Exploiting the Firebase Login Mechanism and APIKEY's

Devendra Nath

About Me

- Security Engineer @KreditBee
- Identified over 100 vulnerabilities in diverse software applications.
- Security Researcher @BugCrowd @hackerone @Intigriti.
- Certified with CRTP (Certifed Red Team Professional) and CompTIA Security+.
- Experienced in Application Security, Network Pentesting, Cloud Security, Active Directory, Android Pentesting and SAST automation.
- Hacking into system since 2019.

Agenda

1. Introduction to Authentication
2. Types of JWTs
3. Exploiting Firebase Services
4. Mitigations
5. References

Introduction to Firebase Authentication

User Authentication:

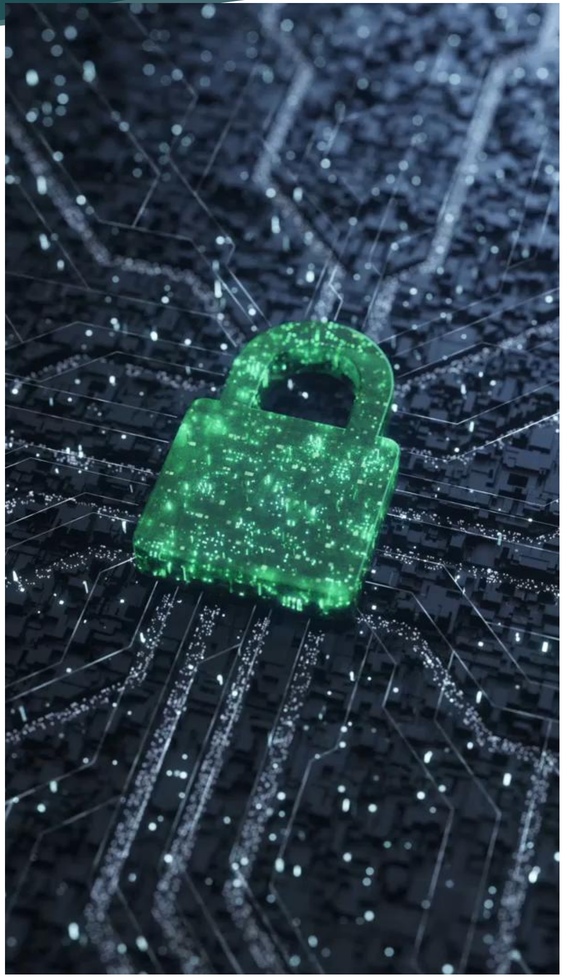
- Users sign in or create an account using Firebase Authentication methods (email/password, social sign-in, etc.).

