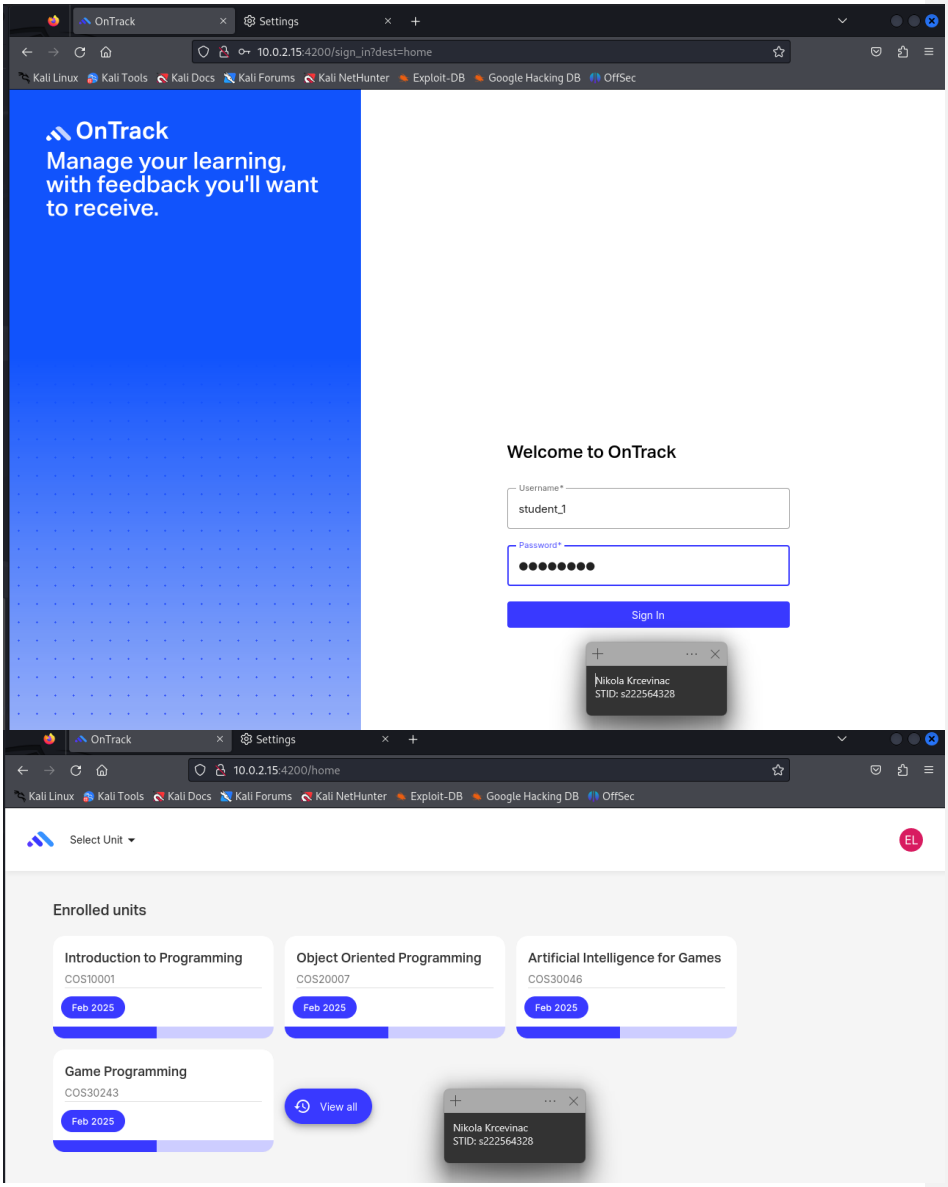


2. In Firefox got to settings preferences and search “proxy”, then click the Network Settings button to them input the same information as shown in the below image:



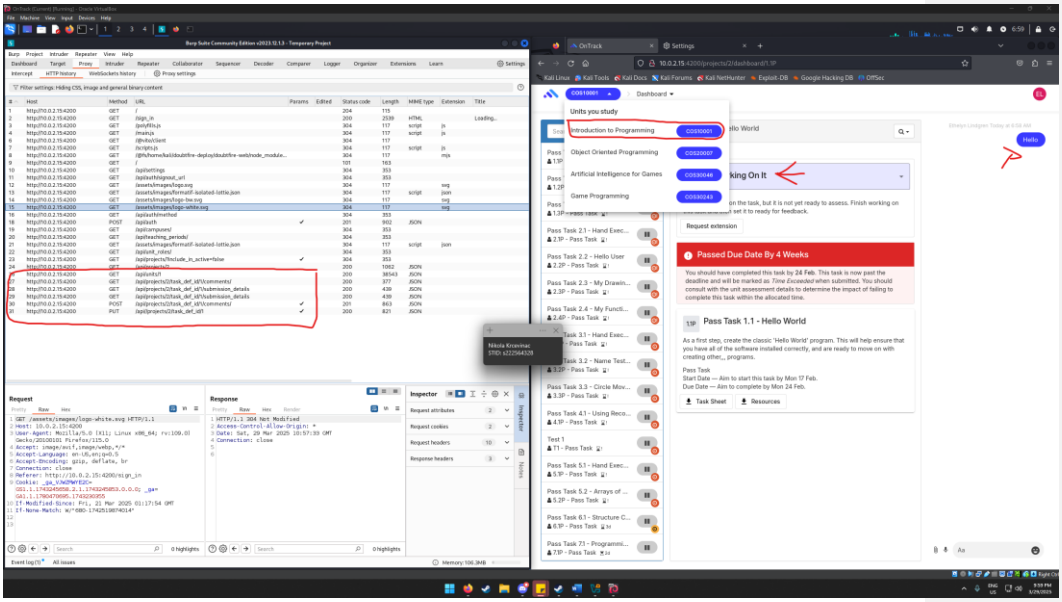
Step 2: Login as Student User

1. Used the student_1 login details and signed in as seen below:

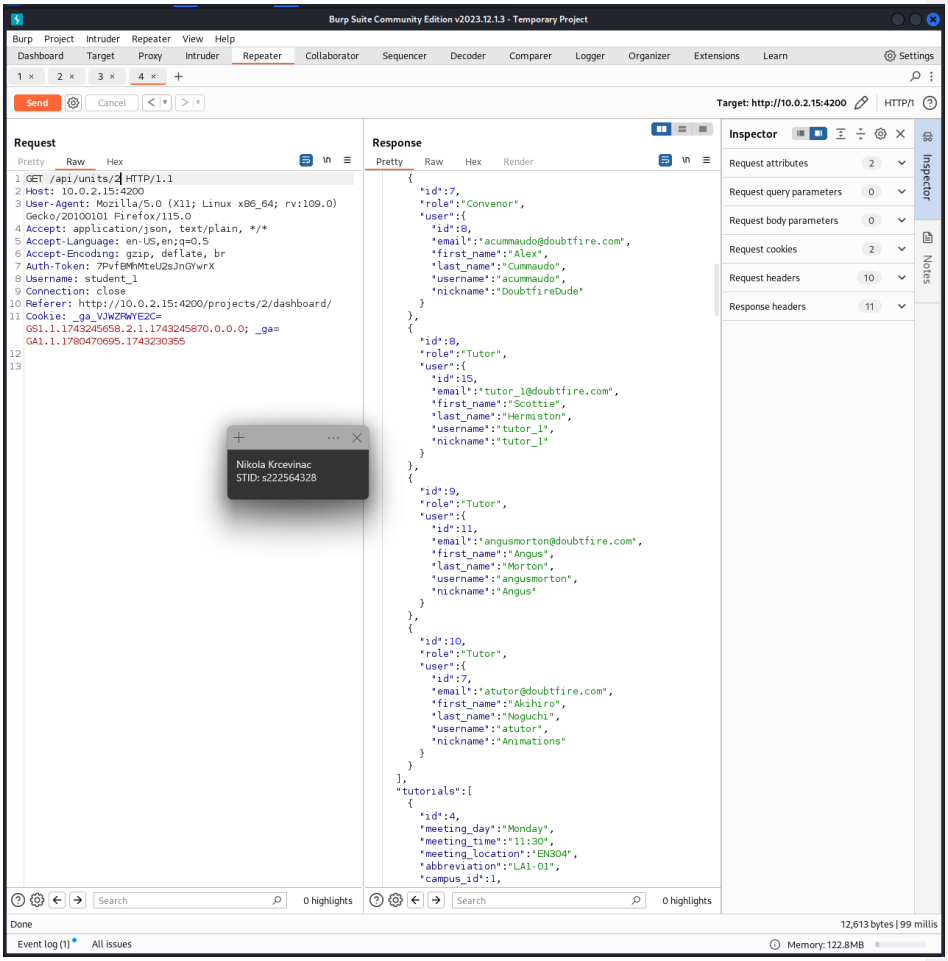


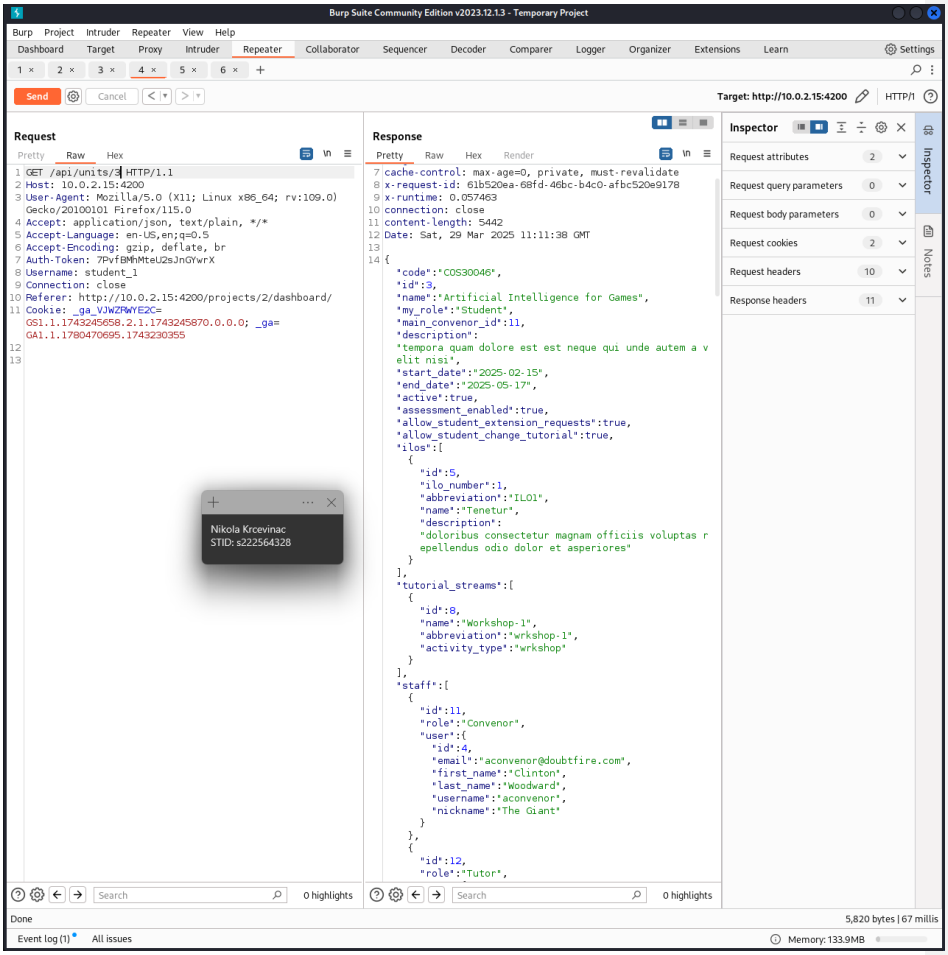
Step 3: Intercepts API Call to Unit Endpoint

1. Below on the left I clicked on Proxy -> HTTP History and then on the right started clicking random buttons, commenting and uploading to create HTTP traffic as seen below:



2. After that I right clicked the “GET /api/units/1” and sent to Repeater in Burp. Below you can see two images where you can cleary seen the names of the Convenor and Tutor visibly, thus showing unauthorized staff data in response:


Burp Suite Community Edition v2023.12.1.3 - Temporary Project
Request
GET /api/units/2 HTTP/1.1
Host: 10.0.2.15:4200
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Auth-Token: 7PvfBMMMteU2sJnGYwrX
Username: student_1
Connection: close
Referer: http://10.0.2.15:4200/projects/2/dashboard/
Cookie: _ga_VJWZPWYE2C=GS1.1.1743245658.2.1.1743245870.0.0.0; _ga=GA1.1.1780470695.1743230355
Nikola Krcevinac
STID: s222564328
Response
Inspector
Request attributes
Request query parameters
Request body parameters
Request cookies
Request headers
Response headers
Done
12,613 bytes | 99 millis
Memory: 122.8MB



Remediation Advice

The application fails to implement proper access controls for sensitive data in its API endpoints.

Mitigations:

- Implement role-based access control (RBAC) on API endpoints to restrict data based on user privileges.
- Ensure sensitive user information (such as staff emails, names, and roles) is only accessible to authorized users (e.g., Admins or the staff themselves).
- Sanitize API responses and avoid over-sharing data that is not required by the frontend for the user's role.
- Conduct regular access control reviews and enforce the least privilege access principles.

References

- OWASP: Insecure Direct Object References (IDOR)
- OWASP Top 10 – Broken Access Control
- Burp Suite Documentation

Contact Details
Name/Teams: Nicholas Krcevinac
Email: s222564328@deakin.edu.au

Pentest Leader Feedback.
Nice work!

6.11: Malicious Code Execution (CVE-2024-4367)

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Wahidullah Hashimi	OnTrack	Pen-Tester	AppAttack	Nicholas Krcevianc	Yes

Was this Finding Successful?
Yes

Finding Description

While testing vulnerabilities for Ontrack, I started by using OWASP ZAP and discovered that it's using a vulnerable version of a JavaScript library (3.11.174). This issue is listed in the CVE (Common Vulnerabilities and Exposures) system as CVE-2024-4367. CVE is a well-known reference point for publicly disclosed security vulnerabilities.

CVE-2024-4367 is a serious flaw in the PDF.js library, which is widely used to view PDFs in web browsers. The problem arises because the library doesn't properly handle certain font data, which opens the door for attackers to inject malicious JavaScript. This vulnerability affects Firefox versions before 126, Firefox ESR versions prior to 115.11, Thunderbird versions earlier than 115.11 and Ontrack.

If a user opens a malicious PDF crafted to exploit this flaw, JavaScript could execute, leading to some pretty severe consequences, like data theft, XSS attacks, or even remote code execution (RCE). It's crucial for anyone using these versions to update their software to avoid falling victim to these types of attacks.

This vulnerability was also discovered in T1 2024. You can find the insights of other penetration tester in AppAttack x ThothTech.pdf on page 56.

Risk Rating

Impact: Major

Likelihood: Moderate

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity.	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally.	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

The business impact of CVE-2024-4367 can be significant, especially for organizations that rely on PDF.js or similar libraries to handle PDF files within their applications. Here are some key areas of concern:

1. Data Breach Risks: If the vulnerability is exploited, attackers could access sensitive user data, like personal information, trade secrets, or intellectual property. This can lead to data theft and damage the company's reputation as customers lose trust.
2. Financial Losses: Exploiting the vulnerability could allow attackers to take control of systems, steal funds, or manipulate transactions. Additionally, the company may face fines for violating data protection regulations like GDPR or CCPA.
3. Reputation Damage: If a successful attack occurs, it can significantly harm the company's reputation. Customers expect their data to be secure, and a breach could result in a loss of trust and reduced business.
4. Legal and Compliance Consequences: Organizations that handle sensitive data could face legal action if they don't take adequate security measures. Failure to patch vulnerabilities may lead to penalties for non-compliance with regulations like GDPR or HIPAA.
5. Operational Disruption: Attackers exploiting the vulnerability could cause system downtime, service outages, or data corruption. This disruption affects business operations and incurs additional costs for recovery.

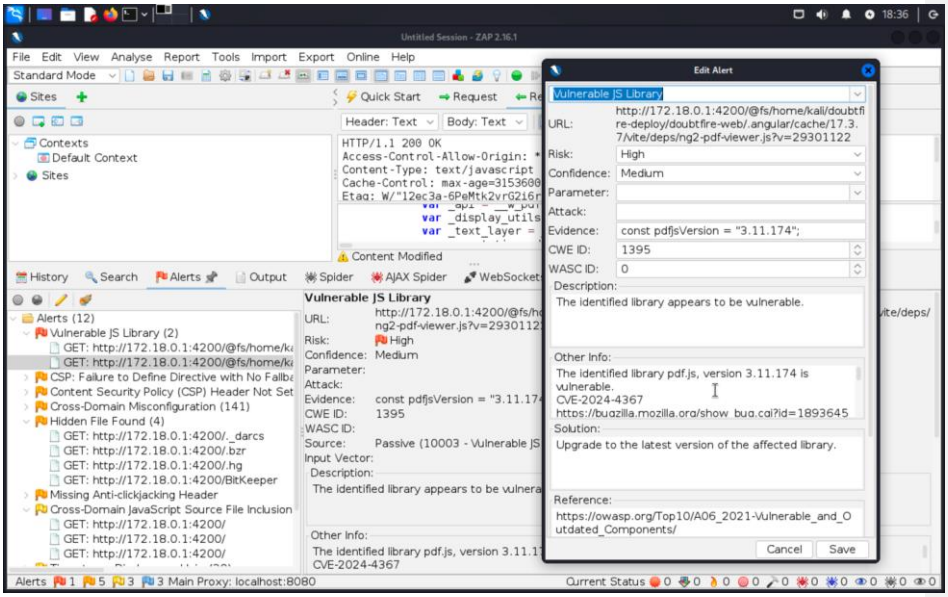
Affected Assets

- Doubtfire API
- OnTrack Frontend
- PDF.js Library
- Firefox (versions earlier than 126)

Evidence

Step 1: [Identify vulnerability]

I used OWASP ZAP to do a deep scan of the web application (Ontrack). As shown in the screenshot below, I found out that the web application uses a vulnerable JS Library.

