Fassbender

INFORMATION SECURITY

202101414-1
Report Security Analysis ZITADEL Console
2021-11-10

Table of Contents

1. Version Control

Version	Date	Author	Rationale
0.1	2021-05-13	Sven Fassbender	First Draft of the Report created
0.2	2021-05-14	Sven Fassbender	Findings Added
0.3	2021-05-15	Sven Fassbender	Findings Added
0.4	2021-05-16	Sven Fassbender	Findings Added
0.5	2021-05-17	Sven Fassbender	Findings and Management Summary Added
1.0	2021-05-17	Sven Fassbender	First Draft for CAOS AG Created
1.1	2021-05-31	Sven Fassbender	Final Report send to CAOS AG, no Changes Requested
1.2	2021-11-10	Livio Amstutz (CAOS AG)	Redacted PII
1.3	2021-11-10	Sven Fassbender	Redacted PII

2. Management Summary

CAOS AG tasked Fassbender Information Security to perform a security analysis of the ZITADEL Console web application. The security analysis took place between May 13th and May 17th, 2021. The analyst was provided with an IAM Administrator account on a publicly accessible instance of the in-scope application. During the analysis the authentication and authorization controls were primarily focused. Furthermore, automated tests were conducted to identify other classes of vulnerabilities. Although, GDPR was not in the primary scope, some potential issues were identified in this area.

The overall level of security is rated as good, solely one issue was identified with a high CVSS score. Missing encryption of data at rest, can result in a full loss of confidentiality, integrity, and availability of the stored data. An attacker with access to the database, such as a regular administrator, can read and alter the information. In a cloud environment, the risk is even higher since administrators of the environment itself can also access the data.

The second highest risk is successful attacks on users who solely rely on username and password as credential. Whilst the platform offers various stronger authentication methods, the usage of this weak authentication method is still supported. Furthermore, the in-place controls to prevent efficient online guessing attacks of weak user passwords deviate from security best practices recommendations.

ZITADEL Console fails to enforce certain IAM policies on the server-side. The policies *allow registration* and *allow username plus password login*, remove those options from the login page user interface. Nevertheless, it was found that performing those actions is still possible if the technical details about the HTTP requests are known to the attacker.

It was found that the session credential (access token) is not terminated server side, whenever the user performs a password change or a manual logout. The likelihood of rogue sessions is increased due to the increased window of opportunity for an attacker. Session credentials must be terminated on the client and the server side, whenever a password change or a logout occurs. An attacker could access the application in the compromised user's context. Furthermore, due to missing re-authentication control for changes in the account section, an attacker could change the users e-mail address and utilize the password forgotten functionality to persist access to the account.

Integrating the user themselves as a security control, by giving them more information and power in the session management could further harden the application. For example, the user should be informed about failed login attempts. Also, an overview of all active sessions with the option to terminate them can help to prevent and detect rogue sessions.

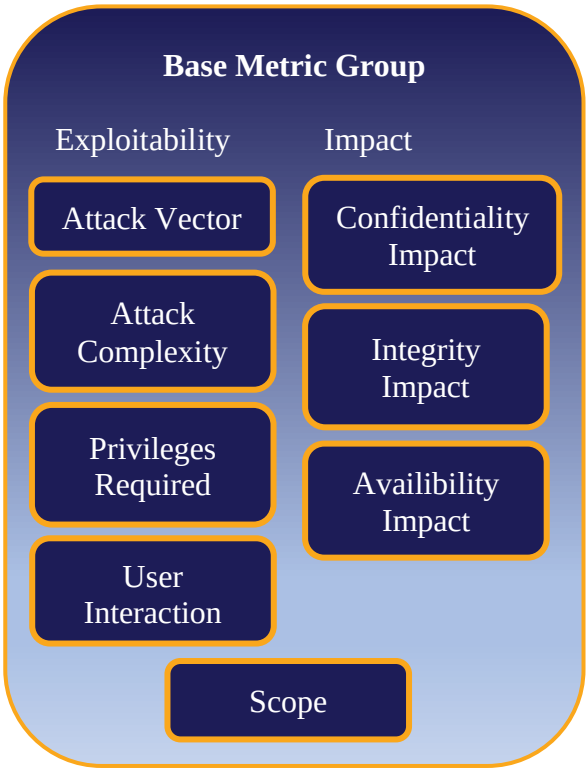
It was not possible to perform a thorough security analysis of the *login with external IDP* feature in the given timeframe.

3. Overview of Findings

No.	Title	CVSS:3.1
8.1	Data not Encrypted at Rest	8.2 (High)
8.2	Failed Login Attempts Not Displayed	6.5 (Medium)
8.3	Missing Protection Against Online Guessing	6.5 (Medium)
8.4	Consider Restriction of Common or Compromised Passwords	6.5 (Medium)
8.5	IAM Policies not Enforced	5.3 (Medium)
8.6	Username Enumeration Possible	4.3 (Medium)
8.7	Verbose Error Messages	4.3 (Medium)
8.8	Consider Usage of Refresh Token	3.3 (Low)
8.9	Logout does not Terminate Session	3.3 (Low)
8.10	Password Change does not Terminate Session	3.3 (Low)
8.11	Missing Re-Authentication for Sensitive Operations	3.3 (Low)
8.12	Overview of Active Sessions not Displayed	3.3 (Low)
8.13	MFA-Bypass Passwordless Authentication	3.1 (Low)
8.14	Username Recovery Option Missing	2.7 (Low)
8.15	Third-Party Hosted Resources Embedded	2.7 (Low)
8.16	WebAuthn Signature Verification	2.5 (Low)
8.17	PII in Application Logs	2.4 (Low)
8.18	Cacheable HTTPS Response with Sensitive Data	2.2 (Low)
8.19	Implicit Grant Type Supported	2.2 (Low)
8.20	Consider Security.txt	0.0 (None)
8.21	Missing Option to Delete Account	0.0 (None)
8.22	Missing Consent GDPR Third-Person Registration	0.0 (None)

4. Vulnerability Scoring

The findings within this report are scored according to the CVSS:3.1 base metric group. This scoring considers the following parameters.



The Common Vulnerability Scoring System (CVSS) is used for communicating the characteristics and severity of software vulnerabilities. The scoring is internationally applied in the industry and by security researchers.

In case that the necessary data for the base metrics is incomplete or varies, the analyst assigned the scores following a worst-case approach.

5. Background

CAOS AG, in the sequel denoted as *client*, tasked **Fassbender Information Security**, further denoted as *contractor*, to perform a security analysis of the **ZITADEL Console** web application.

ZITADEL Console is developed and hosted by the client. The web application is internet facing. The whole service is defined as "Cloud Native Identity and Access Management" (IAM). Whereas the ZITADEL Console is used by the CAOS AG customers to administer their dedicated instance of the IAM.

6. Scope

As scope of the security analysis, the client and contractor agreed on 2021-04-26, to the following:

- ZITADEL Console Web Application
  - https://api.zitadel.app
  - https://console.zitadel.app
  - https://accounts.zitadel.app
  - https://issuer.zitadel.app

The analyst has been provided with a user of the following role, which allowed creation of all subordinate roles:

- IAM Administrator

The security analysis was based on, the latest publicly available OWASP Top 10 web application security risks:

- Injection
- Broken Authentication
- Sensitive Data Exposure
- XML External Entities
- Broken Access Control
- Security Misconfiguration
- Cross-Site Scripting (XSS)
- Insecure Deserialization
- Using Components with Known Vulnerabilities
- Insufficient Logging & Monitoring

Not all the above listed topics were feasible in the given timeframe. Therefore, the contractor, as agreed with the client, prioritized the following topics:

- Broken Authentication
- Broken Access Control

7. Schedule and Place of Services

The offered services were provided in the contractor's premises. The client provided the contractor with a test environment in the specified period. The services were provided in the period between 2021-05-13 to 2021-05-17.

The final report will be delivered latest by 2021-05-28. Delivery object will be an encrypted document, send by e-mail to the following e-mail address.

- florian@caos.ch

8. Findings

This section contains the technical details of all identified findings.

8.1. Data not Encrypted at Rest

Class	Data Protection
Effort	Low
CVSS:3.1	8.2 (High)
Vector String	CVSS:3.1/AV:L/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H

The ZITADEL Console web application does not encrypt data at rest. An attacker with access to the database, can access the stored secret and personal identification information data.