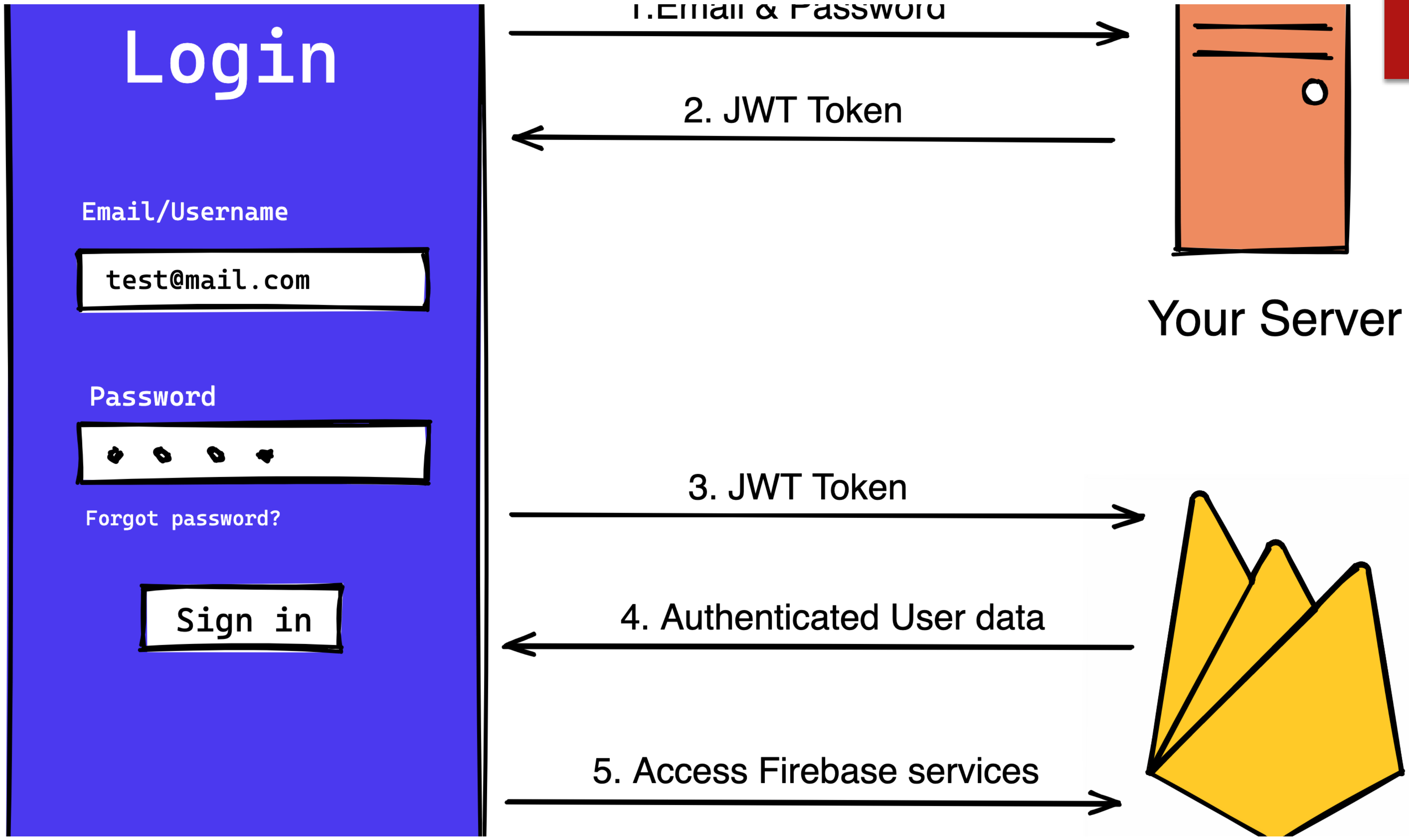
Token Issuance:

- After successful authentication, Firebase issues a JSON Web Token (JWT) to the client.

Token Structure:

- The JWT is a compact, URL-safe means of representing claims between two parties. It consists of three parts: a header, a payload, and a signature.
- Header: Contains the type of the token (JWT) and the signing algorithm used (e.g., HMAC SHA256 or RSA).
- Payload: Contains claims. Claims are statements about an entity (typically, the user) and additional data. Examples include user ID, username, and expiration time.
- Signature: Created by combining the encoded header, encoded payload, and a secret. It's used to verify that the sender of the JWT is who it says it is and to ensure that the message wasn't changed along the way.




Login
Email/Username
test@mail.com
Password
Forgot password?
Sign in
1.Email & Password
2. JWT Token
Your Server
3. JWT Token
4. Authenticated User data
5. Access Firebase services

Types of JWT’s (JSON Web Tokens)

ID Tokens (Access Tokens)

Firebase Authentication issues ID tokens (JWTs) as access tokens when a user successfully signs in.

These ID tokens contain information about the user (claims) and are signed by Firebase using a private key.

ID tokens are short-lived and are used by the client to access protected resources on behalf of the user.

Refresh Tokens

Firebase automatically issues refresh tokens when a user signs in.

Refresh tokens are long-lived compared to ID tokens and can be used to obtain a new ID token when it expires without requiring the user to sign in again.

Custom Tokens

Firebase provides a way to mint custom tokens using the Firebase Admin SDK.
Custom tokens are useful when you have your own authentication system outside of Firebase, and you want to integrate it with Firebase services.
Custom tokens can include custom claims that define roles, permissions, or any additional information relevant to your application.