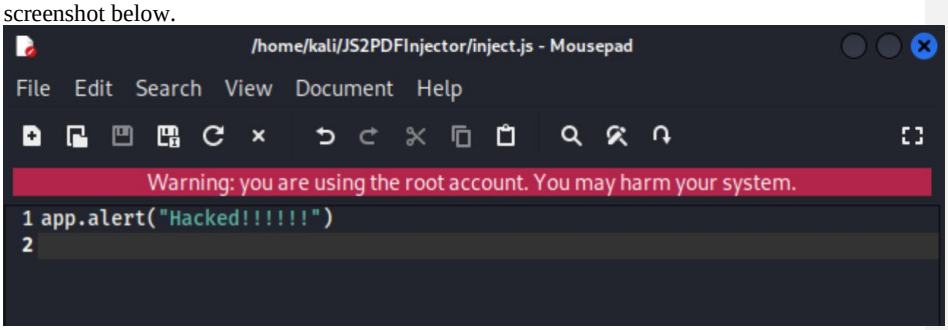


Step 2: [Research]

I did some research on how to exploit this vulnerability and found that I can embed malicious code using JS2PDFInjector tool. When this pdf file is clicked the malicious code will be executed.

Step 3: [Creating the Malicious file]

Clone JS2PDFInjector tool from github in your computer. Create a payload as shown in the screenshot below.

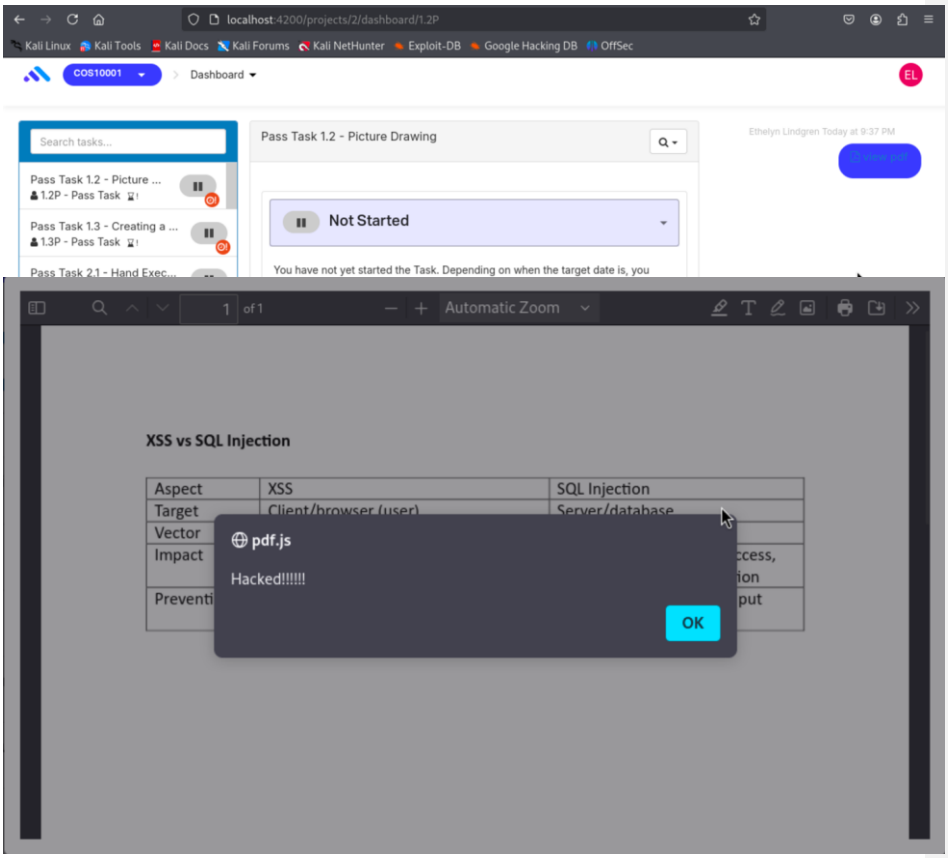


Next, embed this malicious script into a real pdf file using the command shown in the screenshot below.

```
┌──(kali㉿kali)-[~/JS2PDFInjector]
└─$ sudo java -jar JS2PDFInjector-1.0.jar /home/kali/JS2PDFInjector/sub.pdf /
home/kali/JS2PDFInjector/inject.js

[*] Original PDF: /home/kali/JS2PDFInjector/sub.pdf
[*] JavaScript Payload: /home/kali/JS2PDFInjector/inject.js
[*] Output File Path: /home/kali/JS2PDFInjector/js_injected_sub.pdf
[*] Poisoned File Created: /home/kali/JS2PDFInjector/js_injected_sub.pdf
```

Finally, you send this malicious pdf to your marker/tutor. When your marker clicks on this pdf file, the script will be executed. In the screenshot below you can see the executed pdf file.



CVE-2024-4367 Summary:
CVE-2024-4367 is a serious security flaw found in the PDF.js library, which many web browsers use to display PDF files. The issue occurs because the library doesn't properly

check types when dealing with fonts, which could allow attackers to inject and run malicious JavaScript in the PDF. If exploited, this vulnerability could lead to remote code execution (RCE), data theft, or cross-site scripting (XSS) attacks through specially crafted PDF files.

Mitigation Points:

1. **Update PDF.js**: Make sure to update to the latest version of PDF.js that fixes this vulnerability.
2. **Patch Affected Browsers**: Ensure that browsers like Firefox and Thunderbird that rely on vulnerable PDF.js versions are updated to a secure version.
3. **Sanitize PDFs**: Set up processes to scan and clean PDFs for malicious JavaScript before opening them.
4. **Limit JavaScript Execution**: Configure PDF viewers to block or restrict JavaScript execution for added security.

References

OWASP TOP 10:2021, A06:2021 – Vulnerable and Outdated Components. accessible at: https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

Cornerpirate/ JS2PDFInjector, accessible at:

https://github.com/cornerpirate/JS2PDFInjector

National Vulnerability Database (NVD) (2024). CVE-2024-4367 Detail. Accessible at: https://nvd.nist.gov/vuln/detail/CVE-2024-4367

Contact Details

Name: Wahidullah Hashimi
Email: s223397352@deakin.edu.au

Pentest Leader Feedback.

Nicholas Krcevianc:

- Next time please follow the AppAttack Findings Checklist.docx for correct formatting, this time I fixed your style and size.
- Besides that this document is perfect

6.12: Privilege Escalation Chain – BAC and Session Hijacking

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Andrew Duffy	AppAtack	Pen-Tester	OnTrack Web App	Darryl Ooi	No

Was this Finding Successful?
Yes

Finding Description

A Privilege Escalation vulnerability was discovered in the OnTrack web application through exploitation of a chain of related vulnerabilities involving Broken Access Control, exposed administrative UI elements, and session hijacking of a privileged user's token. The application permits a student-level user to render tutor-only UI elements by appending ?tutor=true to the student's task view URL. This enables the display of privileged options, such as task grading functions (Pass, Fail, etc.) which are restricted to tutor or privileged user roles. On its own, this does not permit the student to action those changes, as additional back-end validation appears to block unauthorised, low-level roles from executing these requests.

However, it was demonstrated during testing that when this client-side UI exposure is combined with a previously documented Session Hijacking vulnerability, a student can successfully set up and execute these privileged actions. The low-level user is able to intercept the privileged request and replace the student's token and username with the captured privileged user's token and username. This is then executed and bypasses additional request validation, allowing the student to successfully alter their own assessment status.

The ?tutor=true query was discovered by comparing the provided low-level and privileged user accounts and identifying differences in access and URL requests. Side by side comparison between the two user types identified the query, appending this to the low-level (student) user triggered the same privileged UI elements. Source code or network traffic analysis has the potential to identify the same query for privileged users.

This privilege escalation chain demonstrated below highlights the dangers of allowing privileged UI elements to be rendered in a low-level user's client. On its own, this presents a negligible risk provided that client-side inputs are not trusted without validation and additional vulnerabilities cannot be exploited to bypass compensating user validation controls on the server side. However, as demonstrated, the existence of this coding weakness provided the opportunity to investigate and identify bypasses to these compensation controls allowing the low-level user to execute those privileged commands. In this instance the chain allowed a student to update their own task grades, i.e. – Fail to Pass.

Due to the need to first identify the tutor query as a valid option and then identify that compensating controls need to be bypassed with another vulnerability, that requires capturing a privileged user's session token, whilst it's still valid, the likelihood of this attack being exploited is **Unlikely**. Exploitation requires intent, timing, and a good understanding of web application security, further reducing the likelihood of the attack being identified or successfully executed. The ability for a student to modify their own grades poses a risk to the integrity of the assessment process and highlights the need for additional validation.

Currently this is limited to the single malicious student account limiting impact to that single user resulting in an impact of **Significant** with an overall risk rating of **Medium.**

Risk Rating - Medium

Impact: **Significant**
Likelihood: **Unlikely**

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity.	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally.	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

While the privilege escalation chain requires multiple steps to exploit, including knowledge of client-side behaviours and access to a valid privileged session token, allowing a low-level user to perform actions restricted to higher-privileged users — such as modifying their own assessment outcomes. This directly impacts the integrity of the application by enabling unauthorised manipulation of assessment data. Although compensating controls appear to be in place, the ability to chain client-side exposure with session hijacking highlights weaknesses in layered defences and may reduce stakeholder confidence in the reliability and overall trust in the use of the platform as a fair assessment tool.

Affected Assets

Student Task endpoint -
http://172.18.0.1:4200/projects/[project]/dashboard/[task][?tutor=true] - Allows for rendering of privileged UI elements.
Privileged Authentication/Session Tokens – provides opportunities for session hijacking.

Evidence

Step 1: Set up Burp Suite and environment

1. Open the Firefox browsers, Burp Suite and the Burp Suite inbuilt Chrome Browser. In both browsers, navigate to the application login page at **http://172.18.0.1:4200/sign_in**. Using both Firefox and Burp Suite's browser provides two separate browser instances, helping to prevent session tokens or permissions from being shared or 'bled' between them.

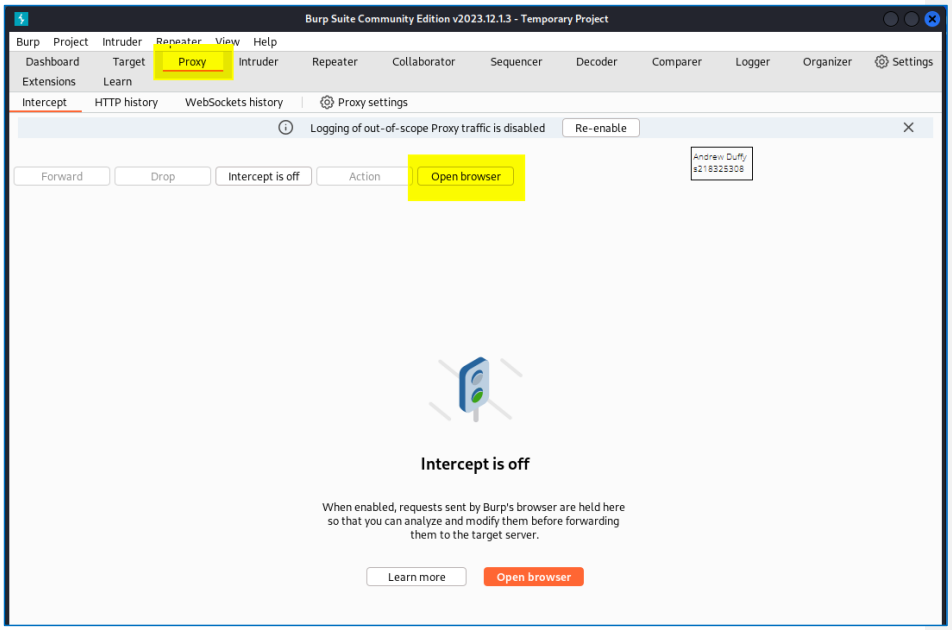
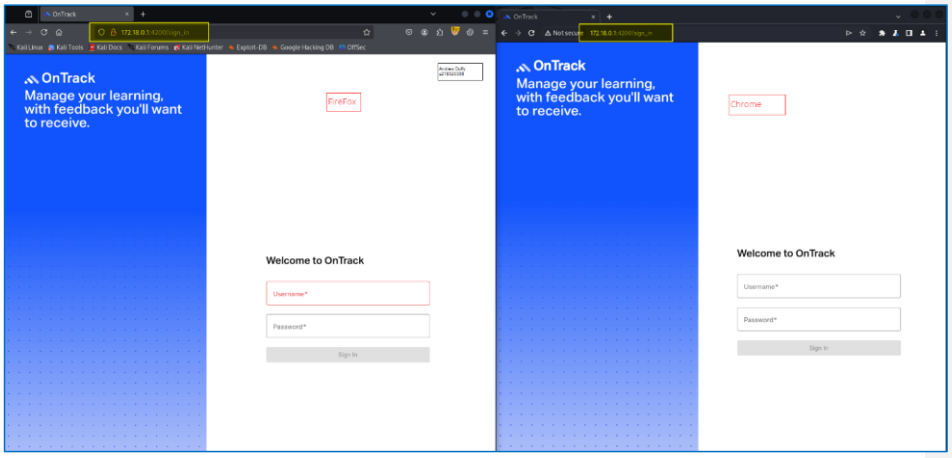


Figure – Setting Burp Suite – Open Chromium Browser



Firgure – Both Browsers ready at login screen

2. The Burp Suite Chromium browser automatically routes traffic through the Burp Suite proxy, so no further configuration is required there. On the Firefox browser, set it to route traffic through the Burp Suite Proxy.

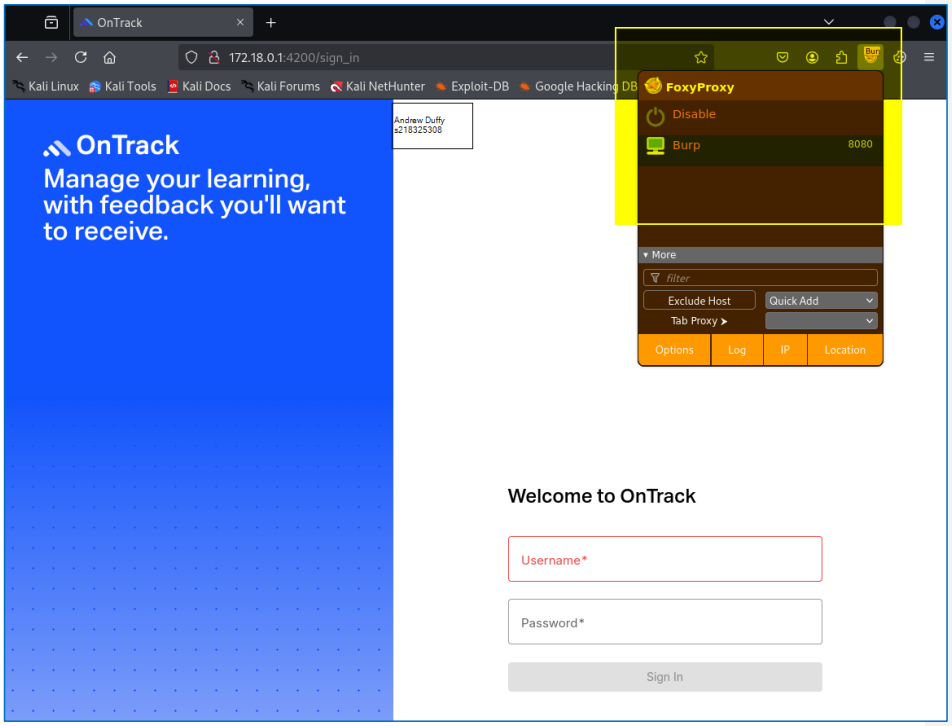


Figure – Firefox browser configured to route traffic through Burp Suite

3. The environment now consists of Burp Suite to intercept and inspect HTTP requests and responses, an instance of the Firefox browser, with traffic routed to Burp Suite and an instance of the inbuilt Burp Suite Chromium browser.

Step 2: Recon and Enumeration

1. With the environment set up, the primary objective was to identify differences in the application between a privileged and unprivileged or low-level user. Identification of these differences give targets for a threat actor to investigate further, trying to obtain access to these privileged resources or actions.

2. Log into the Firefox browser using the student_1 account, this will be the low-level user. Then log into the Chromium browser using atutor account, this will be the privileged user for comparison.

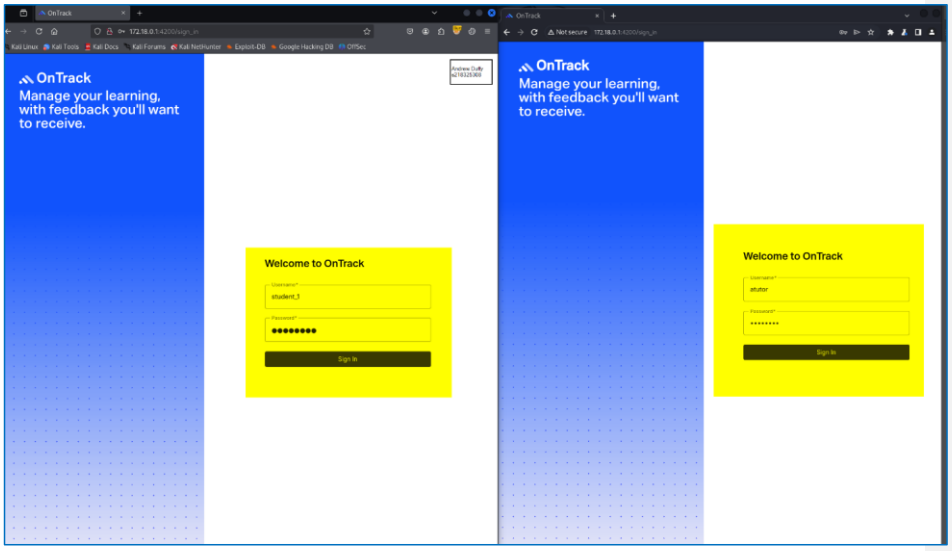


Figure – Login to both user accounts

3. The following endpoint was used for comparison and testing **http://172.18.0.1:4200/projects/18/dashboard/A12.** Comparison between student and tutor account identifies that the tutor has additional UI features which allow the tutor to grade the student's tasks, which are not available to the student. It is also noted that the URL for the tutor includes the **?tutor=true** parameter.