According to security best practices data at rest must be protected by encryption. The encryption ensures the confidentiality of the stored data. Depending on the architecture, different possibilities for encryption of data can be applied or combined. For example:

- Encryption of Hard Disk
- Encryption of Database
- Encryption of Data in Application Layer

It is important to mention, that an attacker with access to the encryption key can of course decrypt the data. Therefore, the encryption key must be stored separate from the encrypted data. Encryption keys should be, generated and stored in a separate hardware module e.g., HSM.

An attacker with access to the data at rest can access the cleartext information. Depending on the information that is accessible, the impact varies. Following is listed some information that may be at risk in the ZITADEL Console web application:

- Personal Identification Information (PII) of users
- Issued Access Token
- OAuth 2.0 Authorization Codes
- Encryption Keys

Recommendation

It is recommended to encrypt data at rest, utilizing encryption keys generated and stored in an independent and secure location e.g., Google Cloud HSM.

8.2. Failed Login Attempts Not Displayed

Class	Session Management
CVSS:3.1	6.5 (Medium)
Vector String	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N

The ZITADEL Console web application does not display failed login attempts after a successful authentication. An attacker may profit of this missing control, since a user will not know, that he was subject to an attack.

According to security best practices, the user should be informed about failed login attempts after a successful authentication. Failed logins can result in different scenarios. In one scenario the attacker is successful, in this case the control of displaying failed login attempts does not help. In another scenario the attack was not (yet) successful. The attacker may return later e.g., once the brute-force protection threshold has been reset or more information on the victim were gathered. In this scenario the user who can see that he has been subject to an attack, can take countermeasures. These countermeasures can be digital or physical. One potential countermeasure could be, to renew the password. Another could be to watch out for shoulder surfing attempts in real-life.

An attacker must have access to the login page of the web application. Furthermore, he must be in the possession of the victim's username (see finding 8.6). To be successful the attacker must know the user's password. It's unlikely that this control helps in a scenario where an attacker has access to the users MFA, the assumption is that in this case the attacker can also sniff or record the user's password.

Recommendation

It is recommended to implement the failed login attempts control to give users the ability to detect that they have been subject to an attack. The following information can be displayed:

- Number of failed login attempts
- Time of the last failed login attempt
- IP-Address and Browser information of the failed login attempts

Furthermore, the user can be assisted with information, on how to react in case that some number of failed login attempts took place.

8.3. Missing Protection Against Online Guessing

Class	Authentication
CVSS:3.1	6.5 (Medium)
Vector String	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N

The ZITADEL Console web application does not protect user accounts with an application layer mitigation mechanism against automated brute-force attacks. An attacker with access to the login page can perform an online brute-force attack to guess a user's password.

Account lockout mechanisms can be, depending on the context of the application, the preferred measure to discourage attackers from online brute-forcing the password of a user. Whilst rate-limiting and firewalling may mitigate some risks on the network layer. The account lockout mechanism protects on the application layer. One issue with account lockout mechanisms, is the potential that a legitimate user locks out himself, by typing the wrong password repeatedly. Furthermore, an attacker may abuse a lockout mechanism to Denial-of-Service the user. To minimize those risks, but also implementing an application layer protection against brute-force attacks, Captchas can be used.

Captchas are online puzzles that are designed to be easily solved by human users, but hard for machines.

An attacker with access to the login page, who knows a user's login name e.g., by guessing (see 8.6) or by shoulder surfing can attempt to brute-force the password. Depending on the role of the compromised account the attacker can take actions available in the user's context. This finding only affects user accounts that have only password authentication enabled. User accounts that are protected by MFA will not get compromised by a guessed password.

Recommendation

It is recommended to implement a CAPTCHA over an account lockout mechanism. The solving of a CAPTCHA could be enforced after e.g., ten failed login attempts. This threshold should be sufficient for users who repeatedly enter wrong passwords but low enough to discourage an attacker from brute-forcing a user's password. The threshold value can be reset after a certain amount of time or after a successful login.

See also recommendations of finding 8.2.

Other possible but more invasive countermeasures for preventing online guessing:

- Account Lockout
- Tar Pit
- Don't allow login solely by password

8.4. Consider Restriction of Common or Compromised Passwords

Class	Session Management
CVSS:3.1	6.5 (Medium)
Vector String	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N

The ZITADEL Console web application does not offer the option for IAM Administrators to restrict the usage of common and compromised passwords. An attacker can be successful using a wordlist based brute force attack to takeover user accounts.

According to security best practices, a user should be prevented from using common, weak, and known to be compromised passwords. Users tend to use or reuse weak or breached passwords. This behavior is especially risky if additional controls such as MFA are not in place. To provide administrators with another control option it can be considered to restrict the usage of those passwords. Even though a "strong" password policy is in place users can still use passwords like *P@ssword123*.

An attacker who wants to get access to a user's account that is protected with a password like *P@ssword123*, likely will be successful within hours. In case, that no other authentication credentials like MFA are required, the attacker can take full control of the user's account. Depending on the role of the compromised account the attacker can take actions available in the user's context.

Recommendation

It is recommended to restrict the usage of common, weak, and compromised passwords.

For example, the password could be compared with the SHA-1 representation of compromised passwords, downloadable from *haveibeenpwned*[1]. If the result of the comparison shows that the hash of the selected password is true, the user should be prompted to select another password.

[1] Haveibeenpwned.com, Passwords, https://haveibeenpwned.com/Passwords, last visited 2021-05-15

8.5. IAM Policies not Enforced

Class	Authentication
CVSS:3.1	5.3 (Medium)
Vector String	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

The ZITADEL Console web application fails to disable user registration and password authentication on the server-side. An attacker who knows the technical details, on how to register a user, can register a user without access to any projects or organization information.

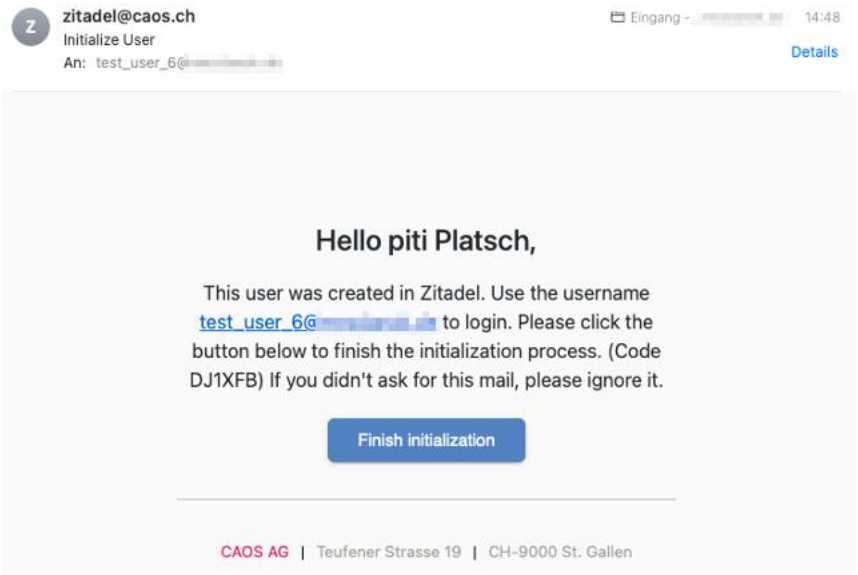
An IAM Administrator has the option to define policies for the IAM. These policies also contain the option to disable the registration feature, as well as disabling login by solely username plus password. This controls does indeed remove the options from the login page of the ZITADEL Console. Nevertheless, during the analysis it was found that it is still possible to register a user and to login with username and password.

