fenrir.pro

Penetration Testing Report

[redacted]

27/02/2024

CONFIDENTIAL

Penetration Testing Report

Audit Scope :
- The [redacted] Hardware Controller V1
- The [redacted] Hardware Controller V2
- [redacted]'s Android Applications (including their operating environnement)
- [redacted]'s Web Applications and endpoints

Audit Duration : 10 days

Audit Specifications : The audit was performed in a *grey box* setup : accounts and access to hardware have been provided by the client.

Audit Denomination : Penetration Test

Document ID : F-PENTEST-202499

Version : v2.1.2

Table of Contents

Confidentiality

This penetration testing audit report is classified as confidential and is intended solely for the use of the auditing company and its client. The information contained herein is proprietary and privileged. It is not to be disclosed, disseminated, or shared with any third parties without explicit written consent from both the auditing company and the audited company.

The confidentiality of this report is paramount to maintain the integrity of the findings, recommendations, and sensitive information revealed during the penetration testing activities. Any unauthorized disclosure, reproduction, or distribution of this report, in whole or in part, is strictly prohibited.

By accessing this report, the recipients acknowledges and agrees to the terms of confidentiality outlined herein. Failure to comply with these confidentiality requirements may result in legal action.

Disclaimer

An audit and specifically a penetration test is a snapshot made at a specific point of your project's lifecycle.

The discoveries and recommandations of this report reflect information discovered during the limited duration of this test and cannot represent modifications or changes made out of this time span. Audits on a limited duration cannot be complete, exhaustive, or assess all of the existing security controls and requirements.

The auditor(s) targeted in priority the weakest and more error-prone mechanisms that could be attacked. We recommend auditing an infrastructure regularly, both (if possible) by an internal and external ethical hacking team to ensure the sustainability of your countermeasures.

Scenarios

During this audit, two scenarios have been used as a base for exploitation methodologies and risk assessment :

1 - Third party accessing [redacted]'s hardware

Assess the risk associated to a malicious third party with no prior access or knowledge of [redacted]'s infrastructure : a patient left alone by a practicioner with the hardware for a limited time, someone attacking [redacted]'s infrastructure from a remote location or gaining access to a Controller's WiFi network for instance.

2 - Malicious client

Assess the risk associated with a client trying to access other clients' data, [redacted]'s intellectual property or other practicioners' confidential patient information.

For each of those scenarios, potential impact on :

- [redacted]'s internal data
- [redacted]**'s clients** data
- **Patient** data

Will be evaluated.

Audit Methodology

The methodology followed for this audit is *fenrir.pro*'s internal methodology and was inspired by the Penetration Testing Execution Standard.

It consists in 3 main phases :

- Reconnaissance
- Scan/Enumeration
- Exploitation

During the **reconnaissance** phase, public information will be gathered to identify people, machines, softwares and networks.

This information will be used in the **scan/enumeration** phase to verify and audit user accounts, IP addresses, TCP/UDP ports and services.

Finally, every potential vulnerability will be tested and attempted to be exploited in the **exploitation** phase.

Executive Summary

[redacted]'s infrastructure includes a very wide array of technology stacks, from low-level harware to Android applications.

During this audit, a global desire to provide easy to use and secure solutions to clients has been noticed, authentication, permissions and encryption of sensitive data was globally solid even though along the vast scope of this audit, some vulnerabilities and missing best-practices have been noticed.

Most of the vulnerabilities discovered make access to [redacted]'s private executables, scripts and intellectual property easier, some of the vulnerabilities discovered could give a practicioner access to some of [redacted]'s private information on the platform and two vulnerabilities could give a practicioner access to patient information for patients from other offices.

The overall security level is evaluated as **improveable**.

Addendum : Now that remediation for the vulnerabilities **V02, V03, V05, V06, V09, V10, V11, V12, V13, V14 and V15** is in place, access to sensitive client and patient information is secure and the overall security level is **acceptable**.

Strong points

- Authentication on the API is solid
- [redacted]'s private technical information has a small online footprint
- No sensitive patient data is exchanged between the tablet and its controllers
- No sensitive patient data is exchanged without strong encryption
- No sensitive client data is exchanged without strong encryption
- No sensitive patient data is store unencrypted

Weak points

- The Hardware Controller V2's WiFi dashboard is accessible online and vulnerable
- The Hardware Controller V2 is open enough for a practicioner to explore or modify it
- The back-end exposes client information to authenticated practicioners

Vulnerability Listing

Name	**Description**	**Criticity (CVSS)**	**ID**
Improper Access Control	The supposedly locked launcher provided by AirWatch's Mobile Device Management solution allows users to open Android's settings via the notification tray and kill the launcher, giving them full access to the device without restriction.	3.8/10	**V01**
XSS Injection (Fixed)	The WiFi dashboard does not correctly sanitizes network SSIDs before displaying them in the network list, making it vulnerable to injection of HTML templates containing JavaScript.	1.8/10	**V02**
Arbitrary Command Execution (Fixed)	The WiFi dashboard offers extra features using a socket.io websocket, including calls to the *ping* and *dns* functions when emitting an event with the same name. Those two functions fail to sanitize user input before using it as parameters for the UNIX *ping* and *dig* commands.	10/10	**V03**
Lack of trafic encryption	Data shared between the Hardware Controller V2 and [redacted]'s Android app is not encrypted, which makes it vulnerable to alteration and replay if an attacker manages to intercept exchanges between the two devices.	5.4/10	**V04**
Lack of authentication/ authorization (Fixed)	Using the remote gateways on controller_gw.[redacted].fr and env-dev.[redacted].fr,	6.5/10	**V05**

	the TCP ports 80 of the controllers V2 are accessible using the Controller's numerical ID. This access is not restricted, except from limited JWT authentication made by the Controller's NGINX server.		
Broken Authorization (Fixed)	Hardware Controller V2 includes a NGINX configuration that decodes and uses values passed in a JWT cookie. The signature of this cookie is not validated by the HTTP server, allowing the information to be modified/forged by the user.	7.2/10	**V06**
No BIOS password	The Hardware Controller V2's BIOS is accessible without a password, which allows booting on an external USB device, for instance to read the Controller's /boot partition's contents and attempt to decrypt its encrypted partition.	5.1/10	**V07**
Root shell access via GRUB	During the boot sequence, GRUB's OS selection menu is displayed. If a keyboard (and a screen) are plugged in the Hardware Controller V2, a user can edit one of the menu's entries to boot on a root shell.	6.7/10	**V08**
Sensitive Data Exposure (Fixed)	On a (authenticated) HTTP GET or PUT request on /user on api2.[redacted].com, a JSON object containing information about the current user is sent by the server. It includes a *password* field containing a bcrypt hash	7.7/10	**V09**

	of the user’s password. Password hashes should be considered **sensitive information** and should never be shared outside of their storage location. This not only gives information about the cryptographic mechanisms used by [redacted]’s back-end but also could turn a low-impact XSS or CSRF injection into a full account takeover if the hash is stolen and then cracked.		
Lack of verification of a requirement (Fixed)	The PUT HTTP route /user on api2.[redacted].com does not actually verifies the length of the password it takes as a parameter. Contrary to other routes and rules the front-end enforces, very weak passwords such as *1* are accepted and can later be used to log onto the dashboard.	4.9/10	**V10**
Insufficient Permissions on HTTP endpoint (Fixed)	The HTTP route GET /patient-data/{patient_id}/interviews/{consultation_id}/pdf which returns PDF contents for a given consultation does not verify the authenticated user's permissions and allows retrieval of reports for all patients on the platform. Even other practicioner's.	7.7/10	**V11**
Insufficient Permissions on HTTP endpoint (Fixed)	The HTTP GET /interviews/{consultation_id}/notes route can be used to retrieve consultation notes that do not concern the currently connected practitioner.	7.7/10	**V12**

