Demo Company
Security Assessment Findings Report

Date: November 19th, 2022

Contact Information

Name	Title	Contact Information
NUWE x Schneider Electric		
Josep Añó Gosp	Josep_glic	Email: josepengineer@gmail.com Github:
Alos	CallMeR00t	Email: carlosalos1995@gmail.com Github:
Sonia	Soni	Email: so.peinadoasensi@gmail.com Github:

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	18.170.119.133

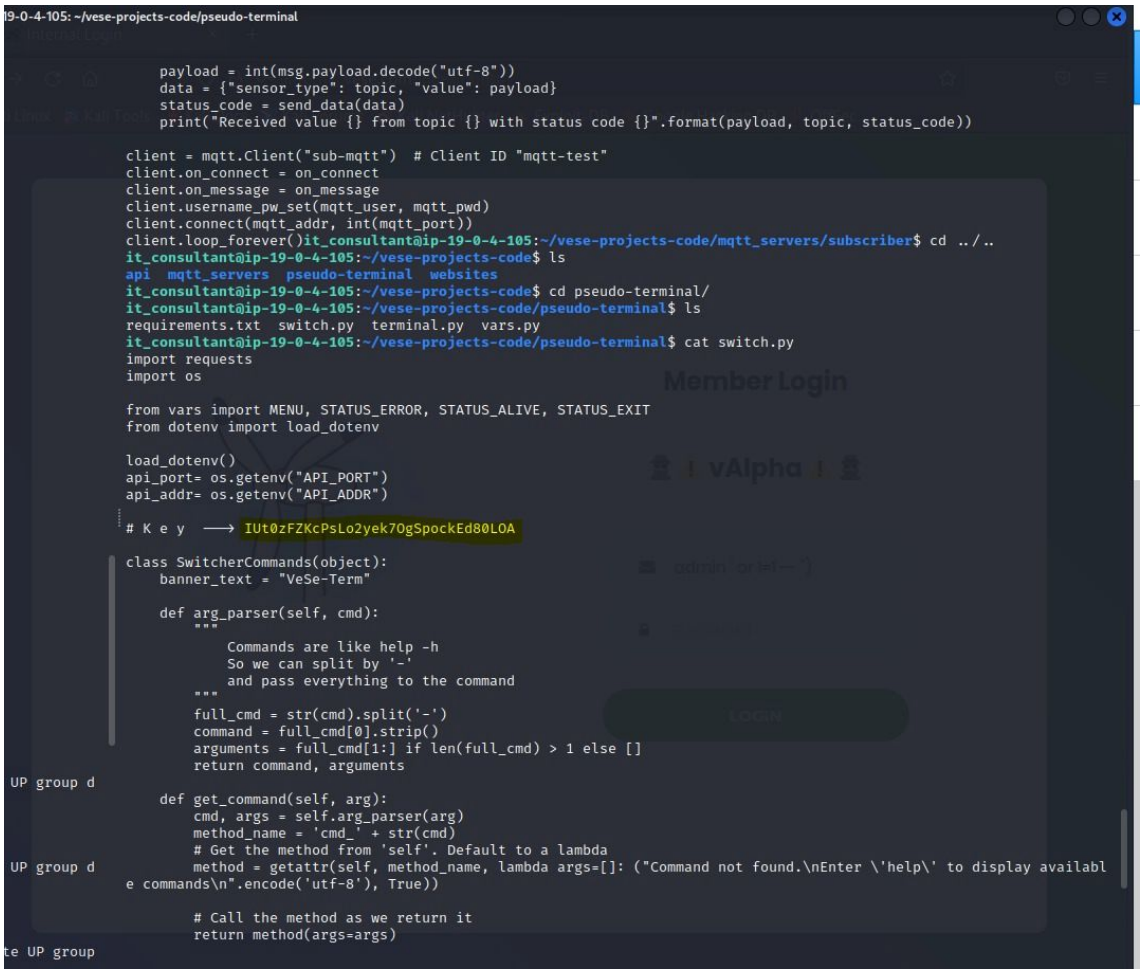
Security Audit Findings

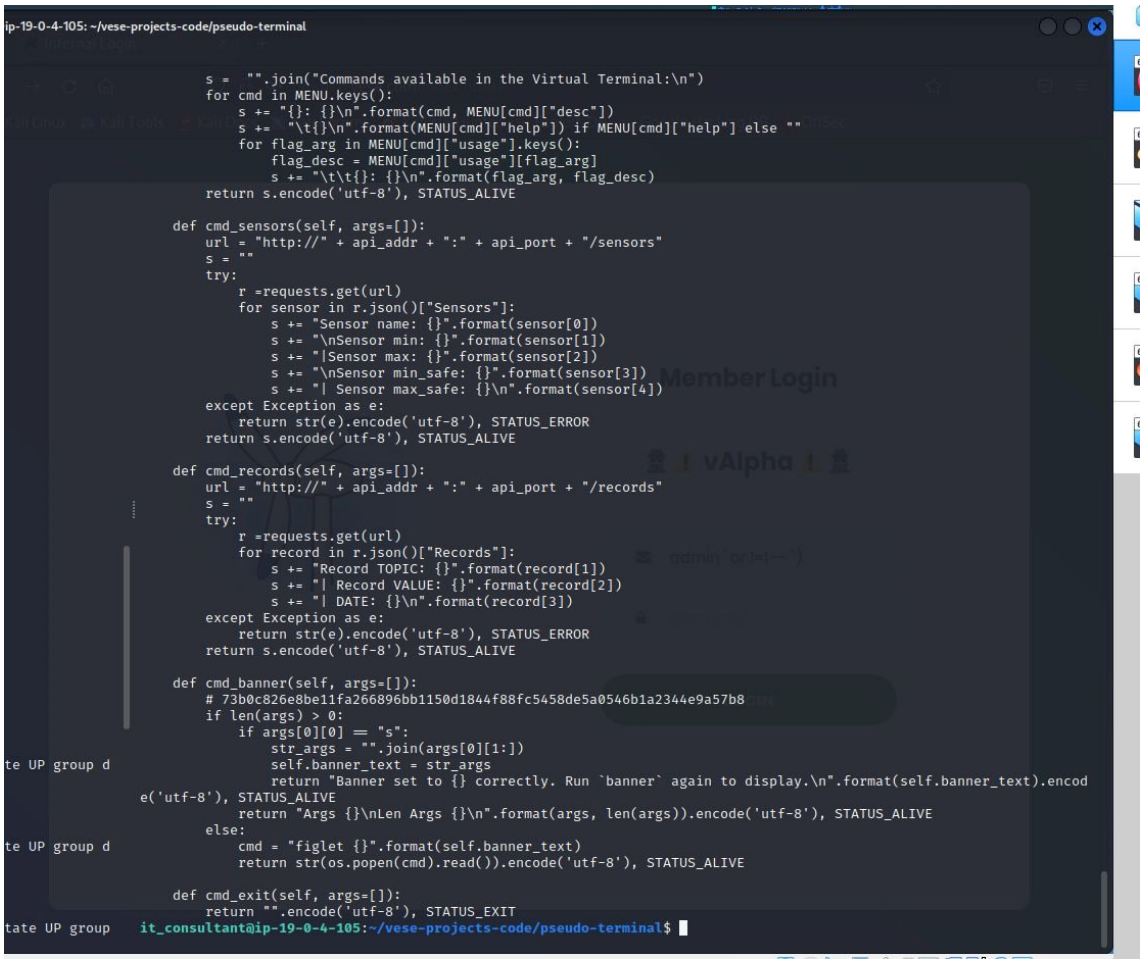
Python script with privileges–switch.py(Severity)

Description:	Vulnerable script in switch.py
Impact:	**High**

Exploitation Proof of Concept.