To register a user, even if the option is disabled, an attacker must send a HTTP POST request like the following:

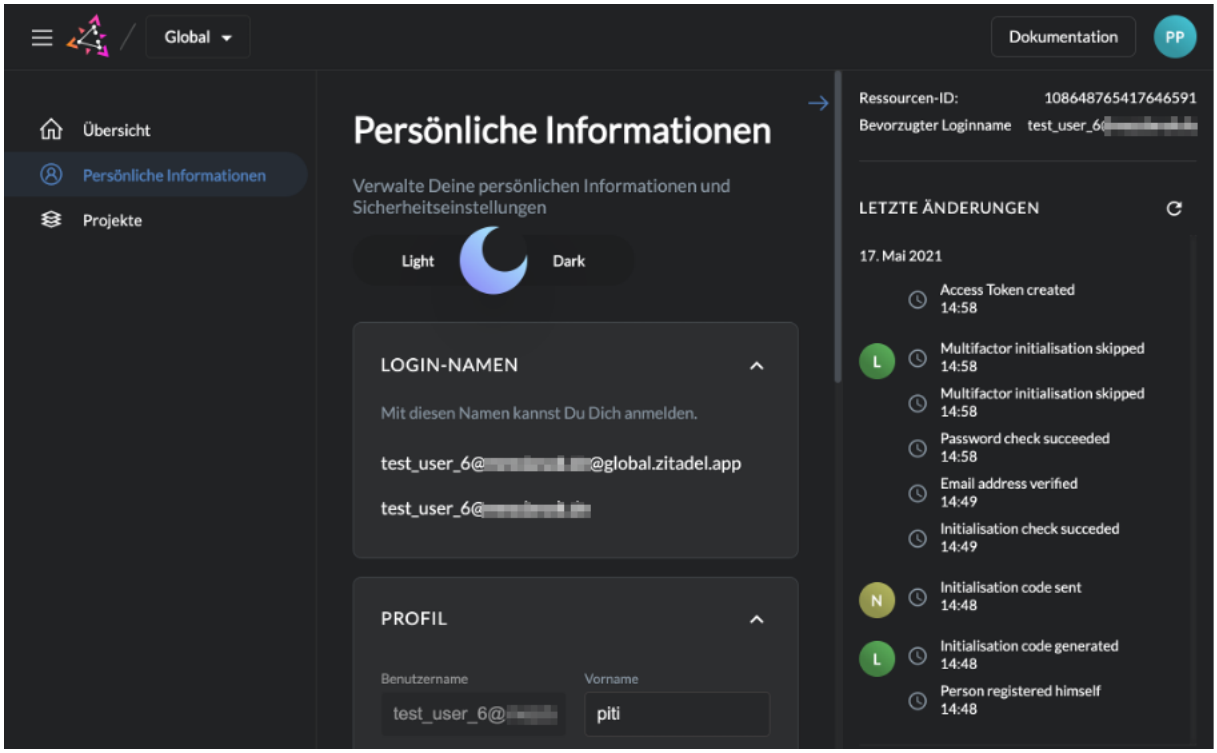
```
POST /register HTTP/1.1
Host: accounts.zitadel.app
Connection: close
Content-Length: 325
Cache-Control: max-age=0
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
Origin: https://accounts.zitadel.app
Content-Type: application/x-www-form-urlencoded
User-Agent: [REDACTED]
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://accounts.zitadel.app/register
Accept-Encoding: gzip, deflate
Accept-Language: de-DE,de;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: [REDACTED]

gorilla.csrf.Token[REDACTED]&authRequestID=108637909065723281&firstname=piti&lastname=Plats
ch&email=test_user_6%40[REDACTED]&language=&gender=&register-password=[REDACTED]&register-
password-confirmation=[REDACTED]&register-term-confirmation=on
```

Shortly after the HTTP request is send, an e-mail message like the following is received.



The initialization process can be finished sending the code, as the e-mail states. Afterwards a login is possible. As can be seen on the following screenshot, it's not possible to access organization or project related information.



An attacker with this level of access, could try to identify vulnerabilities within the system and try to attack other users of the system. During the security analysis, no vulnerabilities were identified, that could be exploited with this level of access.

The same issue is present in case of the *disable login by username and password* policy. As can be seen in the following HTTP request and the subsequent HTTP response, after providing the password, the OAuth flow is started, and a login is possible.

```
POST /password HTTP/1.1
```

```
Host: accounts.zitadel.app
Connection: close
Content-Length: 205
Cache-Control: max-age=0
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
Origin: https://accounts.zitadel.app
Content-Type: application/x-www-form-urlencoded
User-Agent: [REDACTED]
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://accounts.zitadel.app/userselection
Accept-Encoding: gzip, deflate
Accept-Language: de-DE,de;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: [REDACTED]

gorilla.csrf.Token=[REDACTED]&authRequestID=108656148516759600&loginName=test_user_6%40[RED
ACTED]&password=[REDACTED]
```

```
HTTP/1.1 302 Found
Date: Mon, 17 May 2021 14:02:24 GMT
Content-Length: 0
Connection: close
cache-control: no-store
content-security-policy: font-src 'self';manifest-src 'self';default-src 'none';script-src
'self' 'nonce-0onEZRyZUId4ftaLkjJSYoSniiJFGaZ5IYSwpCZuvZA=';img-src 'self';media-src
'none';frame-src 'none';connect-src 'self';object-src 'self';style-src 'self' 'nonce-
0onEZRyZUId4ftaLkjJSYoSniiJFGaZ5IYSwpCZuvZA='
expires: Mon, 17 May 2021 13:02:22 GMT
feature-policy: payment 'none'
location: https://accounts.zitadel.app/oauth/v2/authorize/callback?id=108656148516759600
permissions-policy: payment=()
pragma: no-cache
referrer-policy: same-origin
set-cookie: [REDACTED]
strict-transport-security: max-age=31536000; includeSubDomains
traceparent: 00-465485370ff9371e5154a46e6d7b2872-fa7a8edf6aad4f01-00
vary: Cookie
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 1; mode=block
x-envoy-upstream-service-time: 1578
CF-Cache-Status: DYNAMIC
cf-request-id: 0a1c3ba0a80000073eec095000000001
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-
cgi/beacon/expect-ct"
Report-To:
{"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report?s=8%2BBsPDKGOEo1cGQZ299fkxOdT1
IjC5hX94tcgcF9pO5dKKqiJdKnC02jpjcoiozYUjiiGiMuZCV2ynV4IOVbxC0fO%2Bf5Kz2ZGNsdZF2IN%2FCkIjXHf
g%3D%3D"}],"group":"cf-nel","max_age":604800}
NEL: {"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 650d62143c5a073e-FRA
alt-svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400
```

An attacker who can bypass the policy, can still login as the user. Depending on the role of the compromised account the attacker can take actions available in the user's context.

Recommendation

It is recommended to enforce the policies on the server-side, whenever an IAM Administrator defines this control in the policy.

8.6. Username Enumeration Possible

Class	Authentication
CVSS:3.1	4.3 (Medium)
Vector String	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N

The ZITADEL Console web application login allows enumeration of usernames, due to a warning that is showed after entering a non-existing username. An attacker with access to the login page can perform an online brute-force attack to enumerate existing usernames.

For authentication on the web application at least a username and a password are necessary. Depending on the installation, the user can also use MFA and password less authentication. All authentication methods share, that a valid username is required. Therefore, an attacker must know the username to attack a victims account.

The web application leaks information about existing usernames on the login page. Entering a non-existing username results in an error message like the following.

On the other hand, when a valid username is entered the flow continues and the user must enter additional authentication information.

An attacker with access to the login page can enumerate existing usernames by this behavior. Especially accounts that solely rely on password authentication, can be efficiently attacked with this knowledge. (See finding 8.2 and 8.3)

Recommendation

It is recommended to not disclose information about existing accounts.

The user should be able to enter all authentication information, an error message should be presented not until the verification of all provided credentials fails server side. It is important to ensure that no side channels for enumeration of credentials exist. Sometimes user credentials can be enumerated by timing side channels. It is recommended to make sure that the server response time does not differ, even though one of the provided credentials is valid.

8.7. Verbose Error Messages

Class	Information Disclosure
CVSS:3.1	4.3 (Medium)
Vector String	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N

The ZITADEL Console web application shows verbose error messages in the HTTP Header Grpc-Message. This information can be helpful for an attacker to mount further attack steps.

Detailed technical error messages that are useful during the development of an application are not useful for a user. But an attacker may be able to use this information to conclude which software components or frameworks are used. This information can be useful to search for or discover vulnerabilities in the product.

