# Penetration Test Report:
# *DENIC ID*

HACKMANIT

**Version: 1.2**

Dr.-Ing. Juraj Somorovsky
Phone: (+49)(0)234 / 45930961
E-Mail: juraj.somorovsky@hackmanit.de

July 17, 2019

# Project Information

| | |
|---|---|
| Customer: | DENIC eG<br>Kaiserstraße 75 - 77<br>60329 Frankfurt am Main, Deutschland |
| Contact: | Marcos Sanz |
| Commissioned to: | Hackmanit GmbH<br>Universitätsstraße 150<br>44801 Bochum, Germany |
| Project executive: | Dr.-Ing. Juraj Somorovsky<br>Phone: (+49)(0)234 / 45930961<br>Fax: (+49)(0)234 / 45930960<br>E-Mail: juraj.somorovsky@hackmanit.de |
| Project members: | Karsten Meyer zu Selhausen (Hackmanit GmbH)<br>Dr.-Ing. Vladislav Mladenov (Hackmanit GmbH)<br>Mario Korth (Hackmanit GmbH) |
| Project period: | 11.03.2019 – 19.03.2019 |
| Version of the report: | 1.2 |

This report was technically verified by Dr.-Ing. Vladislav Mladenov.
This report was linguistically verified by David Herring.

Hackmanit GmbH
Represented by: Prof. Dr. rer. nat. Jörg Schwenk, Dr.-Ing. Juraj Somorovsky,
Dr.-Ing. Christian Mainka, Dr.-Ing. Marcus Niemietz
Register court: Bochum, Germany
Register number: 14896

# Contents

# 1 Summary

DENIC ID is the first widely-deployed implementation of the ID4me protocol [1]. ID4me is a novel protocol for federated identity management, of which the two main goals are to provide *(1) Authorization of a user for access to any third party accepting ID4me identifiers* and *(2) Controlled communication of the user's personal information to the third parties accessed by the user* [1]. ID4me is based on well-established standards such as OpenID Connect [12] and Domain Name System (DNS) [4].

Hackmanit GmbH was commissioned to perform a penetration test of DENIC ID. After an initial kick-off meeting at the office of DENIC eG in Frankfurt am Main, the penetration test was performed remotely with a total expense of 11 man-days.

**Weaknesses.** During the penetration test, one weakness classified as *High* and five weaknesses classified as *Medium* were identified. The highest ranked weakness targeted the identity agent. Instead of enforcing that the access token (which is a JWT) contains a signature from the identity authority, the identity agent accepted access tokens which do not contain any signature at all. This allowed an attacker to craft his own access tokens and use them to access the stored personal information of arbitrary users. Another mentionable weakness, which was identified at the identity authority, allowed an attacker to log a victim into his account. Depending on the service provided by the relying party this might result in the victim revealing personal information or files to the attacker.

Both weaknesses described above were already fixed by updates of the identity authority and identity agent during the penetration test. According to DENIC, all remaining weaknesses classified as *Medium* were fixed by the end of June. In our retests, we tested and successfully verified the correctness of three countermeasures (H01, M03, and M05).

Additionally, we strongly recommend fixing the weaknesses classified as *Low* and *Information* as well as to implement the recommendations given in Section 8 to improve the overall security of the tested systems.

**Structure.** The report is structured as follows: In Section 2, the timeline of the penetration test is listed. Section 3 introduces our methodology, and Section 4 explains the general conditions and scope of the penetration test. In section 5, the scenario of the penetration test is described in detail. Section 6 provides an overview of the identified weaknesses and further recommendations. In Section 7, all identified weaknesses are discussed in detail and specific countermeasures are described. Section 8 summarizes our recommendations resulting from observations of the application. Finally, Section 9 lists additional tests that have not revealed any weaknesses.

# 2 Project Timeline

The penetration test was performed between the 11.03.2019 and 19.03.2019 remotely after an initial kick-off meeting in Frankfurt am Main on 11.03.2019. Four penetration testers with different technical backgrounds were involved with a total expense of 11 man-days.

Additionally, weakness M05 was reevaluted during a retest on 07.06.2019.

# 3 Methodology

Among others, the following tools were used for the penetration test:

| Tool | Link |
|---|---|
| Mozilla Firefox | `https://www.mozilla.org/de/firefox/` |
| Google Chrome | `https://www.google.com/intl/de_ALL/chrome/` |
| Burp Suite Professional | `https://portswigger.net/burp` |
| testssl.sh | `https://testssl.sh/` |
| TLS-Attacker | `https://github.com/RUB-NDS/TLS-Attacker` |
| TLS-Scanner | `https://github.com/RUB-NDS/TLS-Scanner` |
| Self-developed tools | - |

**Risk Rating.** Each weakness has its own CVSS 3 base score rating (*Common Vulnerability Scoring System Version 3 Calculator*).[1,2] Based on the CVSS 3 base score, the following weaknesses assessment is performed:

| | |
|---|---|
| 0.0 – 3.9: | Low |
| 4.0 – 6.9: | Medium |
| 7.0 – 8.9: | High |
| 9.0 – 10.0: | Critical |

[1] `https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator`
[2] `http://www.first.org/cvss/cvss-guide`

# 4 General Conditions and Scope

The scope of the black-box penetration test included the ID4me implementation of DENIC, DENIC ID, as well as the userinfo endpoint of the identity agent. Therefore, the test covered the following endpoints:

- Authorization endpoint: `id.test.denic.de/login`
- Token endpoint: `id.test.denic.de/token`
- Client registration endpoint: `id.test.denic.de/clients`
- Introspection endpoint: `id.test.denic.de/token/introspect`
- Revocation endpoint: `id.test.denic.de/token/revoke`
- Userinfo endpoints: `id.test.denic.de/userinfo`, `api-beta.id4me.ionos.com/userinfo`
- End session endpoint: `id.test.denic.de/logout`

Additionally, two web front ends were in the scope of the penetration test: the dashboard at the identity authority, and the front end for the user login and claims confirmation at the authorization endpoint. Therefore, in addition to the endpoints above, the following uniform resource l6ocators (URLs) were in the scope of the penetration test:

- `https://id.test.denic.de/`
- `https://id.test.denic.de/authenticate`
- `https://id.test.denic.de/consent`
- `https://id.test.denic.de/dashboard/*`

The `dashboard` URL provides several additional functions, for example, login, logout, account activation, or management of identities.

This resulted in the following cookies being within the scope of the penetration test:

- sub_sid_current
- sub_sid_XXX
- SESSIONID

# 5 Scenario Description

DENIC ID is an implementation of ID4me [1] – an "Open, Global, Federated Standard For The Digital Identity Management".[3] It is based on established standards such as OpenID Connect and DNS. In contrast to other Single Sign-On (SSO) schemes, ID4me divides the duties of the identity provider (IdP) into two separated entities: an identity agent and an identity authority. The identity agent provides registration services and manages user data. The identity authority is responsible for user authentication and authorization. This role separation results in the following four entities being involved in a login process based on ID4me:

**User** A user utilizing ID4me to log in at an online service. His user account is associated with an ID4me identifier.

**Relying party** An online service which supports logins using an ID4me identifier.

**Identity agent** The entity providing ID4me services to the user. This includes the registration and management of ID4me identifiers as well as storage and distribution of the user's personal data to relying partys in so-called "claims".

**Identity authority** The entity responsible for user authentication and for ensuring that the user authorized the specific relying party to access his personal information.

ID4me identifiers are used to identify the user when he/she wants to log in at a relying party. An ID4me identifier can be any hostname identified by a valid DNS entry which contains a TXT record. This record specifies the responsible identity authority and identity agent.

The process of registering a new ID4me identifier was not in the scope of this penetration test. Therefore, it is not described here. Information on the process can be found in the ID4me documentation [1].

The process of logging in at a relying party using an ID4me identifier is depicted in Figure 1 and described as follows:

1. The user starts the login process with the relying party by providing his ID4me identifier.
2. The relying party queries the DNS for the user's identifier to acquire the responsible identity authority and identity agent.
3. If the relying party is not already registered at the identity authority, it performs Dynamic Client Registration [11] according to the OpenID Connect standard.
4. The relying party redirects the user to the identity authority. The user authenticates at the identity authority and authorizes or rejects access to the claims requested by the relying party on the consent page displayed by the identity authority.

[3] `https://id4me.org/about/`

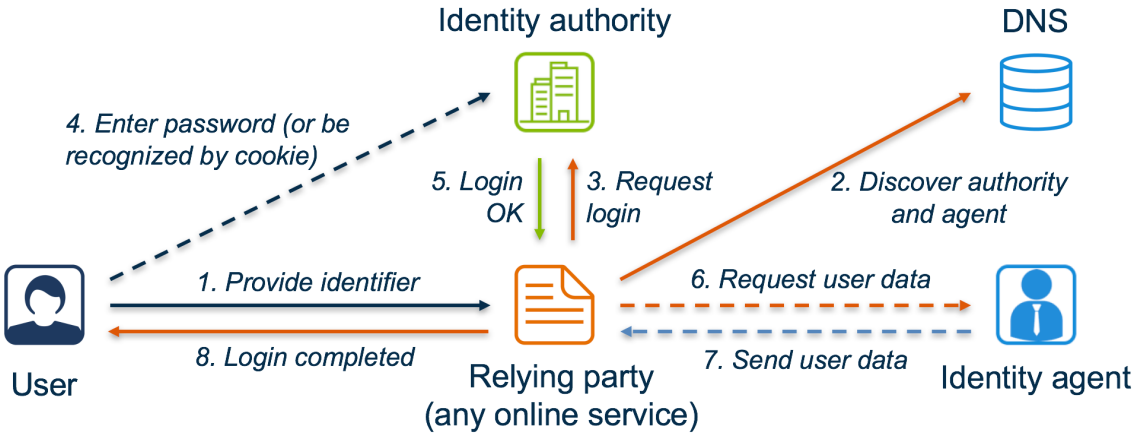


Figure 1: Process of logging in at a relying party using an ID4me identifier. The figure is taken from the official ID4me documentation.[4]

5. The identity authority redirects the user back to the relying party and delivers the Authorization Code to the relying party in this redirection. The relying party redeems the Authorization Code at the token endpoint of the identity authority and receives an access token. Listing 1 provides an example of an access token.

6. If the relying party wants to access claims in addition to the information present in the access token, it queries the userinfo endpoint of the identity authority using the access token. The identity authority makes use of the OpenID Connect distributed claims mechanism[5] and refers the relying party to the identity agent. The relying party queries the userinfo endpoint of the identity agent using the access token.

7. If the access token is valid, the identity agent provides all claims which the relying party is authorized to access according to the `clm` field of the access token. If there is no information stored for a requested claim, the claim is omitted from the identity agent's response.