4. Note the privileged user must go through additional steps to land on the same 'task' page as seen by a student. Navigate to student, then choose a student to view the tasks.

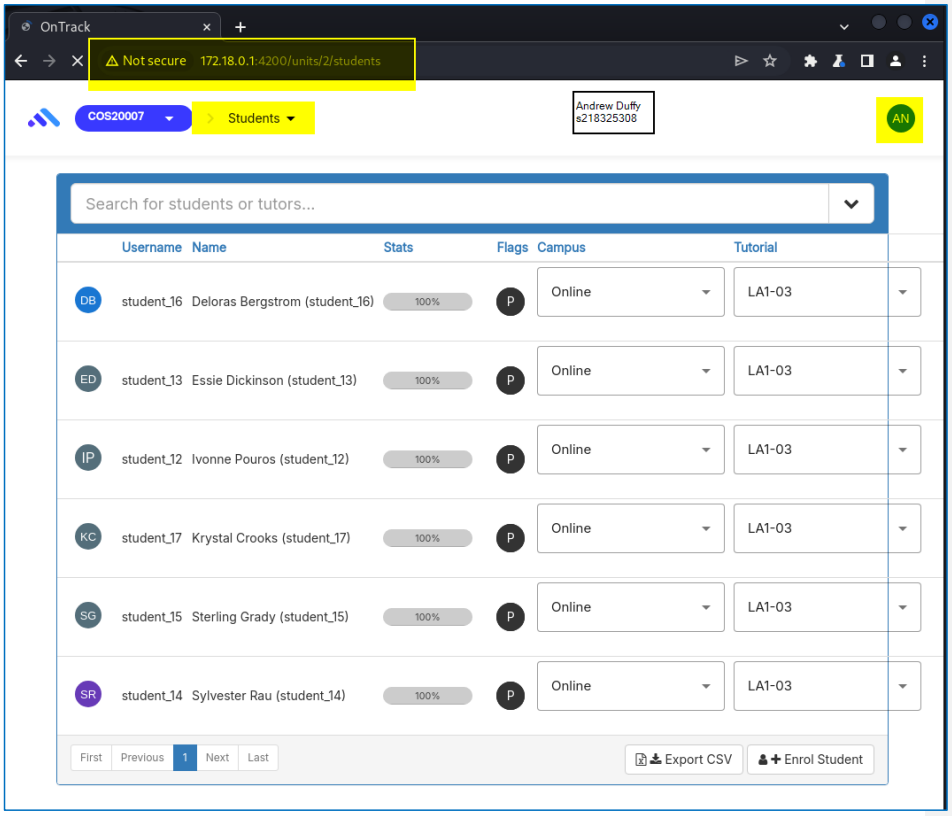


Figure – Navigating to the task page from the tutor

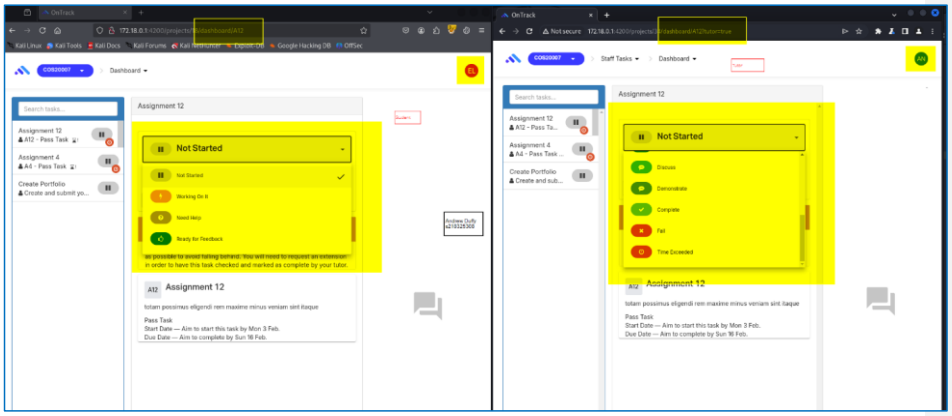


Figure – Logged into both accounts, comparing similar pages

4. Enumerating privileged functionality was done using provided privileged and low level user accounts, although this simplified the process, a motivated actor may still discover and test the **?tutor=true** URL query through other enumeration techniques. As shown below the URL is visible in traffic captured with wire shark.

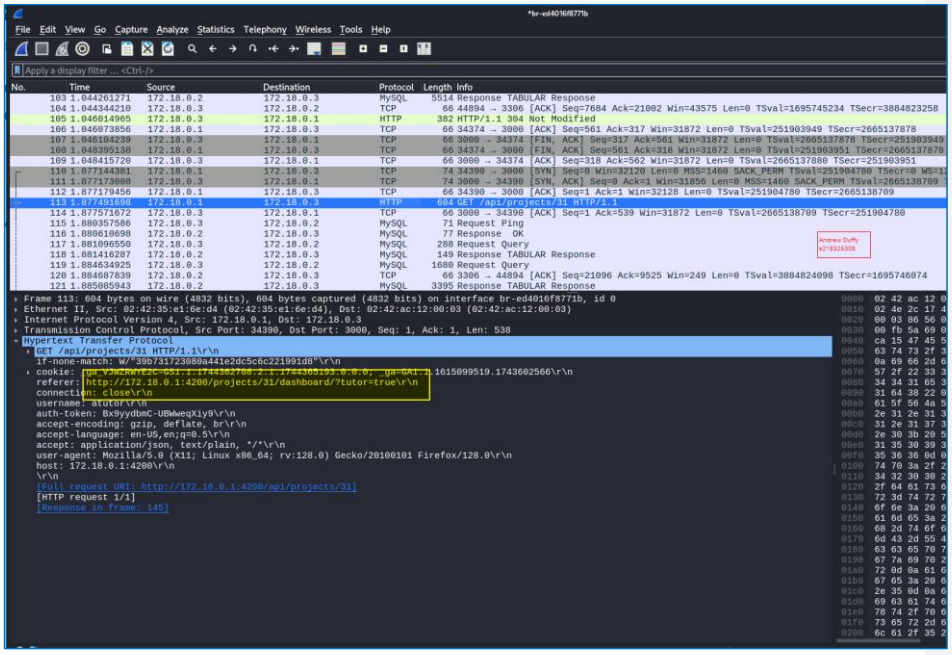


Figure – Wireshark review identify the query

Step 3: Testing the identified functionality

1. Now that a difference has been identified between the privileged and unprivileged user, testing can begin to identify if the unprivileged user can both access and execute these privileged functions. Testing with atutor confirms they can update a student's tasks, IE. to pass, fail etc. Testing will now confirm if a student can enable the tutor UI elements and update their own tasks.

2. Appending the parameter query (**?tutor=true)** to the student's url pressing enter to load th page was sufficient to render the privileged UI elements normally restricted to the tutor and above users.

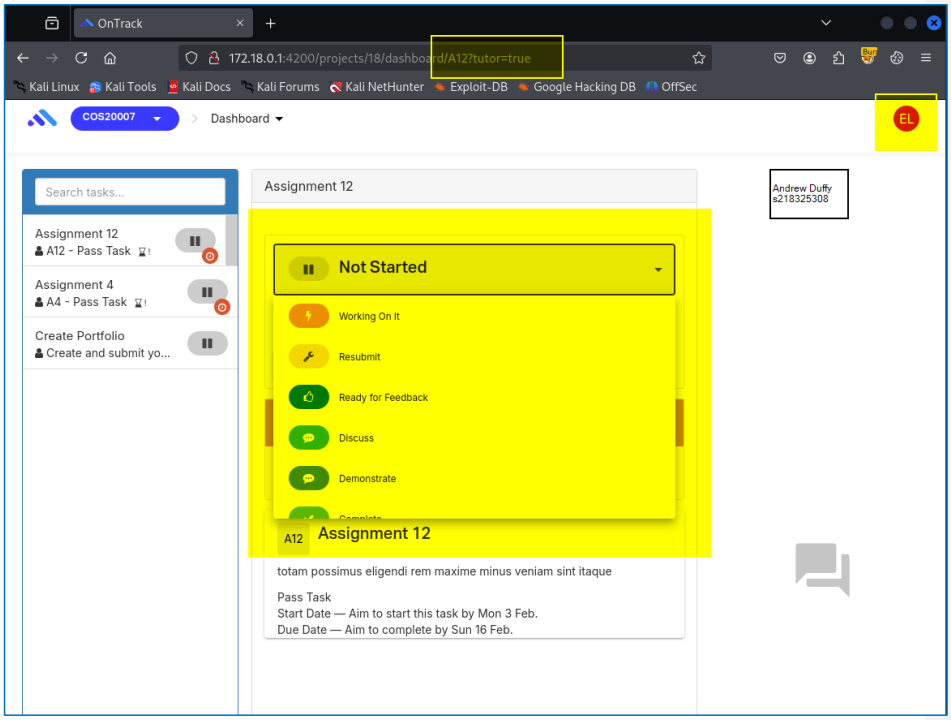


Figure – Privileged UI elements rendering in Student Client Browser

2. Initial attempts by the student level user to change their own grades failed, due to compensating controls and additional validation done by the back end. Although the student can now see the tutor UI elements within their own browser, this is still a restricted function.

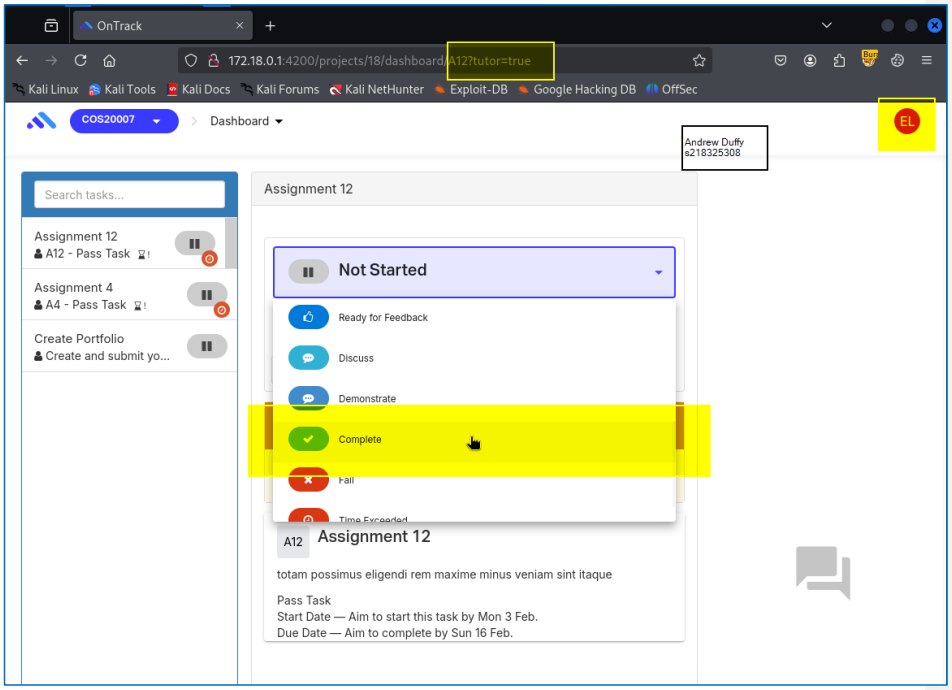


Figure – Student attempting to change own grade to pass

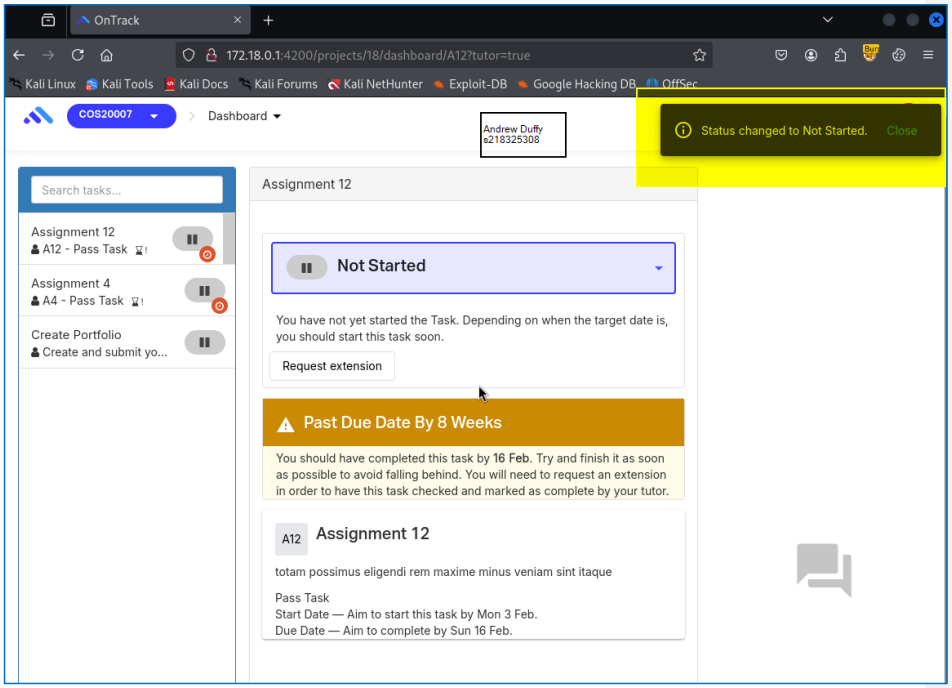


Figure – Failed attempt at changing own grades

3. Rendering administrative UI elements on the client side, even when compensating controls prevent their execution, may be considered poor practice or rated as an informational-level finding. However, as demonstrated in this case, exposing such functionality can entice a malicious user to dig deeper and identify methods to bypass these controls. The remainder of this chain leverages the previously identified session binding vulnerability to demonstrate how an otherwise innocuous finding can be chained into a successful privilege escalation attack.

Step 4: Chaining the vulnerability

1. A privileged session token is required to bypass additional validation to execute the command. Within Burp Suite clear the HTTP Request history if it is full of data, then within the Chromium browser (where the atutor is logged in) refresh the page to capture some traffic. Then back in Burp Suite HTTP history search through the requests to find a valid session token. Save this session token. Note: this information is also available through Wireshark – see Wireshark image above.

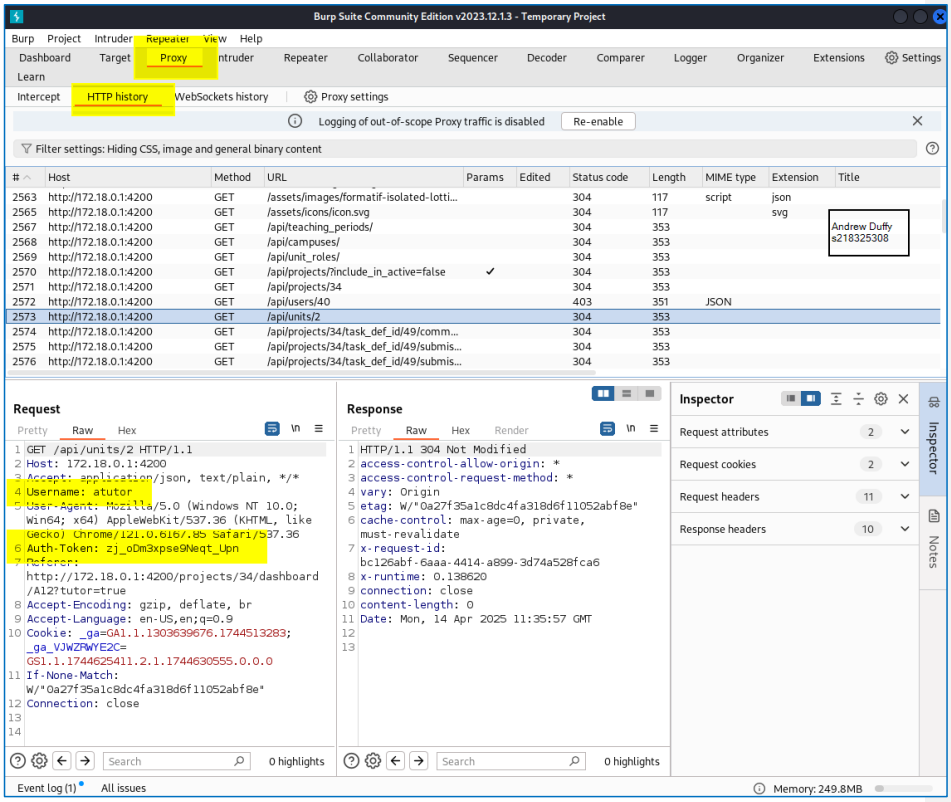


Figure – Capturing privileged users session token

2. Back in the Firefox browser, set up the request to complete a task. Open the drop down so the complete can be selected. Open Burp Suite and turn on intercept traffic.

