Click to prove you're human

Api scopes best practices

Example Business Scenario: APIs and Clients When organizations provide digital services they usually develop multiple APIs and clients that span a number of business areas. In particular, scopes are usually always fixed at design time. It enables users to grant third-party applications limited access without exposing their credentials.OpenID Connect: Built on top of OAuth 2.0, OpenID Connect adds an identity layer for user authentication, making it suitable for single sign-on (SSO) implementations.JWT (JSON Web Tokens): Utilize JWTs for compact and self-contained tokens that securely transmit information between parties, making them ideal for stateless authentication.Implement Rate LimitingRate limiting is an effective strategy to control the number of requests a user or application can make within a specified timeframe:Control Request Rates: Implement rate limiting to control the number of requests a user or application can make within a specified timeframe.Preventing Abuse: Rate limiting safeguards against brute-force attacks and excessive API usage.Fair Usage: It ensures fair and equitable access to your API resources among all users.Secure API KeysAPI keys are a common method of authentication. Prefix scopes enable clients to request scopes that are unknown at design time. You can read more about token exchange behaviors in the following resources: OAuth Token Exchange Flow Implementing Token Exchange Scope Best Practices Summary Scopes represent an area of data and allowed operations on that data. Doing so prevents obviously invalid calls from ever reaching the actual API, which reduces costs in some API architectures. On this page OAuth 2.0 scopes are strings issued to access tokens. This means, you cannot use scopes for dynamic authorization, i.e. you cannot use different scopes for distinct users of your frontend applications. Example Business Scopes A moderately complex client might interact with a number of APIs and use multiple scopes such as those listed in the following table. When you issue scopes to access tokens, you also issue a set of claims. In this case, you need to define custom scopes for … There's a big difference between importing a user's Facebook profile information and posting to their wall. In this scenario, the scopes available to you include those implemented by the OpenID Connect (OIDC) protocol. The threat landscape is dynamic, and continuous learning ensures ongoing protection.What's the role of rate limiting in API security?Rate limiting helps control the number of requests made to an API, preventing abuse, and ensuring fair usage. In this case, you need to know which custom scopes are defined for the API you are calling. The following flow shows the use of token exchange to downscope an access token when an orders API calls a shipping API. Inadequate API security can result in legal consequences and tarnish your organization's reputation. For that, it uses an example to showcase the arguments. For example, if you split a large API into three smaller microservices, for code manageability reasons, you should not need to add any new scopes. To see examples of calling a custom API from an application, read Sample Use Cases: Scopes and ClaimsUnderstand your use case and choose the most restrictive scopes possible. To use a prefix scope you configure a scope name like transaction- in the authorization server. But again, remember that when asked for consent, users can always say no.By default, Auth0 skips user consent for first-party applications, which are applications that are registered under the same Auth0 domain as the API they are calling; however, you can configure your API in Auth0 to require user consent from first-party applications. To learn more, read OpenID Connect Scopes.In an API, to implement access control. To complete your API authorization you also need to follow Claims Best Practices.Gary ArcherProduct Marketing Engineer at CurityYou may also be interested inGet the latest on identity management, API Security and authentication straight to your inbox.Try the Curity Identity Server for Free. Short-lived tokens reduce the risk of prolonged unauthorized access.Refresh Tokens: Implement refresh tokens that allow users to obtain new access tokens without re-authenticating, balancing security with user convenience.Staying informed about the evolving landscape of security threats and best practices is paramount:Continuous Learning: Stay informed about the latest security threats and best practices through participation in security forums, workshops, and continuous education.Regular Updates: Promptly apply security patches and updates to your API and its dependencies to mitigate emerging threats.How to Authenticate API with ApidogbuttonHere are specific instructions based on the most common authentication methods:Authenticate API with ApidogAPI Keys:Navigate to the API settings in Apidog.Access the "Auth" section.Choose "API Key" as the authentication method.Generate a new API key with a descriptive name.Copy the generated API key securely.When making API requests, include the API key in the request header: Authorization: Bearer YOUR_API_KEYAPI KeysOAuth 1.0:In the API settings, select "OAuth 1.0" as the authentication method.Configure OAuth providers (e.g., Google, GitHub) and define scopes.Obtain client ID and client secret from the OAuth provider.Provide these credentials in Apidog's OAuth configuration.Follow Apidog's instructions for generating authorization URLs and handling redirects.OAuth 1.0Basic Authentication:In the API settings, choose "Basic Auth" as the authentication method.Provide the username and password credentials.When making API requests, include the base64-encoded credentials in the request header: Authorization: Basic BASE64_ENCODED_CREDENTIALSJSON Web Tokens (JWTs):In the API settings, select "JWT" as the authentication method.Define the secret key and algorithm for token generation.Generate JWTs using a suitable library or tool.Include the generated JWT in the request header: Authorization: Bearer YOUR_JWTJSON Web TokensKey Benefits of Using ApidogStreamlined API Design and Testing: Apidog simplifies API design, testing, and documentation, ensuring that security considerations are embedded from the outset.Enhanced Security Testing: Apidog's security testing and monitoring capabilities identify vulnerabilities early, reducing the risk of security breaches.Efficient Collaboration: Facilitating team collaboration, Apidog supports secure sharing of API documentation and testing results, fostering a security-aware team culture.Real-time Monitoring and Analytics: Apidog offers real-time insights into API performance and usage, aiding in the rapid detection of security threats.Customizable Access Control: With Apidog, you can establish tailored access controls, aligning with the principle of least privilege.Regular Updates and Support: Apidog remains at the forefront of API security with regular updates and features that embrace the latest security trends.APIs are the backbone of digital connectivity, but their power comes with the responsibility to secure them. High-Privilege Scopes A client can use scopes to temporarily get a high-privilege access token to call particular API endpoints. When a client serves users, those users must authenticate. Different pieces of user information are often stored across a number of online resources. You need to ensure that all components in your system can send or receive scopes in access tokens so that you correctly restrict access to resources. Always Enforce Scopes Your API authorization must always verify that required scopes are present and return a 403 forbidden response to the client when scope verification fails. For example, a client might have access to an order:payment scope but not use it by default. For example, you might do so when you call APIs operated by a subdivision of your organization. The claims enable your detailed API authorization. Scopes and Multiple APIs When you develop multiple APIs and they need to call each other, you can forward the access token from the client to upstream APIs. This approach is common in a small microservices setup, where each API checks for its required scopes, and is the simplest way to use scopes across multiple APIs: Yet when you use multiple APIs, you should avoid designing scopes in an overly technical manner. Use Least-Privilege Scopes Design clients so that they only have access to the data they need, and limit the scope to read-only access when write access is not needed. Avoid the need for access token versioning in your APIs. Avoid deploying frequent scope configuration updates to your authorization server. Similarly, when creating custom scopes for an API, consider what levels of granular access applications may need and design accordingly.Requested scopes versus granted scopesIn certain cases, users get to consent to the access being requested. To learn more, read API Scopes.From an application, to call an API that has implemented its own custom scopes. By only requesting what you need, you are more likely to gain user consent when required since users are more likely to grant access for limited, clearly specified scopes. Another limitation of prefix scopes is that they cannot contain claims. Instead, store them in environment variables or configuration files.Regular Rotation: Enable key regeneration mechanisms, allowing users to refresh their API keys regularly, reducing the risk of compromised keys.Utilize HTTPSHTTPS is a non-negotiable requirement for secure data transmission. APIs must return a 403 forbidden response when an access token has insufficient scope. The flow might include user consent. Scopes define the specific actions applications can be allowed to do on a user's behalf.When an app requests permission to access a resource through an authorization server, it uses the scope parameter to specify what access it needs, and the authorization server uses the scope parameter to respond with the access that was actually granted (if the granted access was different from what was requested).Generally, you use scopes in three ways:From an application, to verify the identity of a user and get basic profile information about the user, such as their email or picture. When you refresh an access token after the configured time, the new access token no longer contains the scope. More commonly, scopes should not usually identify concrete resources. In the following example, the hasScope method might return true if there is an exact match on the order:item scope, or if the parent scope of order is present: public getOrderItems(criteria: Criteria): OrderItem[] { if (!claimsPrincipal.hasScope("order:item")) return repository.getOrderItems(); You could design your APIs to require an additional suffix for data changing commands, in which case the API authorization code for that type of operation could check for a precise scope: public void updateInventoryPrice(item: OrderItem) { if (!claimsPrincipal.hasScope("inventory:price.write")) return repository.createInventoryItem(item); You can optionally use a reverse proxy or API gateway to enforce scopes. Prefix Scopes Although scopes are not usually dynamic, prefix scopes are an exception to that rule. API Authorization Using Scopes APIs verify access token scopes to perform high-level authorization checks. In this case, you need to define custom scopes for your API and then identify these scopes so that calling applications can use them. Keep scopes stable to avoid scope explosion. Use