[4] https://gitlab.com/ID4me/documentation/blob/1a8e464b42ef6f57e75ec3c7f1a23e878dbbe42d/id4me%20Technical%20Overview%20v1.3.pdf
[5] https://openid.net/specs/openid-connect-core-1_0.html#AggregatedDistributedClaims

```
{
    "kid": "FOvy",
    "alg": "RS256"
}.
{
    "sub": "Dvkf7Oy9SEqVgt+hPwY7K3mjInrC91L318KVwtsbfakhhNwqc7ThxOVRiPpj2RS7",
    "id4me.identifier": "pentest.sanz.club",
    "identifier": "pentest.sanz.club",
    "id4me": "pentest.sanz.club",
    "clm": [
        "email",
        "email_verified",
        "preferred_username",
        "name",
        "nickname"
    ],
    "scope": [
        "openid"
    ],
    "iss": "https:\/\/id.test.denic.de",
    "exp": 1552397804,
    "iat": 1552397204,
    "jti": "_E16kF4MmwU",
    "client_id": "sfa4ztr427bsm"
}.
*Signature*
```

Listing 1: Access token issued and signed by the identity authority (decoded).

While generally implementing ID4me, DENIC ID differs from the standard in some crucial aspects. ID4me does not cover the trust relationship between the identity agent and the identity authority; in ID4me every user is allowed to set up and operate his own identity agent. DENIC ID is more specific in this regard and only supports pre-registered identity agents which have a valid contract with the DENIC. Additionally, DENIC ID suggests that a relying party does not trust every identity authority but only a list of predefined authorities. This limits the degrees of freedom provided by ID4me but increases the security by limiting the parties which can participate in the protocol and establishes more trust between these parties.

For the penetration test, DENIC provided us with access to their DENIC ID test environment. This included access to their identity authority (`id.test.denic.de`) which consists of two components: the dashboard (`/dashboard`) and the OpenID Connect endpoints, and an identity agent (`beta.id4me.ionos.de`) and a relying party (`hermes.open-xchange.com`). The identity authority provides the following endpoints:

- Authorization endpoint: `/login`
- Token endpoint: `/token`
- Client registration endpoint: `/clients`
- Introspection endpoint: `/token/introspect`

- Revocation endpoint: `/token/revoke`
- Userinfo endpoint: `/userinfo`
- End session endpoint: `/logout`

The complete configuration of the tested identity authority, including the locations of all relevant endpoints, can be found at `https://id.test.denic.de/.well-known/openid-configuration`.

We were also provided with the following two identifiers already registered in the test environment:

| Identifier | Password | `sub` Claim |
|---|---|---|
| pentest.sanz.club | asdfasdf | Dvkf7Oy9SEqVgt+hPwY7K3mjInrC91L318KVwtsbfakhhNwqc7Thx0VRiPpj2RS7 |
| pentest2.sanz.club | asdfasdf | +V95/459p4c0kvXAOlk2qCALizOEA+DK6e9sjJYVh4Q+2Fp9OgoZSrxek+Zg8RZY |

# 6 Overview of Weaknesses and Recommendations

| Risk Level | Finding | Reference |
|---|---|---|
| H01 | **Signature Exclusion at the Identity Agent:** The identity agent does not enforce the use of a signed JWT. It accepts JWTs with missing signatures. | Section 7.1, page 15 |
| M01 | **Insufficient Clickjacking Protections:** The identity authority does not employ sufficient protections against Clickjacking attacks. | Section 7.2, page 16 |
| M02 | **User Enumeration:** The login process of the dashboard at the identity authority does not prevent user enumeration. | Section 7.3, page 17 |
| M03 | **Missing Binding Between HTTP Parameter `sessionID` and Session Cookies:** The identity authority makes use of different values to reference the session of the user in the current authorization flow but does not bind these values to each other. | Section 7.4, page 18 |
| M04 | **Insufficient Cross-site Request Forgery Protection:** The identity authority does not provide sufficient protection against CSRF attacks. | Section 7.5, page 20 |
| M05 | **Faulty Session Management:** User sessions are not invalidated upon logout on the identity authority. | Section 7.6, page 21 |
| M06 | **Identity Authority Allows HTTP Redirect URIs:** The identity authority allows "web" clients to register redirect URIs which use the HTTP scheme instead of enforcing the use of HTTPS. | Section 7.7, page 22 |
| L01 | **Missing Brute Force Protections:** The identity authority does not enforce any protection mechanisms against brute force attacks, allowing an attacker to try to guess user's passwords. | Section 7.8, page 23 |

| | | |
|---|---|---|
| L02 | **Denial-of-Service Attack at the Identity Agent:** The identity agent attempts to download the complete JWKS file, independently of its size and structure. | Section 7.9, page 23 |
| I01 | **Information Disclosure on the Consent Page:** The identity authority displays a stacktrace on the consent page when the `prompt` parameter of the authorization request is set to "select_account". | Section 7.10, page 24 |
| R01 | **Issue a New Access Token to Access the Identity Agent at Userinfo Endpoint:** The identity authority should issue a new access token in order to access the identity agent at its userinfo endpoint. | Section 8.1, page 26 |
| R02 | **Implement Access Tokens as One-Time-Use Tokens at Identity Agent:** The identity agent should accept an access token only once in order to access its userinfo endpoint. | Section 8.2, page 26 |
| R03 | **Revoke Authorization Code When It Is Redeemed Using False Client Credentials:** The Authorization Code should be revoked when a token request is received using false client credentials. | Section 8.3, page 27 |
| R04 | **Revoke Tokens If the Related Authorization Code Is Redeemed a Second Time:** All relevant tokens should be revoked if the token endpoint receives an already redeemed Authorization Code. | Section 8.4, page 27 |
| R05 | **Revoke Tokens When the User Changes his Password:** All tokens issued for one user should be revoked when he/she changes his/her password at the identity authority. | Section 8.5, page 28 |
| R06 | **Implement Refresh Tokens as One-Time-Use Tokens:** Refresh Tokens should be invalidated when they are redeemed at the token endpoint of the identity authority. | Section 8.6, page 28 |

| | | |
|---|---|---|
| R07 | **Revoke Tokens When a Refresh Token Is Redeemed a Second Time:** All relevant tokens should be revoked if the token endpoint receives an already redeemed refresh token. | Section 8.7, page 28 |
| R08 | **Prevent Concurrent Logins:** The authorization endpoint should prevent concurrent logins. | Section 8.8, page 29 |
| R09 | **Secure Cookies with `HttpOnly` Flag:** Session cookies should be protected by setting the `HttpOnly` flag. | Section 8.9, page 29 |
| R10 | **Secure Cookies with `Secure` Flag:** Cookies should be protected by setting the `secure` flag. | Section 8.10, page 29 |
| R11 | **Enforce HTTP Stricts Transport Security:** Servers should enforce the usage of HTTPS with the `Strict-Transport-Security` HTTP header. | Section 8.11, page 29 |
| R12 | **Restrict Cross-Origin Resource Sharing to Whitelist:** CORS should be restricted to a whitelist of allowed origins at the userinfo endpoint. | Section 8.12, page 30 |
| R13 | **Enable Content Security Policy:** The Content Security Policy should be enabled to increase the application security. | Section 8.13, page 30 |
| R14 | **Set XSS Protection HTTP Header:** The `X-XSS-Protection` HTTP header should be set with the appropriate mode to increase the protection against XSS attacks. | Section 8.14, page 30 |
| R15 | **Disable Referer HTTP Header:** The Referer header should be disabled to prevent potential information leakage. | Section 8.15, page 31 |
| R16 | **Disable Content Type Sniffing:** The `X-Content-Type-Options` HTTP header should be set to prevent content type sniffing. | Section 8.16, page 31 |
| R17 | **Set Cache Control HTTP Headers:** Appropriate cache control HTTP headers should be set to prevent client-side caching. | Section 8.17, page 31 |

| | | |
|---|---|---|
| R18 | **Use Discovery Mechanism at Identity Agent:** The identity agent should use the discovery mechanism to determine the URL of the JWKS file of the identity authority. | Section 8.18, page 32 |

Definitions:

| | |
|---|---|
| **Critical Risk** | Weaknesses classified as *Critical* can be exploited with very little effort by an attacker. They have very large negative effects on the tested system, its users and data, or the system environment. |
| **High Risk** | Weaknesses classified as *High* can be exploited with little effort by an attacker. They have a major negative impact on the tested system, its users and data, or the system environment. |
| **Medium Risk** | Weaknesses classified as *Medium* can be exploited with medium effort by an attacker. They have a medium negative impact on the tested system, its users and data, or the system environment. |
| **Low Risk** | Weaknesses classified as *Low* can be exploited with great effort by an attacker. They have little negative impact on the tested system, its users and data, or the system environment. |
| **Information** | Observations classified as *Information* are usually no weaknesses. Examples of these observations are unusual configurations and possibly unwanted behavior of the tested system. |
| **Recommendation** | *Recommendation* identifies measures that may increase the security of the tested system. Implementation is recommended, but not necessarily required. |

# 7 Weaknesses

In the following sections we list the identified weaknesses. Every weakness has an identification name which can be used as a reference in the event of questions or during the patching phase.

## 7.1 H01 Signature Exclusion at the Identity Agent

| **Exploitability Metrics** | | **Impact Metrics** | |
|---|---|---|---|
| Attack Vector (AV) | **Network** | Confidentiality Impact (C) | **High** |
| Attack Complexity (AC) | **Low** | Integrity Impact (I) | **None** |
| Privileges Required (PR) | **None** | Availability Impact (A) | **None** |
| User Interaction (UI) | **None** | | |
| Scope (S) | **Unchanged** | | |
| Subscore: **3.9** | | Subscore: **3.6** | |
| **Overall CVSS Score: 7.5** | | | |

Table 2: CVSS calculation of weakness H01.

**General Description.** The userinfo endpoint of the identity agent can be used to access the information about a user stored at the identity agent. In order to access this information, the request must contain an access token in the `Authorization` HTTP header. This access token is a JSON Web Token (JWT) issued and validly signed by the identity authority. The identity agent verifies the signature of the JWT and rejects requests containing invalid signatures.

**Weakness.** The identity agent is vulnerable to Signature Exclusion attacks as it processes requests with missing JWT signatures. This allows an attacker to access arbitrary user data at the userinfo endpoint of the identity agent using manipulated JWTs. The identity agent accepts these manipulated JWTs and delivers information about the user specified by the combination of the `iss` and `sub` values if available.

The attacker is able to craft access tokens on their own and use them to access the stored information of arbitrary users, as long as the attacker knows the `sub` values for the corresponding users at the identity authority. The attacker can either use a valid old access token, remove the signature, and adjust the `sub` value and update the timestamps, or build their own access token from scratch. An example of an access token (Base64Url-encoded) which does not contain a signature but is still accepted by the identity agent is given in Listing 2.