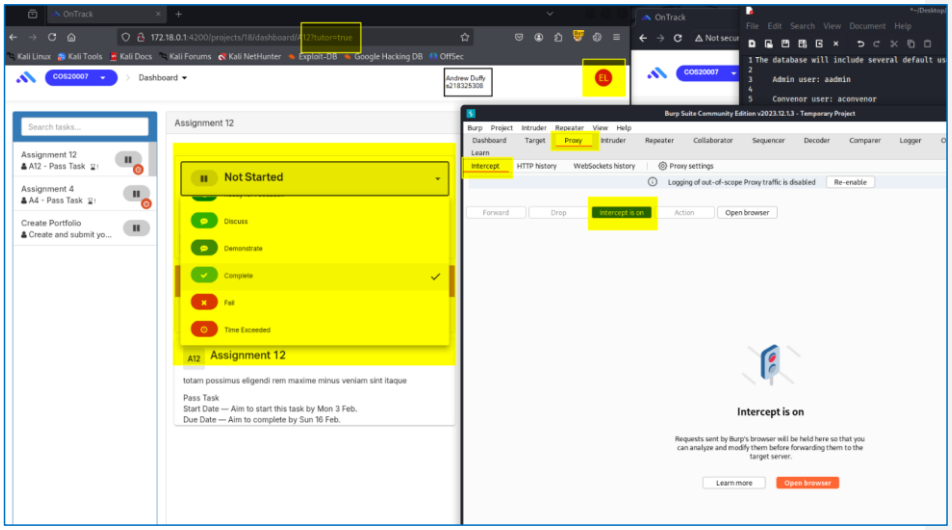


Figure – Student and Burp Suite Set up

3. Intercept the request to update the task by the student and replace the username and session token with the ‘stolen’ username and session token.

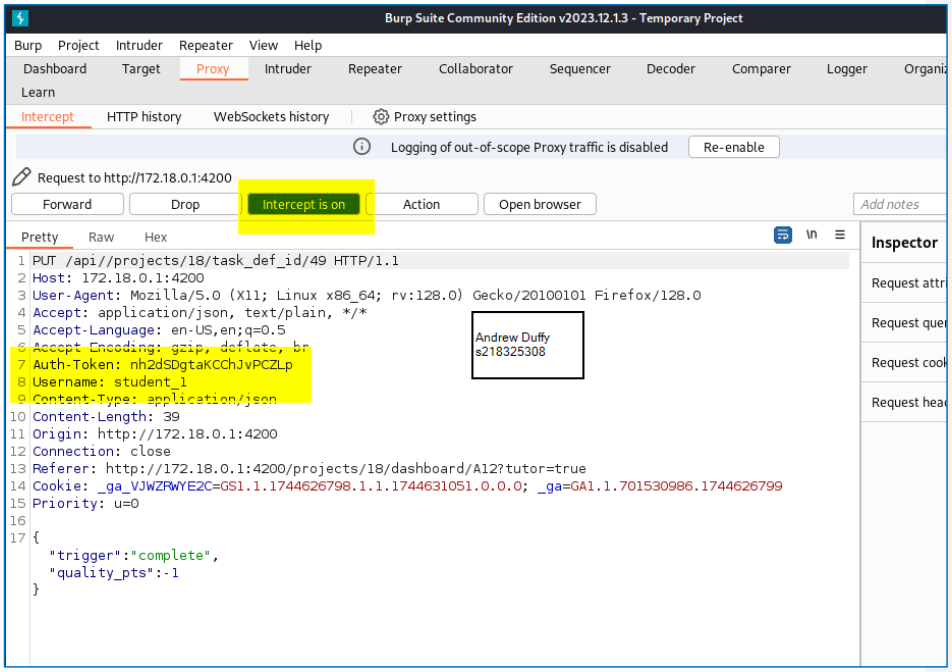


Figure – Intercepted student request to update task

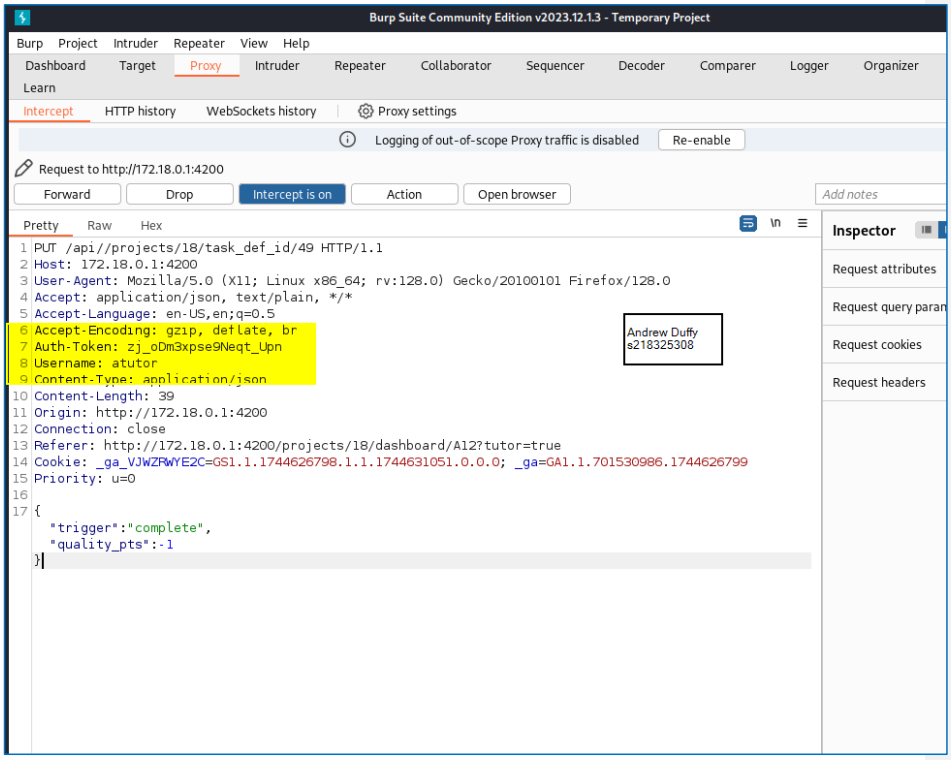


Figure – Request updated with stolen session information

3. Forward the captured traffic or turn off intercept to finalise the request. Take note of the notification change and task update.

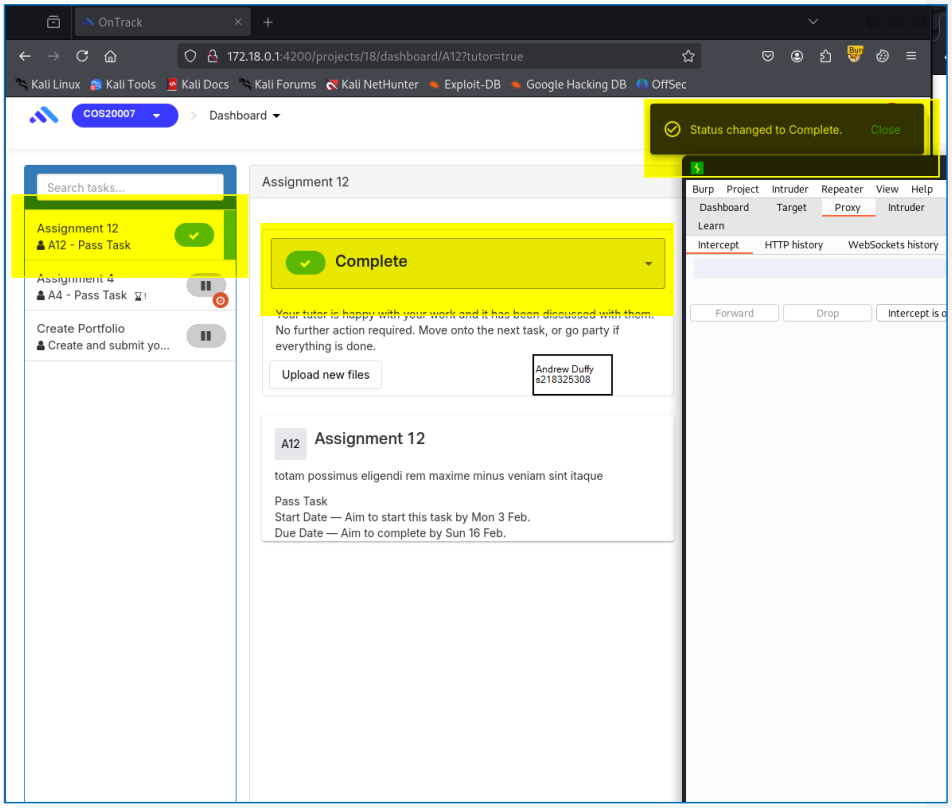


Figure – Successfully updated own task by Student

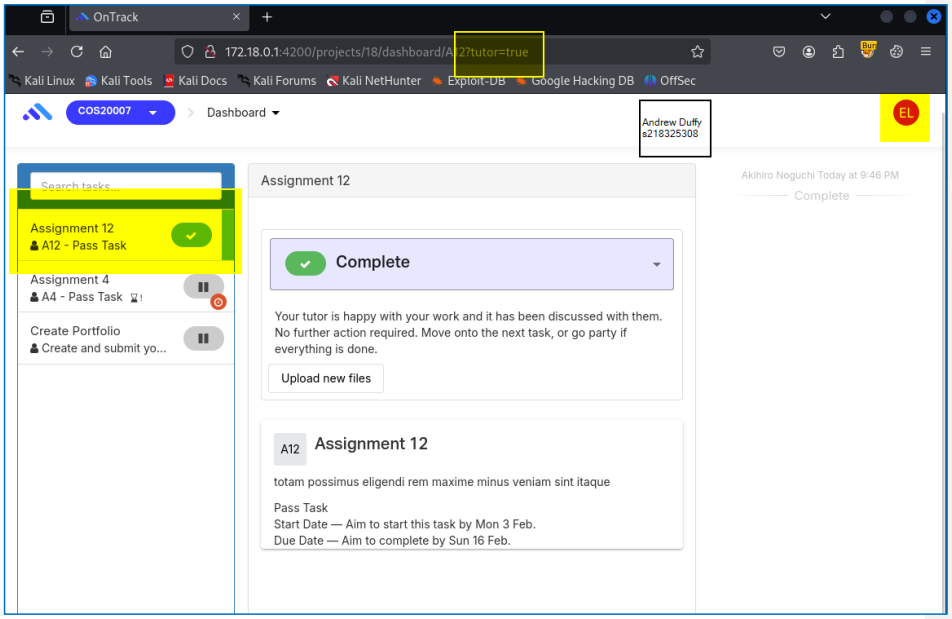


Figure – Successfully updated own task by Student

4. This privilege escalation chain demonstrates how an initially low-impact finding – broken access control leading to exposed privileged UI elements - can act as a foothold for a motivated attacker. In this case, the presence of administrative functionality visible to a student-level user led to further investigation. While the session hijacking vulnerability was ultimately used to bypass backend validation and escalate privileges, the key takeaway is that any secondary vulnerability could have been chained. This reflects typical threat actor behaviour – curiosity and drive triggered by exposed privileged functionality, followed by persistent attempts to achieve an exploit. This attack chain highlights how a relatively low-risk issue, such as improperly restricted UI components, can escalate when paired with other vulnerabilities. At its core, this is a **Broken Access Control vulnerability**, where access to privileged actions is not properly restricted, allowing a low-privileged user to perform actions outside their intended role.

Remediation Advice

The chain demonstrated linking broken access controls on client side rendered administrative functions and poor session token management.

Mitigations:

- Implement role-based access control (RBAC) on both the client and server side to ensure that only authorised users can access and execute privileged functionality.
- Avoid use of client-side parameters (e.g. ?tutor=true) to control access to privileged functionality. Enforce access restrictions in backend only.
- Implement a ‘deny-by-default’ approach to access control, where all privileged actions are restricted unless explicitly granted to the authenticated user’s role.
- Review and implement mitigations related to Session Token vulnerabilities as documented in their respective findings.

References

- OWASP Top 10 – Broken Access Control
- OWASP Cheat Sheet – Authorization
- MITRE CWE-284: Improper Access Control

Contact Details

Name/Teams: Andrew Duffy
Email: s218325308@deakin.edu.au

Pentest Leader Feedback.

Amazing finding, Andrew! The report is also well written. Good work!

6.13: Privilege Escalation via Insecure Cookie Manipulation

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Ashara Konara	AppAttack	Pen-tester	OnTrack	Shehani Wickremasekera	No

Was this Finding Successful?
Yes

Finding Description

The web application hosted at http://172.18.0.1:4200/ suffers from insecure cookie-based role management, where user roles are stored and trusted on the client side (in local storage) without sufficient server-side validation. By inspecting and modifying the role value (e.g., changing from student_1 to admin) using browser developer tools, a user can gain visual access to privileged interfaces such as "Add Users" and "Create Units".

While this manipulation exposes high-level features normally restricted to roles like admin, convenor, or tutor, attempts to execute privileged actions such as submitting forms or saving changes are blocked at the backend, confirming that server-side role enforcement is partially in place.

This issue falls under:

- **Broken Access Control**
- **Insecure Direct Object Reference (IDOR) via Cookies**

Though full privilege escalation was not achieved, the exposure of sensitive functionality to unauthorized users poses a significant risk, especially if paired with future or secondary exploits.

Risk Rating

Impact: Significant
Likelihood: High

Commented [DO1]: - change the impact and likelihood to reflect the corresponding ones which were shaded in the risk matrix.

- additionally, based on your evidence section, this finding should have an impact value of "Minor" at best. Please adjust your risk rating and risk matrix accordingly.

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

	enough to impede regular activity.	to run normally.	able to run normally.	

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

If exploited, this vulnerability allows any user, including students, to impersonate higher-privileged users such as aadmin, aconvenor, or atutor by manipulating the role stored in the browser's local storage. This grants unauthorized visual access to administrative interfaces and privileged functionality, such as user management and course creation panels.

Although actual administrative actions (like modifying data or creating users/units) cannot be executed due to backend validation, the ability to access and view these restricted areas still poses significant risks, including:

- Unauthorized insight into internal system structure and workflow
- Potential leakage of sensitive interface elements and logic
- Increased risk of targeted attacks through recon
- Breach of role-based access control principles
- Reduced trust and perceived security of the platform

While not a full compromise, this flaw weakens security boundaries and could serve as a steppingstone for more advanced exploitation if combined with other vulnerabilities.

Affected Assets

- Web Application at http://172.18.0.1:4200/
- User Role Authorization and Access Logic
- All endpoints are accessible only to privileged roles (admin, tutor, convenor)

Evidence

Step 1: Access the application and log in as a student

Log in using a low-privilege account, such as student_1.
Open **Developer Tools > Application > Storage > Local Storage** and observe a key/value pair that includes the user's role (e.g., role=student or similar).

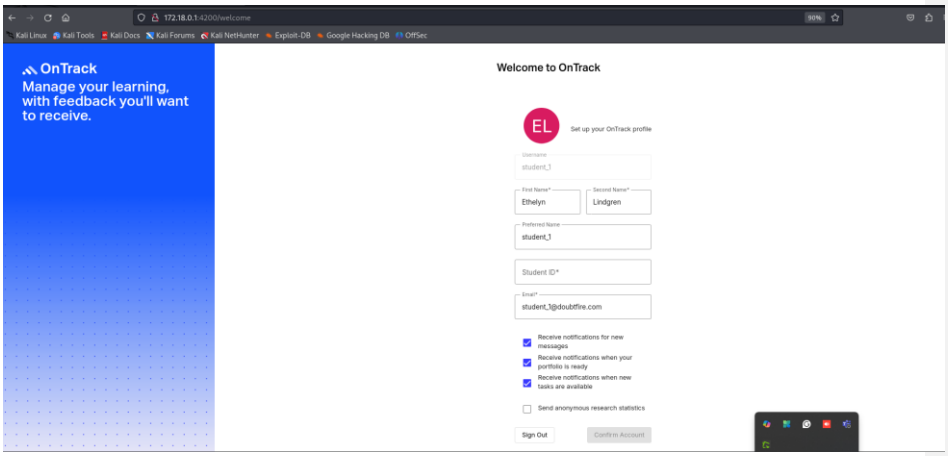


Figure 1: Login as a Student

Step 2: Modify the role in Local Storage (doubtfire_user)

Change the role value directly in **Local Storage** using browser developer tools.
For example, change role=student to role=admin, role=aconvenor, or role=atutor.

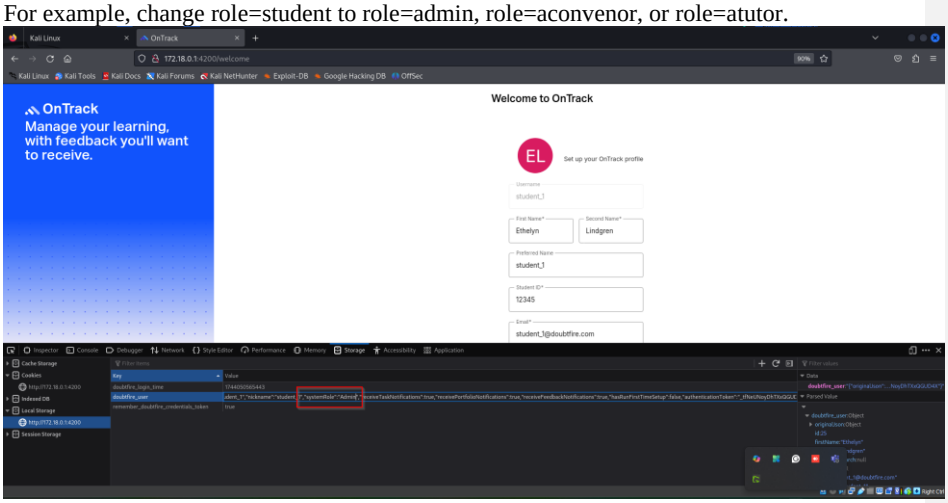


Figure 2: Change the System role to Admin

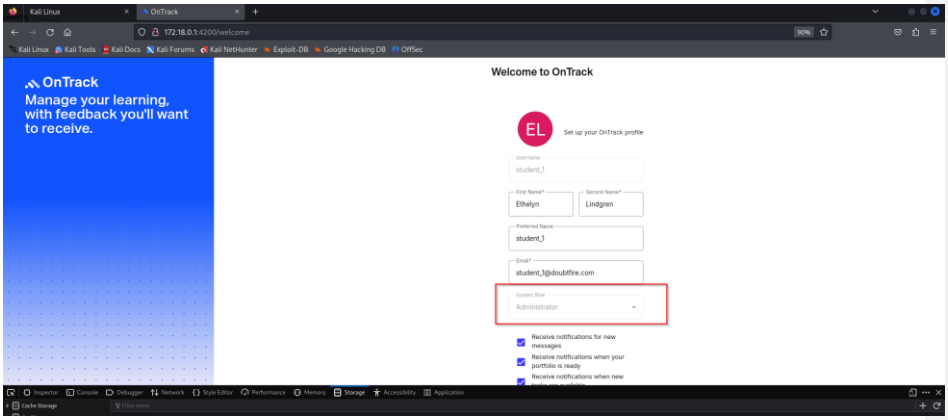


Figure 3: Successfully changed user role

Step 3: Refresh the application or navigate to a privileged page

After modifying the Local Storage data, refresh the browser or navigate to a restricted route. The system grants unauthorized access based on the new role, confirming privilege escalation.