Insufficient Permissions on HTTP endpoint (Fixed)	[redacted] client information is accessible using two API routes (/ backoffice/customers/ and / bds-customers/{customer_id}) using a regular *practicioner* account, for a total of 921 clients. Including 31 (seemingly valid) IBANs and 158 temporary passwords, including 7 that are still in use.	6.3/10	**V13**
Temporary passwords unchanged (Mitigated)	For 7 clients accounts out of the 158 retrieved using the vulnerability **V13**, the temporary password is **still in use**. The web dashboard does not enforces password change when logging in with a temporary password.	6.8/10	**V14**
Sensitive Information in logs (Fixed)	[redacted]'s REDACTED Android application logs patients' names, birthdates and last consultation dates using Android's general logs. Using ADB or a terminal on the tablet it is possible to read those logs and access this information.	6.0/10	**V15**

Risk matrix

Probability / Impact	Low	Moderate	Critical
Likely		V05 V04 V14	V12 V11 V13 V03
Possible		V01	V15 V09 V08 V06
Unlikely	V02	V07 V10	

Figure 1: Risk matrix associated with the vulnerabilities discovered during the audit

Reconnaissance

The reconnaissance phase of a penetration test consists in using public sources (search engines, social networks, online code databases, etc.) to find technical and organizational information about the target.

Subdomains

[redacted]'s project seems to have a small exploitable online footprint, no public code repositories, still, internal technical information can be guessed from the subdomains registered for [redacted]`.fr`, here's a short selection giving useful insight on [redacted]'s infrastructure :

```
docs.[redacted].fr
gitlab.[redacted].fr
firewall.[redacted].fr
router.[redacted].fr
sentry.[redacted].fr
backoffice.[redacted].fr
vpn.[redacted].fr
weblate.[redacted].fr
gitlab2.[redacted].fr
phpmyadmin.[redacted].fr
keycloak2.[redacted].fr
dev2.[redacted].fr
zabbix.[redacted].fr
proxmox.[redacted].fr
[...]
```

Source: https://securitytrails.com/list/apex_domain/[redacted].fr

No public source code or further internal or exploitable technical or organizational information has been discovered during this phase.

Scan

The scanning phase of a penetration test consists in using scanning tools to obtain information about the hosts and their services (ports, version numbers and shared information) that are exposed.

Hardware Controller V1

This controller is not in pairing mode by default, its power button has to be held for a few seconds to make its Bluetooth chip discoverable.

The chipset has a Texas Instrument MAC identifier and is using the protocol 4.0, compatible with both BLE and BR/EDR, it seems to be operating only in BR/EDR mode (or at least, was not in BLE mode during the tests).

When connecting, the encryption mode chosen by the device is 0x00 (Encryption Disabled) even though its chipset is supporting encryption.

The device advertises itself `[redacted]-BOX [redacted]:8d:01`.

The hardware controller does not exchanges any kind of sensitive client/patient/internal data with the [redacted] Android App, it only shares low-level instructions to control third-party medical devices. That being said, not using encryption makes it theorically vulnerable to sniffing and replay attacks.

Note: *It should also be noted that those attacks on Bluetooth require specific hardware (https://www.tarlogic.com/bsam/resources/bluetooth-connection-sniffing/), knowledge and software tools.*

Hardware Controller V2

The hardware controller exposes 7 TCP ports :

- **22** : OpenSSH
- **53** : DNSmasq
- **80** : NGINX
- **111** : RPCbind
- **8000** : Python Flask
- **8080** : Python Flask
- **8082** : Python Flask

The software versions exposed by The hardware controller do not present exploitable known vulnerabilities in this context.

Android Application / Tablet

The tablet is a Samsung Galaxy Tab A8, it does not directly exposes resources on the network it is connected to and is managed by AirWatch's Mobile Device Management software.

In the tablet's launcher configuration, we can find its AirWatch configuration :

- Server URL : ds[redacted].awmdm.com (airwatchmdm)
- Group ID : [redacted]-P-TABLET-CUST
- User : customer@[redacted].fr

Note: *The tablet does mitigate PIN bruteforce, every 5 invalid attempts, the user has to wait 30 seconds before trying again.*

Web Applications

HTTPS configuration - [redacted].com

Grade A

Strong points

- HTTPS enforced
- No deprecated protocol or encryption mechanism proposed by the server

Weak points

- No HSTS header present

The HTTPS configuration for this host is almost perfect, HTTP trafic is redirected to HTTPS, no deprecated protocol is proposed/supported by the server.

Still, it would be advised to add a *"Strict-Transport-Security"* header to force clients to **only** access the API using HTTPS (instead of HTTP) to avoid HTTPS downgrade attacks for users whose trafic is being monitored by a malicious actor.

Resource: *https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security*

HTTPS configuration - api2.[redacted].com

Grade B

Strong points

- HTTPS enforced

Weak points

- No HSTS header present
- TLS 1.0 (deprecated) is offered by the server
- TLS 1.1 (deprecated) is offered by the server

The HTTPS configuration is good for this host, even though we can notice two issues : two deprecated versions of TLS (1.1 and 1.0) are proposed by the server, which might in a Machine-in-the-Middle scenario allow an attacker to retrieve cryptographic information helpful in decrypting HTTPS trafic.

Also, a *"Strict-Transport-Security"* header is missing on this host.

Resource: *Deprecating TLS 1.0 and TLS 1.1 - https://datatracker.ietf.org/doc/rfc8996/*

HTTPS configuration - [redacted].fr

Grade B

Strong points

- HTTPS enforced

Weak points

- No HSTS header present
- TLS 1.0 (deprecated) is offered by the server

The HTTPS configuration is also good for this host, even though a *"Strict-Transport-Security"* header is also missing and TLS 1.0 (deprecated) is proposed by the server.

Resource: *Deprecating TLS 1.0 and TLS 1.1 - https://datatracker.ietf.org/doc/rfc8996/*

HTTPS configuration - client2.[redacted].com

Grade B

Strong points

- HTTPS enforced

Weak points

- No HSTS header present
- TLS 1.0 (deprecated) is offered by the server
- TLS 1.1 (deprecated) is offered by the server

The HTTPS configuration is also good for this host, even though a *"Strict-Transport-Security"* header is also missing and TLS 1.0 and 1.1 (both depreacted) are proposed by the server.

Resource: *Deprecating TLS 1.0 and TLS 1.1 - https://datatracker.ietf.org/doc/rfc8996/*

Web Stack - client2.[redacted].com

The dashboard on client2.[redacted].com uses the following technologies :

Javascript Framework

- Vue.js

Bugtracking

- Sentry

Web Server

- Nginx 1.22.1

CDN / WAF

- cdnjs
- Cloudflare

JavaScript code is minified, source maps are not present (great!) and Vue.js is up to date and configured in production mode.