Note that the attack works independently of the used signature algorithm. The JWT header can either contain the `none` algorithm or the `RS256` algorithm. Therefore, this issue

```
1 eyJraWQiOiJGT3Z5IiwiYWxnIjoiTm9uZSJ9.eyJzdWIiOiIrVjk1XC8ONTlwNGMwa3ZYQU9sazJxQ0
    FMaXpPRUErREs2ZTlzakpZVmg0USsyRnA5T2dvWlNyeGVrK1pnOFJaWSIsImlkNG1lLmlkZW50
    aWZpZXIiOiJwZW50ZXN0Mi5zYW56LmNsdWIiLCJpZGVudGlmaWVyIjoicGVudGVzdDIuc2Fuei5
    jbHViIiwiaWQObWUiOiJwZW50ZXN0Mi5zYW56LmNsdWIiLCJjbG0iOlsibmFtZSIsImdpdmVuX25
    hbWUiLCJmYW1pbHlfbmFtZSIsImVtYWlsIlOsInNjb3BlIjpbIm9wZW5pZCJdLCJpc3MiOiJodHRwczpcL
    1wvaWQudGVzdC5kZW5pYy5kZSIsImV4cCI6MTU1MjQ4NDk3NywiaWF0IjoxNTUyNDg0Mzc3
    LCJqdGkiOiJFQ3hRY21LZEJJNCIsImNsaWVudF9pZCI6InZ5Z3ZxbXdvMmNvbzIifQ.
```

Listing 2: JWT without a signature used as an access token to access user information at the identity agent (encoded). The signature header algorithm is set to `none`.

is not a mere configuration flaw, and blacklisting the `none` algorithm would not prevent the attack. The implementation needs to be updated so that the presence of the signature is always verified.

**Countermeasures.** We strongly recommend enforcing that all access tokens be signed by the issuing identity authority using one of the algorithms specified by RFC7518 (JSON Web Algorithms (JWA)) [2, Section 3.1], except the `none` algorithm. Any request containing an access token without a signature, specifying the usage of the `none` algorithm, or with an invalid signature must be rejected. In all three cases the response should be a general error message, e.g.: `Missing or invalid signature!`

**Retest.** An update of the identity agent was applied during the penetration test. We can confirm that this weakness has been successfully fixed. The identity agent rejects any request containing an access token without a signature, or a header specifying the `none` algorithm.

## 7.2 M01 Insufficient Clickjacking Protections

| **Exploitability Metrics** | | **Impact Metrics** | |
|---|---|---|---|
| Attack Vector (AV) | **Network** | Confidentiality Impact (C) | **High** |
| Attack Complexity (AC) | **Low** | Integrity Impact (I) | **None** |
| Privileges Required (PR) | **None** | Availability Impact (A) | **None** |
| User Interaction (UI) | **Required** | | |
| Scope (S) | **Unchanged** | | |
| Subscore: **2.8** | | Subscore: **3.6** | |
| **Overall CVSS Score: 6.5** | | | |

Table 3: CVSS calculation of weakness M01.

**General Description.** Clickjacking allows an attacker to trick the user into performing clicks, and therefore, actions the user did not intend to perform [7]. To prevent this kind

of attack, several security mechanisms exist, such as framebuster, the `X-Frame-Options` HTTP header, and the Content Security Policy (CSP) option `frame-ancestors`.

**Weakness.** The identity authority does not employ any of the security mechanisms listed above. Therefore, an attacker could use Clickjacking attacks to trick the victim into performing arbitrary actions. For example, the victim can be tricked into consenting to arbitrary claims.

**Countermeasures.** We recommend to employ all the security mechanisms listed above. The reason for this is that older browsers may not support the CSP or the `X-Frame-Options` HTTP header. Therefore, framebuster techniques should be used in addition to these features in order to mitigate Clickjacking attacks. More details are provided in the OWASP Clickjacking Defense Cheat Sheet [8].

## 7.3 M02 User Enumeration

| **Exploitability Metrics** | | **Impact Metrics** | |
|---|---|---|---|
| Attack Vector (AV) | **Network** | Confidentiality Impact (C) | **Low** |
| Attack Complexity (AC) | **Low** | Integrity Impact (I) | **None** |
| Privileges Required (PR) | **None** | Availability Impact (A) | **None** |
| User Interaction (UI) | **None** | | |
| Scope (S) | **Unchanged** | | |
| Subscore: **3.9** | | Subscore: **1.4** | |
| **Overall CVSS Score: 5.3** | | | |

Table 4: CVSS calculation of weakness M02.

**General Description.** Different behavior in the case of existing and non-existing usernames (i.e., the user identifier) allows an attacker to enumerate which accounts exist. This information can be used for further attacks, such as guessing passwords online.

**Weakness.** The login process of the identity authority dashboard first requires the user to enter his identifier. If the identifier exists, the dashboard asks the user for his password. Otherwise, if the identifier does not exist, an error message is displayed. This behavior can be used to determine if an identifier exists at the identity authority.[6]

**Countermeasures.** We recommend to adjust the behavior of the identity authority such that it behaves in the same way in case of an existing and a non-existing identifier.

[6]Due to the design of ID4me a discovery mechanism is necessary. This mechanism is vulnerable to user enumeration in general, as it allows an attacker to determine whether an identifier exists by querying the responsible DNS server. As DNS servers are not in the scope of this penetration test, we cannot consider the existence of a user enumeration weakness using DNS queries when rating this weakness.

Additionally, we recommend displaying CAPTCHAs after a certain number of identifiers (e.g., five) was entered in the login form. This prevents automated user enumeration.

## 7.4 M03 Missing Binding Between HTTP Parameter `sessionID` and Session Cookies

| **Exploitability Metrics** | | **Impact Metrics** | |
|---|---|---|---|
| Attack Vector (AV) | **Network** | Confidentiality Impact (C) | **Low** |
| Attack Complexity (AC) | **Low** | Integrity Impact (I) | **Low** |
| Privileges Required (PR) | **Low** | Availability Impact (A) | **None** |
| User Interaction (UI) | **Required** | | |
| Scope (S) | **Unchanged** | | |
| Subscore: **2.1** | | Subscore: **2.5** | |
| **Overall CVSS Score: 4.6** | | | |

Table 5: CVSS calculation of weakness M03.

**General Description.** During the authorization flow, the identity authority presents a consent page to the user to let him decide whether he grants the requesting client access to his information or not. This page contains a hidden form field called `sessionID`. The value of this form field is used to reference the current authorization flow and prevent Cross-site Request Forgery (CSRF) attacks. The session of the user currently logged in at the identity authority is referenced by the session cookies `sub_sid_current` and `sub_sid_XXX`.

**Weakness.** The values of `sessionID` and these session cookies are not bound to each other. This allows an attacker to bypass the CSRF protection of the `sessionID` value; see M04 for details on the insufficient CSRF protection of the application.

Additionally, the identity authority uses both the value of the `sub_sid_XXX` session cookie and value of the `sessionID` to determine which information should be stored in the access token issued at the end of an authorization flow.

The combination of this incorrect behavior and the missing binding between the value of `sessionID` and the session cookies results in the following CSRF attack which allows an attacker to log in a victim into his/her account:

1. An attacker starts an authorization flow and uses his account at the identity authority (e.g., "pentest.sanz.club") to log in. When the identity authority displays the consent page he does not proceed with the flow but extracts the received `sessionID` parameter.

2. He lures the victim to start a second authorization flow and, if not already logged in, to log in into its account (e.g., "pentest2.sanz.club") at the identity authority.

3. The attacker replaces the value of `sessionID` in the second authorization flow with the value obtained from his first authorization flow and makes the victim confirm the consent page using a CSRF attack.

4. The identity authority creates an Authorization Code and delivers it to the relying party using the victim's user agent (UA). The relying party redeems the Authorization Code at the token endpoint of the identity authority and receives an access token.

The access token received by the relying party contains a mismatch between the `sub` and `id4me.identifier` fields. A decoded example of such an access token is given in Listing 3. While the field `id4me.identifier` references the user account of the victim ("pentest2.sanz.club") the `sub` field's value belongs to the attacker's account. This means the identity authority uses the value of the `sub_sid_XXX` session cookie to determine the value of the `id4me.identifier` field and the value of the `sessionID` to determine the value of the `sub` field.

In the tested scenario, the identity agent behaves correctly and uses the combination of the `iss` and `sub` fields to identify a user account when its userinfo endpoint is called. Therefore, the attack described above results in the victim being logged in the attacker's account at the relying party. The victim might not recognize it is logged in an account different from its own, and uses the services provided by the relying party as it usually would. Depending on these services, the impact of the attack differs. For example, the relying party could provide an upload function for personal files and documents. The victim would upload its private files to the attacker's account, allowing him to access them later.

**Countermeasures.** We recommend binding the `sessionID` value to user's current session at the identity authority. The session is referenced by the session cookies `sub_sid_current` and `sub_sid_XXX`. Every request with inconsistent values should be rejected. This prevents the attack described above and enables `sessionID` to serve as an effective CSRF protection. Additionally, we recommend not using the value of `sessionID` to identify the currently logged in user. Instead, the value should only reference the current authorization flow and the logged in user should only be identified by the session cookies `sub_sid_current` and `sub_sid_XXX`. All user-specific information contained in the access token should only depend on the user identified by session cookies as a way to prevent mismatching information in the access token.

**Retest.** An update of the identity authority was applied during the penetration test. We can confirm that this weakness has been successfully fixed. The identity authority binds the value of `sessionID` to the session cookies `sub_sid_current` and `sub_sid_XXX`. If the request confirming the consent page contains a `sessionID` which does not match the user associated with the session cookies `sub_sid_current` and `sub_sid_XXX`, the identity authority rejects the request and does not issue any Authorization Code.

```
{
  "sub": "Dvkf7Oy9SEqVgt+hPwY7K3mjInrC91L318KVwtsbfakhhNwqc7Thx0VRiPpj2RS7",
  "id4me.identifier": "pentest2.sanz.club",
  "identifier": "pentest2.sanz.club",
  "id4me": "pentest2.sanz.club",
  "clm": [
    "name",
    "given_name",
    "family_name",
    "email"
  ],
  "scope": [
    "openid"
  ],
  "iss": "https://id.test.denic.de",
  "exp": 1552480414,
  "iat": 1552479814,
  "jti": "eop0u3L20VE",
  "client_id": "vygvqmwo2coo2"
}
```

Listing 3: An example body of an access token which was issued by the identity authority when the attack described above is executed (decoded). The value of the `sub` field belongs to a different user account (pentest.club.sanz) than the one specified by the `id4me.identifier`.

## 7.5 M04 Insufficient Cross-site Request Forgery Protection

| **Exploitability Metrics** | | **Impact Metrics** | |
|---|---|---|---|
| Attack Vector (AV) | **Network** | Confidentiality Impact (C) | **Low** |
| Attack Complexity (AC) | **Low** | Integrity Impact (I) | **None** |
| Privileges Required (PR) | **None** | Availability Impact (A) | **None** |
| User Interaction (UI) | **Required** | | |
| Scope (S) | **Unchanged** | | |
| Subscore: **2.8** | | Subscore: **1.4** | |
| **Overall CVSS Score: 4.3** | | | |

Table 6: CVSS calculation of weakness M04.