Python script has root permissions, and it can execute commands.





Remediation

Who:	IT Team
Vector:	Remote, Physical…
Action	Item 1: Deny root permisions Item 2: Encrypt data in plain text, as it poses a high risk.

Security Audit Findings

Python code injection with root permissions – pseudo-terminal/switch.py (Critical)

Description:	Python vulnerability code injection. The script contains commands that use the python library popen, this library allows using execute commands, just using this variable “banner_text” in Telnet connecting to 127.0.0.1:6969. Moreover, this script has root permissions.
Impact:	Critical

Exploitation Proof of Concept.

switch.py file is studied.

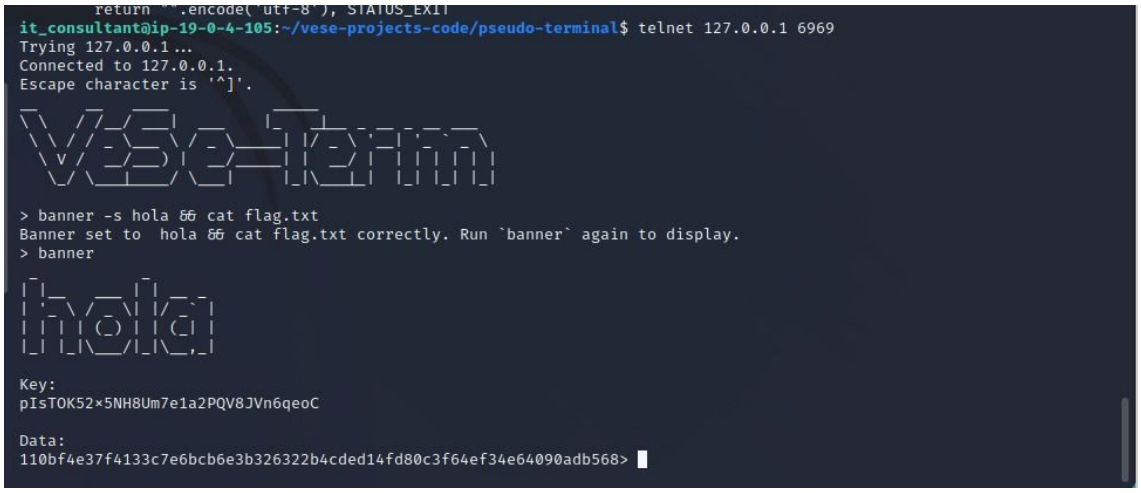
1-Python script has root permissions.

2- Studying the code the library uses the library popen. This allows to execute commands.

3- The place where the popen is executed inside the script is filled with variable strings to create a banner.

4- Using telnet following how the script works, the banner is changed using “banner -s”

5- Adding “string” && “command” to banner, it creates an injection of commands. Allowing using other commands and the root permissions creating an exploit.



Remediation

Who:	IT Team
Vector:	Remote, Physical…

Action:	Item 1: Deny root permisions Item 2: Deleting return condition Item 3: Deny telnet connection Item 4: Put **cmd = False** to avoid execute more commands on the system Additional Recommendations: Block the IP that tries to do the telnet

Security Audit Findings

Lick information – websites/internals/index.html (Severity)

