JWTForge: A JWT Vending Service for Testing, Fuzzing, and Security Research of OAuth2/OIDC Implementations

Abhishek Tiwari

Citation: *A. Tiwari*, "JWTForge: A JWT Vending Service for Testing, Fuzzing, and Security Research of OAuth2/OIDC Implementations", Abhishek Tiwari, 2025. doi:10.59350/6pdmd-3cm41

Published on: November 15, 2025

Summary

JWTForge is a lightweight, open-source JSON Web Token (JWT) ([1]) vending service designed specifically for developers and security researchers who need to test or surface vulnerabilities in the OAuth2 ([2]) and OpenID Connect (OIDC) ([3]) implementations. With one-click, JWTForge can be deployed on Cloudflare Workers ([4]), and it generates realistic, customizable JWT tokens on-demand, making it an essential tool for end-to-end testing, security auditing, and fuzzing authentication systems. Whether you're building a new API, conducting penetration tests, or debugging authentication flows, or researching new attack modes, JWTForge provides the flexibility to generate any token configuration you need - from standard OAuth2/OIDC claims to malformed tokens for edge-case testing.

Statement of Need

Modern web applications depend heavily on OAuth2 ([2]) and OIDC ([3]) for their authentication and authorization needs. However, testing these implementations thoroughly presents several challenges. Using production identity providers (Auth0, Okta, AWS Cognito, Microsoft Entra) for testing and security research can be risky and expensive. One can't easily generate malformed tokens or test edge cases without potentially affecting real users. Developers often resort to creating mock JWT ([1]) tokens which can be time-consuming and often less realistic for identifying vulnerabilities. In addition, security researchers require the ability to generate tokens with expired or future timestamps, invalid signatures, missing required claims, unexpected claim types, and malicious payloads. RFC 8725 ([5]) outlines security best practices for JWT implementations, including recommendations for algorithm validation, claim verification, and protection against common attacks—all of which require comprehensive testing capabilities. Finally, integration and end-to-end tests need consistent, reproducible tokens across different environments without depending on external identity providers.

Currently security researchers use `jwt_tool` ([6]) for validating, forging, scanning and tampering JWTs. It offers testing for known exploits in existing JWTs through brute-forcing, algorithm confusion attacks, and claim manipulation. Unlike `jwt_tool`, which focuses on exploiting and manipulating existing JWTs through algorithm confusion attacks, secret brute-forcing, and signature tampering, JWTForge generates compliant OIDC/OAuth2 tokens from scratch via HTTP API. While both support fuzzing, claim manipulation, and malformed payloads, JWTForge provides full OIDC infrastructure (including JSON Web Key Set (JWKS) and OIDC discovery endpoints) making it ideal for integration testing and controlled security research. Other manual testing tools such as JWT Editor ([7]), SignSaboteur ([8]), and JOSEPH ([9]) are Burp Suite Extensions and allow for editing, signing, and verifying JWT tokens and can be used to perform several well-known attacks against JWT implementations. JWT Cracker ([10]) is another multi-threaded JWT brute-force cracker written in C allowing users to

find the secret key used for creating a JWT token. The tools are complementary i.e. one can use JWTForge to generate baseline tokens, then use these `jwt_tool`, `JOSEPH`, `JWT Editor`, etc. to attack the implementation. Although some of these tools are tailored for manual interaction, not suitable for automated testing.

Features

JWTForge fills this gap by providing a dedicated testing service that generates valid, signed JWT ([1]) tokens with any claim combination. It also provides proper JWKS and OIDC ([3]) discovery endpoints for realistic testing. It supports multiple key types (`RSA`, `EC`) and algorithms (`RS256`, `ES256`). It enables security testing through customizable, even malformed, tokens. JWTForge automatically populates claims based on standard OIDC scopes (`openid`, `profile`, `email`, `address`, `phone`), generating authentic-looking user data via Faker ([11]). This eliminates the tedious task of crafting realistic test personas. In addition, JWTForge allows values in header and payload of JWT tokens to be fuzzed enabling security researchers to perform testing for known vulnerabilities and surface unknown exploits. JWTForge performs randomized fuzzing using BLNS (Big List of Naughty Strings) ([12]), known JWT edge cases and provides malicious values such as Injection payloads (SQL, XSS, command injection, path traversal, etc.) as part of token payload and header.