Web Stack - api2.[redacted].com

[redacted]'s API uses the following technologies :

Web Server

- Nginx 1.22.1

Rendering Engine

- PHP 8.1.13

Note: *This version of NGINX's security updates have ended the 11th of April 2023, it would be safer to update it to a 1.24 or 1.25.*

This version of PHP is associated to a few vulnerabilities (https://vulners.com/nessus/WEB_APPLICATION_SCANNING_113582) it would also be a safe option to update it for a more recent version.

Scanning potential known files on the server using dirb, two accessible configuration files have been found :

- .htaccess
- web.config

They do not contain sensitive configuration information but they are only necessary for the web server, they should not be accessible for the clients.

Exploitation

The “Exploitation” phase of a penetration test consists in using the identified vulnerabilities to simulate and assess a real-world cyberattack’s impact, attempting to exploit weaknesses in the system’s security to gain unauthorized access, execute malicious code, or achieve other compromise scenarios.

Hardware Controller V2

Bootloader/Hardware

A keyboard and screen have been plugged to the V2 controller.

When the machine boots, pressing the “Delete” key of the keyboard will display its BIOS interface.

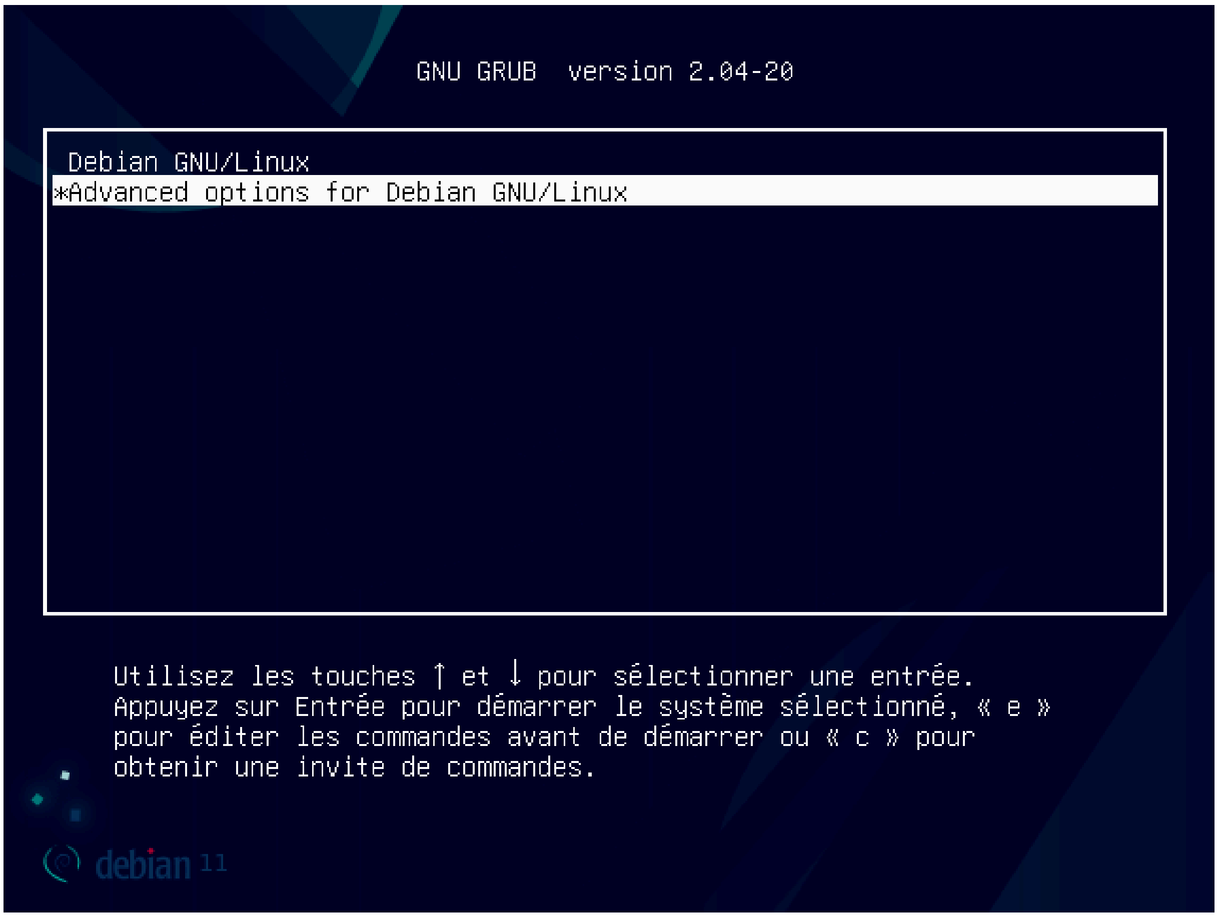
It is accessible without a password and can be used to boot on an external USB device.

Vulnerability	**V07** : No BIOS password
Description	The Hardware Controller V2's BIOS is accessible without a password, which allows booting on an external USB device, for instance to read the Controller's /boot partition's contents and attempt to decrypt its encrypted partition.
Severity	5.1/10
Exploitation	With a USB keyboard plugged-in during the boot sequence, hit the “Delete” key.
Remediation	Practicioners probably don’t need to access their BIOS, restricting access with a strong password would add a useful layer of system hardening.
Resources	https://ubuntu.com/blog/what-is-system-hardening-definition-and-best-practices https://cyber.gouv.fr/publications/recommandations-de-securite-relatives-un-systeme-gnulinux

Remediation: *BIOS admin password addition for the Hardware Controller V2 is under way. It is scheduled for Q2 2024.*

Later in the boot sequence, we can notice that GRUB’s selection menu is displayed for a few seconds.

By editing and booting on one of its entries to force Debian to boot into `/bin/sh` or another shell, it is fairly easy to obtain a root shell on the machine without knowing any of its passwords/passphrases.