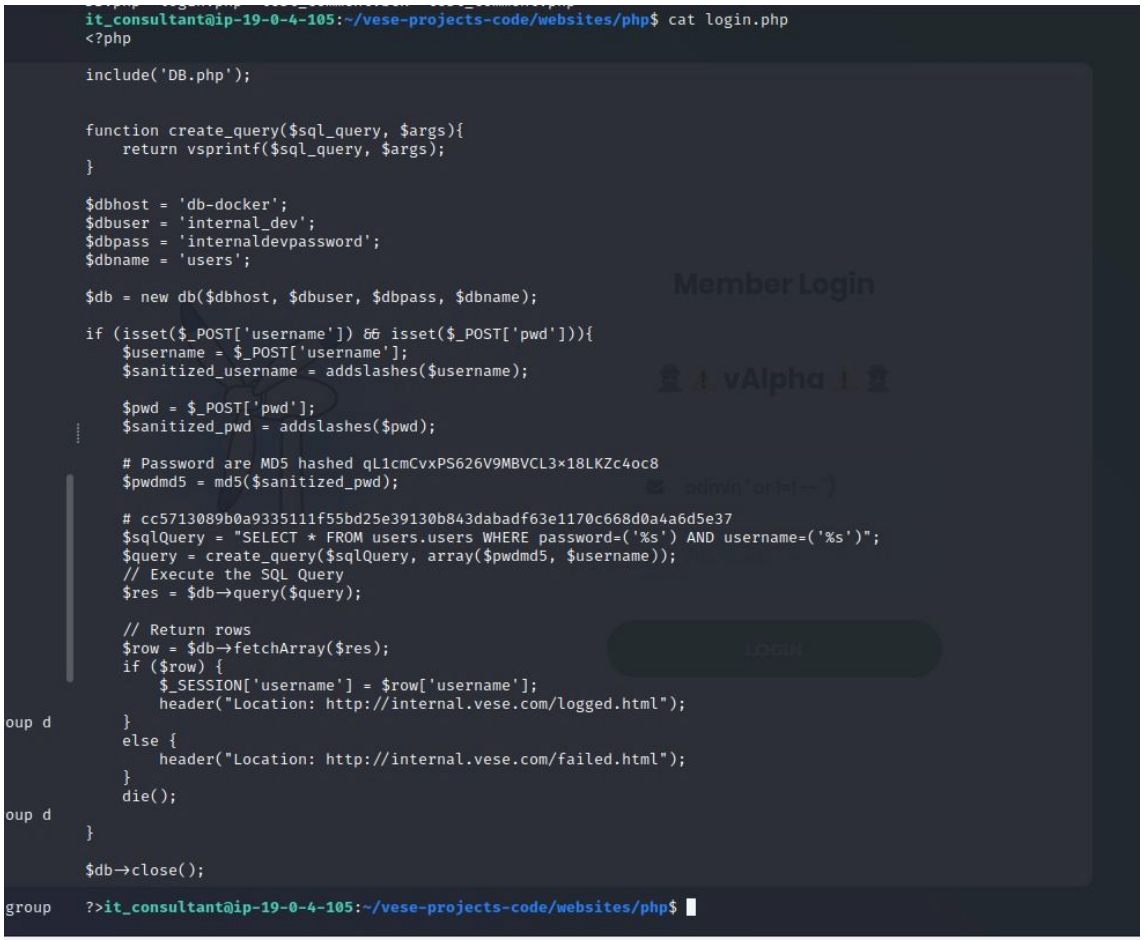
Description:	Licked data in login.php, main index.html.
Impact:	Low

Exploitation Proof of Concept.

1-Index.html inside /websites/internal/ has a form type, it calls a login.php action.

