

## XWeather API: Security Measures

XWeather is an advanced RESTful API that provides hyper-local forecasts and historical weather data in real time, accessible through web, mobile, or native applications. This allows the integration of weather data into Python, JavaScript, iOS, and Android through API key access and dedicated tools for the safe handling of requests and responses (Xweather, 2026).

The most common security threats are:

- Information Leakage: This is one of the major issues because access control is the most neglected aspect, producing a large and anomalous exposure of the API.
- Unauthorized Access: Another major issue is the diversity in authentication logic and its partial implementation. APIs usually use an HTTP header with an autonomous token field for authentication. However, if API implementers use different functions, it is feasible to exploit the implementation differences to obtain unapproved access.
- Lateral movement and injection attacks: In the API's operational mechanism, the data interaction is between the application and the internal components, and it is based on a trust process. In addition, the APIs are connected through the database, message passing, function calls, and forwarding.

If the attacker gains access to only one function or module, they can use the internal functions to exchange data, construct malicious requests, and insert them into the data flow, carrying out a lateral movement attack.

In addition, because most API functions are implemented as web services, they can be attacked through injection. If the API does not filter input data correctly, an attacker

can create malicious code to gain, modify, and delete sensitive information (Zhao, 2024).

Considering these security threats, the following strategies can be implemented:

- To overcome information leakage and unauthorized access, a good solution is the use of a token, in particular, OAuth 2.0 and its extension OpenID Connect (OIDC), which is a framework that allows a user to give third-party applications access to specific resources without sharing the password, because it is based on scopes that define what the app can do. It offers different usage models based on the type of application and security context. In addition, OpenID Connect provides access through an ID token, allowing personalized access, Single Sign-On (SSO), and control of the shared information with the API.
- Even if XWeatherAPI uses HTTPS and TLS protocols, the implementation of mutual TLS could improve security. MTLS requires the client and server to mutually identify themselves using digital certificates.
- Another security principle that can be introduced is the Zero Trust principle, which eliminates implicit trust because every access must be identified, authorized, and encrypted. In addition, it operates microsegmentation that divides the services with logical borders by applying policies. Privileges in this case are minimal because each user/service has access only to what it needs, with controls based on RBAC or ABAC (Jangam et al., 2022)
- I proposed the implementation of a machine learning model based on unsupervised learning, which detects anomalies and suspected activities, such as anomalous requests or multiple accesses by a single user. The model uses techniques such as

clustering and autoencoders to identify abnormal behaviour, including unknown threats. To maintain the efficiency of the model, it will be trained periodically with new data and new threats added. In addition, it will analyse traffic in real time, allowing an automatic response to suspected activities and reducing the burden on security personnel (Ranjan and Dahiya, 2021).

In conclusion, to improve the security of the XWeatherAPI, it will be necessary to upgrade the access measures with a token designed to create a stateless, scalable, and secure system, allowing the division of the authentication to access the resources and support complex systems such as OAuth 2.0 and its extension OpenID Connect. In addition, mutual TLS must be implemented to strengthen security, and the Zero Trust principle must be implemented to eliminate the trust process because every access is controlled, authorized, and encrypted. The introduction of ML, as described above, would be useful for acting quickly in the case of a threat and reducing the burden on security personnel.

## **Reference list**

Jangam, S.K., Karri, N. and Muntala, artha S.R.P. (2022). Advanced API Security Techniques and Service Management. International Journal of Emerging Research in Engineering and Technology, 3(4), pp.63–74. doi:<https://doi.org/10.63282/3050-922X.IJERET-V3I4P108>.

Ranjan, P. and Dahiya, S. (2021). Advanced Threat Detection in API Security: Leveraging Machine Learning Algorithms. *International Journal of Communication Networks and Information Security*, 13(1), pp.185–196. doi:<https://doi.org/2073-607X,%202076-0930>.

Zhao, C. (2024). API Common Security Threats and Security Protection Strategies. *Frontiers in Computing and Intelligent Systems*, 10(2), pp.29–33. doi:<https://doi.org/10.54097/k5djs164>.