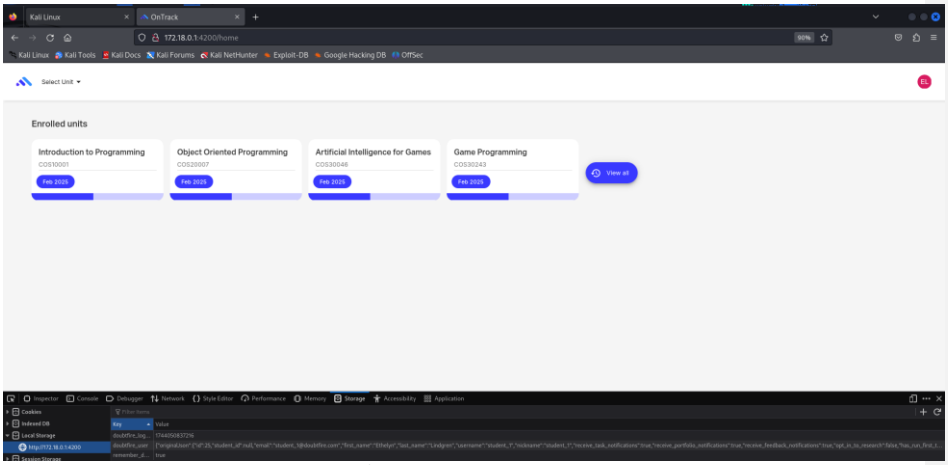


Figure 4: Log as student_1

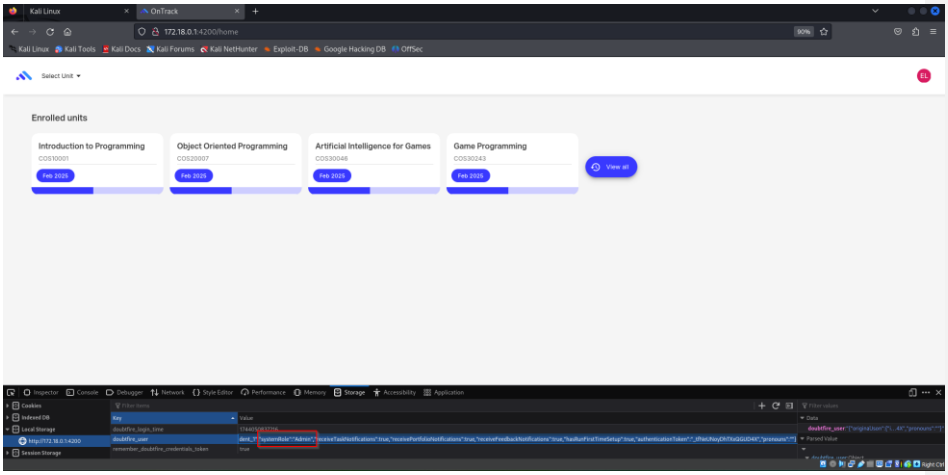


Figure 5: Changed system role again to Admin

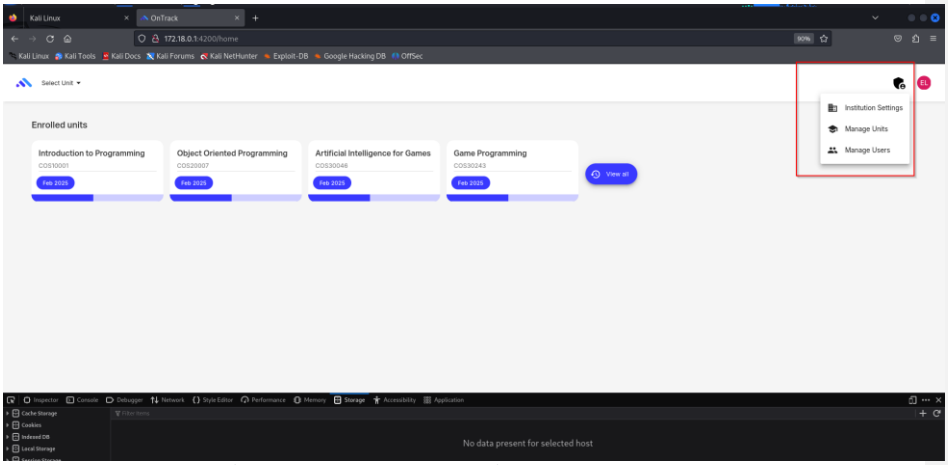


Figure 6: Successfully granted admin access

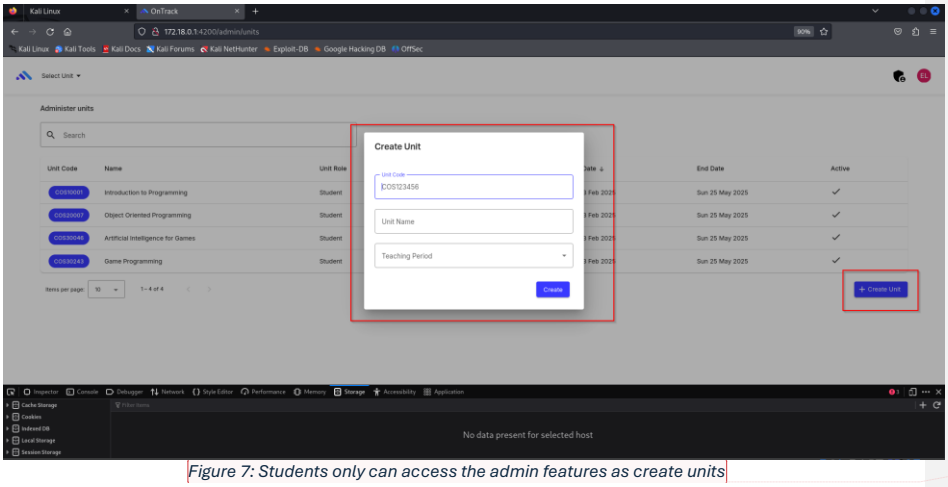


Figure 7: Students only can access the admin features as create units

Commented [DO2]: make it clear that the student can only access the create units feature, but not modify anything.

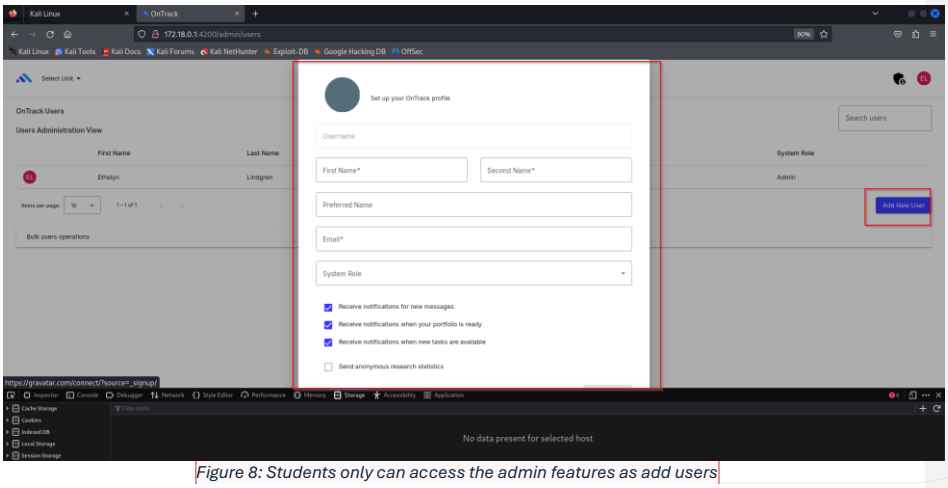


Figure 8: Students only can access the admin features as add users

Commented [DO3]: make it clear that the student can only access the add users feature, but not modify anything.

Remediation Advice

Summary:

The application currently trusts client-side cookies to determine user roles, which can be easily manipulated.

Mitigation Steps:

- Never rely on client-side data (e.g., cookies) for access control decisions
- Store role and session data on the server side, protected from user tampering
- If you use JWTs, ensure they are:
  - Properly signed with a strong secret
  - Validated on every request
- Apply server-side role-based access control (RBAC) logic for all sensitive routes and operations

- Implement logging and alerting suspicious privilege changes or abnormal access behavior

References

OWASP: Broken Access Control
OWASP: Insecure Direct Object Reference (IDOR)
OWASP: Session Management Cheat Sheet

Contact Details

Name: Ashara Shashintha Konara
Email: s223846206@deakin.edu.au

Pentest Leader Feedback.
Shehani Wickremasekera s223841302@deakin.edu.au

- Great work identifying the vulnerability and the document provides a well detailed presentation of the vulnerability.
- There are a few changes which are required in terms of the formatting.
    - The table for the impact criteria and likelihood criteria needs to have the shading for the selected risk rating column.
    - The impact and likelihood should not be highlighted but the font color changed to the correct color.
    - The steps should be black. The title for the steps can remain the same.
    - Please include yes or no for the question of “if this is a retested finding?”
- The references are not accessible. Please include the correct references.

6.14: Lack of Rate Limiting on Login Endpoint

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Fahad Alshamri	Ontrack	Pentester	AppAttack	Darryl Ooi	NO

Was this Finding Successful?
Yes

Finding Description

Using Burp Suite's Intruder module with a publicly available wordlist, I demonstrated that the login endpoint accepts unlimited login attempts without triggering any defenses such as rate limiting, CAPTCHA, or account lockout.
Although no successful credential recovery was achieved, the lack of protections leaves the system vulnerable to brute-force attacks.

Risk Rating

Impact: Major
Likelihood: High

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity.	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally.	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

The absence of rate limiting on OnTrack's login API endpoint allows attackers to launch brute-force attacks without restriction. Over time, this could lead to unauthorized access to student, staff, or administrative accounts. If compromised, an attacker could:

- Access sensitive academic records or student submissions,
- Disrupt platform functionality by impersonating users,
- Initiate privilege escalation attacks,

- Or expose personally identifiable information (PII), leading to privacy breaches and potential regulatory violations.

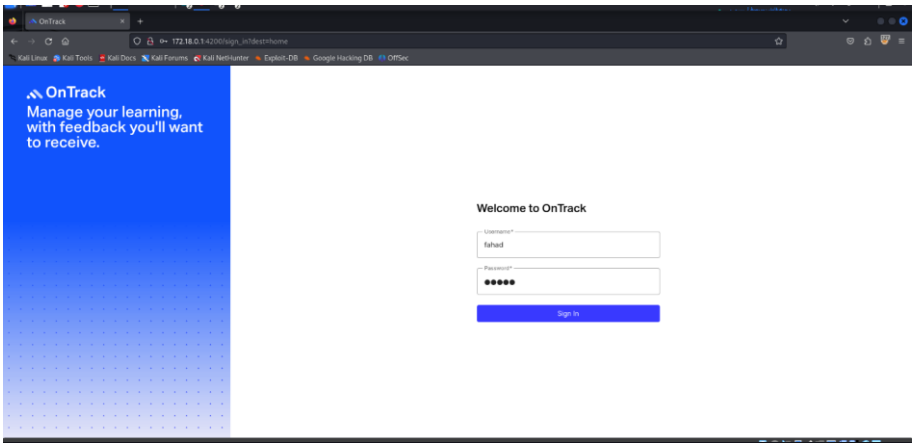
Affected Assets

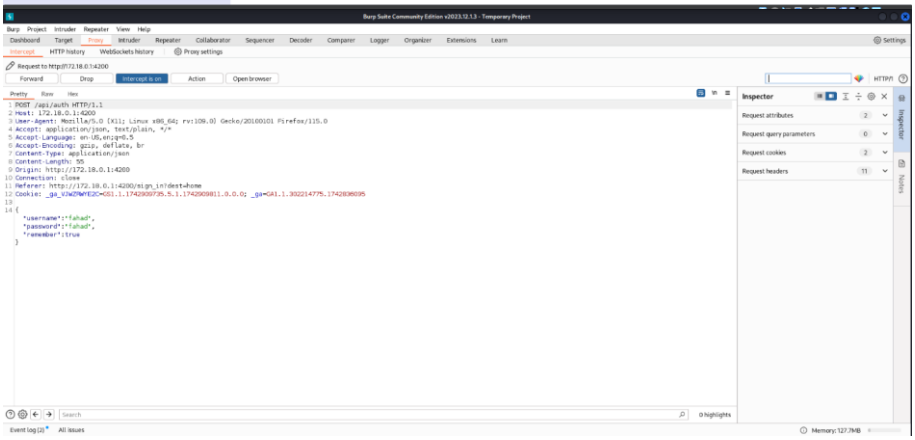
OnTrack API Login Endpoint (/api/auth)
User Credentials and Sessions

Evidence

Step 1: Capture Login Request

Using Burp Suite Proxy, I intercepted a login request while submitting the credentials (fahad / fahad) on the OnTrack login page. The request body was JSON, targeting the /api/auth endpoint.





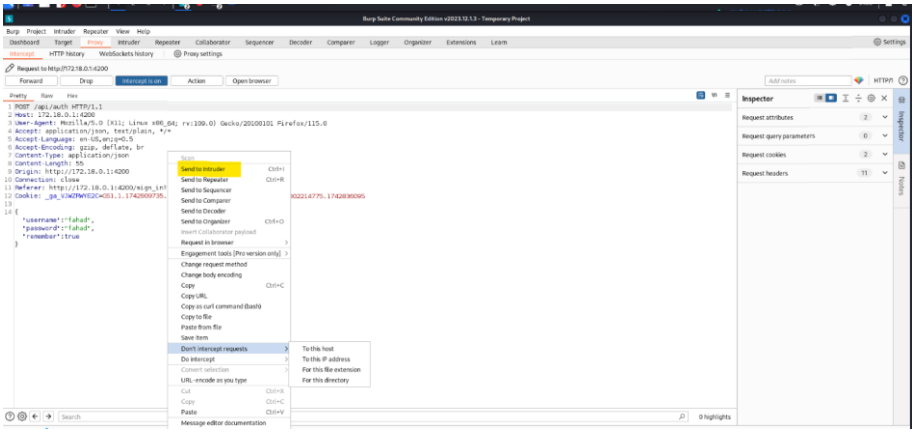
Step 2: Send to Intruder

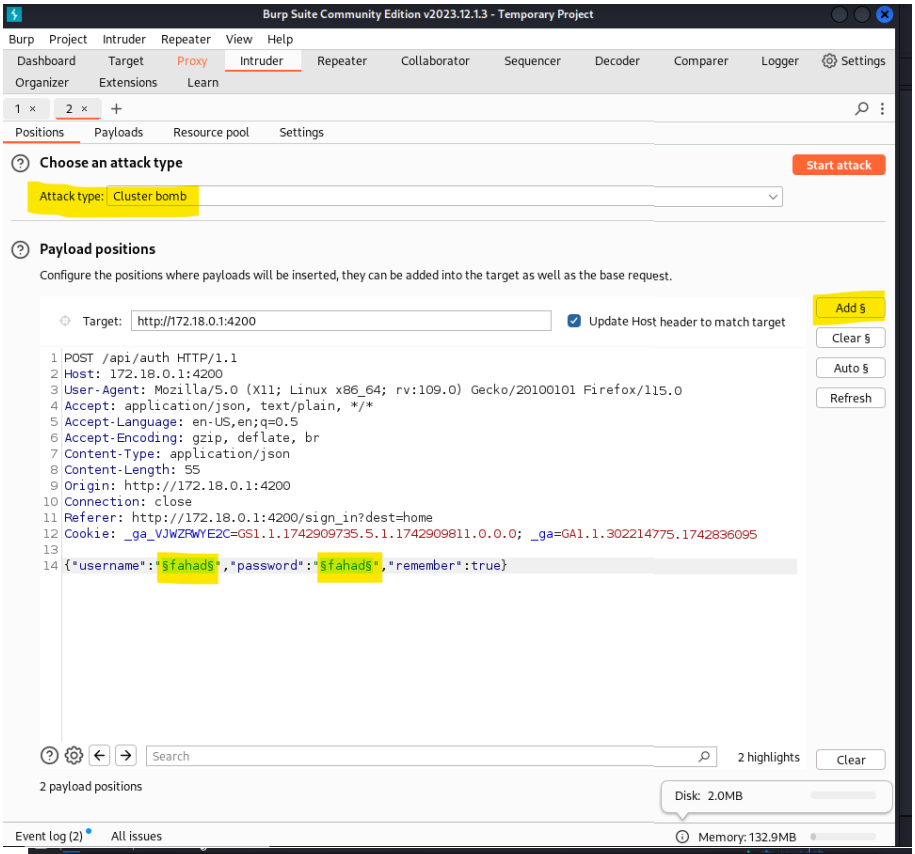
The intercepted login request was forwarded to Burp Suite's Intruder module for brute-force simulation. Within Intruder, payload positions were configured for both the username and password fields in the JSON request body. To test all possible combinations of credentials, the Cluster Bomb attack type was selected. This method allows the tool to iterate through two

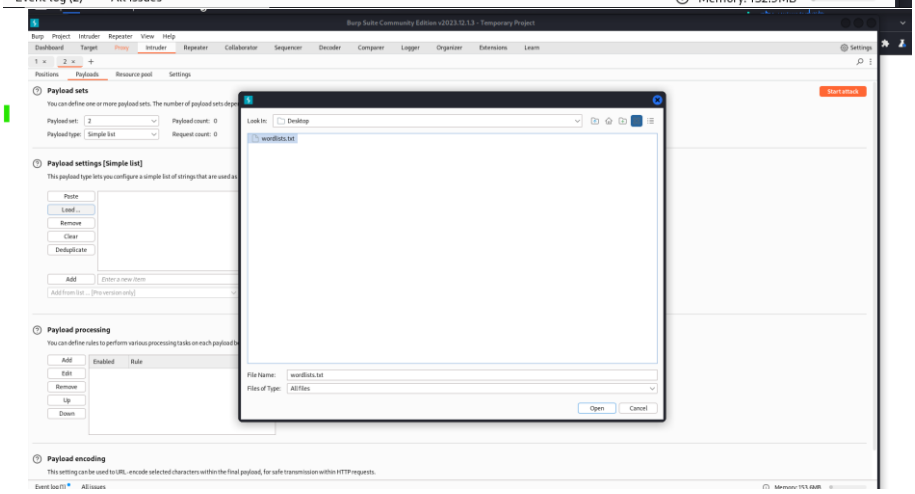
independent payload sets—one for usernames and one for passwords—effectively simulating a wide range of login attempts.