2- Login.php may be vulnerable to brute force attacks doing SQL injection on the login form.

```
it_consultant@ip-19-0-4-105:~/vese-projects-code/websites/internal$ cat index.html
<!DOCTYPE html>
<html lang="en">

<head>
        <title>Internal Login</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <link rel="icon" type="image/png" href="images/icons/favicon.ico" />

        <link rel="stylesheet" type="text/css" href="vendor/bootstrap/css/bootstrap.min.css">

        <link rel="stylesheet" type="text/css" href="fonts/font-awesome-4.7.0/css/font-awesome.min.css">

        <link rel="stylesheet" type="text/css" href="vendor/animate/animate.css">

        <link rel="stylesheet" type="text/css" href="vendor/css-hamburgers/hamburgers.min.css">

        <link rel="stylesheet" type="text/css" href="vendor/select2/select2.min.css">

        <link rel="stylesheet" type="text/css" href="css/util.css">
        <link rel="stylesheet" type="text/css" href="css/main.css">

        <meta name="robots" content="noindex, follow">
</head>

<body>
        <div class="limiter">
                <div class="container-login100">
                        <div class="wrap-login100">
                                <div class="login100-pic js-tilt" data-tilt>
                                        <img src="images/wind.jpeg" alt="IMG">
                                </div>
                                <form class="login100-form validate-form" action="login.php" method="POST">
                                        <span class="login100-form-title">
                                                Member Login
                                                <!--K___E__Y-->
                                                <!--nujnlhrZZKidXugUkCtiUgqDMuoDbnA3-->
                                        </span>
                                        <span class="login100-form-title">
                                                vAlpha
                                        </span>
                                        <div class="wrap-input100 validate-input" data-validate="Username must not b
e empty">

                                                <input class="input100" type="text" name="username" placeholder="Use
rname">

                                                <span class="focus-input100"></span>
                                                <span class="symbol-input100">
                                                        <i class="fa fa-envelope" aria-hidden="true"></i>
                                                </span>
```



```
it_consultant@ip-19-0-4-105:~/vese-projects-code/websites/php$ cat login.php
<?php

include('DB.php');


function create_query($sql_query, $args){
    return vsprintf($sql_query, $args);
}

$dbhost = 'db-docker';
$dbuser = 'internal_dev';
$dbpass = 'internaldevpassword';
$dbname = 'users';

$db = new db($dbhost, $dbuser, $dbpass, $dbname);

if (isset($_POST['username']) && isset($_POST['pwd'])){
    $username = $_POST['username'];
    $sanitized_username = addslashes($username);

    $pwd = $_POST['pwd'];
    $sanitized_pwd = addslashes($pwd);

    # Password are MD5 hashed qL1cmCvxPS626V9MBVCL3×18LKZc4oc8
    $pwdmd5 = md5($sanitized_pwd);

    # cc5713089b0a9335111f55bd25e39130b843dabadf63e1170c668d0a4a6d5e37
    $sqlQuery = "SELECT * FROM users.users WHERE password=('%s') AND username=('%s')";
    $query = create_query($sqlQuery, array($pwdmd5, $username));
    // Execute the SQL Query
    $res = $db->query($query);

    // Return rows
    $row = $db->fetchArray($res);
    if ($row) {
        $_SESSION['username'] = $row['username'];
        header("Location: http://internal.vese.com/logged.html");
    }
    else {
        header("Location: http://internal.vese.com/failed.html");
    }
    die();
}

$db->close();

?>it_consultant@ip-19-0-4-105:~/vese-projects-code/websites/php$
```