Exploiting Firebase Services

Subdomain Discovery via getProjectConfig

Exploiting IdToken using Refresh token

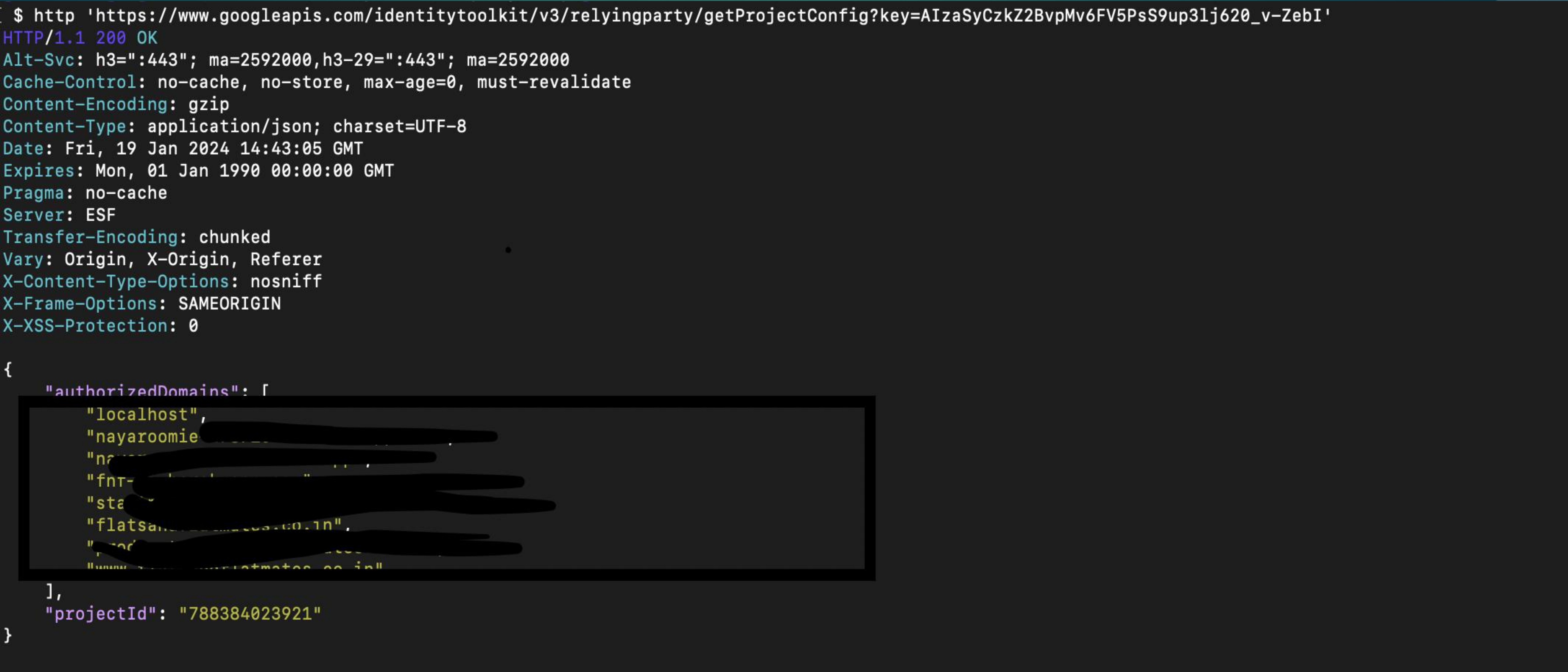
Subdomain Discovery via getProjectConfig

- The concept of "authorized domains" refers to the domains that are allowed to use Firebase Authentication services, such as authentication providers (e.g., email/password, Google Sign-In) and other related features.
- This is a security measure to ensure that authentication requests are only accepted from specified domains, preventing unauthorized access and abuse.
- Authorized domains are relevant when you configure authentication providers that involve redirects, such as OAuth-based providers. For example, when you configure Google Sign-In, you need to specify the authorized domains where the authentication callback can be received.
- The command will fetch you all the subdomains that are integrated with the project and you can use these as active recon.