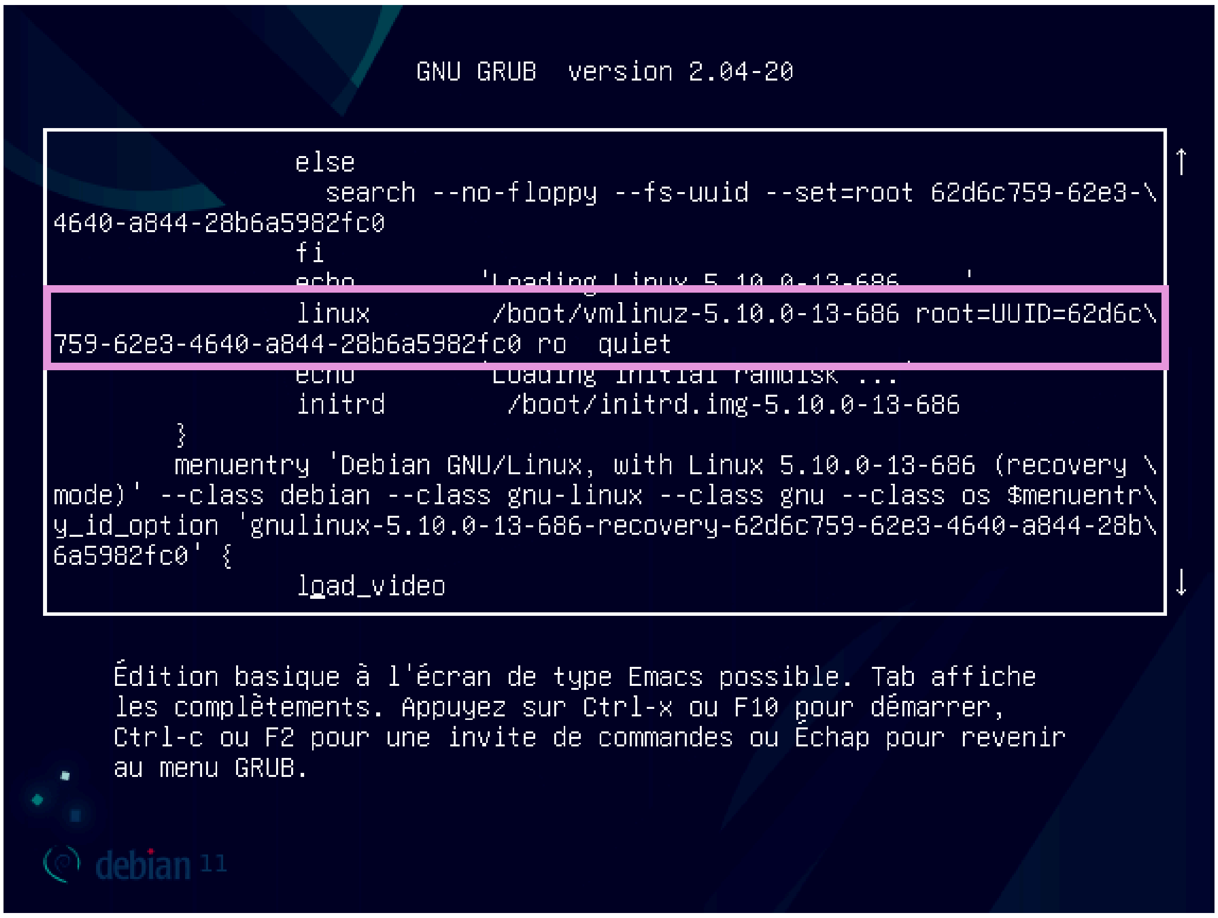


```
GNU GRUB  version 2.04-20

 Debian GNU/Linux
*Advanced options for Debian GNU/Linux

   Utilisez les touches ↑ et ↓ pour sélectionner une entrée.
   Appuyez sur Entrée pour démarrer le système sélectionné, « e »
   pour éditer les commandes avant de démarrer ou « c » pour
   obtenir une invite de commandes.
```

Figure 2: Step 1 : Select any entry in the list

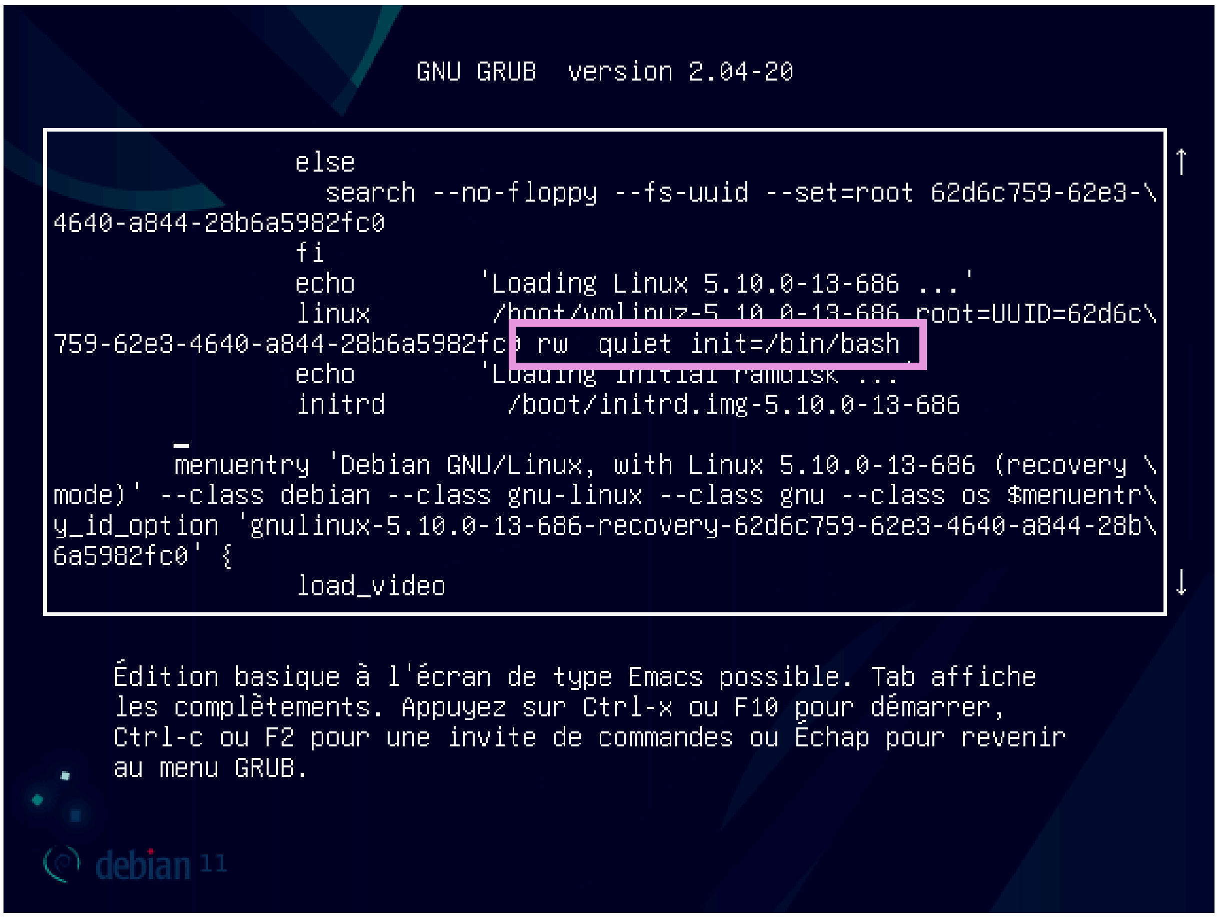


```
                     GNU GRUB  version 2.04-20

                else
                  search --no-floppy --fs-uuid --set=root 62d6c759-62e3-\
4640-a844-28b6a5982fc0
                fi
                echo            'Loading Linux 5.10.0-13-686 ...'
                linux           /boot/vmlinuz-5.10.0-13-686 root=UUID=62d6c\
759-62e3-4640-a844-28b6a5982fc0 ro  quiet
                echo            'Loading initial ramdisk ...'
                initrd          /boot/initrd.img-5.10.0-13-686
        }
        menuentry 'Debian GNU/Linux, with Linux 5.10.0-13-686 (recovery \
mode)' --class debian --class gnu-linux --class gnu --class os $menuentr\
y_id_option 'gnulinux-5.10.0-13-686-recovery-62d6c759-62e3-4640-a844-28b\
6a5982fc0' {
                load_video

    Édition basique à l'écran de type Emacs possible. Tab affiche
    les complètements. Appuyez sur Ctrl-x ou F10 pour démarrer,
    Ctrl-c ou F2 pour une invite de commandes ou Échap pour revenir
    au menu GRUB.
```