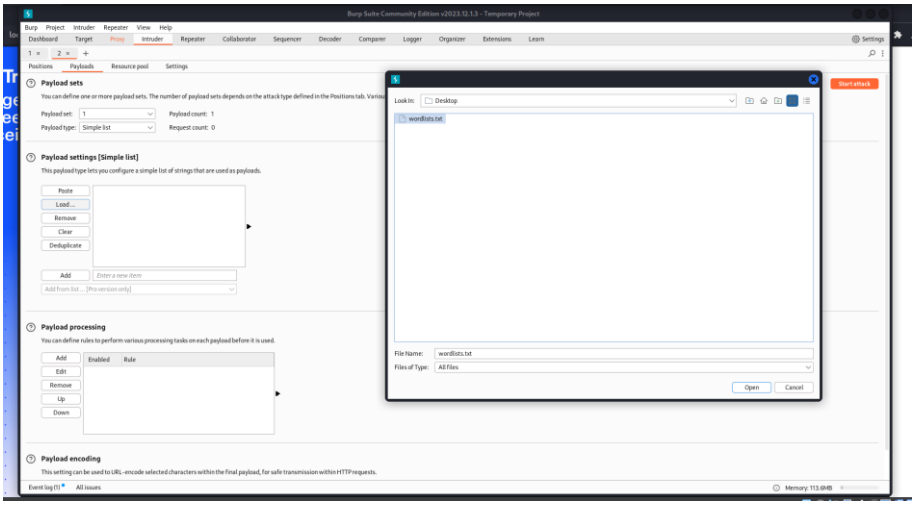
A publicly available wordlist named wordlist.txt was used as the source for both payload sets. This wordlist was obtained from the OpenEthereum GitHub repository and contains a large set of commonly used terms suitable for brute-force testing. The wordlist can be accessed at:

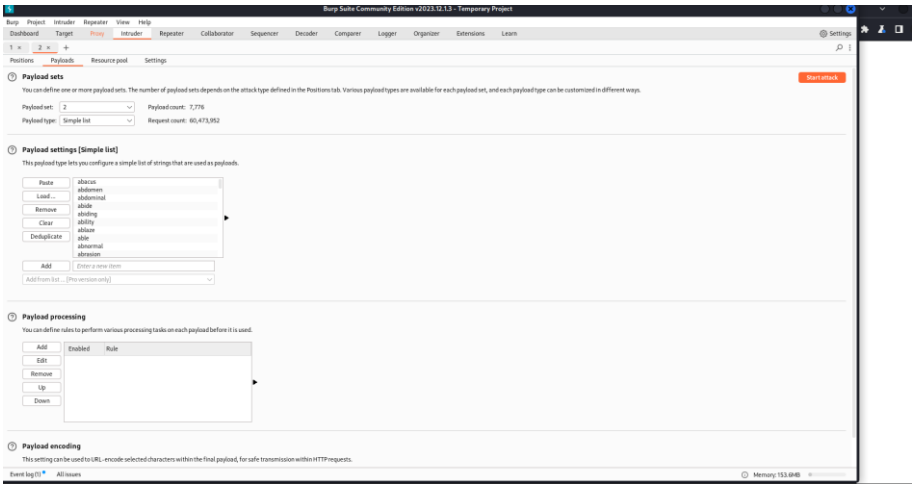
https://github.com/openethereum/wordlist/blob/master/res/wordlist.txt




Burp Suite Community Edition v2023.12.1.3 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Settings
Organizer Extensions Learn
1 × 2 ×
Positions Payloads Resource pool Settings
Choose an attack type
Start attack
Attack type: Cluster bomb
Payload positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.
Target: http://172.18.0.1:4200
Update Host header to match target
Add §
Clear §
Auto §
Refresh
1 POST /api/auth HTTP/1.1
2 Host: 172.18.0.1:4200
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 55
9 Origin: http://172.18.0.1:4200
10 Connection: close
11 Referer: http://172.18.0.1:4200/sign_in?dest=home
12 Cookie: _ga_VJWZRWYE2C=GS1.1.1742909735.5.1.1742909811.0.0.0; _ga=GA1.1.302214775.1742836095
13
14 {"username":"$fahad$","password":"$fahad$","remember":true}
Search
2 highlights
Clear
2 payload positions
Disk: 2.0MB
Event log (2)
All issues
Memory: 132.9MB

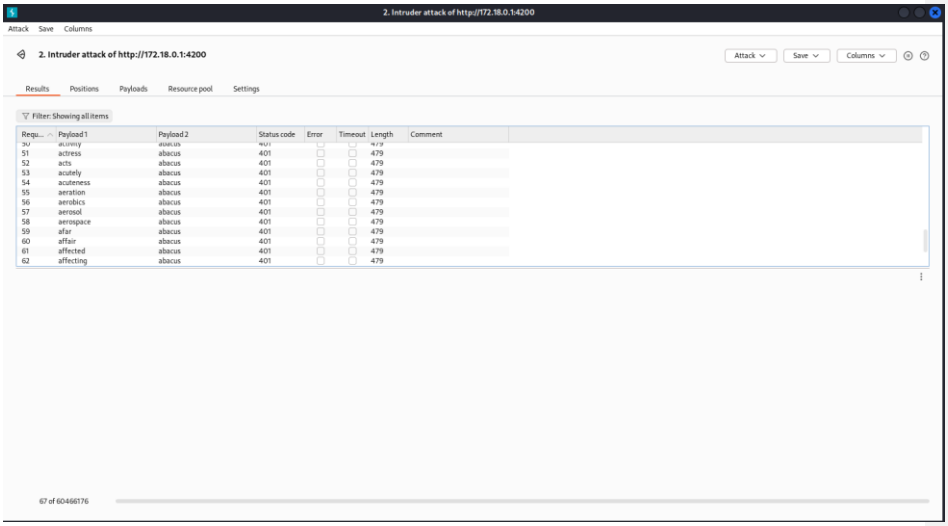






Step 3: Run the Attack

Although the attack using a realistic public wordlist did not reveal valid login credentials, it was observed that the server permitted repeated login attempts without implementing rate limiting or account lockouts, confirming the vulnerability.


2. Intruder attack of http://172.18.0.1:4200
Attack
Save
Columns
2. Intruder attack of http://172.18.0.1:4200
Attack
Save
Columns
Results
Positions
Payloads
Resource pool
Settings
Filter: Showing all items
Requ... Payload 1 Payload 2 Status code Error Timeout Length Comment
51 actress abacus 401 479
52 acts abacus 401 479
53 acutely abacus 401 479
54 acuteness abacus 401 479
55 aeration abacus 401 479
56 aerobics abacus 401 479
57 aerosol abacus 401 479
58 aerospace abacus 401 479
59 afar abacus 401 479
60 affair abacus 401 479
61 affected abacus 401 479
62 affecting abacus 401 479
67 of 60466176

Remediation Advice

To protect against brute force attacks, it is important to implement rate limiting on login attempts and introduce CAPTCHA challenges after a defined number of failures. Accounts should be temporarily locked after several incorrect login attempts to slow down automated attacks. Additionally, the system should actively monitor and alert administrators of repeated login failures. Finally, enforcing strong password policies and enabling multi-factor authentication (MFA) adds an extra layer of protection against unauthorized access.

References

OWASP Brute Force Attack: https://owasp.org/www-community/attacks/Brute_force_attack
OWASP Rate Limiting: https://owasp.org/ASVS/v4.0.3/sections/13.3-Rate-Limiting.html
wordlists: https://github.com/openethereum/wordlist/blob/master/res/wordlist.txt

Contact Details

Name: Fahad Alshamri.
Email: falshamri@deakin.edu.au

Pentest Leader Feedback.

Revision 1

- Please change the blue text to black.
- Additionally, change the orange text in the evidence section to black as well (excluding step headings).
- I would recommend looking in the “4.Ready for Final Report” for what we are looking for in documentation and follow the AppAttack Findings Checklist.docx
- Please provide a valid word list used for your brute force attack. You should not be creating a wordlist with credentials you already know for the web application. Otherwise, this is a proof-of-concept testing rate limiting, and not a successful brute force attack. Please adjust your report accordingly.

Revision 2

- Looks good overall, just a couple of changes required.
- Title the finding lack of rate limiting, as brute force is an attack and not a vulnerability.
- Please make the business impact specific to OnTrack and add more detail for the listed impacts (i.e. unauthorized access, data breaches, and misuse of the system).
- In step 2 of your evidence please add a screenshot of the payloads tab in burp suite to show the payloads. Additionally, in the evidence description, mention where the payloads/wordlists can be obtained (e.g. refer to your wordlist link provided in the references section).
- In your step 3 evidence screenshot, the brute force attack is receiving responses with status code 500. However, responses should have status code 401 instead. Please retest the attack and attach a screenshot with correct status codes received.

6.15: Insecure Transmission of Credentials

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Shehani Wickremasekera	App Attack	Pen Tester / Secure Code Reviewer	OnTrack	Filipe oliveira	No

Was this Finding Successful?
Yes

Finding Description

The application's login feature is accessible over an unprotected HTTP connection (http://172.18.0.1:4200/sign_in). Sensitive information is sent across the network in plaintext without encryption when users enter their login credentials (password and username). This poses a significant security concern, particularly when users visit the website using public or untrusted networks (such as coffee shops, airport Wi-Fi, etc.).

Any attacker on the same network (for example, using ARP spoofing or passive packet sniffing) can intercept and examine login credentials using easily accessible tools like Wireshark or Burp Suite because HTTP does not encrypt data while it is in transit. The credentials that have been intercepted can then be used to impersonate users, access the platform without authorisation, or carry out additional system assaults.

When login forms are found on insecure websites, users and developers are alerted to the problem by contemporary browsers and development tools.

In addition to putting user accounts at serious risk, failing to enforce HTTPS with a valid SSL/TLS certificate may also be against industry best practices and compliance standards (e.g., OWASP, GDPR, PCI-DSS).

Risk Rating

Impact: **Severe**
Likelihood: **High**

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

	impede regular activity.	able to run normally.		

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

Users of the client's learning platform are at risk of credential theft due to this vulnerability. Any hacker keeping an eye on the network, especially on public or unprotected Wi-Fi, can intercept usernames and passwords entered through the login form. If an attacker gains access to administrative accounts, they could:

Access or alter private student information, such as grades, contributions, and personal information.

- Upload or modify scholarly materials or tests.
- Operational disruption occurs when authorised users are locked out.
- To increase privileges or obtain access to other internal systems, move laterally within the system.
- Undermine confidence in the platform's security, which could have an impact on compliance and reputation.

Particularly if the platform handles personally identifiable information (PII) or educational records subject to regulations like FERPA or GDPR, this could result in significant data protection violations and legal repercussions.

Affected Assets

- Web Application Login Page
  http://172.18.0.1:4200/sign_in - This page collects user credentials without encryption.

- User Accounts - All user accounts logging in over HTTP are potentially at risk, including:

  - Students

  - Staff/Administrators

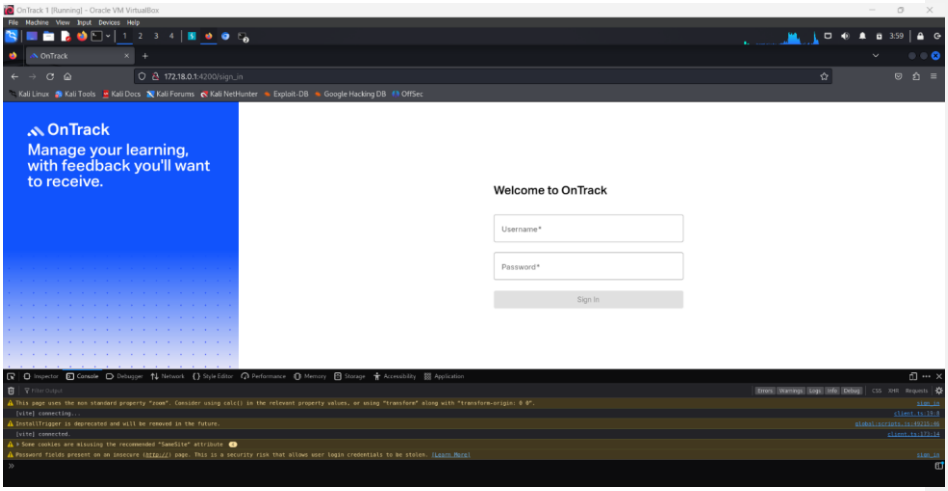
  - Supervisors or content reviewers

- Authentication System & Session Management - Since login credentials are transmitted insecurely, session integrity is also potentially compromised.

Evidence

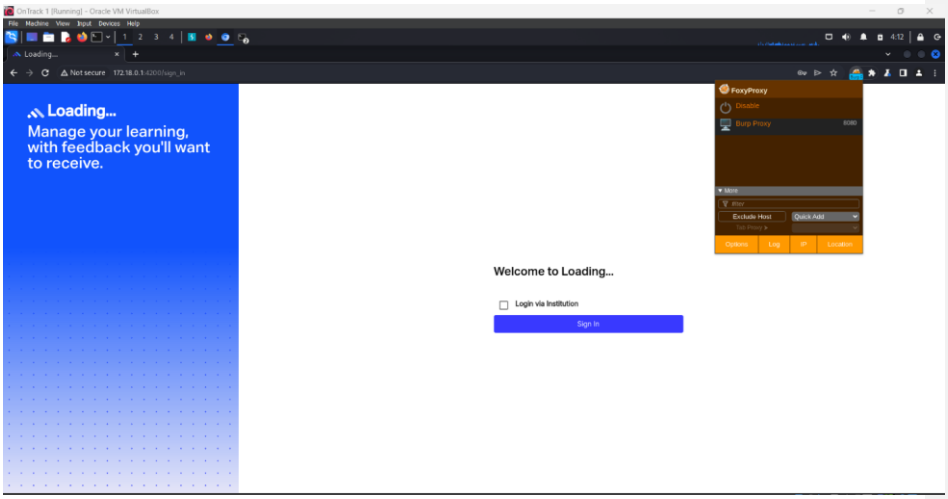
Step 1: Opening the OnTrack Signing in Link and Console Log

The screenshot provided below is of the OnTrack sign in page and the console warning tab which provides an error stating that the site isn't using HTTPS and is simply using HTTP. The initial step was to open the web browser and type in the URL which is http://172.18.0.1:4200/sign_in. The next step was to right click anywhere on the webpage and select "Inspect".



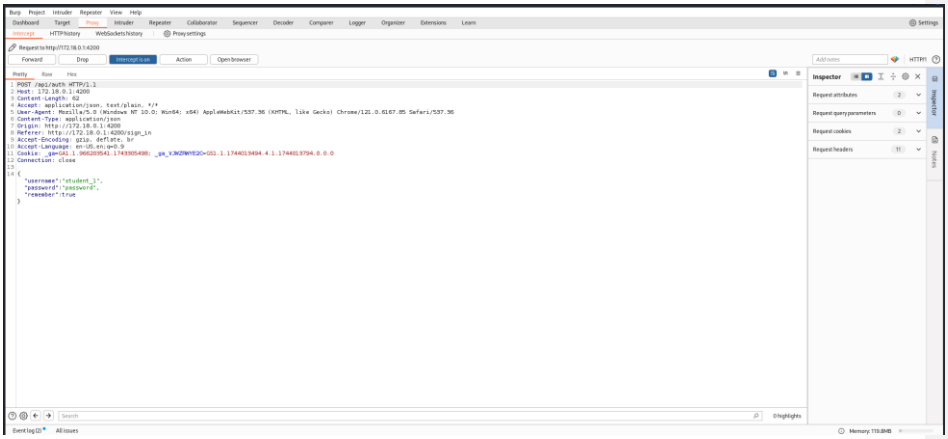
Step 2: Configuring the browser to use Burp Suite

Next was to open Burp Suite and intercept the login request for which the browser had to be configured to use Burp as a proxy (127.0.0.1:8080). This configuration is evident in the screenshot below.