The following HTTP server response shows an exemplary error message.

```
HTTP/2 200 OK
Date: Thu, 13 May 2021 07:45:48 GMT
Content-Type: application/grpc-web+proto
Content-Length: 0
Grpc-Status: 2
Grpc-Message: ID=SQL-M0dsf Message=Project already exists on organisation Parent=(pq:
duplicate key value
(unique_type,unique_field)=('project_names','test_projekt107952890689153079') violates
unique constraint "primary")
X-Envoy-Upstream-Service-Time: 152
Access-Control-Allow-Origin: https://console.zitadel.app
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: *
Cf-Cache-Status: DYNAMIC
Cf-Request-Id: 0a06496ddb00002fa5d2393000000001
Expect-Ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-
cgi/beacon/expect-ct"
Report-To:
{"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report?s=tR7Wv7cc9qoHahc42lmLLF4lxyd0
eSh%2Bi0UEOBNbJSU8SwaqsjNaU6AuNwaV6zHYAAfBxKzhWlYbZlYMi3sovOWlXtFEGvNEW2MfOKBV2nE%3D"}],"gr
oup":"cf-nel","max_age":604800}
Nel: {"report_to":"cf-nel","max_age":604800}
Server: cloudflare
Cf-Ray: 64ea44f62a7f2fa5-FRA
Alt-Svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400
```

As can be seen in the previous example an error generated by a backend database is present in the server response. An attacker can conclude that a *CockroachDB* is in use. Armed with this knowledge, may ease the discovery of SQL-injection vulnerabilities. No SQL-injection vulnerabilities were identified in the given timeframe.

Since the web application is an Open-Source project, this information can also be found in the public repository. Nevertheless, the analyst cannot exclude that the same behavior could lead to exposure of Closed-Source code elsewhere.

Recommendation

It is recommended to show error-codes like in this example *SQL-M0dsf*, that can be linked by the developers to the detailed technical error messages. This error-code can be enriched with a timestamp, that would allow the user to send this information to the developer, who can look up the error in the log files.

8.8. Consider Usage of Refresh Token

Class	OAuth 2.0 Recommendation
CVSS:3.1	3.3 (Low)
Vector String	CVSS:3.1/AV:L/AC:H/PR:L/UI:R/S:U/C:L/I:L/A:N

The ZITADEL Console web application issues access token with an extensive lifetime. An attacker with access to such a token can impersonate the legitimate user for the validity period.

The access token is issued after completion of the OAuth 2.0 authentication flow, which can request multiple credentials of the user for example, username, password, and One-Time-Password. The server's response contains the *access_token* as well as an *id_token*:

```
HTTP/1.1 200 OK
Date: Fri, 14 May 2021 09:31:18 GMT
Content-Type: application/json
Connection: close
access-control-allow-credentials: true
access-control-allow-origin: https://console.zitadel.app
cache-control: no-store
expires: Fri, 14 May 2021 08:31:18 GMT
pragma: no-cache
[…]

{"access_token":"yu8eaTXj3T[REDACTED]erfQpyhG--u8nW46E5h[REDACTED]58P-
g","token_type":"Bearer","expires_in":43199,"id_token":"[REDACTED]"}
```

Consequently, the access tokens are used to authenticate for accessing a protected resource on the ZITADEL API. Therefore, the access token is sent in the Authorization header as Bearer Token.

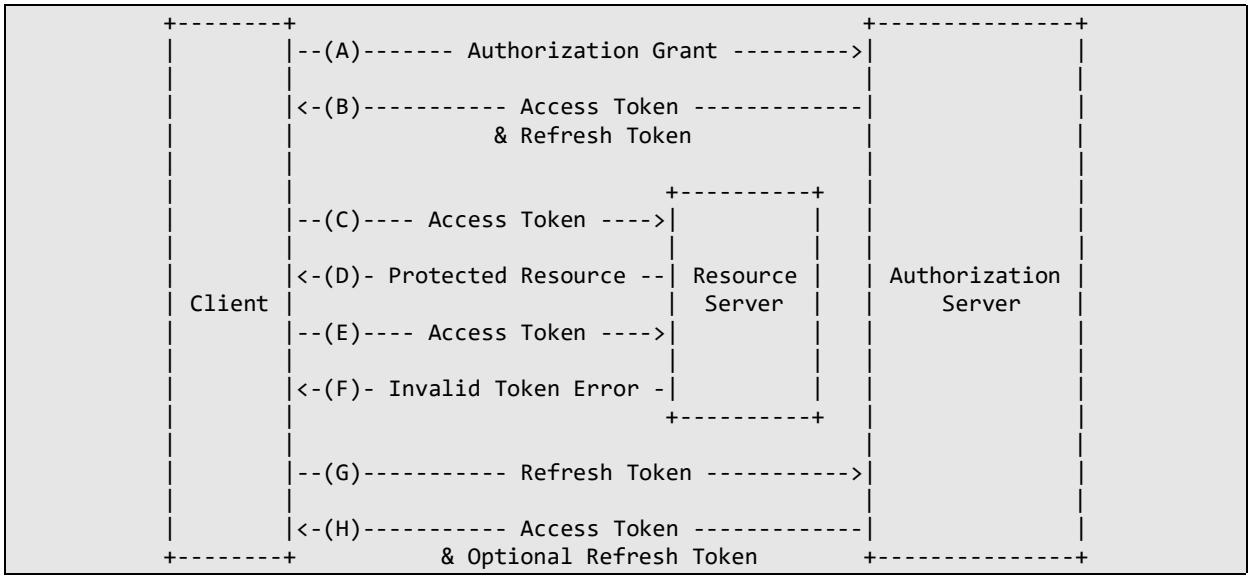
```
POST /zitadel.auth.v1.AuthService/GetMyUser HTTP/1.1
Host: api.zitadel.app
Connection: close
Content-Length: 5
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90"
X-User-Agent: grpc-web-javascript/0.1
Accept-Language: de-DE
sec-ch-ua-mobile: ?0
Authorization: Bearer yu8eaTXj3T[REDACTED]erfQpyhG--u8nW46E5h[REDACTED]58P-g
Content-Type: application/grpc-web+proto
Accept: */*
X-Grpc-Web: 1
x-zitadel-orgid: 107952890689153079
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/90.0.4430.212 Safari/537.36
Origin: https://console.zitadel.app
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Accept-Encoding: gzip, deflate
```

An attacker who gets in the possession of the access token can impersonate the victim for the validity period. The validity period is around 12 hours in the current implementation. That results in a huge window of opportunity. Furthermore, the logout function yet does not

terminate the session server-side (see finding 8.9). An attacker must be able to access the access token to take advantage of this issue.

Recommendation

It is recommended to use refresh and access token. Access token can have a short lifetime and can be invalidated after a certain amount of resource accesses. On the other hand, the refresh token can have a longer lifetime, it's only used every couple of minutes/hours to fetch a new set of access and refresh token. See the following schematic[2]:



This reduces the load on the authorization server, as the resource server does not need to call the authorization server, whenever a protected resource is accessed.

Furthermore, the impact of a compromised access token is lowered, since the window of opportunity for an attacker is smaller, due to the shorter lifetime.

Also, refresh token can help to identify compromised sessions. If an attacker compromises a refresh token and consequently creates a new set of access and refresh token. The legitimate users refresh token, will be rejected by the authorization server once he tries to fetch a new set of tokens. This will result in the need to re-authenticate. The authorization server can show a meaningful error response, which educates the user about the "probably" compromised refresh token. This will allow the user to take actions to secure the account.

Following some considerations[3], for implementing refresh token:

- The scope of the access as well as the refresh token must be limited. Access token must only be allowed to access a protected resource on the resource server. Refresh tokens must only be allowed to create a new set of access and optional refresh token.
- Refresh token must be revoked after being used once (refresh token rotation). (See OAuth-Revocaction[4])

[2] Datatracker.ietf.org, rfc6749, https://datatracker.ietf.org/doc/html/rfc6749#section-1.5, last visited 2021-05-14
[3] Datatracker.ietf.org, rfc6819, https://datatracker.ietf.org/doc/html/rfc6819#section-5.2.2, last visited 2021-05-16
[4] Datatracker.ietf.org, rfc7009, https://datatracker.ietf.org/doc/html/rfc7009#section-2, last visited 2021-05-16

- Whenever a user performs a logout, the refresh token and all access tokens must be revoked server-side. (See OAuth-Revocaction[5])
- Choose rational lifetimes for access and refresh token
- Refresh token should be bound to the *client_id*, to prevent token theft, whenever a new token set is requested *the client_id* must match