Figure 3: Step 2 : Hit "e" to edit it



```
GNU GRUB  version 2.04-20

                else
                  search --no-floppy --fs-uuid --set=root 62d6c759-62e3-\
4640-a844-28b6a5982fc0
                fi
                echo            'Loading Linux 5.10.0-13-686 ...'
                linux           /boot/vmlinuz-5.10.0-13-686 root=UUID=62d6c\
759-62e3-4640-a844-28b6a5982fc0 rw  quiet init=/bin/bash
                echo            'Loading initial ramdisk ...'
                initrd          /boot/initrd.img-5.10.0-13-686

        menuentry 'Debian GNU/Linux, with Linux 5.10.0-13-686 (recovery \
mode)' --class debian --class gnu-linux --class gnu --class os $menuentr\
y_id_option 'gnulinux-5.10.0-13-686-recovery-62d6c759-62e3-4640-a844-28b\
6a5982fc0' {
                load_video

    Édition basique à l'écran de type Emacs possible. Tab affiche
    les complètements. Appuyez sur Ctrl-x ou F10 pour démarrer,
    Ctrl-c ou F2 pour une invite de commandes ou Échap pour revenir
    au menu GRUB.
```

Figure 4: Step 3 : Add init=/bin/sh at the end of the line starting with 'linux'.

Note: *Those are illustration images taken from an external source, but the process is sensibly the same on the* [redacted] *Hardware Controller V2.*

Vulnerability	**V08** : Root shell access via GRUB
Description	During the boot sequence, GRUB's OS selection menu is displayed. If a keyboard (and a screen) are plugged in the Hardware Controller V2, a user can edit one of the menu's entries to boot on a root shell.
Severity	6.7/10
Exploitation	In the GRUB selection menu, hit "E" on a keyboard with an entry selected. Then, add `init=/bin/sh` at the end of the line starting with `linux`. Press F10 to boot on a root shell.
Remediation	Hide this selection menu in GRUB's configuration or make sure entries cannot be edited without a password.
Resources	https://ubuntuhandbook.org/index.php/2020/06/hide-grub-boot-menu-ubuntu-20-04-lts/

	https://askubuntu.com/questions/631317/restricting-on-the-fly-editing-of-grub2-menuentries

Remediation: *Interactive access is going to be disabled on the current bootloader. It is scheduled for Q1 2024.*

Doing this, we notice that the disk partitions other that */boot* are encrypted using LUKS, the passphrase is a 7 lowercase letters long sentence.

It offers a first countermeasure against reverse engineering, but to ensure confidentiality of the [redacted] Hardware Controller V2′s internal scripts, executables, configurations and source code a longer passphrase would be prefered.

Hardware Controller V2 Services

Local Gateway

On the port 80 of the [redacted] Hardware Controller V2 an API allowing interaction with [redacted]'s Android applications is exposed.

It should be noted that exchanges made between the app and The hardware controller are not encrypted, they're using plain HTTP which allowed for interception and modification of exchanges.

For debugging purposes a mock gateway server have been created, its sources are available in the *exploits/fakebox* directory.

As a remediation, using either self-signed certificates or certificates signed by [redacted]'s CA would be a significant evolution.

Note: *It is also important to keep in mind that this service do not exchanges client or patient information except from the client's (not patient) email address.*

Vulnerability	**V04** : Lack of trafic encryption
Description	Data shared between the Hardware Controller V2 and [redacted]'s Android app is not encrypted, which makes it vulnerable to alteration and replay if an attacker manages to intercept exchanges between the two devices.
Severity	5.4/10
Exploitation	In the WiFi parameters of the tablet, configure for the current network a HTTP Proxy whose logs you can monitor. Plain HTTP content with details will be displayed.
Remediation	Configure HTTPS certificates on the local NGINX server of the BoxV2, either using self-signed certificates or certificates signed by [redacted]'s private CA.

Resources	https://www.cloudflare.com/fr-fr/learning/ssl/why-is-http-not-secure/ https://medium.com/@nadinCodeHat/ettercap-and-wireshark-why-is-https-important-fe2169a0da6a

Remediation: [redacted] *is going to add TLS encryption for communication between The hardware controller and the tablet, it is scheduled for Q2/Q3 2024.*

Remote Gateway

When decompiling the Android Application *com.*[redacted]*.REDACTED*, those static strings have been found in the *BoxV2* class:

```
public final class ControllerV2 {
    public static final Companion Companion = new Companion(null);
    public static final String LOCAL_URL = "http://10.42.0.1:5000/";
    public static final String LOCAL_URL_GW = "http://10.42.0.1:5000/controller/";
    public static final String REMOTE_URL = ".[redacted].local/";
    public static final String REMOTE_URL_CTRL = ".[redacted].local/controller/";
    public static final String REMOTE_URL_PREFIX = "https://
controllerstaging.[redacted].fr/";
    public static final String TAG = "ControllerV2";
    [...]
```

The way the strings *REMOTE_URL_POSTFIX_GATEWAY* and *REMOTE_URL_PREFIX* are used in the class's methods gives useful understanding of how controller_gw.[redacted].fr allow for interaction with a box using its ID :

```
    public final String getBoxGatewayUrl(boolean z) {
        if (z) {
            return LOCAL_URL_GW;
        }
        return "https://controller_gw.[redacted].fr/" + this.hostName +
REMOTE_URL_POSTFIX_GATEWAY;
    }

    public final String getBoxDirectUrl(boolean z) {
        if (z) {
            return LOCAL_URL;
        }
        return "https://controller_gw.[redacted].fr/" + this.hostName +
REMOTE_URL_POSTFIX_DIRECT;
    }
    [...]
```

Note: *Decompiled sources are available in the* decompiled/com.[redacted].REDACTED/ *directory.*

It has been tested and it seems that some of the routes on the gateway service of the controllers are accessible without any authentication required.

Vulnerability	**V05** : Lack of authentication/authorization (Fixed)

Description	Using the remote gateways on controller_gw.[redacted].fr and env-dev.[redacted].fr, the TCP ports 80 of the controllers V2 are accessible using the Controller's numerical ID. This access is not restricted, except from limited JWT authentication made by the Controller's NGINX server.
Severity	6.5/10
Exploitation	https://controller_gw.[redacted].fr/ $CONTROLLER_ID.[redacted].intra/
Remediation	Add some kind of authentication (a client SSL/TLS certificate for instance) for this endpoint or alternatively keep it behind a VPN network.
Remediation status	Fixed

Remediation: *This vulnerability has been remediated. Access to a box now requires a valid JWT authentication token on both dev, staging and prod environments.*

Upon inspection of the NGINX configuration of the HTTP server this remote gateway uses, it seems that the server parses the clients JWT tokens contents *without verifying their signature*, which makes editing and altering a custom JWT to fit the webserver's requirements for authentication quite accessible.

The vulnerable code is located in */etc/nginx/njs/jwt.js* on The hardware controller :

```
function jwt(data) {
  var parts = data
    .split(".")
    .slice(0, 2)
    .map((v) => Buffer.from(v, "base64url").toString())
    .map(JSON.parse);
  return { headers: parts[0], payload: parts[1] };
}
```

The token's sections are separated, decoded and interpreted without verification of its signature.