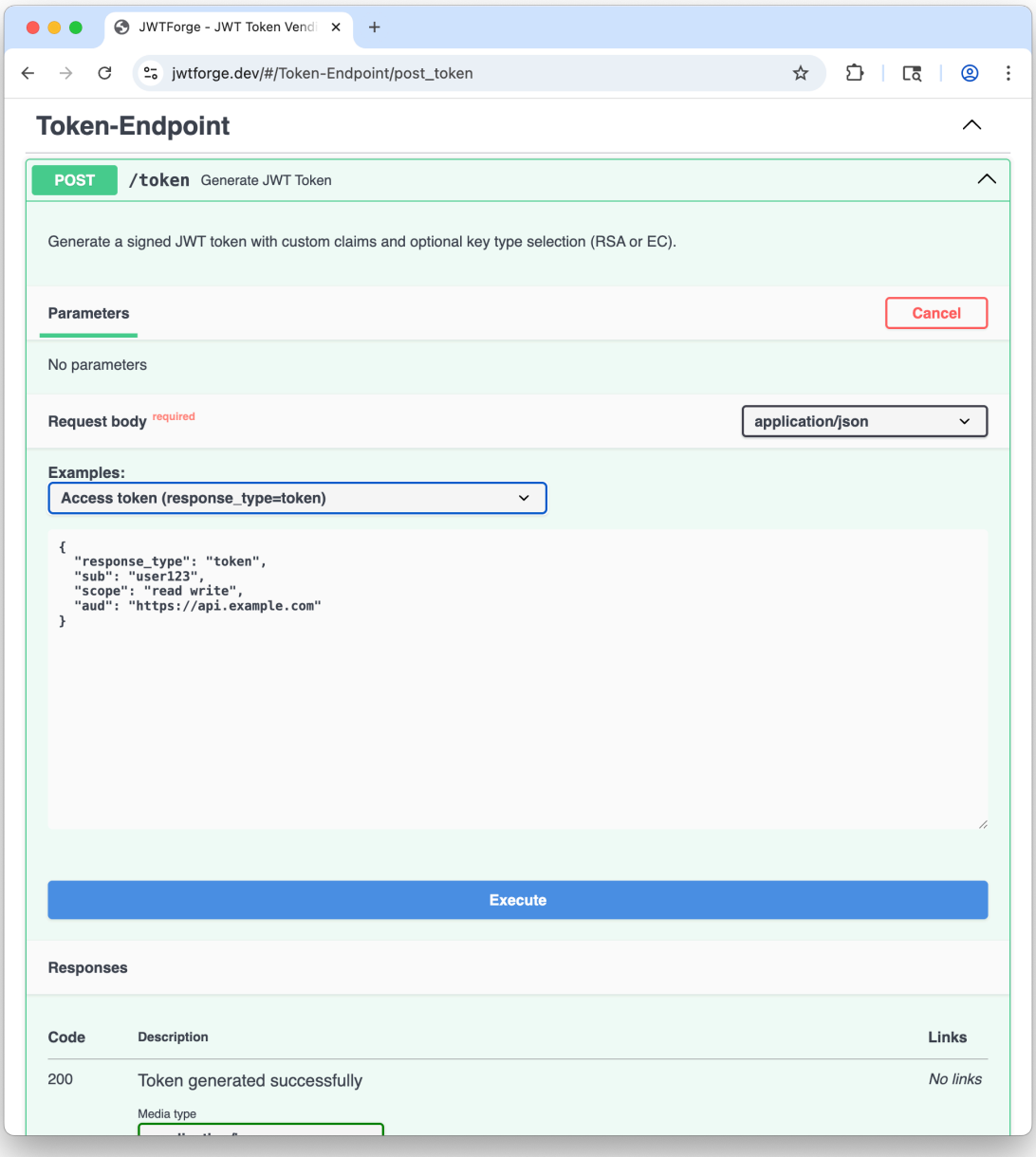

Figure 1: JWTForge Swagger UI for generating tokens. JWTForge provides example token templates for various testing scenarios making it easier for users to generate JWT tokens.

We provide a one-click deploy script enabling users to deploy their own JWTForge vending service on Cloudflare Workers ([4]) for free. Alternatively, users can run JWTForge locally (`npm run dev`) to test authentication flows without internet connectivity or external dependencies. One can also inte-

grate JWTForge into their CI/CD pipeline to generate consistent tokens for automated testing without depending on external identity providers.

Availability

JWTForge is available on GitHub under MIT License which users can deploy to Cloudflare Workers ([4]) platform for free through one-click deployment. Alternatively, a hosted version of service is available at jwtforge.dev. Code examples are also provided as part of repository.

References

[1] M. B. Jones, J. Bradly, and N. Sakimura, "JSON Web Token (JWT)," *RFC Editor*. Internet Engineering Task Force (IETF), 2015. doi: 10.17487/RFC7519.

[2] D. Hardt, "The OAuth 2.0 Authorization Framework," *EFC Editor*. Internet Engineering Task Force (IETF), 2012. doi: 10.17487/RFC6749.

[3] "OpenID Connect Core 1.0," *OpenID Foundation*. 2014. Available: https://openid.net/specs/openid-connect-core-1_0.html

[4] "Cloudflare Workers: Serverless execution environment," *Cloudflare*. Available: https://workers.cloudflare.com/

[5] Y. Sheffer, D. Hardt, and M. B. Jones, "JSON Web Token Best Current Practices," *RFC Editor*. Internet Engineering Task Force (IETF), 2020. doi: 10.17487/RFC8725.

[6] A. Tayler, *jwt_tool: A toolkit for testing, tweaking and cracking JSON Web Tokens*. Available: https://github.com/ticarpi/jwt_tool

[7] *JWT Editor: Burp Suite Extension for editing, signing, verifying, encrypting, and decrypting JSON Web Tokens*. Available: https://github.com/PortSwigger/jwt-editor

[8] *SignSaboteur is a Burp Suite extension for editing, signing, verifying various signed web tokens*. Available: https://github.com/d0ge/sign-saboteur

[9] *JOSEPH: JavaScript Object Signing and Encryption Pentesting Helper*. Available: https://github.com/PortSwigger/json-web-token-attacker

[10] B. Rius, *c-jwt-cracker: Multi-threaded JWT brute-force cracker*. Available: https://github.com/brendan-rius/c-jwt-cracker

[11] *Faker: Generate massive amounts of fake data*. Available: https://fakerjs.dev/

[12] M. Woolf, *Big List of Naughty String*. Available: https://github.com/minimaxir/big-list-of-naughty-strings