Stable Scopes It is possible to use many fine-grained scopes. Use scopes to enable high-privilege access tokens. Therefore, you should handle the finer details of authorization using Claims, another part of the security architecture. One cause of scope explosion is when client-specific concerns leak into scope names, as in the following examples, resulting in duplication: inventory:supplier:2 order:admin-usa:write Configuring excessive scopes in the authorization server can have an adverse impact on your productivity. The new access token's user identity, expiry and other claims can remain the same as those in the original access token. A common way to get started with scopes is to use a combination of the type of resource and the access required on it: Resource TypeAccess LevelScope Valueorderreadorder_read OAuth standards documents do not define how you should use scopes but leaves that to designers of each system. It ensures that only authorized entities can interact with the API.Why is API authentication important?API authentication is crucial to prevent unauthorized access, protect sensitive data, comply with regulations, and maintain the trust of users and clients.What are some common authentication methods for APIs?Common authentication methods include OAuth 2.0, OpenID Connect, JWT (JSON Web Tokens), API keys, and basic authentication.How often should API credentials be rotated?API credentials, such as keys and passwords, should be rotated regularly, typically every 90 days or as per your organization's security policy.What is Apidog, and how does it enhance API security?Apidog is a comprehensive API security tool that simplifies API design, enhances security testing, fosters collaboration, offers real-time monitoring, and supports customizable access control, making it a vital asset in ensuring API security.Can I use Apidog with APIs built on different technologies?Yes, Apidog is adaptable and can be used with APIs built on various technologies, making it a versatile choice for API security.Do I need to stay updated with security practices even after implementing these best practices?Yes, staying informed about evolving security threats and best practices is essential. While usually, the scopes returned will be identical to those requested, users can edit granted scopes (both during initial consent and sometimes after, depending on the resource), thereby granting an app less access than it requested. The authorization server should enable you to configure a time-to-live for scopes. For example, an API may need to allow read-write access to some resource properties and read-only access to others. After user authentication and consent, the authorization server issues the granted scopes to the access token. You should activate user consent at the authorization server when a third-party client requests access to APIs. The user may grant the third-party access to some scopes and not others. Scopes Design When you design scopes for real-world systems you set boundaries on where clients can use access tokens. Employ these practices:Input Sanitization and Validation: Implement input sanitization to remove or neutralize potentially harmful characters or code.Type and Length Checks: Ensure that input data matches the expected format and length to prevent injection attacks and data corruption.Employ Robust Access ControlFine-grained access control is crucial for limiting user access based on roles and permissions:Role-Based Access Control (RBAC): Implement fine-grained access control by assigning roles and permissions to users.Principle of Least Privilege: Follow the principle of least privilege, where users are granted only the minimum access necessary for their tasks.Regularly Rotate CredentialsRegular credential rotation is a proactive measure to mitigate the risk of compromised keys:Scheduled Rotation: Frequently update API credentials, including keys and passwords, on a predetermined schedule.Automated Alerts: Implement automated alerts to notify users when it's time to change their credentials, ensuring timely rotation.Monitor and Log AccessComprehensive monitoring and logging provide insights into API activity, aiding in early threat detection:Real-Time Monitoring: Establish real-time monitoring systems to detect unusual patterns or potential security breaches in API access.Audit Trails: Maintain detailed audit logs of API access, which are essential for security audits, compliance, and forensic analysis.Use Token ExpiryToken expiration is a critical measure to limit the use of stolen tokens:Short-Lived Tokens: Set expiration times for tokens to limit the use of stolen tokens. These rules need to use dynamic behavior based on properties of the authenticated user. APIs act as gateways, facilitating interactions between different software applications and systems. It's an effective defense against brute-force attacks and excessive API consumption.Can I apply these best practices to existing APIs, or are they for new ones only?These best practices can be applied to both existing and new APIs. It's never too late to enhance the security of your APIs.Is API authentication the only aspect of API security?No, API security encompasses multiple aspects, including authentication, authorization, encryption, and monitoring. The Introduction to Scopes explains how APIs use scopes to restrict access to resources. Token refresh can either use all scopes granted or a subset of them. Clients then silently refresh access tokens within the same authenticated user session. The only scopes left in the access token are those that the shipping API needs. The source API can use OAuth token exchange to get a new access token with different scopes to send to the target API. When a client receives an access token, the scopes in the token