**General Description.** Cross-site Request Forgery (CSRF) is an attack in which an attacker tricks his victim into performing authenticated commands that changes the application state [6]. The attack is possible since browsers automatically attach cookies to every application HTTP request, regardless of the request origin. Therefore, it's impossible for the server application to distinguish between a valid user-initiated request and an invalid request which was not initiated with the user's consent.

**Weakness.** The dashboard of the identity authority does not apply any CSRF protection. For instance, an attacker could abuse this to disable the victim's account at the identity authority. Additionally, the CSRF protection mechanism present on the authorization endpoint is flawed, as the `sessionID` is not bound to the session cookies `sub_sid_current` and `sub_sid_XXX`. This allows an attacker to bypass the CSRF protection using a `sessionID` obtained with his own account. Details for this weakness can be found in M03.

**Countermeasures.** We recommend adding CSRF protection to all parts of the identity authority which allow to execute crucial actions. We also recommend binding the value of `sessionID` to the user's current session at the identity authority, which is referenced by the session cookies `sub_sid_current` and `sub_sid_XXX`, and reject every request which contains values not bound to each other. This enables `sessionID` to serve as an effective CSRF protection and prevents the attack described in M03.

Additionally, adding the SameSite flag to cookies should be considered, where applicable. In the *strict* mode, `SameSite` cookies are only sent if the request's origin is the website itself. In the *lax* mode, `SameSite` cookies are also sent if the user follows a regular link.

## 7.6 M05 Faulty Session Management

| **Exploitability Metrics** | | **Impact Metrics** | |
|---|---|---|---|
| Attack Vector (AV) | **Physical** | Confidentiality Impact (C) | **High** |
| Attack Complexity (AC) | **Low** | Integrity Impact (I) | **None** |
| Privileges Required (PR) | **None** | Availability Impact (A) | **None** |
| User Interaction (UI) | **Required** | | |
| Scope (S) | **Unchanged** | | |
| Subscore: **0.7** | | Subscore: **3.6** | |
| **Overall CVSS Score: 4.3** | | | |

Table 7: CVSS calculation of weakness M05.

**General Description.** Proper session management requires that sessions are invalidated upon logout. OWASP states that "if a session can still be used after logging out then the lifetime of the session is increased and that gives third parties that may have intercepted the session token more (or perhaps infinite, if no absolute session expiry happens) time to impersonate a user."[7] Users might want to log out at the identity authority for different reasons and should be able to terminate their sessions. One of the more obvious reasons is the use of public computers on which users might not use the private mode. Being unable to log out increases the risk that the user's session is compromised and an attacker takes over the user's account [5].

[7] `https://owasp-aasvs.readthedocs.io/en/latest/requirement-3.2.html`

**Weakness.** The identity authority does not invalidate the session of the user if the user states that he/she is not the person displayed. Additionally, the identity authority does not provide any way to the user to log out of his/her current session.

**Countermeasures.** We recommend to not only unset the cookies in the user's browser, but also to invalidate the session on the identity authority upon logout, and when the user states he/she is not the owner of the account displayed ("not-me"). Additionally, an explicit logout function should be provided to the user, and the identity authority might want to implement the end session endpoint in order to allow relying partys to initiate a logout at the identity authority.

**Retest.** During a retest on 07.06.2019, we reevaluated this weakness. We discovered that the "not-me" button now properly invalidates the users session at the authorization endpoint. However, there was still no explicit logout option which the user can use to log out from the identity authority.

## 7.7 M06 Identity Authority Allows HTTP Redirect URIs

| **Exploitability Metrics** | | **Impact Metrics** | |
|---|---|---|---|
| Attack Vector (AV) | **Network** | Confidentiality Impact (C) | **Low** |
| Attack Complexity (AC) | **High** | Integrity Impact (I) | **Low** |
| Privileges Required (PR) | **None** | Availability Impact (A) | **None** |
| User Interaction (UI) | **Required** | | |
| Scope (S) | **Unchanged** | | |
| Subscore: **1.6** | | Subscore: **2.5** | |
| **Overall CVSS Score: 4.2** | | | |

Table 8: CVSS calculation of weakness M06.

**General Description.** During the process of dynamic client registration, it is possible to specify the type of client by using the parameter `application_type`. Valid values for the parameter are "native" and "web". According to the ID4me standard, "native" clients must use custom URI schemes or the HTTP scheme with the hostname `localhost` as `redirect_uri`. "Web" clients, however, must not use the HTTP scheme but rather the HTTPS scheme, and are not allowed to use the `localhost` hostname.

**Weakness.** When registering a "web" client, the identity authority does not enforce the use of the HTTPS scheme for the `redirect_uri` but also allows the use of the HTTP scheme.

**Countermeasures.** We recommend enforcing the use of the HTTPS scheme for redirect URIs of "web" clients. Client registration requests for "web" clients containing a redirect

URI, which uses the HTTP scheme or the hostname `localhost`, should be rejected by the identity authority.

## 7.8 L01 Missing Brute Force Protections

| **Exploitability Metrics** | | **Impact Metrics** | |
|---|---|---|---|
| Attack Vector (AV) | **Network** | Confidentiality Impact (C) | **Low** |
| Attack Complexity (AC) | **High** | Integrity Impact (I) | **None** |
| Privileges Required (PR) | **None** | Availability Impact (A) | **None** |
| User Interaction (UI) | **None** | | |
| Scope (S) | **Unchanged** | | |
| Subscore: **2.2** | | Subscore: **1.4** | |
| **Overall CVSS Score: 3.7** | | | |

Table 9: CVSS calculation of weakness L01.

**General Description.** Brute force attacks rely on the sheer amount of tries to find the correct input. To prevent these kind of attacks, countermeasures such as CAPTCHAs and account lockouts exist.

**Weakness.** The identity authority does not prevent brute force attacks on user passwords. An attacker can make unthrottled guesses for a user's password, and if the password is correctly guessed, the attacker can log into the user's account.

**Countermeasures.** We recommend implementing CAPTCHAs to prevent automated brute force attacks. We discourage the implementation of account lockouts since these can be abused for Denial of Service (DoS) attacks.

## 7.9 L02 Denial-of-Service Attack at the Identity Agent

| **Exploitability Metrics** | | **Impact Metrics** | |
|---|---|---|---|
| Attack Vector (AV) | **Network** | Confidentiality Impact (C) | **None** |
| Attack Complexity (AC) | **High** | Integrity Impact (I) | **None** |
| Privileges Required (PR) | **None** | Availability Impact (A) | **Low** |
| User Interaction (UI) | **None** | | |
| Scope (S) | **Unchanged** | | |
| Subscore: **2.2** | | Subscore: **1.4** | |
| **Overall CVSS Score: 3.7** | | | |

Table 10: CVSS calculation of weakness L02.

**General Description.** The identity agent attempts to access the JWKS file of an identity authority upon receiving an access token at the userinfo endpoint in order to verify the signature of the access token. Instead of using the discovery mechanism to determine the JWKS URL, the identity agent directly requests the JWKS file at `*IdentityAuthority URL*/jwks.json`.

**Weakness.** During the penetration test, it was possible to provide a 2.5 GB image file as the jwks.json to the identity agent. The identity agent started to download the image file and did not terminate the connection after a particular time or after a certain amount of downloaded data occured. This behavior could result in a Denial-of-Service attack when an attacker makes multiple requests to the userinfo endpoint, tricking the identity agent into downloading multiple large files and consuming network, memory, and other resources of the identity agent.

**Countermeasures.** We recommend only downloading a reasonable amount of data when the identity agent attempts to access the JWKS file of an identity authority, and terminating the connection if either the maximum amount of data is exceeded, or the download is not finished after a reasonable amount of time.

**Retest.** An update of the identity agent was implemented during the penetration test. We can confirm that this weakness has been successfully fixed. The identity agent terminates the connection after a certain period of time instead of trying to download the JWKS files for an unlimited amount of time.

## 7.10 101 Information Disclosure on the Consent Page

**General Description.** The authorization request to the identity authority can contain an optional parameter called `prompt`. This parameter is used by the relying party to indicate to the identity authority how it should handle active sessions of the user. Valid values are: “login”, “consent”, “none”, and “select_account”.

**Weakness.** If the authorization request to the identity authority contains the parameter `prompt` and is set to “select_account”, the consent page delivered by the identity authority contains a stacktrace. The stacktrace reveals internal data to the public. An example of the displayed stacktrace is depicted in Listing 4.

**Countermeasures.** We recommend never displaying internal error messages like stacktraces to the user.

```
FreeMarker template error (DEBUG mode; use RETHROW in production!):
The following has evaluated to null or missing:
==> knownSession.id [in template "de/denic/domainid/authendpoint/view/login.ftl" at line 25, column 111]
...
----
FTL stack trace ("~" means nesting-related):
  - Failed at: ${knownSession.id} [in template "de/denic/domainid/authendpoint/view/login.ftl" at line 25, column 109]
----
Java stack trace (for programmers):
----
freemarker.core.InvalidReferenceException: [... Exception message was already printed; see it above ...]
  at freemarker.core.InvalidReferenceException.getInstance(InvalidReferenceException.java:134)
  at freemarker.core.EvalUtil.coerceModelToTextualCommon(EvalUtil.java:467)
  at freemarker.core.EvalUtil.coerceModelToStringOrMarkup(EvalUtil.java:389)
  at freemarker.core.EvalUtil.coerceModelToStringOrMarkup(EvalUtil.java:358)
  at freemarker.core.DollarVariable.calculateInterpolatedStringOrMarkup(DollarVariable.java:100)
  at freemarker.core.DollarVariable.accept(DollarVariable.java:63)
  at freemarker.core.Environment.visit(Environment.java:366)
  at freemarker.core.IteratorBlock$IterationContext.executedNestedContentForCollOrSeqListing(IteratorBlock.java:317)
  at freemarker.core.IteratorBlock$IterationContext.executeNestedContent(IteratorBlock.java:271)
  at freemarker.core.IteratorBlock$IterationContext.accept(IteratorBlock.java:242)
  at freemarker.core.Environment.visitIteratorBlock(Environment.java:642)
  at freemarker.core.IteratorBlock.acceptWithResult(IteratorBlock.java:107)
  at freemarker.core.IteratorBlock.accept(IteratorBlock.java:93)
  at freemarker.core.Environment.visit(Environment.java:330)
  at freemarker.core.Environment.visit(Environment.java:336)
  at freemarker.core.Environment.process(Environment.java:309)
  at freemarker.template.Template.process(Template.java:384)
  at io.dropwizard.views.freemarker.FreemarkerViewRenderer.render(FreemarkerViewRenderer.java:77)
  at io.dropwizard.views.ViewMessageBodyWriter.writeTo(ViewMessageBodyWriter.java:81)
  at io.dropwizard.views.ViewMessageBodyWriter.writeTo(ViewMessageBodyWriter.java:29)
  at org.glassfish.jersey.message.internal.WriterInterceptorExecutor$TerminalWriterInterceptor.invokeWriteTo(
      WriterInterceptorExecutor.java:265)
  at org.glassfish.jersey.message.internal.WriterInterceptorExecutor$TerminalWriterInterceptor.aroundWriteTo(
      WriterInterceptorExecutor.java:250)
  at org.glassfish.jersey.message.internal.WriterInterceptorExecutor.proceed(WriterInterceptorExecutor.java:162)
  ...
```

Listing 4: Stacktrace displayed on the consent page when the prompt parameter of the authorization request is set to “select_account”.