Step 3: Intercepting the Sign-In page and Obtaining the Credentials

The next step was to intercept the login page by entering the credentials. As you can see the credentials are visible on Burp Suite. This is proof that the credentials are not encrypted and shared via HTTP.



Remediation Advice

- Enforce HTTPS: Use a valid TLS certificate to set up the web server to serve all content over HTTPS.
- Redirect HTTP to HTTPS: Make sure that all HTTP traffic is automatically transferred to HTTPS by implementing redirection rules.
- Implement HTTP Strict Transport Security (HSTS): To make sure that modern browsers only allow HTTPS access, include the HSTS header (Strict-Transport-Security).
- Secure Cookies: To stop leaks and JavaScript access, make sure authentication cookies have the Secure and HttpOnly settings set.

- Conduct Frequent Security Audits: Check the online application frequently for out-of-date protocols or incorrect SSL/TLS setups.

References

1. Transport Layer Security - OWASP Cheat Sheet Series
2. Strict-Transport-Security - HTTP | MDN
3. Proxy intercept - PortSwigger
4. CaptureSetup - Wireshark Wiki

Contact Details

Name – Shehani Natasha De Sayrah Wickremasekera
Email – s223841302@deakin.edu.au

Pentest Leader Feedback.

Filipe Oliveira S222478779@deakin.edu.au

- Great work
- Please just change the steps themselves (not the titles) to black text
- Great references, and very clear steps.

6.16: Client-Side Template Injection (CVE-2023-26116)

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Wahidullah Hashimi	OnTrack	Pen-Tester	AppAttack	Shehani Wickremasekera	No

Was this Finding Successful?
Yes

Finding Description

Upon testing for vulnerabilities in Ontrack, I used OWASP ZAP and discovered that it's using a vulnerable version of AngularJS (1.5.11), which is affected by CVE-2023-26116. This vulnerability affects AngularJS versions prior to 1.8.0 and can lead to two main security risks:

1. **Regular Expression Denial of Service (ReDoS):** This issue occurs when a system is forced to process a regular expression pattern match that takes an excessive amount of time, leading to Denial of Service (DoS). ReDoS can overwhelm the system and block access for legitimate users. However, when I attempted to perform a proof of concept (PoC) for this vulnerability on Ontrack, it was unsuccessful. This could be due to mitigations in place (such as input sanitization) or specific conditions required for the attack that were not met.
2. **Client-Side Template Injection (CSTI):** This vulnerability allows attackers to bypass expression sandboxing in AngularJS versions <1.8.0, enabling them to execute arbitrary JavaScript code in the browser. This is particularly dangerous when user input is interpolated into the DOM without proper sanitization. When I tested for this vulnerability on Ontrack, the PoC was successful, meaning an attacker could potentially inject malicious scripts into the application. This vulnerability can lead to Cross-Site Scripting (XSS), where attackers can steal sensitive data, such as user credentials or cookies, or manipulate the page to perform actions on behalf of the user.

The successful CSTI vulnerability highlights a critical security risk in Ontrack, while the ReDoS vulnerability did not appear to be exploitable in this specific instance.

Risk Rating

Impact: Major
Likelihood: Moderate

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

	impede regular activity.	able to run normally.		

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

The CVE-2023-26166 vulnerability in Ontrack poses serious risks to both the application and the education sector as they are the main users. Here's a breakdown:

Client-Side Template Injection (CSTI):

- **Security Breach:** Attackers can bypass AngularJS's sandboxing, executing malicious code in users' browsers, leading to data theft, session hijacking and potentially full account compromise.
- **Legal Risks:** Exposed sensitive data could lead to legal issues and penalties under data protection laws.
- **Operational Disruption:** Exploiting the vulnerability could disrupt business operations, causing downtime and lost productivity.
- **Financial Loss:** A breach could lead to significant expenses, including legal fees, regulatory fines, and the cost of incident response and recovery.

Regular Expression Denial of Service (ReDoS):

- **System Unavailability**: While the PoC didn't succeed, ReDoS could still cause slowdowns or crashes, blocking legitimate users from accessing Ontrack.

Affected Assets

Doubtfire API
Ontrack Web Application
AngularJS Library

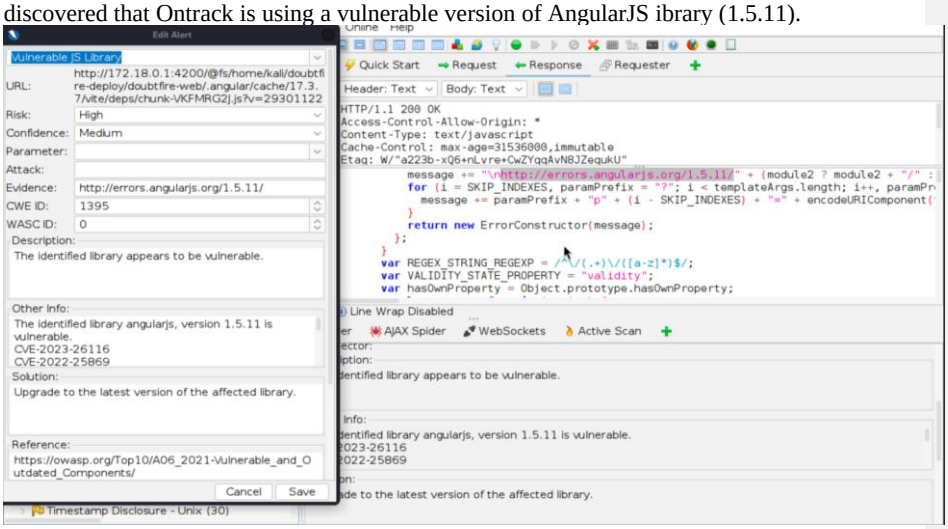
Evidence

Here is step by step instruction on how to exploit this vulnerability.

Step 1: Identify Vulnerability

Use OWASP ZAP to do a deep scan of Ontrack. After the scan is finished, ZAP will display vulnerabilities that exist in the web application. As shown in the screenshot below, ZAP

discovered that Ontrack is using a vulnerable version of AngularJS library (1.5.11).

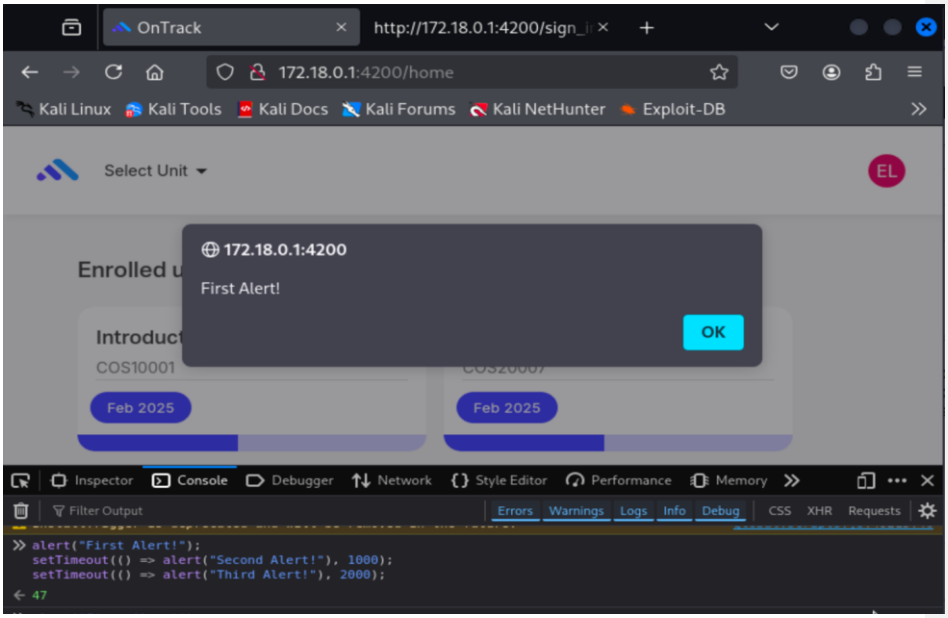


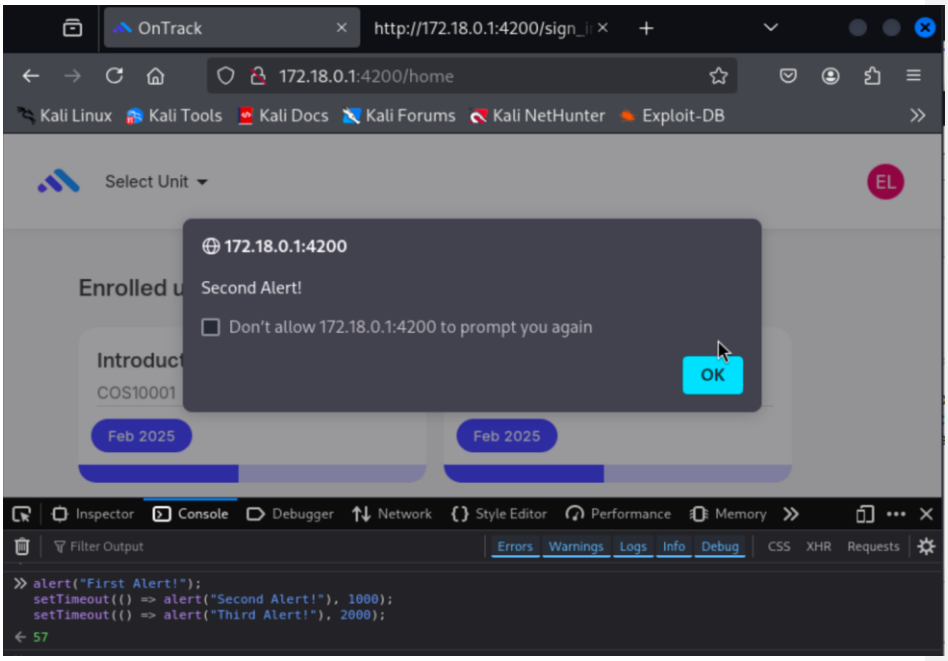
Step 2: Research

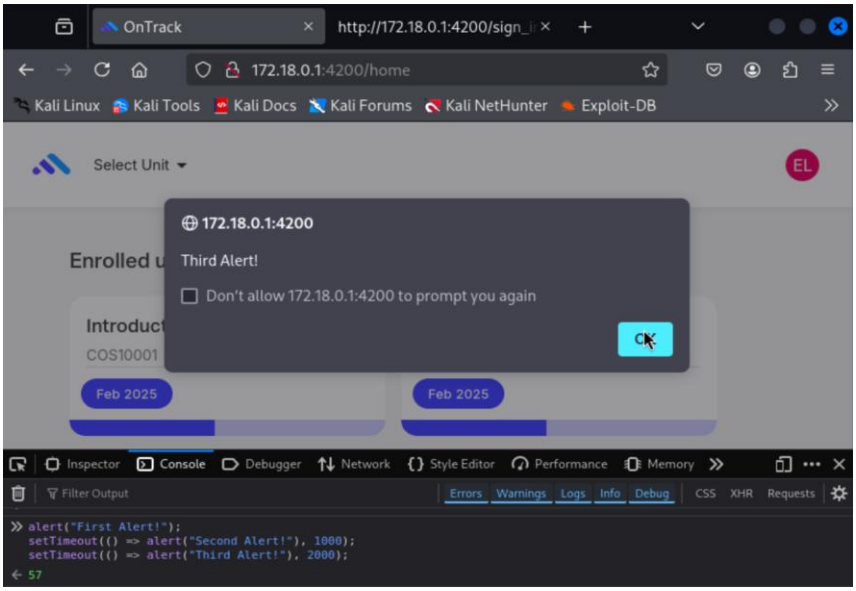
Once the vulnerability is identified, begin researching it in depth and explore different methods for exploiting it.

Step 3: Exploitation

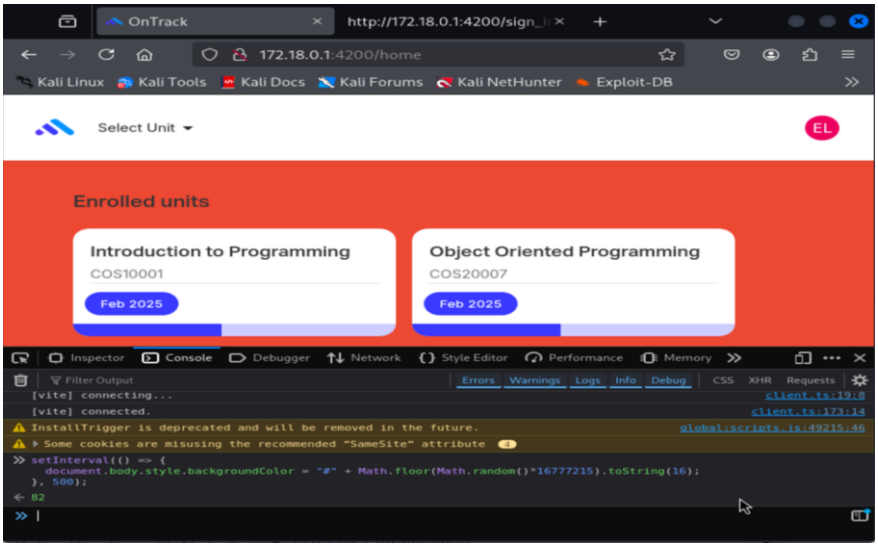
Now that you know about different methods of exploiting vulnerability, begin writing your script and testing your script on the web application to prove the existence of vulnerability. So, in this case open the developer console in your browser with Ontrack running and write a simple script “alert ("First Alert!"); setTimeout(() => alert("Second Alert!"), 1000); setTimeout(() => alert("Third Alert!"), 2000);” to trigger a series of alerts that interact with each other.

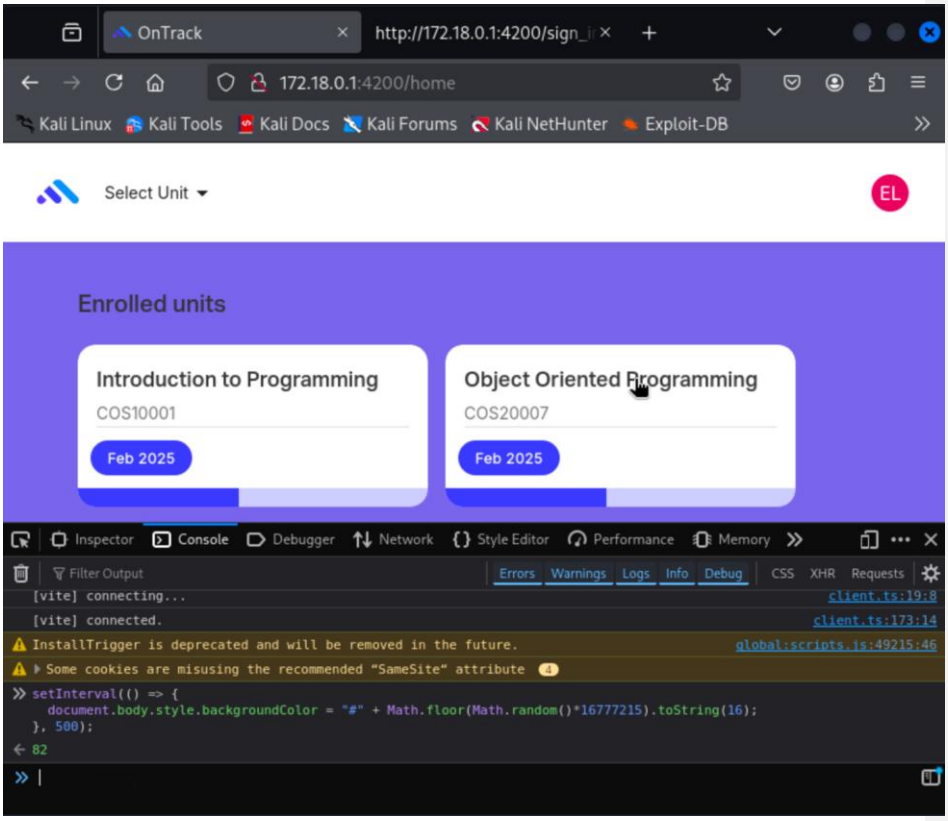

OnTrack
http://172.18.0.1:4200/sign_
172.18.0.1:4200/home
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Select Unit
EL
172.18.0.1:4200
First Alert!
OK
Enrolled u
Introduct
COS10001
Feb 2025
Feb 2025
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Filter Output
Errors
Warnings
Logs
Info
Debug
CSS
XHR
Requests
alert("First Alert!");
setTimeout(() => alert("Second Alert!"), 1000);
setTimeout(() => alert("Third Alert!"), 2000);
47


OnTrack
http://172.18.0.1:4200/sign_
172.18.0.1:4200/home
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Select Unit
EL
172.18.0.1:4200
Second Alert!
Don't allow 172.18.0.1:4200 to prompt you again
OK
Enrolled u
Introduct
COS10001
Feb 2025
Feb 2025
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Filter Output
Errors
Warnings
Logs
Info
Debug
CSS
XHR
Requests
alert("First Alert!");
setTimeout(() => alert("Second Alert!"), 1000);
setTimeout(() => alert("Third Alert!"), 2000);
57



To further test this vulnerability, use this script “setInterval(() => {
document.body.style.backgroundColor = "#" +
Math.floor(Math.random()*16777215).toString(16);
}, 500);” to create a colour changing background for Ontrack. As shown in the screenshot below the background colours are changing in Ontrack.





Summary

Ontrack is using AngularJS version 1.5.11, which is vulnerable to Client-Side Template Injection (CSTI). I successfully exploited this by injecting JavaScript directly into the browser's console, triggering alert messages and a colour-changing background. This confirmed that the application allows arbitrary code execution, posing a serious risk of session hijacking, data theft, and user manipulation.