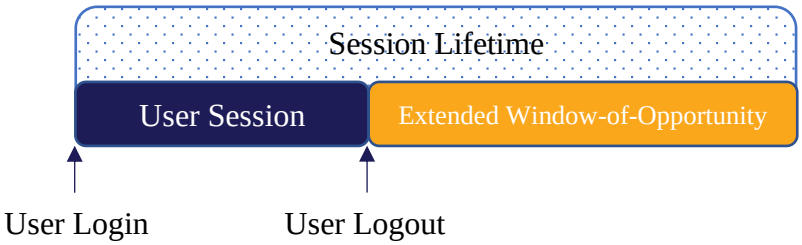
[5] Datatracker.ietf.org, rfc7009, https://datatracker.ietf.org/doc/html/rfc7009#section-2, last visited 2021-05-16

8.9. Logout does not Terminate Session

Class	Session Management
CVSS:3.1	3.3 (Low)
Vector String	CVSS:3.1/AV:L/AC:H/PR:L/UI:R/S:U/C:L/I:L/A:N

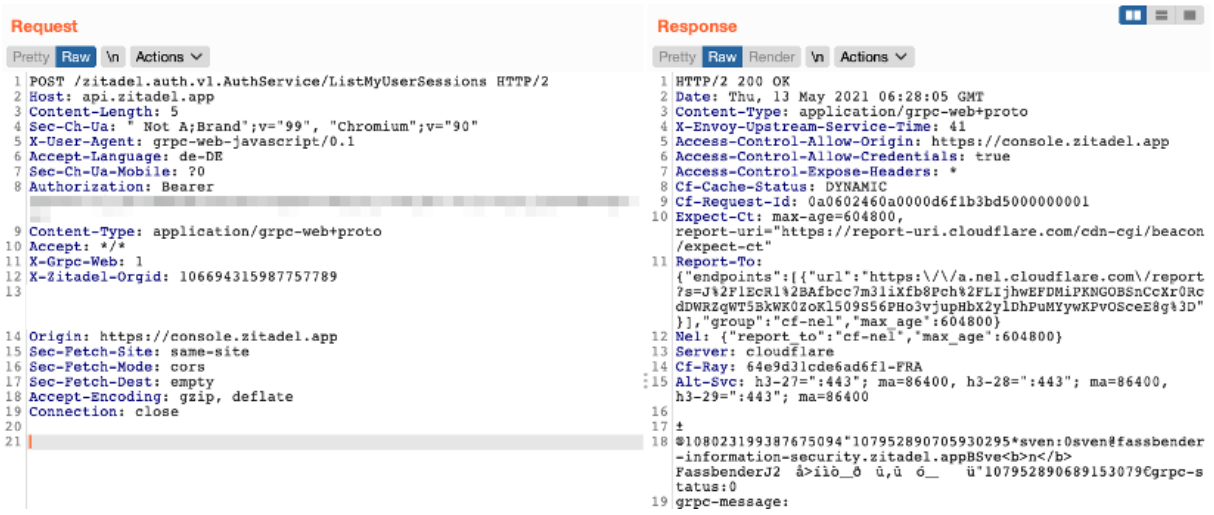
The ZITADEL Console web application fails to terminate a user session on a manual logout action. The window of opportunity, for an attacker is extended since the user is not able to terminate the session.

A manual logout by the user should result in a termination of the issued credential (access token) on the client as well as on the server side. Not terminating the credential server side leads to an extended window of opportunity for an attacker.



A manual logout in the web application leads to client-side termination of the credentials. Nevertheless, it was found that the credentials still can be used to authenticate on the ZITADEL API.

The following exemplary screenshot shows a HTTP request and response that were send, after a manual logout was performed in the user interface of the application.



As can be seen in the previous screenshot, the credential in the HTTP Authorization header is still accepted by the ZITADEL API. The protected resource *ListMyUserSessions* can be accessed, and the server response contains the requested information.

An attacker who got in the possession of a user credential (access token) can take advantage of the extended window of opportunity. He can impersonate the victim for the whole lifetime

of the access token. (Also see findings 8.8 and 8.12) Depending on the role of the compromised account the attacker can take actions available in the user's context.

Recommendation

It is recommended to terminate the issued session on a manual logout by the user. The credential must be deleted on the client side. Furthermore, the credential must be deleted or revoked on the server side.

It can also be considered, to terminate all existing sessions of that user, when a manual logout is performed. This would also terminate sessions, that somehow exist in parallel. Sometimes users forget to manually perform a logout. Depending on the requirements of the application, this approach can destroy some business logic, therefore this control is recommended optionally.

In case that a cryptographically signed credential e.g., JSON Web Token (JWT) is used, a direct termination of the credential is not possible. In general, the validation of JWT basically takes place by validating the signature and the expiry time encoded in the JWT body. In this case, a session identifier value should be issued within the JWT. The check of the credential must then also contain a verification of the session identifier. Once the user performs a manual logout, the session identifier must be invalidated server side.

8.10. Password Change does not Terminate Session

Class	Session Management
CVSS:3.1	3.3 (Low)
Vector String	CVSS:3.1/AV:L/AC:H/PR:L/UI:R/S:U/C:L/I:L/A:N

The ZITADEL Console web application does not prompt the user with the option to terminate the session when a password change was successful. An attacker who is in the possession of a user session can benefit from a larger window of opportunity.

According to security best practices, a user should be prompted with the opportunity to terminate all other active sessions after a successful password change. This control can be helpful for a user who wants to take back control of compromised sessions. (See also finding 8.12)

An attacker must be in the possession of another user's access token, to take advantage of this issue. The access tokens are available in the session storage of a user's web browser. The access tokens have an extensive lifetime and are not revoked on a logout. (See findings 8.8 and 8.9) Depending on the role of the compromised account the attacker can take actions available in the user's context.

Recommendation

It is recommended to prompt the user with the option to terminate other active sessions, after a successful password change.

8.11. Missing Re-Authentication for Sensitive Operations

Class	Authentication
CVSS:3.1	3.3 (Low)
Vector String	CVSS:3.1/AV:L/AC:H/PR:L/UI:R/S:U/C:L/I:L/A:N

The ZITADEL Console web application does not force a user to authenticate again, before performing sensitive operations in the user's context. An attacker who somehow got in the possession of an access token can make changes in the user account settings.

According to security best practices, a user should be forced to re-authenticate whenever a sensitive operation is performed. Sensitive operations can not only affect a single user, such as change of e-mail address or change of password but also multiple users, such as change of password policies disable of two-factor-authentication and so on.

For example, this control is also recommended by the W3C in the WebAuthn specification[6], as can be seen in the following excerpt.

> *When initiating a registration ceremony, interrupt the user interaction after the e-mail address is supplied and send a message to this address, containing an unpredictable one-time code and instructions for how to use it to proceed with the ceremony. Display the same message to the user in the web interface regardless of the contents of the sent e-mail and whether or not this e-mail address was already registered.*

An attacker must be in the possession of another user's access token, to take advantage of this issue. The access tokens are available in the session storage of a user's web browser. The access tokens have an extensive lifetime and are not revoked on a logout. (See findings 8.8 and 8.9) Depending on the role of the compromised account the attacker can take actions available in the user's context.

Recommendation

To strengthen the security of the web-application and decrease the impact of compromised tokens a re-authentication for sensitive operations should be considered.

The analyst recommends implementing the re-authentication especially for users with administrative permissions, such as IAM-Administrator, as the impact of a compromised token is higher, than for users with less permissions.

[6] W3C, WebAuthn, https://www.w3.org/TR/webauthn-2/#sctn-username-enumeration, last visited 2021-05-15

8.12. Overview of Active Sessions not Displayed

Class	Session Management
CVSS:3.1	3.3 (Low)
Vector String	CVSS:3.1/AV:L/AC:H/PR:L/UI:R/S:U/C:L/I:L/A:N

The ZITADEL Console web application does not display a user's active sessions. An attacker with access to a compromised user account can profit from this missing control since a user does not have the ability to detect the compromised state.

According to security best practices web applications should implement the possibility for a user to see and manage active sessions. This control not only empowers a user to terminate old sessions, but also potentially detect and end a compromised state. The active session overview could also have some meta data related to the session, such as browser type or IP-address that helps the user to identify the origin of the session.

Implementing the session overview control can decrease the window of opportunity for an attacker.

An attacker must be in the possession of a compromised account, to take advantage of this issue. Depending on the role of the compromised account the attacker can take actions available in the user's context. The effort for an attacker to fully compromise a user account, depends on the settings and the users' choices.

Recommendation

It is recommended to implement a session overview for users of the web application. The user should be able to manually terminate a session in this view. To allow tracking of the session's origin meta data such as IP-address, start-time and browser type can be presented.