# 8 Recommendations

In the following sections we provide our recommendations to improve the security of the tested system.

## 8.1 R01 Issue a New Access Token to Access the Identity Agent at Userinfo Endpoint

**General Description.** When the relying party accesses the userinfo endpoint of the identity authority using an access token, the identity authority makes use of the distributed claims feature and informs the relying party that the claims can be accessed at the userinfo endpoint of the identity agent. In the current scenario, the relying party uses the same access token to access the userinfo endpoint of the identity agent afterwards. The identity agent does not use token introspection but directly validates the signature and timestamps in the access token. If the access token is revoked at the identity authority, the identity agent is not aware of this fact.

The following scenario adjustment could enable the identity authority to make sure revoked access tokens cannot be used to access the identity agent. Instead of reflecting the access token used to access its userinfo endpoint, the identity authority issues a new access token and provides it to the relying party. This new access token needs to contain a random and unique value for the `jti` field and an `aud` field matching the identity agent. The identity agent must properly validate all security parameters of the access token (`iss`, `aud`, `jti`, timestamps, and the signature). In combination with R02, this adjustment ensures that the relying party is not able to directly access the userinfo endpoint of the identity agent; it is forced to firstly access the userinfo endpoint of the identity authority every time it wants to access user claims.

**Recommendation.** During the penetration test, DENIC informed us that in the future, it is planned that when the relying party accesses the userinfo endpoint of the identity authority, a new access token will be issued. We recommend implementing this adjustment and issue a new as described above.

## 8.2 R02 Implement Access Tokens as One-Time-Use Tokens at Identity Agent

**General Description.** When the relying party accesses the userinfo endpoint of the identity agent, the identity agent does not use token introspection. The identity agent does not know whether the access token has been revoked at the identity authority and provides the requested claims to the relying party based upon the access token validity (i.e., if it is not expired and the signature is valid). However, the following scenario adjustment could enable the identity agent to ensure that revoked access tokens cannot be

used to access its userinfo endpoint. Instead of allowing an unlimited number of requests to access the user claims within the validity period of an access token, each access token should be accepted only once. As described in R01, the access token intended to be used to access the userinfo endpoint of the identity agent should contain a random and unique `jti` value. The identity agent can then store this value for the validity period of an access token in order to validate the freshness of additional access tokens. In combination with R01, this adjustment ensures that the relying party needs to firstly access the userinfo endpoint of the identity authority every time it attempts to access user claims, instead of being able to directly access the userinfo endpoint of the identity agent. This enables access token revocation to be effective in the analyzed scenario.

**Recommendation.** We recommend only accepting an access token and provide the requested claims after successfully verifying that no access token with the same `jti` has been received before.

## 8.3 R03 Revoke Authorization Code When It Is Redeemed Using False Client Credentials

**General Description.** In order to redeem an Authorization Code at the identity authority, valid client credentials are necessary. The credentials must belong to the relying party which the Authorization Code is intended for. A token request which contains valid client credentials for a relying party but an Authorization Code which was issued for another client might indicate that the Authorization Code has been compromised. Therefore, the Authorization Code should be invalidated in order to mitigate a possible attack.

**Recommendation.** We recommend invalidating the Authorization Code if the token endpoint of the identity authority receives a token request with invalid client credentials (i.e., the client credentials do not match the specific Authorization Code).

## 8.4 R04 Revoke Tokens If the Related Authorization Code Is Redeemed a Second Time

**General Description.** When an Authorization Code is sent to the token endpoint of the identity authority, tokens are issued to the sender and the Authorization Code is invalidated. Redeeming the same Authorization Code at the token endpoint again is not possible. However, this second attempt to redeem an Authorization Code might indicate that the Authorization Code has been compromised. The identity authority cannot determine whether the relying party or an attacker initiated the first or second token request. Therefore, all tokens issued in response to the first token request should be revoked in order to mitigate a possible attack.

**Recommendation.** We recommend revoking all tokens related to an Authorization Code if a token request containing a previously redeemed Authorization Code is received at the token endpoint of the identity authority.

## 8.5 R05 Revoke Tokens When the User Changes his Password

**General Description.** A user can use the dashboard of the identity authority to change the password for his account. However, all tokens issued prior to the password change are not revoked and can still be used to access both the userinfo endpoints of the identity authority and the identity agent.

**Recommendation.** We recommend revoking all current tokens related to a user who changes his password at the identity authority.

## 8.6 R06 Implement Refresh Tokens as One-Time-Use Tokens

**General Description.** When a refresh token is sent to the token endpoint of the identity authority, a new access token and a new refresh token are issued to the sender. However, the old refresh token is not invalidated and can be redeemed to obtain new tokens multiple times. This increases the attack surface because a leaked refresh token can be used to obtain new valid access and refresh tokens even if the leaked refresh token has been used by the victim prior to the leak.

**Recommendation.** We recommend invalidating refresh tokens after they have been used to obtain a new access and refresh token at the token endpoint of the identity authority. Further information and recommendations on the protection of refresh tokens can be found in the "OAuth 2.0 Security Best Current Practice"' [3, section 4.12].

## 8.7 R07 Revoke Tokens When a Refresh Token Is Redeemed a Second Time

**General Description.** Receiving multiple requests at the token endpoint containing the same refresh token might indicate that the refresh token has been compromised. Therefore, the identity authority should revoke all access tokens and refresh tokens related to the refresh token when it receives a request containing this specific refresh token for the second time.

**Recommendation.** We recommend revoking all tokens related to a refresh token if the identity authority receives a request containing an already redeemed refresh token. Further information and recommendations on the protection of refresh tokens can be found in the "OAuth 2.0 Security Best Current Practice" [3, section 4.12].

## 8.8 R08 Prevent Concurrent Logins

**General Description.** If multiple valid sessions can exist for the same user at the same time, this behavior is called concurrent logins. Concurrent logins make it harder to detect if an account has been compromised and should be avoided unless absolutely necessary [5].

**Recommendation.** We recommend preventing concurrent logins at the authorization endpoint by invalidating all other active sessions upon a new login of one user.

## 8.9 R09 Secure Cookies with `HttpOnly` Flag

**General Description.** In most cases, the user's session on the server is associated with a unique identifier. This identifier is usually stored within a cookie in the client's browser and attached to every request to the corresponding website. An attacker, who wants to obtain the user's session, could exploit a Cross-site scripting (XSS) vulnerability in the target application and use the injected JavaScript code to steal the user's cookie. To prevent an attacker from stealing the cookie, the `HttpOnly` flag was introduced. It prevents JavaScript from accessing cookies which were set using this flag.

**Recommendation.** We recommend always setting sensitive cookies (including session cookies) with the `HttpOnly` flag. In particular, this affects the session cookies `sub_sid_current` and `sub_sid_XXX`.

## 8.10 R10 Secure Cookies with `Secure` Flag

**General Description.** Cookies and their sensitive content might be leaked over insecure (unencrypted) connections. The `secure` cookie flag instructs browsers to only send cookies over encrypted `https` connections.

**Recommendation.** We recommend always using the `secure` flag unless it is inevitable that the service can be used via an insecure `http` connection.

## 8.11 R11 Enforce HTTP Stricts Transport Security

**General Description.** HTTP Strict Transport Security (HSTS) was introduced to prevent browsers from sending unsecured HTTP requests. It is enabled by setting the `Strict-Transport-Security` HTTP header. This header contains a time value which indicates for how long the website should only be accessed using HTTPS [9].

**Recommendation.** We recommend enabling HSTS on all pages by setting the following HTTP header with an appropriate time-interval:

```
Strict-Transport-Security: max-age=[time-interval]; includeSubDomains
```

Additionally, it should be considered to apply for inclusion in Google Chromium's HSTS preload list.[8] This list is included in major browsers and specifies domains which must only be accessed using HTTPS. The advantage of the list is that for the first visit of the user HSTS is already enforced by the browser.

## 8.12 R12 Restrict Cross-Origin Resource Sharing to Whitelist

**General Description.** Cross-origin resource sharing (CORS) is a mechanism to share resources across domain boundaries. However, overly permissive CORS settings can lead to security issues as JavaScript can access the respective endpoints from any domain. All tested endpoints which have no web front end respond with valid CORS HTTP headers and set the `Access-Control-Allow-Origin` header value to the request's `Origin` HTTP header value.

**Recommendation.** We recommend implementing a whitelist of allowed origins instead of using the `Origin` header for the `Access-Control-Allow-Origin` value.

## 8.13 R13 Enable Content Security Policy

**General Description.** The Content Security Policy (CSP) is a security mechanism that is used to instruct the browser which actions are allowed on the website and which are not. This covers locations from which resources might be loaded or which might frame the website. Additionally, it can be used to allow or deny loading specific types of resources.

**Recommendation.** We recommend that the application applies a strict CSP to prevent attacks such as XSS and Clickjacking.

## 8.14 R14 Set XSS Protection HTTP Header

**General Description.** To prevent exploitation of missed XSS vulnerabilities at the web application, browser vendors implemented XSS filters. These filters can be enabled and configured for different modes with specific HTTP headers [10].

**Recommendation.** We recommend adding the `X-XSS-Protection` HTTP header to all responses to enable the browsers XSS filter. We recommend that the HTTP header is used with the `mode=block` option if possible [10].

[8] `https://hstspreload.org`

## 8.15 R15 Disable Referer HTTP Header

**General Description.** The `Referer` HTTP header is attached to almost every HTTP request by the browser. It is set to the URL of the page which the user's browser is coming from. This can cause information leaks as parameters contained in the URL can be leaked to another party [10]. In the case of DENIC ID the `Referer` which is attached to the request of a relying party's logo will leak the `nonce` and `state` of the authorization flow. This could be a potential security issue if the relying party's logo is hosted on a different website. To prevent information leaks through the `Referer` header, the `Referrer-Policy` header exists. It can be used to disable the `Referer` header completely.

**Recommendation.** We recommend adding the HTTP header `Referrer-Policy: no-referrer` to every response to disable the usage of the `Referer` HTTP header [10].