1. Stage 2.Production   3.S3-buckets

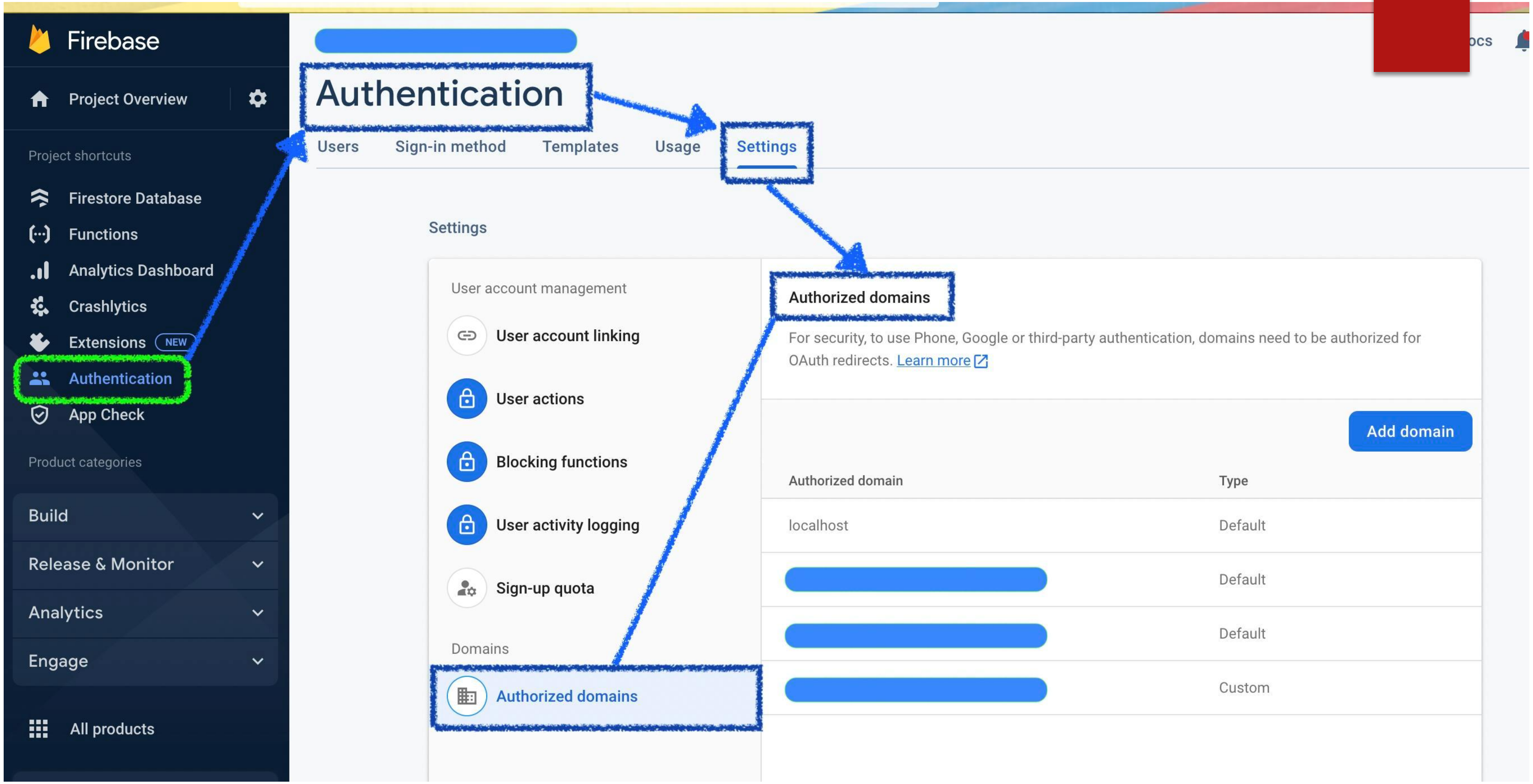
```
http "https://www.googleapis.com/identitytoolkit/v3/relyingparty/getProjectConfig?key=<api_key>" | jq -r
'.authorizedDomains[] | select(test("amazonaws.com"))'  >> domains.txt
```