Remediation

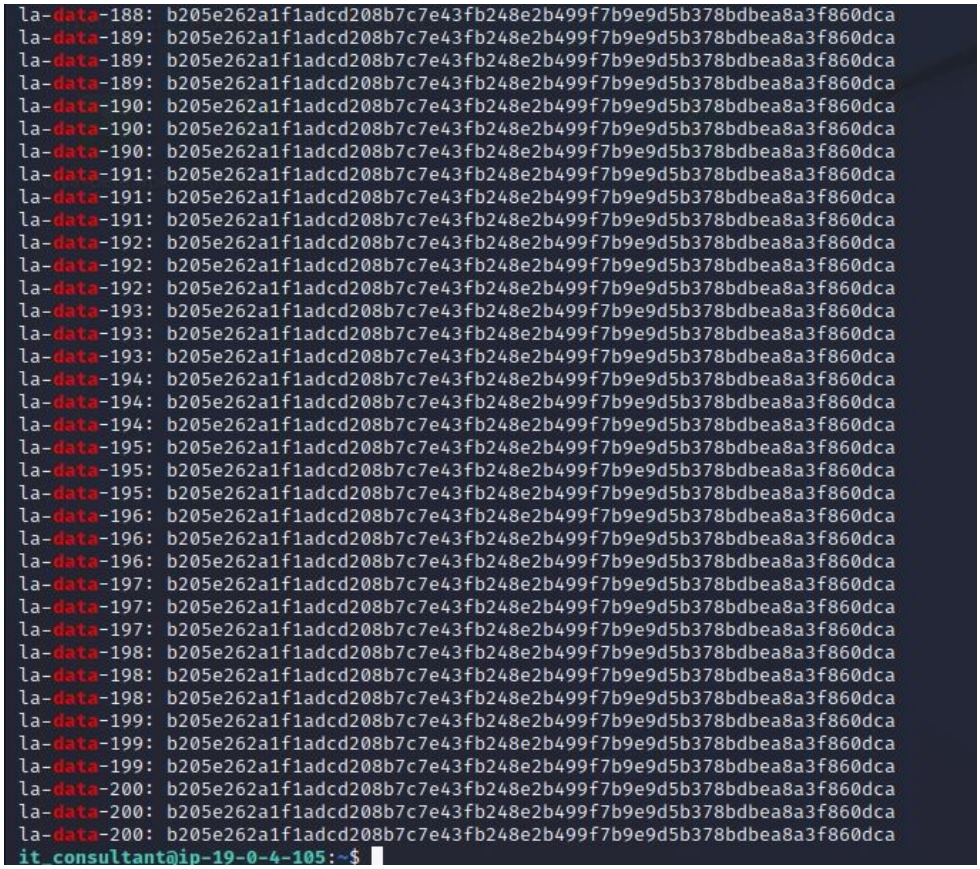
Who:	IT Team
Vector:	Remote, Physical…
Action:	Item 1: Deny root permisions Item 2: Add lock after several attempts. For example: after X failures, lock the account for X minutes. Item 3: Encrypt data in plain text, as it poses a high risk.