## 8.16 R16 Disable Content Type Sniffing

**General Description.** To provide a better user experience, browsers attempt to determine the content type of the presented document (in particular Internet Explorer). This so-called content type sniffing can result in the actually stated content type (`Content-Type` HTTP header) being ignored. Therefore, if a server returns JavaScript Object Notation (JSON) data and sets the correct content type (`Content-Type: application/json`), the browser might still determine that the presented data looks like Hypertext Markup Language (HTML) and try to render it. This could result in security vulnerabilities such as XSS. To prevent content type sniffing, the HTTP header `X-Content-Type-Options` can be used [10].

**Recommendation.** We recommend adding the HTTP header `X-Content-Type-Options: nosniff` to all responses to prevent content type sniffing.

## 8.17 R17 Set Cache Control HTTP Headers

**Description.** On some pages, the application uses the `Expires` HTTP header with the value `epoch`, resulting in those pages being immediately invalidated in the browser cache. However, the application does not use the HTTP headers `Cache-control` and `Pragma` which can be used to prevent any caching by the browser.

**Recommendation.** We recommend adding the HTTP headers `Cache-control: no-store`, `Pragma: no-cache`, and `Expires: -1` on all pages which should not be cached. This will prevent the browser from caching any of those pages.

## 8.18 R18 Use Discovery Mechanism at Identity Agent

**General Description.** When the identity agent receives an access token, it needs to access the JWKS file of the identity authority which issued the access token in order to verify the signature. However, the identity agent does not use the discovery mechanism to determine the URL of the identity authority's JWKS file. Instead, it tries to access the JWKS file at the URL `*IdentityAuthority*/jwks.json` directly. Depending on the configuration of the identity authority, the JWKS file might be located at a different URL and this URL might change in the future. Therefore, the identity agent should use the discovery mechanism to determine the correct location of the JWKS file and use the URL provided by the identity authority to access the JWKS file afterward.

**Recommendation.** We recommend ensuring that the identity agent always uses the discovery mechanism to determine the metadata and URLs of the identity authority instead of trying to access endpoints or files at the identity authority based on URLs hardcoded or obtained earlier.

# 9 Further Evaluations

In this section, we list further evaluations which we conducted in our penetration test. It provides useful information for future security evaluations.

## 9.1 OpenID Connect Parameters

The following tests for common OpenID Connect parameters were conducted:

- It is not possible to change the flow from authorization code flow to the implicit flow or hybrid flow using the `response_type` parameter. The identity authority rejects requests with values other than "code" for the `response_type` parameter and responds with different error messages.
- Different valid values for the `display` parameter ("page", "popup", "touch" or "wap") result in the same consent page being delivered by the identity authority. Invalid values (e.g., "hackmanit") result in the request being rejected with the error message: `HTTP/1.1 404 Not Found ... {"code":404,"message":"HTTP 404 Not Found"}`
- The parameter `response_mode` seems to be ignored by the identity authority. Different values both valid ("form_post", "query", and "fragment") and invalid (e.g., "hackmanit") all result in the Authorization Code being delivered as a query parameter.
- Besides the weakness described in I01, other valid values for the `prompt` parameter are handled correctly:
  - If the value is "login", the identity authority prompts the user to log in even when the user had a valid session before.
  - If the value is "consent", the identity authority displays a consent page to the user directly if the user has a valid session.
  - If the value is "none", the identity authority redirects the user back to the relying party with an error message: `error_description=Consent+required&error=consent_required`

## 9.2 Authorization Code

The following tests for the Authorization Code were conducted:

- The Authorization Code can only be redeemed once. If an already redeemed Authorization Code is sent to the token endpoint of the identity authority a second time, it is rejected with the error message: `HTTP/1.1 400 Bad Request ... {"error_description":"Invalid or expired authorization code, redirection URI mismatch, or PKCE verification failure","error":"invalid_grant"}`

- The Authorization Code can only be redeemed using the client credentials of the relying party it was intended for. If the token request does not contain valid client credentials, or the client credentials do not match the relying party the Authorization Code was intended for, the identity authority rejects the request with the error message: `HTTP/1.1 400 Bad Request ... {"error_description":"Invalid or expired authorization code, redirection URI mismatch, or PKCE verification failure","error":"invalid_grant"}`

## 9.3 Access Token

The following tests for the access token were conducted:

- The access token is supposed to be valid for 600 seconds. Sending a request to the userinfo endpoint of the identity authority or identity agent shows that the access token is not accepted after it has expired.

- The identity authority allows storing the selection of claims a user wants to grant to a specific relying party permanently by checking the "Remember for future logins with this client?" box on the consent page. These granted claims can be changed or revoked in the dashboard of the identity authority. This results in the access token no longer being valid at the userinfo endpoint of the identity authority. As the scenario does not make use of token introspection, the identity agent can only determine whether an access token is valid or not based on the contained timestamps. Therefore, the access token is still considered to be valid at the userinfo endpoint of the identity agent and can still be used to access user information after the granted claims have been changed in the dashboard of the identity authority. If recommendations R01 and R02 will be implemented, the identity agent will not be accessible with revoked access tokens anymore.

- The `claims` parameter in the authorization request contains a JSON document. This document specifies which claims the relying party would like to access using the access token. It is possible to specify that the claims should be contained directly in the ID token instead of being accessed at the userinfo endpoint later. However, when the authorization request contains a JSON document requesting this behavior, the ID token issued by the identity authority later does not contain the requested claims and the access token does not contain any claims either. The userinfo endpoints of both the identity authority and identity agent do not deliver any claims when invoked with the obtained access token.

- Adding unknown claims (e.g., "hackmanit") to the JSON document in the `claims` parameter in the authorization request does not result in an error. However, unknown claims are not contained in the access token issued by the identity authority and, therefore, not provided by the userinfo endpoint of the identity agent.

## 9.4 Refresh Token

The identity authority issues a refresh token in addition to the access token and ID token if the authorization request does not contain the `claims` parameter. To be able to use refresh tokens later, the relying party needs to register the "refresh_token" grant type during its client registration by adding it to the `grant_types` field in the client registration request. The following tests for the refresh token were conducted:

- A refresh token can only be redeemed at the token endpoint of the identity authority using the correct client credentials. If the request does not contain client credentials, invalid client credentials, or valid client credentials which do not belong to the relying party the refresh token was issued for, the identity authority rejects the request with the error message: `HTTP/1.1 401 Unauthorized ... {"error_description":"Invalid client: Possible causes may be missing \/ invalid client_id, missing client authentication, invalid or expired client secret, invalid or expired JWT authentication, or a mismatch between registered and submitted client authentication method","error":"invalid_client"}`

## 9.5 Client Registration Endpoint

The following tests for the client registration endpoint were conducted:

- When registering a new client the parameters `preferred_client_id` and `preferred_client_secret` are ignored and cannot be used to decide which client id and client secret the identity authority assigns to the client.
- When the parameter `application_type` is set to "native", the `redirect_uri` must use the HTTP scheme and the hostname `localhost`. Registering a "native" client using the HTTPS scheme, other hostnames (including `localhost.com`), or malicious redirect URIs (e.g., `http://localhost.attacker.com` or `http://localhost:x@attacker.com`), is not possible.
- When the parameter `application_type` is set to "web", the `redirect_uri` must not use the hostname `localhost`. Registering a "web" client using the hostname `localhost` is not possible. However, the identity authority does not enforce the use of the HTTPS scheme for the `redirect_uri` of "web" clients (see M06)
- It is not possible to register a client using the `javascript:` scheme in the `redirect_uri` parameter.
- When registering a new client, it is possible to define which grant types the client should be authorized to use. However, the identity authority does not assign the grant types "implicit", "client_credentials", "password", "urn:ietf:params:oauth:grant-type:jwt-bearer", or "urn:ietf:params:oauth:grant-type:saml2-bearer" to the client even when they are present in the field `grant_types`

in the client registration request. The only grant types the identity authority allows clients to specify are "authorization_code", and the combination of "authorization_code" and "refresh_token".

- The client registration endpoint can additionally be used to view the information stored for a registered client. In order to access this information, a GET request containing the registration access token for the specific client needs to be sent to `*IdentityAuthority*/clients/*clientid*`. Requests without a registration access token, or with the registration access token of a different client, are rejected with an `HTTP/1.1 401 Unauthorized` error message.

- The client registration endpoint can additionally be used to update certain information about registered clients. In order to update the information, a PUT request containing the registration access token for the specific client, and a JSON document containing the new information, needs to be sent to `*IdentityAuthority*/clients/*clientid*`. Using such requests, it is possible to update information like the redirect URI or logo URL, but not the client id. Requests without a registration access token, or with the registration access token of a different client, are rejected with an `HTTP/1.1 401 Unauthorized` error message.

- The client registration endpoint can additionally be used to delete registered clients. In order to delete a client, a DELETE request containing the registration access token for the specific client needs to be sent to `*IdentityAuthority*/clients/*clientid*`. Requests without a registration access token, or with the registration access token of a different client, are rejected with an `HTTP/1.1 401 Unauthorized` error message. After deleting the client, it is no longer possible to access the consent page, token endpoint, or userinfo endpoint using the client's client id.

## 9.6 End Session Endpoint

We were not able to successfully invoke the end session endpoint. If it is used with a valid `sessionID`, it responds with the error `Access denied by resource owner or authorization server`. If it is used with an invalid `sessionID`, the endpoint responds with `Log-out fails cause required backend resource not available any more`.

## 9.7 Introspection Endpoint

We were not able to successfully invoke the introspection endpoint. It always denied the access. While accessing it with valid client credentials resulted in the error message `Client not registered for https://id.test.denic.de/token/introspect scope`, accessing it with an access token resulted in the error `Insufficient scope`.

## 9.8 Revocation Endpoint

The following tests for the revocation endpoint were conducted:

- To revoke an access token, the request needs to contain the client credentials of the client which the access token was issued to. It is not possible to revoke access tokens issued to other clients.
- After an access token was revoked, it could not be used to access the userinfo endpoint of the identity authority. Due to design issues, the access token can still be used to access the identity agent since the access token is a self-containing JWT, and the identity agent is not supposed to perform token introspection. If recommendations R01 and R02 will be implemented, the identity agent will not be accessible with revoked access token anymore.