As long as the *payload* section of the token contains the fields **user**, **date** and **"valid": true**, the token will be considered valid by the server.

Such a token have been created for the user **root** using jwt.io.

payload

```
{
  "sub": "1234567890",
  "user": "root",
  "date": 1516239022,
  "valid": true
}
```

corresponding JWT

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwidXNlciI6InJvb3QiLCJkYXRlIjoxNTE2MjM5MDIyLCJ2YWxpZCI6dHJ1ZX0.
FAKE_SIGNATURE_NOT_VERIFIED

Vulnerability	**V06** : Broken Authorization (Fixed)
Description	Hardware Controller V2 includes a NGINX configuration that decodes and uses values passed in a JWT cookie. The signature of this cookie is not validated by the HTTP server, allowing the information to be modified/forged by the user.
Severity	7.2/10
Exploitation	Forge your own JWT token using *jwt.io* or *cyberchef* for instance, then send it to the server in a Authorization header.
Remediation	Either add another layer of JWT verification before sending requests to controllers or make sure the existing NGINX server verifies the signature (which can be done directly in NGINX).
Resources	https://docs.nginx.com/nginx/admin-guide/security-controls/configuring-jwt-authentication/
Remediation status	Fixed

Remediation: *This vulnerability has been remediated. JWT token signatures are now verified on all environment (dev, staging and production).*

This vulnerability could pose a significant risk as the controllers' dashboard can be accessed via the remote gateway with a gateway url such as :

```
https://controller_gw.[redacted].fr/[redacted]-$\{CONTROLLER_ID\}.[redacted].local/
dashboard
https://env-dev.[redacted].fr/[redacted]-$\{CONTROLLER_ID\}.[redacted].intra/dashboard
```

Note: *The production version of this endpoint (controller.[redacted].fr) correctly verifies JWT token signatures.*

The remote gateway works using **WireGuard** as a VPN client/server infrastructure.

A wireguard configuration file is present on The hardware controller in the default location */etc/wireguard/wg0.conf*.

```
[Interface]
Address = 10.11.12.94/32
ListenPort = 51192
PrivateKey = [redacted]/[redacted]+[redacted]=
MTU = 1360

[Peer]
PublicKey = 3IEXYyRydyUQooFffRlkl1uiLoSa8s9JPDlqFTVXE34=
AllowedIPs = 10.51.0.1/32, 10.13.0.0/24, 10.21.0.0/16, 172.16.75.0/24, 10.45.0.0/16
Endpoint = 162.[redacted].[redacted].[redacted]:51192
PersistentKeepalive = 25
```

When modifying the configuration file to allow scanning of the whole 10.51.0.0/16 range, no machine except from our own and the server answered TCP discovery requests on this range (*cf. scan_output/nmap_10.10.0.0_16_sT_T5_aftermodifwg0.txt*).

A few [redacted] machines were accessible on the other ranges (*cf. scan_output/nmap_sT_10.11.0.0_24.txt*) but none of them exposed client/patient information or seemed to expose internal [redacted] information.

The VPN configuration is safe and regularly updated by **puppet**, along with other critical configuration files of the Hardware Controller V2.

WiFi Dashboard

The WiFi dashboard exposed on the port **8081** of the Hardware Controller V2 is a Python Flask application, its source code has been retrieved from the controller and audited.

Two vulnerabilities have been identified :

a XSS injection in the Wireless networks list, at the line 200 of the file `wifi/templates/index.html` :

```
        if ('rsn' in val) {
          type = "lock"
        }
        var cell2 = row.insertCell(1);
        var cell3 = row.insertCell(2);
        var celllock = row.insertCell(3);
        var cell4 = row.insertCell(4);
        cell1.innerHTML = `<input type="radio" name="ssid" value="${val.ssid}" />
  <input type="radio" name="type" value="${type}" hidden/>`;
        console.log(val)

        cell2.innerHTML = val.ssid
```

The *SSID* value is not sanitized and allows injection of HTML templates containing JavaScript code.

It has been tested by creating a new Access Point whose SSID is `"><script>alert(1)</script>`. The 802.11 32 characters limitation for SSIDs makes injection harder to exploit but short enough XSS payloads can be created : https://github.com/terjanq/Tiny-XSS-Payloads.

As a remediation, either using a JavaScript framework including input sanitization or a JavaScript sanitization library (such as https://jsxss.com/en/index.html) would be recommended.

Vulnerability	**V02** : XSS Injection (Fixed)
Description	The WiFi dashboard does not correctly sanitizes network SSIDs before displaying them in the network list, making it vulnerable to injection of HTML templates containing JavaScript.
Severity	1.8/10
Exploitation	Create a new WiFI network using an AP or a mobile device with `"><script>alert(1)</script>` as a SSID.
Remediation	Sanitize SSID names, either using a whole JavaScript framework or a sanitization library.
Resources	https://jsxss.com/en/index.html https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
Remediation status	Fixed

Remediation: *This vulnerability has been remediated. DOMPurify (https://github.com/cure53/DOMPurify) now correctly sanitizes the SSIDs displayed. No further XSS injection have been discovered.*

Also, a **more critical** UNIX command injection using two features exposed by the socket.io websocket.

```python
def bash_command(cmd: str, logger: str = "", status: bool = False) -> str:
    out = []
    process = subprocess.Popen(["/bin/bash", "-c", cmd], stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
    if process.stdout is not None:
        for out_line in iter(lambda: process.stdout.readline(), b""):  # type:
ignore
            out.append(str_line := out_line.decode())
            logging.debug(str_line)
            socketio.emit(logger, {"log": str_line})
    if not status:
        return "".join(out)
    else:
        code = process.wait()
        out, err = process.communicate()
        return "".join(out), str(out), str(err), int(code)
```

The main issue originates from the line 45 of *wifi.py* in the *bash_command()* function.

Every call made to this function containing user-provided input is inherently unsafe, the call to *subprocess.Popen* does not verify the contents of the *cmd* variable, which is built from user input for the *dns* and *ping* calls.

Exploitation can be done using *command substitution* with backquotes/backticks (`) or `$()` to encapsulate another command which will be executed before the main command.

For instance on a page of the dashboard, open the JavaScript console and type `socket.emit('dns', '$(whoami))'`. The result of the `whoami` command will be sent back in the same websocket.

Vulnerability	**V03** : Arbitrary Command Execution (Fixed)
Description	The WiFi dashboard offers extra features using a socket.io websocket, including calls to the *ping* and *dns* functions when emitting an event with the same name. Those two functions fail to sanitize user input before using it as parameters for the UNIX *ping* and *dig* commands.
Severity	10/10
Exploitation	On a page of the dashboard, open the JavaScript console and type `socket.emit('dns', '$(whoami))'`. The `whoami` command will be executed, then its output used as a parameter for the initial `dig` command.
Remediation	Only allow specific inputs to be passed to unix commands through Popen, with an allowlist (best solution), a regex or/and make sure you sanitize inputs using shlex.quote (cf. resource) for instance.
Resources	https://semgrep.dev/docs/cheat-sheets/python-command-injection/

Remediation: *This vulnerability has been remediated. Command execution has been tested again and is not possible anymore on this API.*

Interestingly, we can notice that this web application is running as `root`, which gives us full unrestricted access to the controller.