Remediation Advice

- **Upgrade AngularJS to v1.8.0 or later** – The issue is fixed in newer versions.
- **Avoid using {{userInput}} in the DOM** – Never bind untrusted user input directly into the template.
- **Enable AngularJS expression sandboxing** – If upgrading isn't immediately possible, apply AngularJS's strict contextual escaping (SCE).
- **Input validation and sanitization** – Ensure all inputs are properly escaped and validated.

References

Herodevs. (n.d.). CVE-2023-26116 - Vulnerability Directory. Available at: https://www.herodevs.com/vulnerability-directory/cve-2023-26116 [Accessed 7 Apr. 2025].

Synk Security (n.d.) Regular Expression Denial of Service (ReDoS). Available at: https://security.snyk.io/vuln/SNYK-JS-ANGULAR-3373044 [Accessed 7 Apr. 2025].

NVD (n.d.) CVE-2023-26116 Detail. Available at: https://nvd.nist.gov/vuln/detail/cve-2023-26116 [Accessed 7 Apr. 2025].

Contact Details

Wahidullah Hashimi
s223397352@deakin.edu.au

Pentest Leader Feedback.

Shehani Wickremasekera s223841302@deakin.edu.au

- Good job identifying the vulnerability and documenting it.
- A well detailed report.
- References have all the links and are accessible.
- The only changes which need to be made are the titles of the steps. Please remove the brackets and capitalize the first letter in Step 1. Also, please add a space between the summary paragraph and the Remediation Advice as well as the other paragraph and sections need to be spaced out as well. Apart from that everything else looks good.

6.17: Unauthenticated API Endpoint Disclosure

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Wishwa Mahanama	Appattack	Pen-tester	OnTrack	Shehani Wickremasekera	No

Was this Finding Successful?
Yes

Finding Description

While testing the OnTrack application, I discovered that the endpoint /api/auth/method could be accessed without logging in. This endpoint reveals a small piece of backend information — specifically, it tells what type of login method the system uses:
{"method":"database","redirect_to":null}

Even though this might look like a simple message, it's not something that should be available to everyone. Attackers can use this kind of information to better understand how the system works internally and plan more targeted attacks.
I found this by using Burp Suite to watch the traffic between the browser and the server while I was on the login page. I noticed this request in the background and tested it directly in Burp Repeater and in a browser. Both tests showed that you don't need to be logged in to get this information.

Risk Rating

Impact: Significant
Likelihood: Moderate

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage	Risk that holds minor form of impact, but not significant	Risk that holds enough impact to be somewhat of a threat. Will	Risk that holds major impact to be of threat. Will cause	Risk that holds severe impact and is a threat. Will cause

and regular activity can continue.	enough to be of threat. Can cause some damage but not enough to impede regular activity.	cause damage that can impede regular activity but will be able to run normally.	damage that will impede regular activity and will not be able to run normally.	critical damage that can cease activity to be run.

Likelihood				
Rare	**Unlikely**	**Moderate**	**High**	**Certain**
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

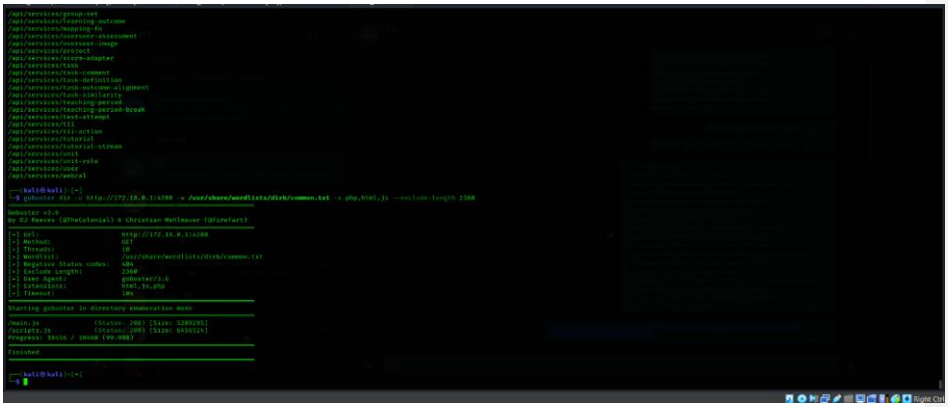
Disclosing authentication mechanisms allows attackers to tailor their attacks more effectively. For example, knowing that the system uses "database" authentication may encourage brute-force or credential-stuffing attacks, where attackers try common username and password combinations.
If an attacker knows the system relies on a database for authentication, they may also attempt SQL injection or database enumeration to retrieve credentials, user roles, or sensitive personal information. This could lead to full database compromise if chained with another vulnerability, such as SQL injection or broken access control.
Although this endpoint seems minor, it exposes unnecessary backend details to unauthenticated users. In the worst case, this could lead to lateral movement within the system or privilege escalation when combined with other exposed endpoints

Affected Assets

- OnTrack Web Application (http://localhost:4200)
- Endpoint: /api/auth/method

Evidence

Step 1: Step 1: Identify API endpoints for testing, I used the following methods
Gobuster for Directory Enumeration:I ran Gobuster on the target application to discover any exposed files or paths. This revealed main.js and scripts.js, which likely contain frontend logic and API calls

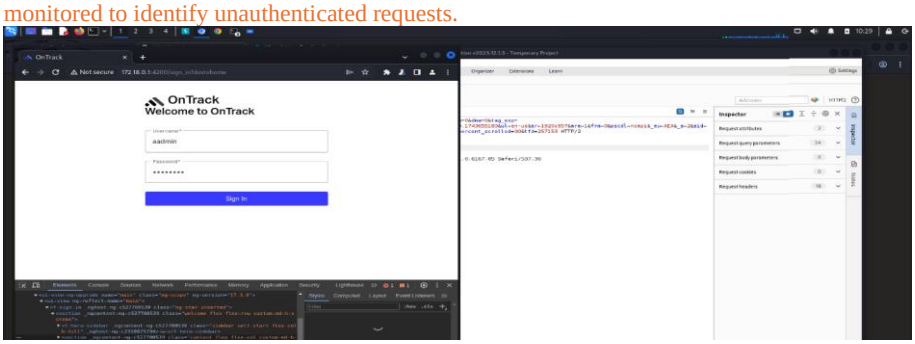


JavaScript File Analysis: I ran multiple grep commands on the JavaScript files to find references to API endpoints:

grep -oP "/api/[^\"']+" main.js

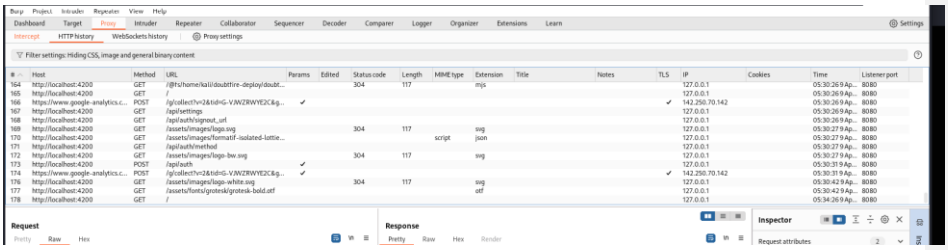
strings main.js | grep -i "/api/"

Step 2: Used Burp Suite's browser to access the OnTrack login interface. HTTP traffic was monitored to identify unauthenticated requests.



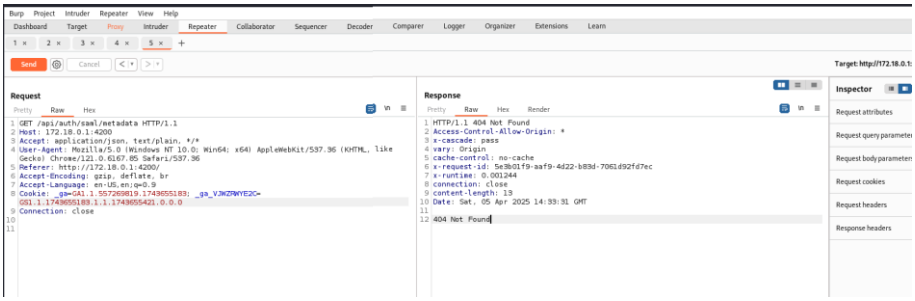
Step 3: Identify a Target Endpoint

This screenshot from Burp Suite's HTTP History shows various API and static asset endpoints that were automatically captured while browsing the OnTrack login page



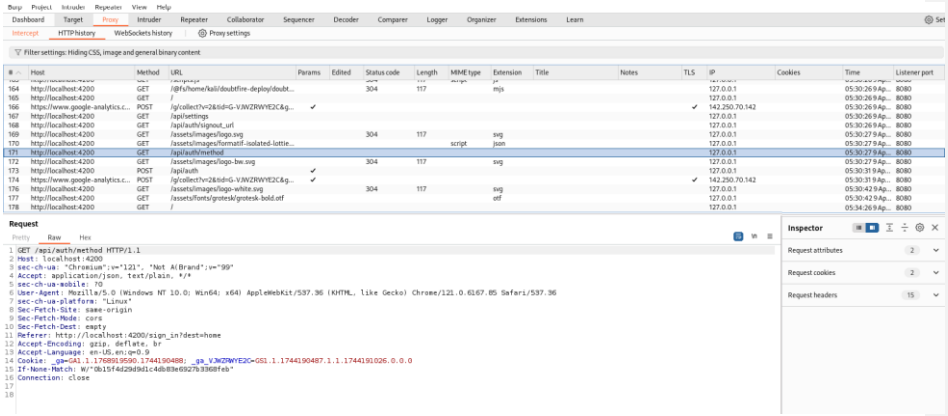
Step 4: Attempt Unauthorized API Access

Sent a request to /api/auth/saml/metadata using Burp Repeater. Got a 404 Not Found — endpoint likely does not exist.



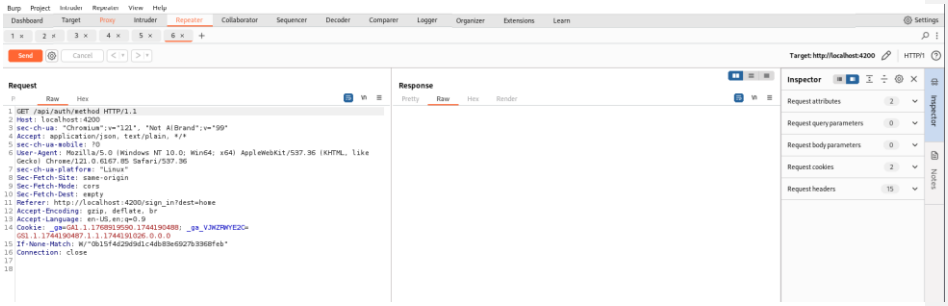
Step 4: Reproduce the Vulnerability

This screenshot from Burp Suite's Proxy history shows that the /api/auth/method endpoint was called automatically by the frontend during the login page load. I was not logged in at this time, yet the server responded with a 200 OK, indicating the endpoint is accessible to unauthenticated users.



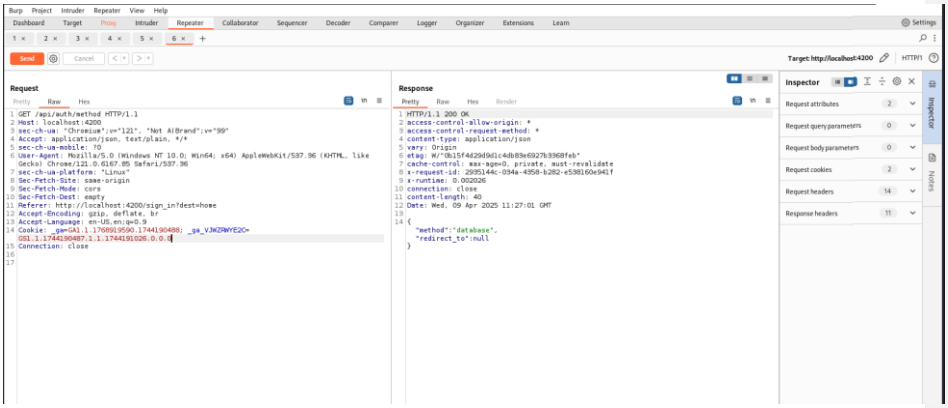
Proxy History: Identifying the Endpoint

In this screenshot, I manually sent a GET request to /api/auth/method using Burp Suite Repeater. I made sure not to include any authentication headers or valid session cookies. Despite that, the server still responded, confirming the endpoint does not require authentication.



Repeater Request Without Authentication

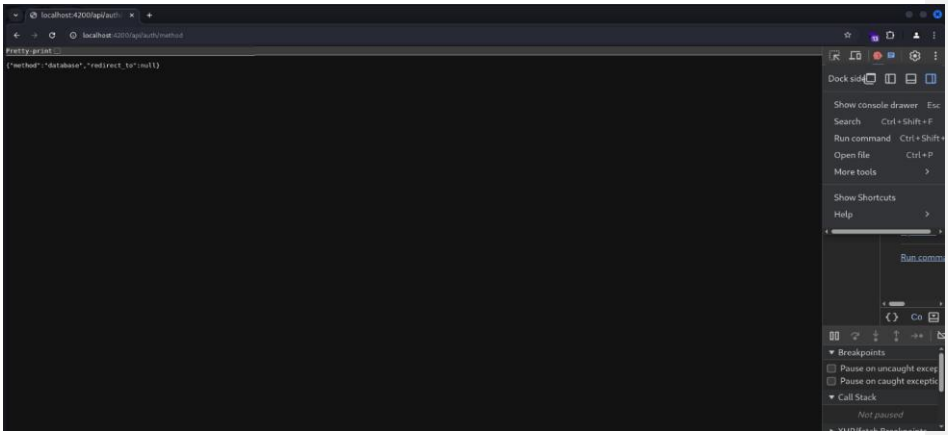
Here, the server responds to the unauthenticated Repeater request with a 200 OK status and a JSON body revealing backend configuration:
{"method": "database", "redirect_to": null}
This confirms an information disclosure vulnerability, as backend login method details are exposed to unauthenticated users.



Confirmed Response with Configuration Data

Step 5: Confirm in Browser = Exploitation

This step shows the actual exploitation of the vulnerability. I opened the /api/auth/method endpoint directly in a browser without logging in, and it returned sensitive configuration data. This proves the endpoint is accessible by anyone and confirms the information disclosure vulnerability.



Remediation Advice

The /api/auth/method endpoint should not be accessible without logging in. It gives away important system information that attackers can use. This type of access should be protected so only logged-in users can reach it.

Recommended Fixes:

- Add proper access control to make sure only authenticated users can access this endpoint.
- If someone tries to access it without logging in, the server should return a 401 Unauthorized error.
- Double-check the frontend code to make sure it's not exposing backend system details to users who aren't logged in.

Remediation Code (Assuming Node.js + Express)

```
// Check if user is logged in
function isAuthenticated(req, res, next) {
  if (req.session && req.session.user) {
    next(); // user is logged in, go to next
  } else {
    res.status(401).json({ error: 'Unauthorized' }); // block if not logged in
  }
}

// Protect the endpoint so only logged-in users can access it
app.get('/api/auth/method', isAuthenticated, (req, res) => {
  res.json({
    method: 'database',
    redirect_to: null
  });
});
```

References

OWASP API Security Top 10 – API6:2019
PortSwigger – Burp Suite Guide
OWASP Broken Access Control
Express.js Middleware Guide

Contact Details

Name: Wishwa Mahanama
Email: s223767042@deakin.edu.au

Pentest Leader Feedback.

Shehani Wickremasekera s223841302@deakin.edu.au

- Great work identifying the vulnerability and a well detailed report.
- There are a few amendments which need to be made to the report which has been listed below.
    - Please ensure the Impact and Likelihood are color coded correctly.
    - In the risk criteria table (Impact and Likelihood criteria), ensure the columns have been colored to match the rating.
    - Please fill in the column which contains the question “is this a re-tested finding?”. Answer should be yes or no.
    - Please ensure the documents font style and size are the same. (For example: The risk rating seems to have different font style/size).
    - Ensure all steps have a heading and the explanation of the step is in black (Keep the step’s heading color as it is). Also, first provide the description of the step followed by the screenshot. - Observe the other vulnerability reports in the Final Report folder which will help you address this matter.
    - For the remediation advice, please provide the code which could be used to fix this vulnerability.
- The references look good and are accessible.

Appendix A: Authors

Finding	Technical QA	Reporting QA
Table of Contents & Sections		DARRYL JUN OOI
Document Formatting		DARRYL JUN OOI
Statement of Confidentiality		DARRYL JUN OOI
Executive Summary		DARRYL JUN OOI
Technical Summary	Pen-Testing Team members contributions	DARRYL JUN OOI
Data Impact Visual		DARRYL JUN OOI
OnTrack Vulnerability Findings	Files from All Team Members of Pen-Testing Team	DARRYL JUN OOI
Appendix A: Authors		DARRYL JUN OOI
Appendix B: Limitations		DARRYL JUN OOI

Appendix B: Limitations

During the transition from Stage 1 to Stage 2 of the security assessment, which was planned to begin on April 28th, it was determined that Stage 2 would not proceed as initially scheduled. This decision was based on the following factors:

1. **Progress on Vulnerability Remediation**: By the time of the transition, the OnTrack team had only addressed one of the vulnerabilities identified during Stage 1. As a result, there had been limited progress in the remediation of security issues.
2. **Lack of New Functionality**: The version of the OnTrack application available during the transition did not introduce any new features or functionality compared to the version tested in Stage 1. This resulted in minimal differences between the current version being tested and the version intended for Stage 2.

Given these factors, it was concluded that proceeding with Stage 2 would not provide additional value. Instead, the decision was made to focus efforts on continuing the penetration testing of the current version of OnTrack for the remainder of the trimester.