## 9.9 Token Endpoint

The following tests for the token endpoint were conducted:

- The identity authority enforces that registered clients can only use the grant types they were authorized to during client registration. Request with values different from these grant types are rejected with the error message: `HTTP/1.1 400 Bad Request ... {"error_description":"The client is not authorized to use this grant type","error":"unauthorized_client"}`
- The identity authority enforces that the parameter `redirect_uri` is present and not empty. Additionally, a strict match between the value of the parameter and the redirect URI registered during client registration (e.g., `https%3A%2F%2F4bgm4tygv3t0vz6h7oof555x2o8ew3.burpcollaborator.net`) is applied. Requests containing the redirect URI of a different client, as well as the following manipulated redirect URIs, were sent to the token endpoint:
  - `https%3A%2F%2Fattacker.com#https%3A%2F%2F4bgm4tygv3t0vz6h7oof555x2o8ew3.burpcollaborator.net`
  - `https%3A%2F%2Fattacker.com#https%3A%2F%2F4bgm4tygv3t0vz6h7oof555x2o8ew3.burpcollaborator.net`
  - `https%3A%2F%2Fattacker.com%23https%3A%2F%2F4bgm4tygv3t0vz6h7oof555x2o8ew3.burpcollaborator.net`
  - `https%3A%2F%2F4bgm4tygv3t0vz6h7oof555x2o8ew3.burpcollaborator.net:x@attacker.com`
  - `https%3A%2F%2F4bgm4tygv3t0vz6h7oof555x2o8ew3.burpcollaborator.net%3Ax%40attacker.com`
  - `http%3A%2F%2F4bgm4tygv3t0vz6h7oof555x2o8ew3.burpcollaborator.net`

- `https%3A%2F%2F4bgm4tygv3t0vz6h7oof555x2o8ew3.burpcollaborator.net.attacker.de`
- `https%3A%2F%2F4bgm4tygv3t0vz6h7oof555x2o8ew3.burpcollaborator.net/hackmanit`
- `https%3A%2F%2F4bgm4tygv3t0vz6h7oof555x2o8ew3.burpcollaborator.net%2Fhackmanit`
- `https%3A%2F%2F4bgm4tygv3t0vz6h7oof555x2o8ew3.burpcollaborator.net/../`
- `https%3A%2F%2F4bgm4tygv3t0vz6h7oof555x2o8ew3.burpcollaborator.net%2F..%2F`
- `https%3A%2F%2F4bgm4tygv3t0vz6h7oof555x2o8ew3.burpcollaborator.net:8443`
- `https%3A%2F%2lF4bgm4tygv3t0vz6h7oof555x2o8ew3.burpcollaborator.net%3A8443`

All requests containing a redirect URI different from the registered one were rejected with the error message: `HTTP/1.1 400 Bad Request ... {"error_description":"Invalid or expired authorization code, redirection URI mismatch, or PKCE verification failure","error":"invalid_grant"}`

## 9.10 Userinfo Endpoints

In addition to the vulnerabilities discovered, the following tests for the userinfo endpoints were conducted:

- Both userinfo endpoints do not allow any HTTP method except GET to retrieve user information. The methods POST, PUT, PATCH, and DELETE are not allowed at all, while HEAD and OPTIONS do not return any user information.
- The userinfo endpoint of the identity authority can only be accessed using a valid access token. Both the timestamps and the signature of the JWT are validated. Manipulating the body of the access token (e.g., by changing the subject `sub`, adding or removing claims) invalidates the signature and results in the request being rejected. The identity authority is not vulnerable to signature exclusion attacks by removing the signature or specifying the `none` algorithm in the header of the access token.
- As described in weakness H01 the identity agent does not enforce that the access token contains a signature. However, if it contains a signature, it has to be a valid one. Requests containing an access token with an invalid signature are rejected by the identity agent.

- When a request containing two `Authorization` HTTP headers, which both contain an access token, is sent to the userinfo endpoint of the identity authority, the second header is ignored and the access token from the first header is used for further processing.

- Sending a request to the userinfo endpoint of the identity agent containing two `Authorization` HTTP headers which both contain a access token results in an error message independently of the validity of, or user related to, the two access tokens.

- From our observations, we assume that the identity agent uses the `identifier` parameter of the access token and queries the DNS to determine which identity authority is responsible for the identifier. Afterwards, it compares the `iss` field of the DNS response with the `iss` field in the access token. If they match, the JWKS file of the identity authority is accessed and used to verify the signature of the access token. This allowed us to operate our own identity authority, and craft and self-sign our own access tokens. We conducted the following tests with self-signed access tokens:

  - The identity agent seems to make use of the combination of the `iss` and `sub` fields to identify a user account. Therefore, it is not possible to access the information of arbitrary users by self-signing an access token which uses the victim user's `sub` value. The identity agent will combine the `sub` value with the `iss` value of our own identity authority. This combination represents a different user account than the one of the victim.

  - If the access token contains two `iss` fields, the identity agent does not reject the request, but uses the value of the second field for all further processing of the access token. Therefore, it is not possible to trick the identity agent into using our JWKS file to verify the signature and another `iss` value to identify the user account.

  - If the access token contains an empty `iss` field or no `iss` field at all, the identity agent rejects the request and does not provide any user information.

  - The identity agent always tries to access the JWKS file and use the key with the matching key id (if present in the document) to verify the signature of the access token. Keys or different URLs to specify the key location present in the header of the access token itself are ignored.

## 9.11 Updating Stored Claims

The identity agent allows updating the claims stored for a user account using the `https://mw-beta.id4me.ionos.com/claims` endpoint. To access this endpoint, a valid access token is required. The following tests for the `https://mw-beta.id4me.ionos.com/claims` endpoint were conducted:

- The endpoint cannot be accessed with a request containing an empty `Authorization` HTTP header or a request without any `Authorization` HTTP header. The identity agent rejects these requests with an error message: `HTTP/1.1 500 ... {"status":500,"error":"Internal Server Error","message":"org.springframework.http.ResponseEntity cannot be cast to java.util.Map","timeStamp":"Thu Mar 14 13:12:42 CET 2019","trace":null}`
- In contrast to the attack described in weakness H01, this endpoint is not vulnerable to signature exclusion attacks. If the access token contains no signature or the contained signature is invalid, the identity agent responds with an error: `HTTP/1.1 403 ... {"general":{},"fields":[{"name":"error","description":"Forbidden.","msgId":"Forbidden."}]}`. Specifying the `none` algorithm in the header of the access token results in the same error message.
- The endpoint was accessed using different HTTP methods:
  - PATCH: The PATCH method is regularly used when the endpoint is accessed using the update function of the web interface of the identity agent.
  - PUT: The PUT method is working with the same parameters as a PATCH request and results in the same response as the PATCH request.
  - DELETE: The DELETE method can be used to delete the claims stored for the user account. All claims are deleted and an overview of the deleted claims is contained in the response of the endpoint.
  - GET: The GET method deletes all claims and responds with an updated `updated_at` value independently of the GET parameters contained in the request.

## 9.12 Open Redirects

We evaluated all URLs in the scope of this penetration test and their corresponding parameters for open redirects. This includes the `redirect_uri` parameter which is strictly validated against the registered redirect URIs of the specific client using simple string comparison. We were not able to identify any open redirect.

## 9.13 Cross-site Scripting

We evaluated all parameters reflected to the end-user for XSS vulnerabilities. All parameters that were reflected at some point were placed either into the HTML context or into the attribute context. Therefore, all payloads aimed either to directly inject HTML elements, or to escape from attribute values and inject additional attributes. The default payload for manually evaluating was `"'<>` to simply check if any of the characters were injected into the website unencoded. We discovered that the application uses either URL

encoding or HTML entity encoding for all reflected values. As this prevents us from injecting arbitrary HTML code or new attributes and event handlers; we were not able to execute arbitrary JavaScript code.

## 9.14 XML-based Attacks

We evaluated if any endpoint in the scope of the penetration test processes eXtended Markup Language (XML) data contained in the request. We used different test payloads for XML DoS and XML External Entity (XXE) attacks. However, no endpoint processed the XML data and no vulnerability based on XML could be identified.

## 9.15 TLS Configuration

We tested the TLS configuration of both the identity agent and the identity authority with testssl.sh and TLS-Scanner. While the servers use different TLS configurations, their configurations are both secure and they are not vulnerable to any relevant attack. Both servers support TLS 1.0 or higher and secure cryptographic algorithms.

