CVE-2022-0482 Report

Mija Pilkaite - 112550807

CVE-2022-0482 is a notable security flaw that was discovered in the Easy!Appointments scheduling software, specifically impacting versions prior to 1.4.3. This vulnerability, classified under CWE-863 (Incorrect Authorization), allows unauthorized individuals to access personally identifiable information pertaining to events without having to authenticate. Reported by Francesco Carlucci on January 30th, 2022, exploiting this vulnerability poses a significant threat as it could enable exposure of sensitive data, undermining the affected system's confidentiality and integrity.

To recreate an environment vulnerable to CVE-2022-0482, a deliberative methodology must be employed. An isolated controlled environment like a virtual machine or Docker container should be prepared to host the software. We will use Ubuntu 22.04 Server. Initial steps involve downloading a vulnerable version of Easy!Appointments, namely 1.4.2 or earlier from the project's GitHub repository (https://github.com/alextselegidis/easyappointments). We use wget command to get the source file. We change the PHP version in “docker/server/Dockerfile” to 8.0. Next, we must configure the “config.php” file according to the documentation found from the official website (https://easyappointments.org/docs.html#1.4.2/docker.md). We run the command:

```
docker compose up -d
```

to start the Docker containers. Now, the Docker container can be entered running:

```
docker exec -it <container_name> bash’
```

We move on by installing all the needed add-ons: Git, Node.js, Composer.

We use ‘apt install’ for installing Git and Node.js:

```
apt install git
apt install npm
```

whereas the Composer can be installed running:

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"

php -r "if (hash_file('sha384', 'composer-setup.php') ===
'e21205b207c3ff031906575712edab6f13eb0b361f2085f1f1237b7126d785e826a450292b6cfd1d64d92e6563bbde0
2') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composersetup.php');
} echo PHP_EOL;"

php composer-setup.php

php -r "unlink('composer-setup.php');"
```

Now, we can enter the Docker container again and install our project dependencies:

```
npm install
php composer.phar install
```

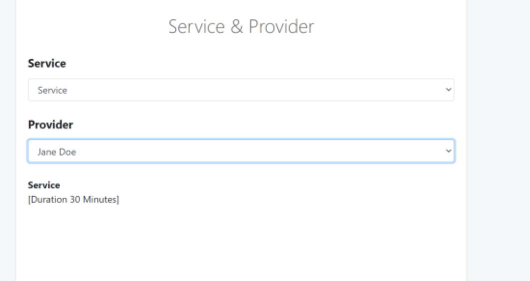
Finally, we can access the Easy!Appointments website on our machine using http://localhost.

Now, as our environment is set up, we must move on to finding our strategies for exploiting the vulnerability, which is going to be our next part. For right now, we must play around and make sure the setup is working.

The vulnerability lies within the fact that personal data can be exposed to people without proper credentials. The backend API provides functionality for data management, such as retrieving a list of appointments within a given time frame, accessible through the endpoint: /index.php/backend_api/ajax_get_calendar_events. However, this endpoint lacks security measures like authentication or permissions checks. To make a POST request, only "startDate", "endDate", and "csrfToken" are needed. Since the csrfToken is obtainable by any user who visits the public form, and it's also valid for backend access, this vulnerability allows potential attackers to extract private information about appointments in JSON format from the backend API.

In order to exploit the vulnerability, we start by launching the application and creating the admin profile. Then, we can access that by logging in at http://localhost/index.php/backend.

Then, coming back to http://localhost we can create an appointment:



Service & Provider

Service

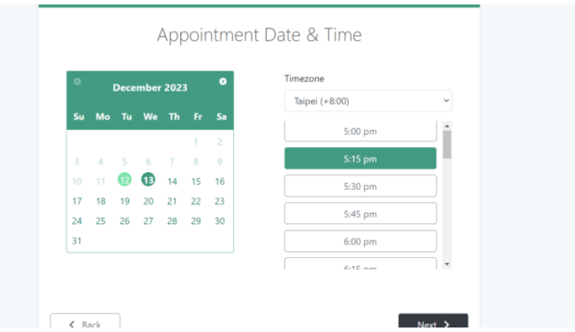
Service

Provider

Jane Doe

Service
[Duration 30 Minutes]

We choose the date and time:



Appointment Date & Time

December 2023

Su	Mo	Tu	We	Th	Fr	Sa
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Timezone

Taipei (+8:00)

5:00 pm

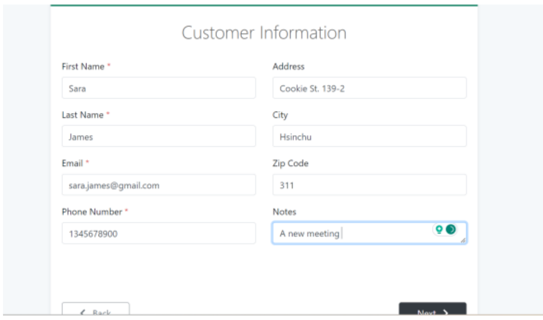
5:15 pm

5:30 pm

5:45 pm

6:00 pm

Fill out the information of a client requesting a meeting:



Customer Information

First Name *
Sara

Last Name *
James

Email *
sara.james@gmail.com

Phone Number *
1345678900

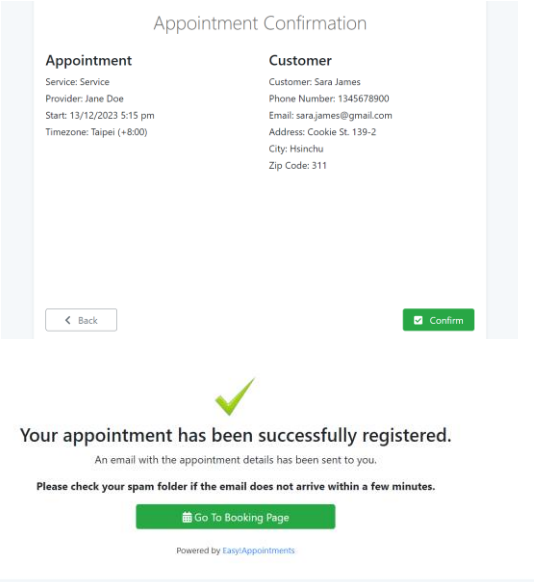
Address
Cookie St. 139-2

City
Hsinchu

Zip Code
311

Notes
A new meeting

After the appointment has been made, one should receive a confirmation message:



Appointment Confirmation

Appointment

Service: Service
Provider: Jane Doe
Start: 13/12/2023 5:15 pm
Timezone: Taipei (+8:00)

Customer

Customer: Sara James
Phone Number: 1345678900
Email: sara.james@gmail.com
Address: Cookie St. 139-2
City: Hsinchu
Zip Code: 311

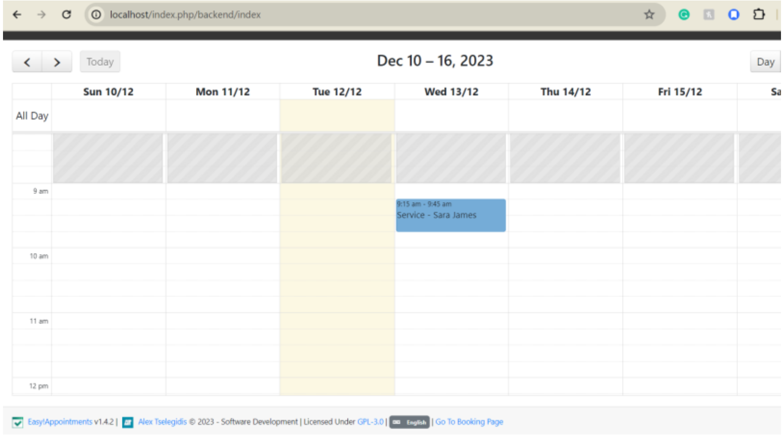
Your appointment has been successfully registered.

An email with the appointment details has been sent to you.

Please check your spam folder if the email does not arrive within a few minutes.

Powered by Easy!Appointments

Now, if we check on the backend (admin) side, the appointment should show up in our schedule (the discrepancy in time is because of the time zones as mine is still set to home).



localhost/index.php/backend/index

Dec 10 – 16, 2023

9:15 am - 9:45 am
Service - Sara James

Easy!Appointments v1.4.2 | Alex Tselegidis © 2023 - Software Development | Licensed Under GPL-3.0 | English | Go To Booking Page

Now, we can run the script to exploit the vulnerability.

An assailant could easily obtain their CSRF token from the application's homepage and then proceed to access the unsecured API endpoint (/index.php/backend_api/ajax_get_calendar_events). By iterating over various dates, they could extract all the appointment and user data stored in the system. Additionally, the HTTP response disclosed a wealth of sensitive information that could be exploited maliciously:

- Comprehensive details of all clients (including full name, email, phone number, address, etc.).
- Appointment hashes, which could be used to cancel appointments and compromise data integrity.
- Information about the service provider, including hashed passwords (with the extent of the impact being uncertain).

Thus, we can exploit the vulnerability. The script can be run using the command:

```
cve-2022-0482.py [-h] [--startDate STARTDATE] [--endDate ENDDATE] hostname

```

Thus, after running the script, we can obtain the following information:

```
POST request response:
{
    "appointments": [
        {
            "id": "1",
            "book_datetime": "2023-12-12 08:18:46",
            "start_datetime": "2023-12-13 09:15:00",
            "end_datetime": "2023-12-13 09:45:00",
            "location": null,
            "notes": "A new meeting ",
            "hash": "N53EagdGAwf7",
            "is_unavailable": "0",
            "id_users_provider": "2",
            "id_users_customer": "4",
            "id_services": "1",
            "id_google_calendar": null,
            "provider": {
                "id": "2",
                "first_name": "Jane",
                "last_name": "Doe",
                "email": "jane@example.org",
                "mobile_number": null,
                "phone_number": "+1 (000) 000-0000",
                "address": null,
                "city": null,
                "state": null,
                "zip_code": null,
                "notes": null,
                "timezone": "UTC",
                "language": "english",
                "id_roles": "2",
                "services": [
                    "1"
                ],
                "settings": {
                    "username": "janedoe",
                    "password": "90245039aa524b37ea43742896af2870337e942566402e644b2a860ed3d48636",
                    "salt": "9b048590adcc4d78fc47ab8c2279bba7631ecbb96b61f686c32e5bde5aa1a9b7",
                    "working_plan": "{\"monday\":{\"start\":\"09:00\",\"end\":\"18:00\",\"breaks\":[{\"start\":\"14:30\",\"end\":\"15:00\"}]},\"tuesday\":{\"start\":\"09:00\",\"end\":\"18:00\",\"breaks\":[{\"start\":\"14:30\",\"end\":\"15:00\"}]},\"wednesday\":{\"start\":\"09:00\",\"end\":\"18:00\",\"breaks\":[{\"start\":\"14:30\",\"end\":\"15:00\"}]},\"thursday\":{\"start\":\"09:00\",\"end\":\"18:00\",\"breaks\":[{\"start\":\"14:30\",\"end\":\"15:00\"}]},\"friday\":{\"start\":\"09:00\",\"end\":\"18:00\",\"breaks\":[{\"start\":\"14:30\",\"end\":\"15:00\"}]},\"saturday\":{\"start\":\"09:00\",\"end\":\"18:00\",\"breaks\":[{\"start\":\"14:30\",\"end\":\"15:00\"}]},\"sunday\":{\"start\":\"09:00\",\"end\":\"18:00\",\"breaks\":[{\"start\":\"14:30\",\"end\":\"15:00\"}]}}",
                    "working_plan_exceptions": null,
                    "notifications": "1",
                    "google_sync": "0",
                    "google_token": null,
                    "google_calendar": null,
                    "sync_past_days": "30",
                    "sync_future_days": "90",
                    "calendar_view": "default"
                }
            },
            "service": {
                "id": "1",
                "name": "Service",
                "duration": "30",
                "price": "0.00",
                "currency": "",
                "description": null,
                "location": null,
                "availabilities_type": "flexible",
                "attendants_number": "1",
                "id_service_categories": null
            },
            "customer": {
                "id": "4",
                "first_name": "Sara",
                "last_name": "James",
                "email": "sara.james@gmail.com",
                "mobile_number": null,
                "phone_number": "1345678900",
                "address": "Cookie St. 139-2",
                "city": "Hsinchu",
                "state": null,
                "zip_code": "311",
                "notes": null,
                "timezone": "Asia/Taipei",
                "language": "english",
                "id_roles": "3"
            }
        }
    ],
    "unavailability_events": []
}

```

Thus, we can see that the meeting we have set up was easily exposed and the private data of our mock appointment can be retrieved without any special credentials.

Then, we can retrieve the nginx logs, that are stored in our docker container at:

```
/var/log/nginx/application.access.log
```

During the log analysis, we can see that one can identify the attack by this line:

```
- [12/Dec/2023:09:08:53 +0000] "POST /index.php/backend_api/ajax_get_calendar_events HTTP/1.1" 200 2262 "-" "python-requests/2.31.0"

```

We can recognize the line as suspicious as the IP address is unrecognised. We can also see that the return was successful, and 2262 bytes of data were returned. If we had more data in our database, most likely the amount of data would be even greater. We can also see that python-requests part is rather suspicious as it lets us suspect that a script has been executed.

Thus, the CVE-2022-0482 is a high severity vulnerability as it exposes private information before checking the proper credentials. Even though the patch was quickly released and the vulnerability has been resolved in the later versions, EasyAppointments could not issue an automatic update nor send a notification to the users, thus there are still many users that might have their data exposed and the danger is not completely rectified.

Bibliography

National Vulnerability Database. n.d. 10 10 2023. <https://nvd.nist.gov/vuln/detail/CVE-2022-0482>.

huntr by ProtectAI. n.d. 10 10 2023. <https://huntr.com/bounties/2fe771ef-b615-45ef-9b4d625978042e26/>.

DockerHub. n.d. 10 10 2023. < https://hub.docker.com/r/vanhack/easyappointments>.

GitHub. n.d. 10 10 2023. < https://github.com/alextselegidis/easyappointments>.