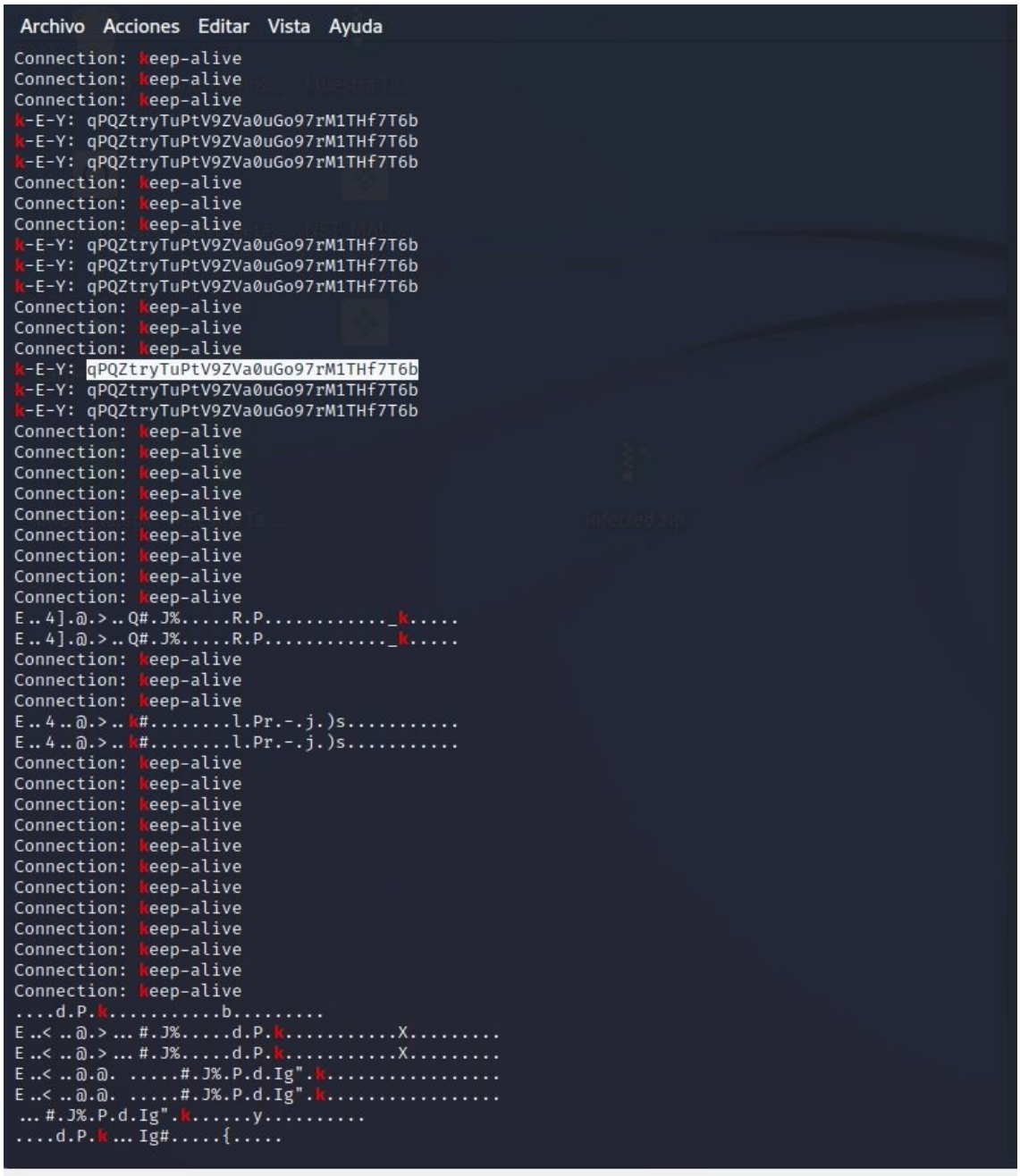
Security Audit Findings

Pcap command– dump_2022_11_19, tcpdump.pcap (Severity)

Description:	Pcap, analyzes the traffic of the entire attack
Impact:	Informational

Exploitation Proof of Concept.





Remediation

Who:	IT Team
Vector:	Remote, Physical…
Action:	Item 1: Deny root permisions Item 2: Encrypt data in plain text, as it poses a high risk.

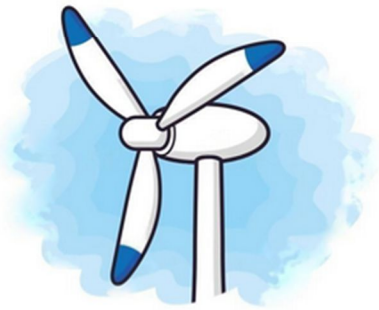
Security Audit Findings

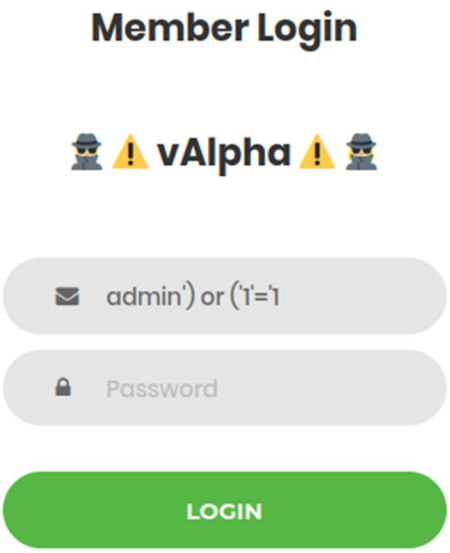
SQL injection–http://internal.vese.com(Severity)