represent the current API privileges of the client. Scopes and Time to Live You usually configure clients to use access tokens with a short expiry time, such as 15 minutes. ScopeAPI Access GrantedopenidAn OpenID Connect scope that represents the user's identity.profileAn OpenID Connect scope that represents the user's name-related details.customer:benefitsAccess to customer benefits.inventoryAccess to inventory details.orders:writeAccess to create orders and then view or update them afterwards.shipping:writeAccess to change delivery information. Always use HTTPS to encrypt data transmitted between clients and your API:Data Encryption in Transit: Always use HTTPS to encrypt data transmitted between clients and your API. The forwarding API sends a token exchange request to the token endpoint of the authorization server. Upscoping requires care so always think through the security implications, especially if user consent is active. Ensure their security by following these practices:Encryption and Secure Storage: Encrypt API keys during transmission and storage to prevent unauthorized access.Environment Variables for Storage: Avoid hardcoding API keys in application code. Instead, you should only need to add new scopes occasionally, such as when you expand APIs to cover a new business area: Avoid frequently upgrading clients to use new scopes. To enable this, APIs should separate data both by business area and data sensitivity, so that you expose parts of your data to different clients. Consider adding new scopes only when you add new API business areas. You should control which clients can gain access to which API privileges. As an application developer, you should be aware of this possibility and handle these cases in your app. The scope granted is always for a concrete resource and you can visualize the full scope in a consent screen: Prefix scopes can be useful in advanced scenarios, such as Financial-grade use cases. This comprehensive guide outlines ten essential best practices for API authentication, emphasizing the role of tools like Apidog in enhancing and ensuring API security.Empower your API security with Apidog's advanced features including streamlined design and testing, robust security testing, efficient collaboration tools, real-time analytics, and flexible access control.Elevate your API protection today – Checkout This Button Below buttonThe Importance of API AuthenticationSafeguarding Digital AssetsAPIs serve as gateways to your digital assets, including sensitive data and critical functionalities. User RoleAuthorization RuleCustomerA customer can only view benefits and orders associated to their own user ID.CustomerA customer with a higher subscription levels can access additional inventory.AdministratorAn administrator may have access to all data, but with regional restrictions.Supplier UserA supplier business partner can only view inventory for their own company's supplier ID. API authentication is the first line of defense, but a holistic security approach is recommended. The following example shows how an API can make a token exchange request to get a new access token with different scopes and, if required, a different audience: curl -X POST \-H 'Content-Type: application/x-www-form-urlencoded' \-d 'grant_type=urn:ietf:params:oauth:grant-type:token-exchange' \-d 'client_id=forwarding-api' \-d 'client_secret=myS3cret' \-d 'subject_token=56acc3f6-b9ef-4a34-a9d4-f9d7a27a505b' \-d 'subject_token_type=urn:ietf:params:oauth:token-type:access_token' \-d 'scope=shipping.write' \-d 'audience=shipping-api' \ It is also possible to use token exchange to upscope access tokens. One possible convention is to make read-only access the default and then add a write suffix when higher privilege is needed: ScopeAccess GrantedorderRead only access to full order information.order:itemsRead only access to details about order items.inventory:writeThe ability to create, change or delete an entire inventory item.inventory:price:writeThe ability to change the price details for an inventory item. Changing Scopes with Token Exchange Forwarding access tokens may not always be the most secure option. The teams working in a business area should be able to understand the scopes you design. When the user needs to make a payment, the client could issue an authorization redirect to the authorization server and request the high-privilege scope. Your main API authorization may also need to enforce dynamic rules such as those listed below. When you make an authorization request with such a scope, the scope is only issued to access tokens for the configured time. More generally, when APIs call each other you choose a token sharing approach. However, this critical role also makes them a target for cyber attacks, data breaches, and unauthorized data access. Utilize proven authentication protocols and mechanisms to ensure the highest level of security:OAuth 2.0: Employ OAuth 2.0 for secure authorization and delegation of access. Usually though, it is better to keep scopes simple when you get started, then add more scopes later when you identify the need. For example, a system that sells products to online customers might use various components like those shown here, targeted at different roles of user. The client then receives an access token with which it can call a high-privilege API endpoint. Ensuring the security of APIs, particularly through robust authentication practices, is not just a technical necessity but a business imperative. Get up and running in 10 minutes.Start Free Trial In our digitally interconnected world, Application Programming Interfaces (APIs) have become fundamental to software development and communication. Robust API authentication serves as the initial and fundamental defense against unauthorized access, ensuring