8.13. MFA-Bypass Passwordless Authentication

Class	Session Management
CVSS:3.1	3.1 (Low)
Vector String	CVSS:3.0/AV:P/AC:L/PR:L/UI:R/S:U/C:L/I:L/A:N

The ZITADEL Console web application allows an attacker to login without providing the MFA credential, in a specific edge case scenario. An attacker with physical access to the passwordless credential can bypass the MFA requirement by solely providing the password.

It is possible to use *passwordless* authentication alongside other authentication options. It is possible to have *passwordless* authentication available in parallel with password plus MFA authentication.

During the analysis an edge case was identified, that would allow an attacker to login without providing the MFA. The attacker must be in the possession of a valid *credentialAssertionData*. This data is only issued, if *passwordless* authentication is enabled in the IAM policies. If enabled the *userselection* HTTP response will contain the necessary data, as can be seen in the following excerpt:

```
[…]
    <input type="hidden" name="authRequestID" value="108639252417407016"/>
    <input type="hidden" name="credentialAssertionData" value="[REDACTED]"/>
    <input type="hidden" name="credentialData"/>
[…]
```

In case that in this exact moment the IAM Administrator disables the *passwordless* authentication option (Edge Case), the client is still in possession of the *credentialAssertionData*.

This data is used by the client to create the following HTTP request:

```
POST /login/passwordless HTTP/1.1
Host: accounts.zitadel.app
Content-Type: application/x-www-form-urlencoded
Origin: https://accounts.zitadel.app
Accept-Encoding: gzip, deflate
Cookie: [REDACTED]
Connection: close
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: [REDACTED]
Referer: https://accounts.zitadel.app/userselection
Content-Length: 1175
Accept-Language: de-ch

gorilla.csrf.Token=[REDACTED]&authRequestID=108639252417407016&credentialAssertionData=[RED
ACTED]&credentialData=[REDACTED]
```

The server requires asks the user to enter the password now, since *passwordless* authentication is disabled, but entering the MFA is not required.

This very rare edge case is unlikely to ever be exploited in practice. Nevertheless, an attacker who is in the possession of the username, the password and the *passwordless* credential could use this to login without providing the MFA.

Recommendation

It is recommended to enforce MFA in case that *passwordless* authentication is disabled by the administrator.

8.14. Username Recovery Option Missing

Class	Authentication
CVSS:3.1	2.7 (Low)
Vector String	CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:N/I:N/A:L

The ZITADEL Console web application lacks an option to recover a forgotten username. A user who forgets his username, cannot recover it independently, this results in a limited Denial-of-Service.

For authentication on the web application at least a username and a password are necessary. Depending on the installation, the user can also use MFA and password less authentication. All authentication methods share, that a valid username is required.

Sometimes users forget their username for a specific application. Especially when the username is generated or specified by another entity. The loss of the username results in, at least a temporary Denial-of-Service since the user must contact the administrator or support to recover the username.

In this case a Denial-of-Service leads to limited availability of the service for a single user.

Recommendation

It is recommended to implement a functionality, that enables a user to recover a forgotten username by himself.

For example, the user could be asked to enter his e-mail address. The username could then be sent by e-mail to the users e-mail address. It’s important to make sure, that by this functionality no information is leaked to attackers. Therefore, the web application should respond in the same way, independent if an existing or non-existing e-mail address is entered. That also applies to the response timing of the server. The response time should be the same in both scenarios.

8.15. Third-Party Hosted Resources Embedded

Class	Information Leakage
CVSS:3.1	2.7 (Low)
Vector String	CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:L/I:N/A:N

The ZITADEL Console web application embeds resources of third parties. This leads to information leakage for the application users. A user of the application can be tracked by third parties.

Information leakages by third-party content arise, when a resource of a third-party domain is embedded in the web application. (e.g., by *href*) The user's browser will automatically fetch the resources to display the web-content correctly. The third-party service can track the user at least by his IP-address.

The browsing the following URLs results in server responses that include third party content:

- https://console.zitadel.app/auth/callback?code=-bUSVJQMyIRxhtLJEFruFsE4o9g7kkR_I221Z99VFieF-w&state=aGcyRC1nWlZkZ2hVR2hENkpTSXdaMFZ2SzVQdjd3MnIuc1pidDA4dG84MGdq;7bd1bf9c-ba3c-49f0-ad38-ae129a20aa1f
- https://console.zitadel.app/signedout?state=

As can be seen in the following server response, *Cascading Style Sheets* (CSS) are embedded from a third-party service. Furthermore, the *Content-Security-Policy* header lists all hosts that are used for third party content hosting.

```
HTTP/2 200 OK
Date: Thu, 13 May 2021 06:26:03 GMT
Content-Type: text/html; charset=utf-8
Cache-Control: public, max-age=43200, s-maxage=604800
Content-Security-Policy: font-src 'self' fonts.gstatic.com maxst.icons8.com;manifest-src
'self';connect-src 'self' *.zitadel.app fonts.googleapis.com fonts.gstatic.com
maxst.icons8.com;script-src 'self' 'unsafe-eval';style-src 'self' 'unsafe-inline'
fonts.googleapis.com maxst.icons8.com;img-src 'self';media-src 'none';frame-src
'none';default-src 'none';object-src 'none'
Expires: Thu, 13 May 2021 18:26:03 GMT
Feature-Policy: payment 'none'

[…]
<link rel="stylesheet" href="https://maxst.icons8.com/vue-static/landings/line-
awesome/line-awesome/1.3.0/css/line-awesome.min.css">
[…]
```

Not only users of the web application may be tracked by the third-party companies but also the quality of the web application itself can be affected. Especially privacy aware users, tend to use blockers for known tracking domains, which prevents the browser from fetching the necessary contents. This results in the web application being displayed in a bad style.

Recommendation

It is recommended to host all necessary content within the trustworthy domain of the web application.

8.16. WebAuthn Signature Verification

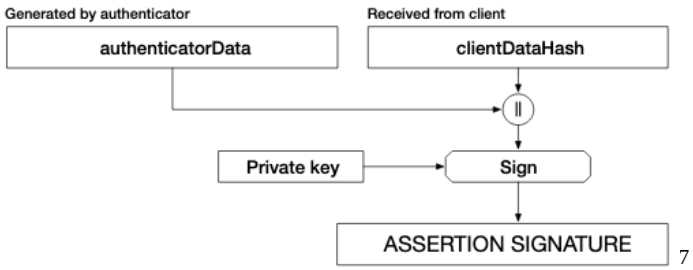
Class	Authentication
CVSS:3.1	2.5 (Low)
Vector String	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:L

The ZITADEL Console web application has an issue when verifying the WebAuthn signature. An attacker could alter the signed data.

ZITADEL Console offers the option to use password less authentication. The authentication flow is based on the WebAuthn client authentication. WebAuthn authentication does not require a user credential, instead it's based on a challenge-response procedure.

```
{"id":"dOL8SdIbTjqgsOZIYsdDtKxCchI","rawId":"dOL8SdIbTjqgsOZIYsdDtKxCchI","type":"public-
key","response":{"authenticatorData":"I68Y7cP-coJ3qFzGJTeSF6GwMIka4iU8-
dG9ISmfAdgFAAAAAA","clientDataJSON":"[REDACTED]","signature":"[REDACTED]","userHandle":"MTA
4MTc2OTI1NjI4ODg2NDI2"}fQ%3D%3D
```

As can be seen in the previous excerpt, the client provides an assertion signature to enable verification of the integrity of the value *clientDataJSON*. This signature should be verified server side with the client public key that was exchanged during the registration process.



[7]

The previously shown, JSON serialized data is send in a HTTP request Base64 encoded like the following.

```
POST /login/passwordless HTTP/1.1
Host: accounts.zitadel.app
Content-Type: application/x-www-form-urlencoded
Origin: https://accounts.zitadel.app
Accept-Encoding: gzip, deflate
Cookie: [REDACTED]
Connection: close
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 14_5_1 like Mac OS X) AppleWebKit/605.1.15
(KHTML, like Gecko) Version/14.1 Mobile/15E148 Safari/604.1
Referer: https://accounts.zitadel.app/loginname
Content-Length: 1179
Accept-Language: de-ch

gorilla.csrf.Token=[REDACTED]&authRequestID=108189069061713312&credentialAssertionData=[RED
ACTED]
```

During the analysis it was found that it is possible to e.g., modify the last character of the signature string. As can be seen in the following example, the character *A* was changed to *E*.