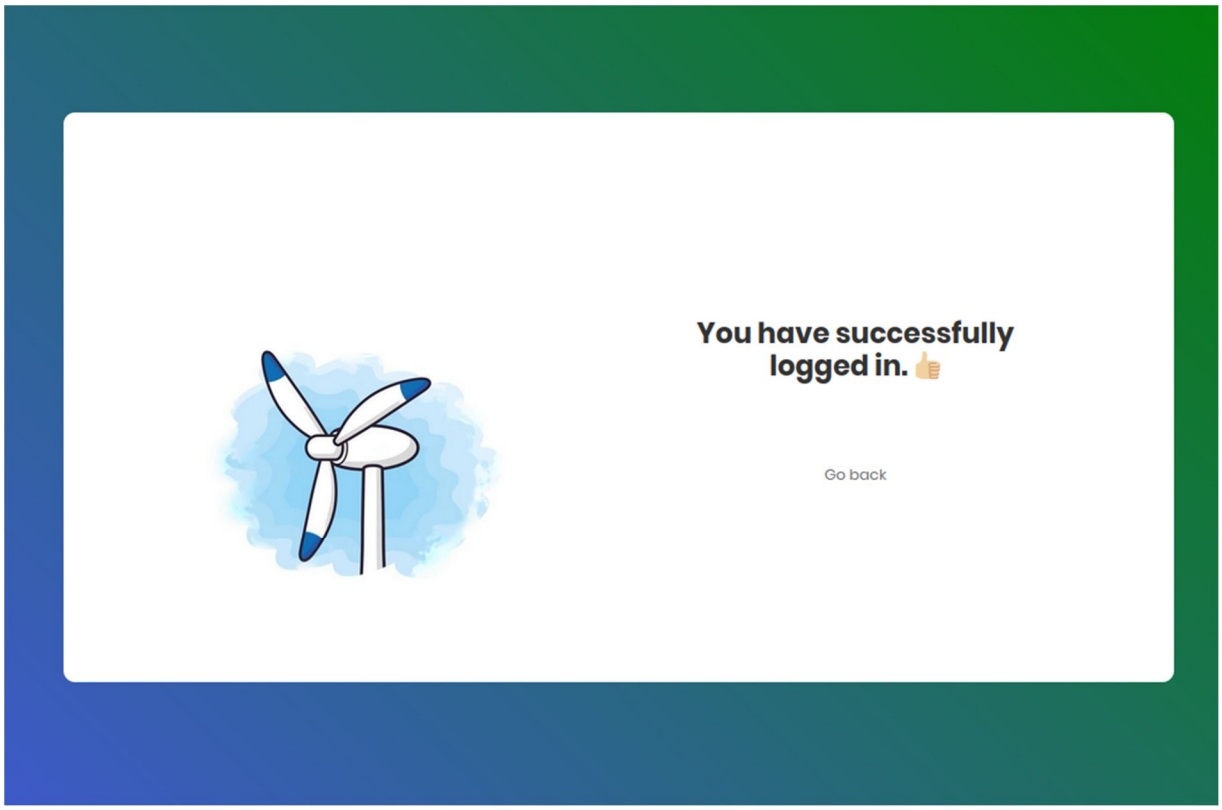
Description:	Maridb database is vulnerable to SQL injection in login in the website http://internal.vese.com
Impact:	Critical

Exploitation Proof of Concept.

1- **Login as admin') OR ('1'='1 creates SQL Injection**
2- **This allows to introduce any kind of SQL command to obtain information needed**







Remediation

Who:	IT Team
Vector:	Remote, Physical…
Action:	Item 1: Update security version of Mariadb Item 2: Not allow multiple login tries Item 3: Block Ips who tries to login multiple time Item 4: Block Ips by regex which try to SQL injection

Exploitation Paths

1- **Attackers get initial access via API web using SQL Injection in http://internal.vese.com**
2- **Obtain persistence inside the script test_comment.php, wih a shell reverse when parameters are introduced as fields name, mail, and comment in contact field in http://vese.com**
3- **They can execute commands with root privileges using a vulnerability in python code injection using the function “popen”, which allows executing code in OS. The script terminal.py had root privileges and was running.**