Finding Name: 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.

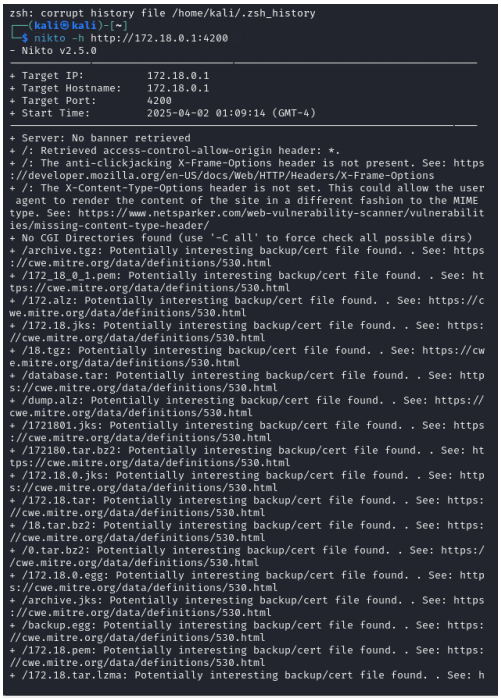
Affected Assets

- **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.

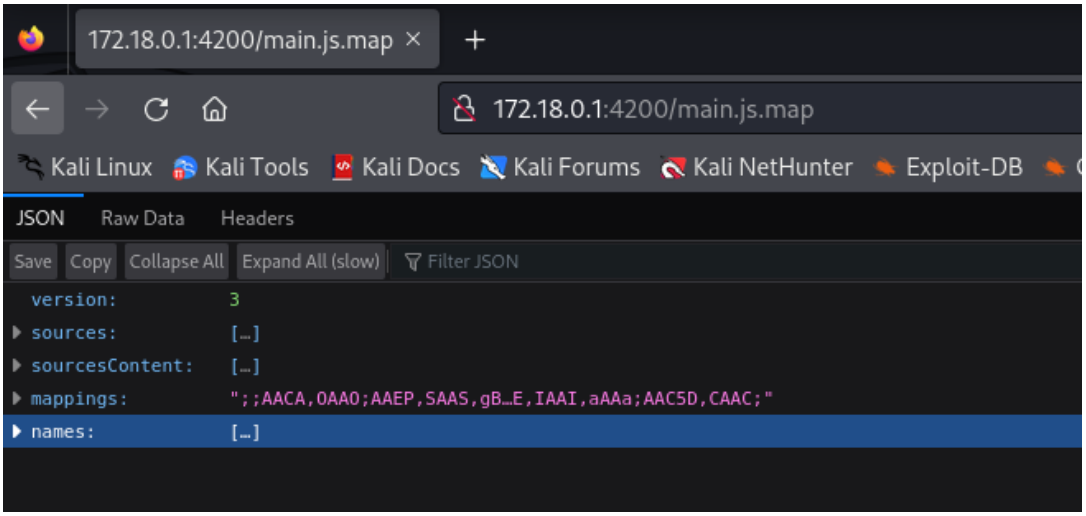


Step 2: Accessing Potentially Exposed File

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**