[7] W3C, WebAuthn, https://www.w3.org/TR/webauthn-2/#sctn-op-get-assertion, last visited 2021-05-15

Even though the signature string was modified the server did not return an error message but continued with the authentication.

```
{"id":"dOL8SdIbTjqgsOZIYsdDtKxCchI","rawId":"dOL8SdIbTjqgsOZIYsdDtKxCchI","type":"public-
key","response":{"authenticatorData":"[REDACTED]","clientDataJSON":"[REDACTED]","signature"
:"[REDACTED]A","userHandle":"MTA4MTc2OTI1NjI4ODg2NDI2"}fQ%3D%3D

{"id":"dOL8SdIbTjqgsOZIYsdDtKxCchI","rawId":"dOL8SdIbTjqgsOZIYsdDtKxCchI","type":"public-
key","response":{"authenticatorData":"[REDACTED]","clientDataJSON":"[REDACTED]","signature"
:"[REDACTED]E","userHandle":"MTA4MTc2OTI1NjI4ODg2NDI2"}fQ%3d%3d
```

It could be, that this finding is a false positive, the analyst cannot finally exclude that it is related to potential partial Base64 encoding of the signature string.

An attacker who can alter the assertion data, could alter the challenge-response or the origin. This could result in a temporary Denial-of-Service for a single user. The attacker must be able to modify the TLS protected data on transit, or directly on the victim's device.

Recommendation

It is recommended to verify this issue with the developer of the respective library, that takes care of the signature verification.

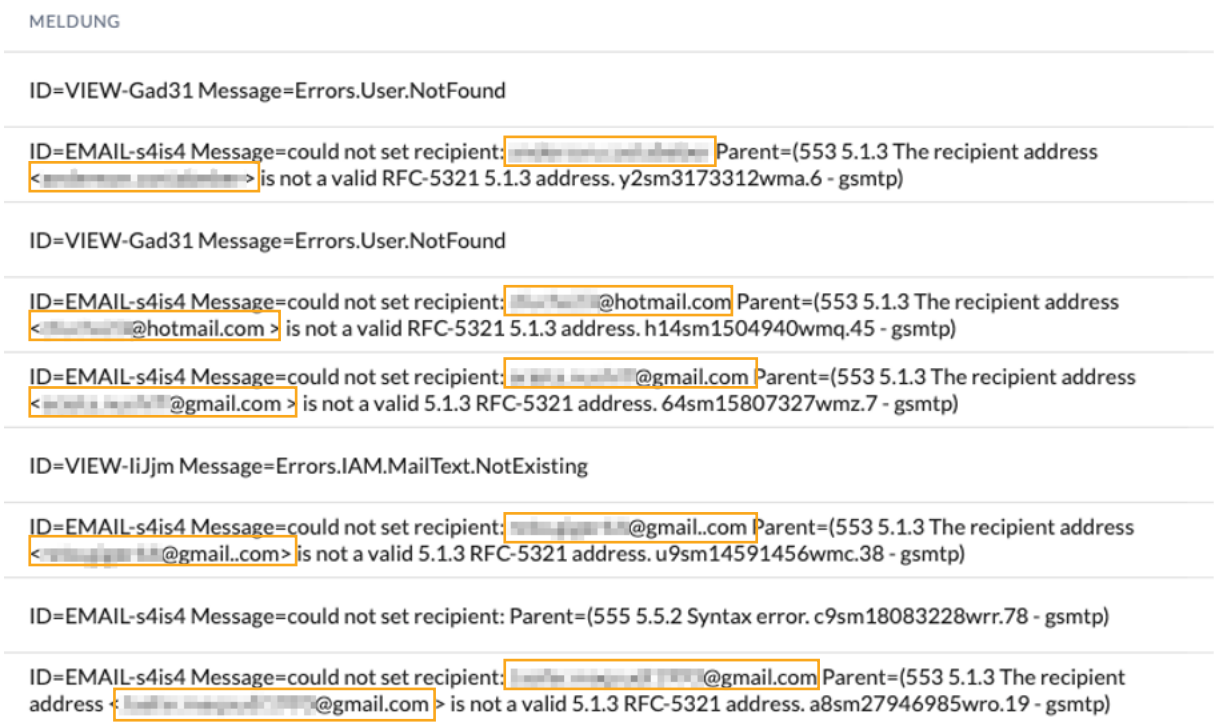
8.17. PII in Application Logs

Class	Data Protection
CVSS:3.1	2.4 (Low)
Vector String	CVSS:3.1/AV:N/AC:L/PR:H/UI:R/S:U/C:L/I:N/A:N

The ZITADEL Console web application logs contain personal identification information (PII). An attacker with access to the application logs can read this information.

According to security best practices, application logs should not contain PII. A recommended security control is that application logs are stored and backed up on an application independent storage. This storage can be centralized within a company. In case that this storage gets compromised the attacker can access the logged PII.

During the analysis it was found that the application logs can contain PII, such as e-mail addresses and phone numbers.



As can be seen on the previous screenshot, this information can also be accessed by users with the role IAM Administrator on the user interface.

An attacker with access to the log files can read this information.

Recommendation

It is recommended to prevent PII from being logged in the application logs. Instead, a generic placeholder should be logged.

8.18. Cacheable HTTPS Response with Sensitive Data

Class	Information Leakage
CVSS:3.1	2.2 (Low)
Vector String	CVSS:3.1/AV:L/AC:H/PR:L/UI:R/S:U/C:L/I:N/A:N

The ZITADEL Console web application allows caching of server responses with sensitive information. An attacker with access to the victim's browser can read the cached data.

According to security best practices and efforts to prevent leakage of Personal Identification Information (PII), server responses which contain person related information should not be cached. By default, web browsers cache content to speed up the browsing experience of the user. Nevertheless, the web server can instruct the browser not to do so, by sending the following http headers in the response:

```
Cache-control: no-store
Pragma: no-cache
```

The following requests and the associated server responses were found to be affected by the issue:

- https://api.zitadel.app/zitadel.auth.v1.AuthService/GetMyUser

```
HTTP/2 200 OK
Date: Thu, 13 May 2021 05:19:55 GMT
Content-Type: application/grpc-web+proto
X-Envoy-Upstream-Service-Time: 32
Access-Control-Allow-Origin: https://console.zitadel.app
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: *
Cf-Cache-Status: DYNAMIC
Cf-Request-Id: 0a05c3dc5f0000082c8f917000000001
Expect-Ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-
cgi/beacon/expect-ct"
Report-To:
{"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report?s=Yhr0%2B429YFbFcPKYuSYuMGOYtA
kH07rkyxINfJIjlF%2BakO9fVG54ow5Nv%2F7eGI51omZswkjXy0iCV%2BjahkjCG0bc%2BC1A0JKh3cgWFLGDWjM%3
D"}],"group":"cf-nel","max_age":604800}
Nel: {"report_to":"cf-nel","max_age":604800}
Server: cloudflare
Cf-Ray: 64e96f409f5d082c-CDG
Alt-Svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400

[…]
107952890689153079"──────────────────────────────────────────────────────────────────────
sven*0sven@[REDACTED]@[REDACTED]:M
(
──────────────────────────────────────────────────────────────────────
Sven
Fassbender"Sven Fassbender*──────────────────────────────────und
[REDACTED]
```

- https://api.zitadel.app/zitadel.auth.v1.AuthService/ListMyProjectOrgs

```
HTTP/2 200 OK
Date: Thu, 13 May 2021 05:19:55 GMT
Content-Type: application/grpc-web+proto
X-Envoy-Upstream-Service-Time: 120
Access-Control-Allow-Origin: https://console.zitadel.app
```

```
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: *
Cf-Cache-Status: DYNAMIC
Cf-Request-Id: 0a05c3dc570000082c48360000000001
Expect-Ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-
cgi/beacon/expect-ct"
Report-To:
{"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report?s=pLDvqtDzGkEoR%2F%2Bs45Js1PEQ
c7MsS3OP9w9eM6xscnBjiy1FNBd1n060NtSqi1QSD5%2BBtil4ntWv4RLTUW%2Fcg7KKO6c%2BlEpb%2BCNGo7e3BKA
%3D"}],"group":"cf-nel","max_age":604800}
Nel: {"report_to":"cf-nel","max_age":604800}
Server: cloudflare
Cf-Ray: 64e96f408f4e082c-CDG
Alt-Svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400

[…]
74[REDACTED]
86[REDACTED]
10[REDACTED]
97[REDACTED]
82[REDACTED]
[…]
```