Android Application Execution Environment

The tablet locks users in a restricted environment in which only selected applications and parameters are accessible.

Using the tablet, it has been noticed that AirWatch's launcher is not actually 100% permeable, there is at least one way to exit the launcher only with UI interactions.

Here are the steps followed during the audit to exit the launcher and dump an APK file for [redacted]'s application that is installed on the tablet.

Vulnerability	**V01** : Improper Access Control
Description	The supposedly locked launcher provided by AirWatch's Mobile Device Management solution allows users to open Android's settings via the

	notification tray and kill the launcher, giving them full access to the device without restriction.
Severity	3.8/10
Exploitation	Exploitation steps given in the audit details.
Remediation	Replace the launcher or modify its configuration to make sure a user cannot evade from the locked environment.

Remediation: [redacted] *is going to disable access to system parameters via Android notifications. This fix is scheduled for Q1/Q2 2024.*

It is also quite simple to get the WPA2 pre-shared key of a [redacted] Hardware Controller V2 from a connected tablet since AirWatch's launcher uses the system WiFi settings panel :

A QR code containing the network's SSID and PSK is then displayed, which allows connecting with the [redacted] Hardware Controller V2 with another device. The only requirement is to know the tablet's PIN code.

Note: *ADB and developer options are disabled on the tablet, Android version is maintained up to date, the tablet is not rooted and the global configuration does not present exploitable vulnerabilities.*

Android Application

The application does not store sensitive information on globally accessible parts of the tablet, its configuration does not allow exchange of patient information via a unencrypted protocol. The codebase is not obfuscated and quite comfortable to decompile and read (*cf. decompiled/ com.*[redacted]*.REDACTED*).

If the application did not present vulnerabilities by itself, it logs patients names, birthdates, languages and last consultation.

As the tablet logs can be retrieved using ADB, physical access to the tablet plugged to a computer via USB or network access via ADB over TCP could give access to patient information :

Logs for a test patient whose names are "toto42 toto42"

```
#> adb -s R9PTB1ERSJD logcat | grep toto42
01-09 15:32:44.766 22936 23259 D LocalServiceImpl: updatePatient firstName=toto42,
lastName=toto42, birdthdate=1993-11-11, language=FRENCH, fileNumber=6,
lastConsultation=null
01-09 15:36:20.929 22936 23590 D RemoteServiceImpl: serviceCall : addPatient toto42
toto42 localId = 3
01-09 15:36:21.350 22936 23591 W RemoteServiceImpl: IOException when addPatient
toto42 toto42 localId = 3
01-09 15:37:12.781 22936 22984 D RemoteServiceImpl: serviceCall : addPatient toto42
toto42 localId = 3
[...]
```

Vulnerability	**V15** : Sensitive Information in logs (Fixed)
Description	[redacted]'s REDACTED Android application logs patients' names, birthdates and last consultation dates using Android's general logs. Using ADB or a terminal on the tablet it is possible to read those logs and access this information.
Severity	6.0/10
Exploitation	In a shell from a machine connected to the tablet `adb logcat | grep 'updatePatient'`.
Remediation	Remove every log containing sensitive information from the application's source code.
Resources	https://developer.android.com/privacy-and-security/risks/log-info-disclosure?hl=fr
Remediation status	

Remediation: *This vulnerability has been remediated.* [redacted] *Exam does not logs patient information anymore in its latest release.*

Dashboard / API Exploitation

Authentication

Authentication on the platform uses the `POST /login` API route, with an **email** and **password** JSON parameters.

The API uses a standard *Authorization* header with a JWT value, JWT signatures are correctly verified and seem to use a strong *RSASHA256* keypair.

No SQL injection was found on this endpoint. Requests are rate-limited by the back-end, only 1 request per second is allowed, which makes bruteforce and dictionary attacks much less likely to succeed in a reasonable timespan.

Password lost

The "lost password" feature uses the user's email address, it uses a single-use token with satisfying entropy (probably the SHA256 hash of a random value).

The API routes does not allow modification of another user's password and the token is correctly invalidated directly after its first use. They cannot be used to enumerate existing email adresses on the platform, the same respose is sent by the server if the user's email is valid or not.

Password rules are correctly enforced.

User profile

Current user's information is fetched using a HTTP GET request on */user* on the API.

Surprisingly, the JSON object returned by the back-end contains the current user's password hash (probably a salted bcrypt hash given its format).

Vulnerability	**V09** : Sensitive Data Exposure (Fixed)
Description	On a (authenticated) HTTP GET or PUT request on /user on api2.[redacted].com, a JSON object containing information about the current user is sent by the server. It includes a *password* field containing a bcrypt hash of the user's password. Password hashes should be considered **sensitive information** and should never be shared outside of their storage location. This not only gives information about the cryptographic mechanisms used by [redacted]'s back-end but also could turn a low-impact XSS or CSRF injection into a full account takeover if the hash is stolen and then cracked.
Severity	7.7/10
Exploitation	Looking at the "Network" section of a user's browser debug tools, they can see a HTTP call `/user` on the API. It contains a JSON object with a bcrypt password hash.
Remediation	A password hash is a very sensitive piece of information. Once retrieved (for instance with a theorical XSS injection on the dashboard), it can be cracked and used to find an actual user's

	password. It also has been noticed that some users of the platform do use weak (old, supposedly temporary) passwords, which makes cracking an actual issue.
Remediation status	Fixed

Remediation: *This vulnerability has been remediated. The version 1.15.11.0 of the API correctly fixes this issue.*

Modifying a user's information uses a HTTP *PUT* request on the same endpoint, permissions are correctly handled but password rules are much less restrictive on this route, using "1" or even an empty string as a password is allowed by the back-end.

Vulnerability	**V10** : Lack of verification of a requirement (Fixed)
Description	The PUT HTTP route /user on api2.[redacted].com does not actually verifies the length of the password it takes as a parameter. Contrary to other routes and rules the front-end enforces, very weak passwords such as *1* are accepted and can later be used to log onto the dashboard.
Severity	4.9/10
Exploitation	Using cURL or the "replay" feature of your browser's network tools, change the password value in the query made to the remote server.
Remediation	Make sure the password rules are the same on every API endpoint.
Remediation status	Fixed

Remediation: *This vulnerability has been remediated. The version 1.15.11.0 of the API fixes this issue, the route now correctly enforces password requirements.*

This HTTP route cannot be used to obtain or modify other user's information. It does not allow modification of a user's permissions or roles and does not allow account takeover by changing the user's email address by another existing one.

Patients/Consultations

Exporting, searching, editing and accessing patients and consultations data using the dashboard and [redacted]'s API has proven to be robust.

The search filters seem resistant to SQL injection, permissions are correctly handled except for two specific endpoints which exposed patient information to all practitioners' accounts :

- `GET /patient-data/{patient_id}/interviews/{consultation_id}/pdf`
- `GET /interviews/{consultation_id}/notes`

The first route allows, when incrementing the numerical *patient_id* and *consultation_id* value (starting from 1) to retrieve patient names and consultation details in the PDF format.