that only trusted entities gain entry to your digital assets.Safeguarding Digital AssetsEnsuring Data Privacy and ComplianceIn today's regulatory landscape, data privacy and compliance with laws such as GDPR, HIPAA, and CCPA are paramount. It ensures data confidentiality and protects against man-in-the-middle attacks.Certificate Validation: Regularly validate and update SSL/TLS certificates to maintain a secure connection.Validating input data is essential to prevent injection attacks and data corruption. The client then asks for a scope containing a specific ID at runtime. Verifying scopes is only an entry-level check and not a complete API authorization solution. Therefore, in the general case, the scopes granted may be a subset of those requested by the client. Design scopes so that they work for end-to-end flows that use multiple APIs. Use token exchange to downgrade scopes when calling less trusted upstream APIs. Conclusion Scopes provide high-level access control for your APIs. You should be able to scale your use of scopes to many APIs with good manageability. Users may upload and store photos with a service like Flickr, keep digital files on Dropbox, and store contacts and events in Google Calendar or on Facebook.Often, new applications will want to make use of the information that has already been created in an online resource. The convention is to use a trailing hyphen in the name. If you are requesting scopes, make sure you ask for enough access for your application to function, but only request what you absolutely need. Use Hierarchical Scopes When data is hierarchical you can use hierarchical scopes, as in the below examples, where colon characters represent subresources: ScopeGrants access toorderFull information about orders.order:itemInformation about items within an order.order:paymentAccess to order payment details.order:shipping:addressInformation about where to deliver orders.order:shipping:statusInformation about the delivery status of orders. Scopes do not provide a full API authorization solution. Use the following guidelines when you design scopes to secure your APIs: Always use scopes in APIs and enforce them at every API endpoint. Although it often makes sense to consider read and write permissions, this approach has some limitations. This article explains how to manage scopes at scale, avoid common problems, and use some advanced scope techniques. Once user authentication completes, the authorization server can present a consent screen to the user before it issues the access token. The grant usually lasts for an entire authenticated user session, during which the client silently refreshes access tokens. Avoid Scope Explosion Without care, there is a risk that you could create a large number of scopes that are difficult to maintain over time, leading to scope explosion. Third-party applications, which are external applications, require user consent.In an API, to implement access control. It could also send the user back through the authorization flow to ask for additional permissions. Different API endpoints can each require their own scopes. An API endpoint that requires an insurance scope would immediately deny access if that scope is not present in an access token for a user, even if that user has insurance rights in another application. Use Scopes for Coarse-Grained Access Control In the example business scenario you might enforce these high-level user authorization rules with scopes: Use Claims for Fine-Grained Access Control Scopes only enable entry-point API authorization. The authorization server then returns an access token to the client containing the user's identity. Most commonly, you downscope access tokens. Effective API authentication goes beyond security; it's a compliance necessity, ensuring that data is handled by the law.Best Practices for API AuthenticationUse Strong Authentication MethodsAuthentication is the foundation of API security, and strong methods are essential. Enable user consent when a user needs to grant scope-based access to a third-party client. By default, the authorization server uses the scopes that a user granted for its refresh token logic that issues new access tokens. Scopes in Clients You need to configure each client with scopes to restrict its API privileges. For example, your app could warn the user that they will see reduced functionality. This may not be the desired behavior for high-privilege scopes. Consider using API business areas as scopes. To do so, the application must ask for authorization to access this information on a user's behalf. For example, you might have a user facing application that uses an orders scope to call an Orders API but not want that application to know about an invoices scope used by another API once order processing work is complete. Are you establishing a user's identity or asking the user to allow you to interact with their data? By implementing the 10 API Authentication Best Practices and integrating Apidog, you ensure robust protection for your APIs. This proactive approach not only safeguards your digital assets but also enables your APIs to thrive in a secure and efficient environment.What is API authentication?API authentication is the process of verifying the identity of users or applications accessing an API. These high-level scopes are stable and sufficient to set boundaries on where clients can use access tokens. For example, you may have three business areas of marketing, insurance and finance and use scopes to represent each of them. To use token exchange you must register the forwarding API as a client of the authorization server and grant it access to the scopes that it will request. Failing to secure these gateways can expose your organization to data breaches, financial losses, and reputational damage.