The results of testssl.sh are provided in Listing 5 and Listing 6.

```
Testing protocols via sockets except NPN+ALPN

SSLv2 not offered (OK)
SSLv3 not offered (OK)
TLS 1 offered
TLS 1.1 offered
TLS 1.2 offered (OK)
TLS 1.3 not offered
NPN/SPDY h2, http/1.1, acme-tls/1 (advertised)
ALPN/HTTP2 h2, http/1.1 (offered)

Testing cipher categories

NULL ciphers (no encryption) not offered (OK)
Anonymous NULL Ciphers (no authentication) not offered (OK)
Export ciphers (w/o ADH+NULL) not offered (OK)
LOW: 64 Bit + DES, RC[2,4] (w/o export) not offered (OK)
Triple DES Ciphers / IDEA offered (NOT ok)
Average: SEED + 128+256 Bit CBC ciphers offered
Strong encryption (AEAD ciphers) offered (OK)


Testing robust (perfect) forward secrecy, (P)FS -- omitting Null Authentication/Encryption, 3DES, RC4

PFS is offered (OK) ECDHE-RSA-AES256-GCM-SHA384 ECDHE-RSA-AES256-SHA
ECDHE-RSA-CHACHA20-POLY1305
ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA
Elliptic curves offered: prime256v1 secp384r1 secp521r1 X25519


Testing server preferences

Has server cipher order? yes (OK)
Negotiated protocol TLSv1.2
Negotiated cipher ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
Cipher order
```

```
37 TLSv1: ECDHE-RSA-AES128-SHA ECDHE-RSA-AES256-SHA AES128-SHA AES256-SHA
38 ECDHE-RSA-DES-CBC3-SHA DES-CBC3-SHA
39 TLSv1.1: ECDHE-RSA-AES128-SHA ECDHE-RSA-AES256-SHA AES128-SHA AES256-SHA
40 ECDHE-RSA-DES-CBC3-SHA DES-CBC3-SHA
41 TLSv1.2: ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES256-GCM-SHA384
42 ECDHE-RSA-CHACHA20-POLY1305 ECDHE-RSA-AES128-SHA
43 ECDHE-RSA-AES256-SHA AES128-GCM-SHA256 AES256-GCM-SHA384
44 AES128-SHA AES256-SHA ECDHE-RSA-DES-CBC3-SHA DES-CBC3-SHA
45
46
47 Testing server defaults (Server Hello)
48
49 TLS extensions (standard) "next protocol/#13172" "session ticket/#35"
50 "renegotiation info/#65281"
51 "application layer protocol negotiation/#16"
52 Session Ticket RFC 5077 hint (no lifetime advertised)
53 SSL Session ID support yes
54 Session Resumption Tickets: yes, ID: no
55 TLS clock skew Random values, no fingerprinting possible
56 Signature Algorithm SHA256 with RSA
57 Server key size RSA 2048 bits
58 Server key usage Digital Signature, Key Encipherment
59 Server extended key usage TLS Web Server Authentication, TLS Web Client Authentication
60 Serial / Fingerprints 03CC50B3831362F7ABED840DB869BAEAC811 / SHA1 7C3467E08EE097DB532E50EB0A4EC4D
       28457F35B
61 SHA256 4D4A06CED3FEF289D47FEDA4E22910CF9B2B63EE8914938B8BB0025FA5818A81
62 Common Name (CN) id.test.denic.de (CN in response to request w/o SNI: mTRAEFIK DEFAULT CERT)
63 subjectAltName (SAN) id.test.denic.de
64 Issuer Let's Encrypt Authority X3 (mLet's Encrypt from mUS)
65 Trust (hostname) Ok via SAN and CN (SNI mandatory)
66 Chain of trust Ok
67 EV cert (experimental) no
68 "eTLS" (visibility info) not present
69 Certificate Validity (UTC) 82 >= 30 days (2019-03-06 15:50 --> 2019-06-04 15:50)
70 # of certificates provided 2
71 Certificate Revocation List --
72 OCSP URI http://ocsp.int-x3.letsencrypt.org
73 OCSP stapling not offered
74 OCSP must staple extension --
75 DNS CAA RR (experimental) not offered
76 Certificate Transparency yes (certificate extension)
77
78 Testing vulnerabilities
79
80 Heartbleed (CVE-2014-0160) not vulnerable (OK), no heartbeat extension
81 CCS (CVE-2014-0224) not vulnerable (OK)
82 Ticketbleed (CVE-2016-9244), experiment. not vulnerable (OK), reply empty
83 ROBOT not vulnerable (OK)
84 Secure Renegotiation (CVE-2009-3555) not vulnerable (OK)
85 Secure Client-Initiated Renegotiation not vulnerable (OK)
86 CRIME, TLS (CVE-2012-4929) not vulnerable (OK)
87 BREACH (CVE-2013-3587) no HTTP compression (OK) - only supplied "/" tested
88 POODLE, SSL (CVE-2014-3566) not vulnerable (OK)
89 TLS_FALLBACK_SCSV (RFC 7507) Downgrade attack prevention supported (OK)
90 SWEET32 (CVE-2016-2183, CVE-2016-6329) VULNERABLE, uses 64 bit block ciphers
91 FREAK (CVE-2015-0204) not vulnerable (OK)
92 DROWN (CVE-2016-0800, CVE-2016-0703) not vulnerable on this host and port (OK)
93 make sure you don't use this certificate elsewhere with SSLv2 enabled services
94 https://censys.io/ipv4?q=4D4A06CED3FEF289D47FEDA4E22910CF9B2B63EE8914938B8BB0025FA5818A81 could help
       you to find out
95 LOGJAM (CVE-2015-4000), experimental not vulnerable (OK): no DH EXPORT ciphers, no DH key detected with
       <= TLS 1.2
96 BEAST (CVE-2011-3389) TLS1: ECDHE-RSA-AES128-SHA
97 ECDHE-RSA-AES256-SHA
98 AES128-SHA AES256-SHA
99 ECDHE-RSA-DES-CBC3-SHA
100 DES-CBC3-SHA
```

```
VULNERABLE -- but also supports higher protocols TLSv1.1 TLSv1.2 (likely mitigated)
LUCKY13 (CVE-2013-0169), experimental potentially VULNERABLE, uses cipher block chaining (CBC) ciphers with
    TLS. Check patches
RC4 (CVE-2013-2566, CVE-2015-2808) no RC4 ciphers detected (OK)
```

Listing 5: testssl.sh scan of id.test.denic.de

```
Testing protocols via sockets except NPN+ALPN

SSLv2 mnot offered (OK)
SSLv3 mnot offered (OK)
TLS 1 not offered
TLS 1.1 not offered
TLS 1.2 moffered (OK)
TLS 1.3 not offered
NPN/SPDY http/1.1 (advertised)
ALPN/HTTP2 http/1.1 (offered)

Testing cipher categories

NULL ciphers (no encryption) not offered (OK)
Anonymous NULL Ciphers (no authentication) not offered (OK)
Export ciphers (w/o ADH+NULL) not offered (OK)
LOW: 64 Bit + DES, RC[2,4] (w/o export) not offered (OK)
Triple DES Ciphers / IDEA not offered (OK)
Average: SEED + 128+256 Bit CBC ciphers offered
Strong encryption (AEAD ciphers) offered (OK)


Testing robust (perfect) forward secrecy, (P)FS -- omitting Null Authentication/Encryption, 3DES, RC4

PFS is offered (OK) ECDHE-RSA-AES256-GCM-SHA384
ECDHE-RSA-AES256-SHA384 ECDHE-RSA-AES256-SHA
ECDHE-RSA-AES128-GCM-SHA256
ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA
DHE-RSA-AES128-GCM-SHA256 DHE-RSA-AES128-CCM8
DHE-RSA-AES128-CCM DHE-RSA-AES128-SHA256
DHE-RSA-AES128-SHA
Elliptic curves offered: prime256v1 secp384r1 secp521r1 X25519 X448
DH group offered: Unknown DH group (2048 bits)

Testing server preferences

Has server cipher order? yes (OK)
Negotiated protocol TLSv1.2
Negotiated cipher ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
Cipher order
TLSv1.2: ECDHE-RSA-AES256-GCM-SHA384 ECDHE-RSA-AES128-GCM-SHA256
ECDHE-RSA-AES256-SHA384 ECDHE-RSA-AES256-SHA
ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA
DHE-RSA-AES128-GCM-SHA256 DHE-RSA-AES128-CCM8 DHE-RSA-AES128-CCM
DHE-RSA-AES128-SHA256 DHE-RSA-AES128-SHA


Testing server defaults (Server Hello)

TLS extensions (standard) "renegotiation info/#65281" "server name/#0"
"EC point formats/#11" "session ticket/#35"
"next protocol/#13172" "max fragment length/#1"
"application layer protocol negotiation/#16"
"encrypt-then-mac/#22"
"extended master secret/#23"
Session Ticket RFC 5077 hint 10800 seconds, session tickets keys seems to be rotated < daily
SSL Session ID support yes
Session Resumption Tickets: yes, ID: yes
TLS clock skew Random values, no fingerprinting possible
Signature Algorithm SHA256 with RSA
```

```
61 Server key size RSA 2048 bits
62 Server key usage Digital Signature, Key Encipherment
63 Server extended key usage TLS Web Server Authentication, TLS Web Client Authentication
64 Serial / Fingerprints 04C4BAD16928B01B55A137CC4EF0EA4E / SHA1 62C19E9B102D1C8929CDFA1856F59E299272
       DA39
65 SHA256 89237DA7647C13157728CCD3E627C4B9F620D76C0AA1869F3C76A83FF63E50BD
66 Common Name (CN) *.id4me.ionos.com (CN in response to request w/o SNI: *)
67 subjectAltName (SAN) *.id4me.ionos.com
68 Issuer GeoTrust RSA CA 2018 (DigiCert Inc from US)
69 Trust (hostname) Ok via SAN wildcard and CN wildcard (SNI mandatory)
70 Chain of trust Ok
71 EV cert (experimental) no
72 "eTLS" (visibility info) not present
73 Certificate Validity (UTC) 643 >= 60 days (2018-12-17 00:00 --> 2020-12-16 12:00)
74 # of certificates provided 2
75 Certificate Revocation List http://cdp.geotrust.com/GeoTrustRSACA2018.crl
76 OCSP URI http://status.geotrust.com
77 OCSP stapling not offered
78 OCSP must staple extension --
79 DNS CAA RR (experimental) not offered
80 Certificate Transparency yes (certificate extension)
81
82
83 Testing vulnerabilities
84
85 Heartbleed (CVE-2014-0160) not vulnerable (OK), no heartbeat extension
86 CCS (CVE-2014-0224) not vulnerable (OK)
87 Ticketbleed (CVE-2016-9244), experiment. not vulnerable (OK)
88 ROBOT Server does not support any cipher suites that use RSA key transport
89 Secure Renegotiation (CVE-2009-3555) not vulnerable (OK)
90 Secure Client-Initiated Renegotiation not vulnerable (OK)
91 CRIME, TLS (CVE-2012-4929) not vulnerable (OK)
92 BREACH (CVE-2013-3587) no HTTP compression (OK) - only supplied "/" tested
93 POODLE, SSL (CVE-2014-3566) not vulnerable (OK)
94 TLS_FALLBACK_SCSV (RFC 7507) No fallback possible, no protocol below TLS 1.2 offered (OK)
95 SWEET32 (CVE-2016-2183, CVE-2016-6329) not vulnerable (OK)
96 FREAK (CVE-2015-0204) not vulnerable (OK)
97 DROWN (CVE-2016-0800, CVE-2016-0703) not vulnerable on this host and port (OK)
98 make sure you don't use this certificate elsewhere with SSLv2 enabled services
99 https://censys.io/ipv4?q=89237DA7647C13157728CCD3E627C4B9F620D76C0AA1869F3C76A83FF63E50BD could help
       you to find out
100 LOGJAM (CVE-2015-4000), experimental not vulnerable (OK): no DH EXPORT ciphers, no common prime detected
101 BEAST (CVE-2011-3389) no SSL3 or TLS1 (OK)
102 LUCKY13 (CVE-2013-0169), experimental potentially VULNERABLE, uses cipher block chaining (CBC) ciphers with
       TLS. Check patches
103 RC4 (CVE-2013-2566, CVE-2015-2808) no RC4 ciphers detected (OK)
```

Listing 6: testssl.sh scan of api-beta.id4me.ionos.com

# References

[1] Vittorio Bertola. ID4me Technical Overview. https://id4me.org/documents/, June 2018.

[2] M. Jones. JSON Web Algorithms (JWA). RFC 7518 (Proposed Standard), May 2015.

[3] T. Lodderstedt, J. Bradley, A. Labunets, and D. Fett. OAuth 2.0 Security Best Current Practice. Draft-ietf-oauth-security-topics-12, 2019. https://tools.ietf.org/html/draft-ietf-oauth-security-topics-12.

[4] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034 (INTERNET STANDARD), November 1987.

[5] OWASP. Session management cheat sheet. https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Session_Management_Cheat_Sheet.md.

[6] OWASP. Cross-site request forgery (csrf), 2016. https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF).

[7] OWASP. Clickjacking, 2018. https://www.owasp.org/index.php/Clickjacking.

[8] OWASP. Clickjacking Defense Cheat Sheet, 2019. https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Clickjacking_Defense_Cheat_Sheet.md.

[9] OWASP. HTTP Strict Transport Security Cheat Sheet, 2019. https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.md.

[10] OWASP. OWASP Secure Headers Project, 2019. https://www.owasp.org/index.php/OWASP_Secure_Headers_Project#xxxsp.

[11] N Sakimura, J Bradley, and M Jones. Openid connect dynamic client registration 1.0, 2013.

[12] Natsuhiko Sakimura, J Bradley, M Jones, B de Medeiros, and C Mortimore. Openid connect core 1.0, 2014.