Gathering Subdomains



```
$ http 'https://www.googleapis.com/identitytoolkit/v3/relyingparty/getProjectConfig?key=AIzaSyCzkZ2BvpMv6FV5PsS9up3lj620_v-ZebI'
HTTP/1.1 200 OK
Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Encoding: gzip
Content-Type: application/json; charset=UTF-8
Date: Fri, 19 Jan 2024 14:43:05 GMT
Expires: Mon, 01 Jan 1990 00:00:00 GMT
Pragma: no-cache
Server: ESF
Transfer-Encoding: chunked
Vary: Origin, X-Origin, Referer
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 0

{
    "authorizedDomains": [
        "localhost",
        "nayaroomie[redacted]
    ],
    "projectId": "788384023921"
}
```



Settings

User account management

- User account linking
- User actions
- Blocking functions
- User activity logging
- Sign-up quota

Domains

- Authorized domains

Authorized domains

For security, to use Phone, Google or third-party authentication, domains need to be authorized for OAuth redirects. Learn more

Add domain

Authorized domain	Type
localhost	Default
[redacted]	Default
[redacted]	Default
[redacted]	Custom

Firebase REST API’s for User Login & Sign-Up

1. Login:

```
 curl 'https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=[API_KEY]’  -H 'Content-Type: application/json'
--data-binary '{"email":"[user@example.com]","password":"[PASSWORD]","returnSecureToken":true}’
```

2. SignUp:

```
 curl 'https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=[API_KEY]’  -H 'Content-Type: application/json' --data-binary
'{"email":"[user@example.com]","password":"[PASSWORD]","returnSecureToken":true}’
```

3. Email update:

```
 curl 'https://identitytoolkit.googleapis.com/v1/accounts:update?key=[API_KEY]’  -H 'Content-Type: application/json' --data-binary
'{"idToken":"[GCIP_ID_TOKEN]","email":"[user@example2.com]","returnSecureToken":true}’
```

4. Password Update:

```
 curl 'https://identitytoolkit.googleapis.com/v1/accounts:update?key=[API_KEY]’  -H 'Content-Type: application/json’  --data-binary
'{"idToken":"[GCIP_ID_TOKEN]","password":"[NEW_PASSWORD]","returnSecureToken":true}'
```

Exploiting IdToken using Refresh token

1. Login:

```
curl 'https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=[API_KEY]'  -H 'Content-Type: application/json' --data-binary '{"email":"[user@example.com]","password":"[PASSWORD]","returnSecureToken":true}'
```

- The returnSecureToken will give the refreshToken upon Successful Login.
- To refresh an expired idToken we need APIKEY of the firebase Project and the refreshToken, in the response we will get the idToken.
- can we also use another company's APIKEY to refresh the expired idToken using refreshToken?? YES, we can! which is a business logic bug in the firebase

Mitigations

- Custom Claim should be implemented which should track all the token’s and destroy them once custom token is revoked.
- User Enumeration can be avoided by opting the firebase security settings.
- Sign-in Method can be restricted and Admin SDK can signup new user.
- Use revokeRefreshToken() once user logout’s
- verifyIdToken() can check if the ID Token is revoked or not