Note: *Permissions on* `/patient-data/{patient_id}/interviews/{consultation_id}` *are safe, only the* `/pdf` *endpoint is vulnerable.*

Vulnerability	**V11** : Insufficient Permissions on HTTP endpoint (Fixed)
Description	The HTTP route GET /patient-data/{patient_id}/interviews/{consultation_id}/pdf which returns PDF contents for a given consultation does not verify the authenticated user's permissions and allows retrieval of reports for all patients on the platform. Even other practicioner's.
Severity	7.7/10
Exploitation	Iterating on IDs from 1 to 15000 on `HTTP GET /patient-data/{patient_id}/interviews/{consultation_id}/pdf` to dump PDF consultation reports.
Remediation	Make sure the permissions associated with this route are consistent with the ones on `HTTP GET /patient-data/{patient_id}/interviews/{consultation_id}`
Remediation status	Fixed

Remediation: *This vulnerability has been remediated. The version 1.15.11.0 of the API fixes this issue, the route now correctly verifies user permissions.*

The second one returns a JSON object containing consultation notes, iterating on the *consultation_id* (starting from 1) allows to retrieve all of the consultation notes stored for all practitioners and patients.

Vulnerability	**V12** : Insufficient Permissions on HTTP endpoint (Fixed)
Description	The HTTP GET /interviews/{consultation_id}/notes route can be used to retrieve consultation notes that do not concern the currently connected practitioner.
Severity	7.7/10
Exploitation	Iterating on IDs from 1 to 15000 on `HTTP GET /interviews/{consultation_id}/notes` to dump consultation notes.

Remediation	Make sure the permissions associated with this route are consistent with the ones on `HTTP GET /patient-data/{patient_id}/interviews/{consultation_id}`
Remediation status	Fixed

Remediation: *This vulnerability has been remediated. The version 1.15.11.0 of the API fixes this issue, the route now correctly verifies user permissions.*

Admin/Back-Office sections

The Vue.js application's routes enforce permission verification for administrative routes that have to be restricted.

The front-end was not found to be vulnerable, its only downside is that its sources contain references to its development version `https://dev2.`[redacted]`.fr` which runs an un-minified version of the app with mapfiles accessible, with unobfuscated code easily accessible.

Most of the administrative/back-office routes tested on the back-end returned a `HTTP 403` response and correctly verified that the current user was actually allowed to access resources, except from a few ones which unfortunately allowed enumeration of valuable client information.

> **Note**: *Every upload section of the practicioner dashboard also has been tested, none seemed exploitable, filetypes and filenames are correctly verified, no LFI/RFI or abitrary file upload has been found.*

`GET /backoffice/customers/`

This route seems to be a transient buffer, it returns a JSON object containing two tables : *created* and *updated*.

The first time the route was requested, it returned valid data for a customer not related to the current practicioner's account in its *created* table :

```
[
   {
      "sales_term":"3",
      "payment_method":"5",
      "title":null,
      "firstname":null,
      "lastname":null,
      "phone":null,
      "contact_mail":"real@client.se", #redacted for client confidentiality
      "brand":null,
      "tva":"XXXX6962-0910" #redacted for client confidentiality,
      "iban":null,
      "sync_token":"3",
      "company_name":"100 Street Stockholm Norrmalmstorg", #redacted for client
confidentiality
      "billing_address":{
         "address":"Sankt Eriksgatan", #redacted for client confidentiality
         "city":"Stockholm",
         "country":"Sweden",
         "zip_code":"11234"
      },
      "delivery_address":{
         "address":"=G2",
         "city":"=I2",
         "country":"Sweden",
         "zip_code":"=H2"
      },
      "backoffice_id":"1323",
      "parent_id":29,
      "updated_at":"2024-01-15T16:42:49Z",
      "created_at":"2024-01-15T16:42:49Z",
      "id":901
   }
]
```

`GET /backoffice/customers/`

This route returns client data, about the same as `/backoffice/customers/` except that the returned JSON object contains a few additional fields :

```
{
   "id":116,
   "parent_id":114,
   "backoffice_id":1041,
   "geolocalization":null,
   "title":null,
   "firstname":null,
   "lastname":null,
   "company_name":"OPTICAL CLERMONT", #redacted
   "shop_name":"OPTICAL CLERMONT", #redacted
   "status":"Client",
   "group":"OPTICAL CLERMONT", #redacted
   "market_type":"Opti",
   "brand":"OPTICAL", #redacted
   "chain_id_v1":null,
   "shop_id_v1":965,
   "central_code":null,
   "tva":null,
   "iban":null,
   "bic":null,
   "legal_representative":null,
   "phone":"040000001", #redacted
   "contact_mail":"clermont@optical.fr", #redacted
   "tmp_password":"Wlmv6DAJ",
   "billed_parent":1,
   "billing_address":{
      "city":"CLERMONT FERRAND",
      "address":"rue de Clermont", #redacted
      "country":"FRANCE",
      "zip_code":"63100"
   },
   "delivery_address":{
      "city":"CLERMONT FERRAND",
      "address":"rue de Clermont",
      "country":"FRANCE",
      "zip_code":"63100"
   },
   "sync_token":"1",
   "sales_term":1,
   "payment_method":4,
   "created_at":"2023-02-07T16:59:20Z",
   "updated_at":"2024-01-15T16:42:53Z",
   "deleted_at":null,
   "parent_organization_id":null,
   "organization_id":null,
   "shop_id":null
}
```

All IDs for this route have been enumerated, which allowed to retrieve : 31 valid IBANs for 31 current clients and 158 non-null *tmp_password* fields values.

Vulnerability	**V13** : Insufficient Permissions on HTTP endpoint (Fixed)
Description	[redacted] client information is accessible using two API routes (/backoffice/customers/ and /bds-customers/{customer_id}) using a regular *practicioner* account, for a total of 921 clients. Including 31 (seemingly valid) IBANs and 158 temporary passwords, including 7 that are still in use.
Severity	6.3/10
Exploitation	Exploit available in *exploits/dump_customers.sh*.
Remediation	If this route must be accessible for the practicioners accounts, make sure it ONLY returns information about the clients they must be aware of, not all of [redacted]'s clients.
Remediation status	Fixed

Remediation: *This vulnerability has been remediated. The version 1.15.11.0 of the API fixes this issue, the route now correctly verifies user permissions.*

Note: *Some other routes were accessible with a practicioner account under* `/backoffice`, `/bds` *and* `/admin` *but they probably have a lower impact :* `/backoffice/items`, `/backoffice/process-accounting`, `/backoffice/import-accounting`, `/stats/admin/`, `/stats/practionner/shop/{shop_id}`, `/stats/opto/shops/{shop_id}/{other_unknown_id}`

The 158 temporary passwords were tested and 7 have not been changed and can be used to connect on the dashboard using the client's *contact_mail*.

Vulnerability	**V14** : Temporary passwords unchanged (Mitigated)
Description	For 7 clients accounts out of the 158 retrieved using the vulnerability **V13**, the temporary password is **still in use**. The web dashboard does not enforces password change when logging in with a temporary password.
Severity	6.8/10
Exploitation	Exploits available in *exploits/get_tmpcreds.sh* and *exploits/testcreds.sh*.
Remediation	Make sure the users connecting using temporary passwords are forced to change them, **both** on the dashboard and the Android Application.

Remediation status	Fixed

Remediation: *This vulnerability is considered remediated. As temporary passwords are not available anymore thanks to the fix of V13. It would be advised additionally to force current users with weak passwords to change them and deactivate unused accounts.*