- https://api.zitadel.app/zitadel.auth.v1.AuthService/ListMyUserSessions

```
HTTP/2 200 OK
Date: Thu, 13 May 2021 05:20:02 GMT
Content-Type: application/grpc-web+proto
X-Envoy-Upstream-Service-Time: 27
Access-Control-Allow-Origin: https://console.zitadel.app
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: *
Cf-Cache-Status: DYNAMIC
Cf-Request-Id: 0a05c3f8210000082c9b282000000001
Expect-Ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-
cgi/beacon/expect-ct"
Report-To:
{"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report?s=yZnvhaNpt9KvSflKirN9S5JyfEVQ
3A0E7hw2W7VN2twNvJXg9tGgAG0yFpMXXRG9m1uC8MeyG%2Fjp9MEkzVhGmVqWYqFU2YU5ezQmbpgdi%2Bw%3D"}],"
group":"cf-nel","max_age":604800}
Nel: {"report_to":"cf-nel","max_age":604800}
Server: cloudflare
Cf-Ray: 64e96f6d0dd0082c-CDG
Alt-Svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400

[…]
108023199387675094"107952890705930295*──────────────────────────────────────────
sven:0sven@[REDACTED]BSven FassbenderJ3ðä>
íìò□ð□û,
[…]
```

As can be seen in the previous examples, Cloudflare has set the *Cf-Cache-Status* HTTP header to *DYNAMIC*. (Marked blue) That means by default Cloudflare does not store the resource but that also shows that no explicit setting is in place to instruct Cloudflare not to store the resource. If the application sends a no-cache header, Cloudflare will respect the setting and will not store the asset. This will be recognizable by the *Cf-Cache-Status* HTTP header *BYPASS*.

An attacker must have access to a victim's web browser to access the cached data. The extracted data may be used by the attacker to conduct further attack steps. Depending on the

Cloudflare handling the information could also end up in the Cloudflare cache. Every entity with access to this cache could extract the information.

Recommendation

It is recommended to implement the following HTTP headers, in all server responses that may contain sensitive information:

```
Cache-control: no-store
Pragma: no-cache
```

After implementing the respective HTTP headers, it is recommended to also verify the Cloudflare HTTP header *Cf-Cache-Status*, which should have the value *BYPASS*.

8.19. Implicit Grant Type Supported

Class	Authentication
CVSS:3.1	2.2 (Low)
Vector String	CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:U/C:L/I:N/A:N

The ZITADEL Console web application supports the implicit grant type. The implicit grant type can lead to the leakage of access token and is therefore not recommended.

During the analysis it was found that the implicit grant type is theoretically allowed. This information was identified in the OpenID configuration file:

- https://issuer.zitadel.app/.well-known/openid-configuration

```
"grant_types_supported":["authorization_code","implicit","urn:ietf:params:oauth:grant-type:jwt-bearer"],
```

The implicit grant type was not used during the normal usage of the ZITADEL Console web application. Nevertheless, it is recommended to follow the recommended security best practices.

According to security best practices[8] the implicit grant type should not be used by clients. That leads to the conclusion that this grant type should also not be accepted by the server. The implicit grant flow returns the access token in the URI parameter of the 302-redirect location, like the following example:

```
HTTP/1.1 302 Found
[…]
Location: https://xxx/login#access_token=[REDACTED]
[…]
```

In this case the client web browser would follow the HTTP redirect as following:

```
GET /login#access_token=[REDACTED]
Host: xxx
[…]
```

As can be seen in the previous example, the access token would get exposed in the HTTP GET request as parameter. GET parameters can be visible in application or proxy log files, therefore it is not recommended to transfer sensitive information in GET parameters. If the access token would be sent as GET parameter, it would result in an information leakage.

An attacker with access to the respective log, could access the information and mount further attack steps. In case of the access token, the attacker could impersonate the legitimate user on the ZITADEL API. Depending on the role of the compromised account the attacker can take actions available in the user's context.

Recommendation

It is recommended evaluate the possibility, not to support the implicit grant type.

[8] Datatracker.ietf.org, OAuth 2.0 Security Best Current Practice, https://datatracker.ietf.org/doc/html/draft-ietf-oauth-security-topics#section-2.1.2, last visited 2021-05-16

8.20. Consider Security.txt

Class	Informal
CVSS:3.1	0.0 (None)
Vector String	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N

The ZITADEL Console web application does not offer a security.txt file. An independent security researcher might have a hard time to identify the right contact person within the organization to report security issues.

Security researchers discover vulnerabilities from time to time, without having a contract with the operator of a web-service. A common hurdle in a responsible disclosure process is identifying the responsible party, which takes care of the disclosure process. Around the year 2017, the proposed standard security.txt was invented. To encourage researchers in reporting discovered vulnerabilities and streamline this process, it should be present in the ZITADEL Console web-application.

The security.txt file should be hosted on the following path:

- https://console.zitadel.app/.well-known/security.txt

Detailed information on the file itself can be found here[9]. The file itself can be generated using this web-service[10].

Recommendation

It is recommended to host a security.txt file with the necessary information for security researchers. Linking the GitHub hosted Security Policy[11] can be considered.

[9] Github.com, security-txt, https://github.com/securitytxt/security-txt, last visited 2021-05-14
[10] Securitytxt.org, https://securitytxt.org/, last visited 2021-05-14
[11] Github.com, ZITADEL Security Policy, https://github.com/caos/zitadel/blob/main/SECURITY.md, last visited 2021-05-14

8.21. Missing Option to Delete Account

Class	DSGVO
CVSS:3.1	0.0 (None)
Vector String	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N

The ZITADEL Console web application lacks the option, for a user to delete his account. Whilst the deletion of an account can be requested at the support, it should be considered to empower the user to do so on his own.

According to the General Data Protection Regulation (GDPR) Art. 17[12] the user has the right to erasure ('right to be forgotten').

> *b) the data subject withdraws consent on which the processing is based according to point (a) of Article 6(1), or point (a) of Article 9(2), and where there is no other legal ground for the processing;*

To lower the potential administrative overhead for the operators of the web-service, it can be considered to give the end-user the option to delete an account and all related data. Furthermore, the withdrawal of consent could be targeted with this functionality.

The analyst cannot determine the potential damage that can result in case of a GDPR violation. Therefore, the impact is set to *None*.

Recommendation

The analyst recommends implementing the delete functionality in the "edit Account" section, to empower an end-user and thus strengthen then informal self-determination of the individual.

It is recommended implementing a re-authentication step before account related critical operations like this can take place. (See finding 8.11)

[12] GDPR Art. 17, Right to erasure, https://gdpr-info.eu/art-17-gdpr/, last visited 2021-05-14

8.22. Missing Consent GDPR Third-Person Registration

Class	GDPR
CVSS:3.1	0.0 (None)
Vector String	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N

The ZITADEL Console web application does not request the user consent, in case that another person created the account in the web interface. A user could use the web application without GDPR consent.

According to General Data Protection Regulation (GDPR) Art.7[13] the operator of the web application shall be able to demonstrate that the user has given his consent for processing of his personal data.

In case that an administrator creates a user account within the application, the user will receive an e-mail with instructions on how to set the user password. This process does not contain a step where the user must give his consent for processing of personal data.

Another issue within the actual design is, that an administrator can enter personal information about the user, such as e-mail address, name, forename, and gender. This may hurt the users right for informal self-determination.

The analyst cannot determine the potential damage that can result in case of a GDPR violation. Therefore, the impact is set to *None*.

Recommendation

It is recommended to consult an GDPR-Expert if the actual practice is in line with the GDPR requirements.

In doubt it is recommended to remove the option for an administrator to enter person related information. Instead, users could be animated to use the registration functionality of the web application. Open registrations can pose a risk to a company's IAM, therefore additional countermeasures must be implemented to limit the group of people who can register. To make sure that only persons who are part of the company can register, a domain whitelist can be implemented. This whitelist can be maintained by the administrator.

Another, less scalable approach is to only generate initialization codes, which can be manually distributed by the administrator of the platform.

[13] GDPR Art. 7, Conditions for consent, https://gdpr-info.eu/art-7-gdpr/, last visited 2021-05-15