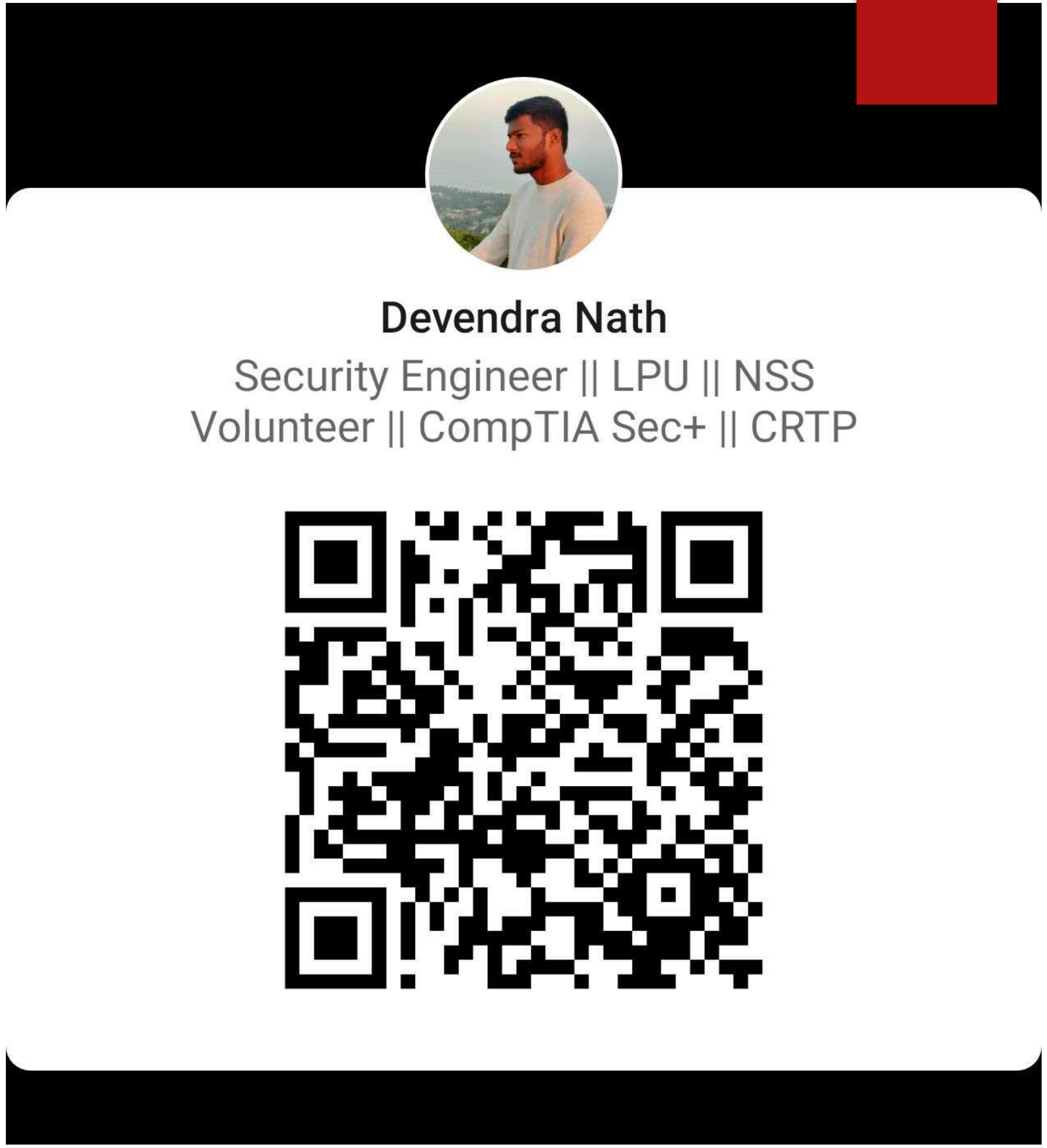
REFERENCES

- https://www.comparitech.com/blog/information-security/firebase-misconfiguration-report/
- https://www.linkedin.com/pulse/firebase-common-security-misconfigurations-clear-gate-gbtwf/
- https://firebase.google.com/support/release-notes/admin/node#version_570_-_january_04_2018
- https://medium.com/@DEVEN99/firebase-idtoken-creation-via-refresh-token-using-another-organization-apikey-bug-or-intended-ca7246472181
- https://medium.com/@DEVEN99/exploring-firebase-authentication-unveiling-nuances-in-apikey-interactions-across-key-9cb225a6fbb4
- https://medium.com/@DEVEN99/securing-firebase-authentication-mitigating-vulnerabilities-and-best-practices-593981e61b98

Thank You

▶ You can reach out to me if you have any doubts

- LinkedIn: https://www.linkedin.com/in/devendra-nath-b343a5190/
- Twitter: https://twitter.com/Asva_Shelby
- GitHub: https://github.com/sharathdev19
- Email: devendranathreddy214@gmail.com



Devendra Nath

Security Engineer || LPU || NSS Volunteer || CompTIA Sec+ || CRTP